An Elitist Multiobjective Simulated Annealing

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

Subhamoy Chakraborti

Under the supervision of

Dr. Sanghamitra Bandyopadhyay
Machine Intelligence Unit

Indian Statistical Institute
203, Barrackpore Trunk Road
Kolkata-700108.

July 7, 2003

Indian Statistical Institute

203 Barrackpore Trunk Road
Kolkata 700 108

Certificate of Approval

This is to certify that this thesis titled **"An Elitist Multiobjective Simulated Annealing"** submitted by Subhamoy Chakraborti toward partial fulfillment of requirements for the degree of M.Tech in Computer Science at Indian Statistical Institute, Kolkata embodies the embodies the work done under my supervision.

Sanghamitra Bandyopadhyay

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

Acknowledgements

I would like to take this opportunity to thank Dr. Sanghamitra Bandyopadhyay for the continuous guidance, support as well as criticism that I received throughout the last year, and I feel fortunate enough to have her as a mentor.
I also take pleasure in thanking my classmates, friends and last but not the least, my mother, whose continuous support helped me sail through the rough days.

Subhamoy Chakraborti
Indian Statistical Institute
Kolkata 700 108

CONTENTS

Chapter 1

Introduction

Throughout our life, we make decisions, with or without conscious thought. This decision may be as simple as selecting the color of the dress that we are going out with or as difficult as those involved in designing a missile. The former decision may be taken in a fraction of a second, while the latter one might take several years. The main goal of this latter kind of decision-making is to minimize cost as well as maximize gain, where gain might be defined in different ways while dealing with different kinds of problems. In other words, problems related to optimization of different criteria are widely prevalent in real-life. Development of optimization algorithms has therefore been of great challenge in computer science. The problem is compounded by the fact that in many situations one may need to optimize several objectives simultaneously. These problems are known as multiobjective optimization problems. The present work deals with development of some such complex *multiobjective optimization algorithms*.

1.1 MultiObjective Optimization Problem

Taking the example from [15], we may think of the case of purchasing a *car*. The purchaser wishes to satisfy the following criterion: *minimizing* the cost, insurance premium and weight and *maximizing* the feel good factor while in the car. The purchaser also wants the car to have a good stereo system, seats for six adults and a mileage of 20kmpl. If we view this situation in mathematical model, the available cars are the problem's *decision variables*, the conditions to be met are the constraints and the process of minimizing and maximizing the criterion is called *optimization*. An *objective function* based on the decision variables is used to determine an associated vector representing how *well* some particular vehicle satisfies the criterion. Because multiple objectives are simultaneously considered, this problem is known as MultiObjective Optimization Problem (MOOP).

In the same manner, in most of the real world problems we face, we have to simultaneously optimize two or more different objectives, which are often competitive in nature. Finding a single solution in these cases is very difficult, if not impossible. In this kind of problems, one way of thinking might be to optimize each criterion separately. In some earlier works, efforts were made to convert the multiobjective problem to a single objective problem. But it may so happen that optimizing one objective lead to some unacceptable low value of the other objective(s). Thus we need to treat all the objectives together, which needs a detailed analysis.

In the world of management, this type of problem is known as multiple criterion decision making (MCDM). To make things clear, we provide another example of decision making involved in selecting mode of transport. Suppose we want to travel from a place A to another place B. If we avail a bus, we have to pay Rs. 5.00. If we go by a minibus, we go a bit quicker, but have to expend Re. 1.00 more. If we want to go faster we need to pay Rs.10.00 for traveling in super fast bus. And if we are in real hurry, we have to go by a taxi, which will take Rs.50.00. Now if the cost is the only objective of this decision-making process, then the optimal solution is getting the bus. Again if time were the only factor everyone would have availed taxi. But if we do not have any such constraints, then we can avail any mode of transport and in real world, this problem becomes a true multiobjective kind of problem. Because here between any two solutions, one is better than the other in terms of one objective, but at the same time it is worse in terms of the other objective.

Thus we are not in a position to get a single solution, which would be the *best*. In general, in MOOP we can hardly have a single solution; rather most of the time, we have to settle for a set of alternative optimal solutions. They are optimal in the sense that no other solutions in the search space are superior to them when all the objectives are considered.

The set of solutions of an MOOP consists of all the decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degradation in another - these vectors are known as *Pareto optimal*.

Webster's dictionary defines the term *effective* as the production or the power to produce an acceptable result; *efficient* is defined as acting in such a way as to avoid resource loss or waste in functioning. The goal of any algorithm that intends to solve the MOOP should be to *achieve the Pareto-optimal set effectively and efficiently*.

Evolutionary algorithms like genetic algorithms etc. are widely used as effective search and optimization tools in many problem domains. Simulated annealing is another such optimization technique that is based on the principle of statistical mechanics. Conventionally these algorithms have been applied for single criterion optimization. In recent times, attempts have been made to modify these algorithms, so that they are able to perform multiobjective optimization. These attempts have targeted the evolutionary algorithms more because of their population based nature. Only limited attempts have been made in case of simulated annealing since it is essentially a point by point search. In this dissertation, we try to alleviate the disparity in the literature by developing an efficient multiobjective simulated annealing algorithm.

1.2 Choice of Simulated Annealing

Evolutionary algorithms are one of the popular search methods. They mimic the metaphor of natural biological evolution [37]. Evolutionary algorithms operate on a population and by applying the principle of survival of the fittest, they produce better and better approximations to a solution. Simulated Annealing (SA) is another powerful technique for finding good solutions to a wide variety of combinatorial optimization problems. The concept of simulated annealing originates from the principles of the annealing procedure, which is a physical process. As MOOPs have many solutions, till now evolutionary algorithm (EA) has been the natural choice [2] for solving complex MOOPs instead of Simulated Annealing (SA). In the literature, there are very few SA-based MOOP-solving algorithms. However, this dissertation attempts to use SA in the area of MOOPs. The reason behind this choice of SA is the powerfulness of this tool, which stems from its good selection technique and annealing scheme. Another reason why SA performs well is annealing, that is, the gradual temperature reduction technique [5].

1.3 Quality Measure

Another issue in the study of MOOP that is gaining much importance nowadays is that of assessment of the *quality* of the solution obtained by a specific algorithm, i.e., once an 'Observed Pareto set' is obtained by an algorithm, it is usually of great interest to know to what extent the observed solution set represents the *true Pareto frontier*. In single-objective optimization, quality can be simply defined by means of the objective function: the smaller (or greater) the value, the better the solution. In contrast, quality is itself multiobjective in the presence of several criteria in MOOP. Because of this, there is not even a universally accepted definition of "optimum"' as in single-objective optimization, which makes it difficult to even compare results of one method to another. Normally the decision about what the *best* answer is corresponds to the so-called (human) decision maker.

As it is often not clear what we mean by quality in case of several optimization criteria - closeness to Pareto front, uniform distribution on the front or coverage, until recently graphical plots were used for comparing the outcomes of Multiobjective Evolutionary Algorithms (MOEA) and other Multiobjective Optimization (MOO) strategies [15]. But visual and intuitive quality assessment can often be misleading and in some cases impossible when the number of design objectives is more than three.

As suggested in [19], we may notice that there are two distinct goals in any MOOP. One of them is to find solutions as close to the Pareto Optimal front as possible and the other one is to try to disperse them as much as possible on the obtained non-dominated front. In some sense these two goals are orthogonal to each other, as the first one searches for points towards the Pareto optimal front and the latter one along the front. A good algorithm that intends to solve MOOP, should try to achieve both of these two goals. We can also think of another goal in this context: an algorithm should try to maximize the number of solutions obtained.

1.4 Goal of the Dissertation

This dissertation attempts to provide an insight into the world of MOOP, discuss some basic concepts dealing with MOOP. Also it gives a review of the existing *Measures* to check the nicety of a solution and introduces two new concepts regarding this area. The other goal of this work is to introduce a new elitist algorithm that solves MOOP. The basic tool that has been used in this work to solve MOOP is simulated annealing, which has not been explored to a great extent in this area till date. The proposed algorithm has been tested on different problems, and also compared with two relevant existing algorithms: Pareto Archived Evolution Strategy (PAES) and Multiobjective Simulated Annealing (MOSA). The comparison has been done using three existing measures Error Ratio, Spacing and Spread. The problems have also been tested using our proposed measure. The results show that the proposed algorithm performs quite satisfactorily compared to these two existing algorithms. Also a different kind of problem called Minimal Deceptive Problem (MDP), that deceives the optimization algorithms from finding pareto optimal solutions, has been explored in this work and an instance of MDP has been proposed and tested.

1.5 Scope of the Present Work

The remainder of this document is organized as follows: in Chapter 2, we discuss the basic theories of MOOP. After that we give a brief introduction to simulated annealing and the reasons behind its choice as the basic tool in our work. Chapter 4 deals with some existing relevant works, viz. SPEA [4], MOSA [5] and PAES [10]. We give the outline of our proposed algorithm in the next chapter, i.e., Chapter 5. Chapter 6 is devoted to measures. In Chapter 7, we present the performance comparisons of our proposed algorithm with respect to MOSA and PAES. The scope and possible direction of future work in this field has been explored in Chapter 8. References are provided at the end.

Chapter 2

Issues in MultiObjective Optimization Problem

2.1 Introduction

In this chapter, we introduce the basics of multiobjective optimization problem (MOOP). For any farther discussion on MOOP, we need to be conversant with terminologies like dominance relation, pareto optimality. In Section 2.2, we first formally define multiobjective optimization problem. Then in Section 2.3, we introduce the concepts of dominance relation and pareto optimality. Using a simple figure, we try to explain these concepts. We extend these ideas in Section 2.4 and Section 2.5 to discuss nondominated set, strong and weak dominance. Also we have used the concepts of ranking in comparing the solution set obtained by different algorithms while solving traveling salesman problem. For this reason, ranking has been defined in Section 2.6. Section 2.7 concludes this chapter.

2.2 Formal Definition of MOOP

An MOOP has more than one objective function, which are to be optimized simultaneously. Like single objective optimization problem, the problem has a number of constraints that constitute the feasible solution space. A general MOOP can be described as a vector function f that maps a tuple of m parameters (decision variables) to a tuple of n objectives.

Formally:

$$\min/\max y = f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$$

$$\text{where } x = (x_1, x_2, x_3, \ldots, x_m) \in X$$

$$y = (y_1, y_2, y_3, \ldots, y_n) \in Y$$

Here x is the decision vector, X is the parameter space, y is the objective vector and Y is the objective space. It is often referred to as vector optimization because a vector of objectives f = (f_1, f_2, …, f_M), instead of a single objective is being optimized. These multiple objectives often conflict with each other. The opposing objectives place a partial, rather than total ordering of the search space.

2.3 Dominance Relation and Pareto Optimality

An important concept of multiobjective optimization is that of *domination*. Most multiobjective optimization techniques use the concepts of dominance relation and Pareto optimality. We give the formal definitions here corresponding to a maximization problem. The definitions are easily extended to minimization problems.

Dominance Relation

Let us consider two vectors *a and b* $\in X$.
Then *a is said to dominate b* iff
$\forall i \in \{1,2,...n\} : f_i(a) \geq f_i(b) \wedge \exists j \in \{1,2,...n\} : f_j(a) > f_j(b)$
i.e., for all vector functions f_i, *a* has a higher or equal value than that of *b* and also there exists at least one vector function f_j for which *a*'s value is strictly greater than that of *b*.

We can explain this using a two-objective optimization problem. It has five different solutions, as shown in the figure below.
We also assume that the objective function f_1 needs to be maximized while the objective function f_2 needs to be minimized. Five solutions having different values of the objective functions are shown in Figure 1.

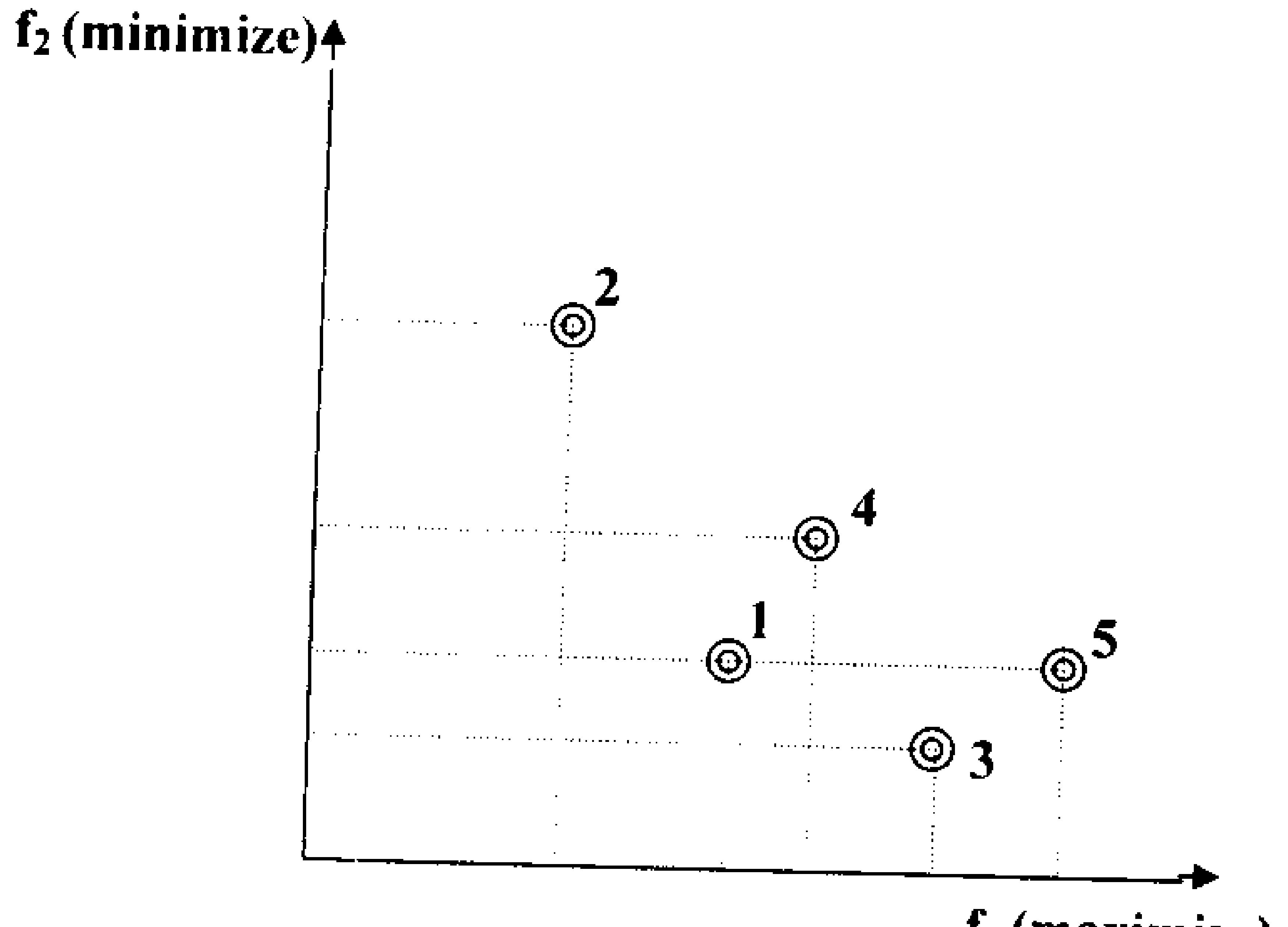

FIGURE 1: Example of dominance, pareto optimality

Now we can use the definition of *domination* to decide which solution is better between any two given solutions in terms of both objectives. Considering solution 1 and solution 2, we find that solution 1 is better than solution 2 in objective function f_1 as well as in f_2. Thus we may write that solution 1 is better than solution 2, i.e., solution 1 *dominates* solution 2. Again if we compare solution 1 and solution 5, solution 5 is better than solution 1 in terms of objective function f_1 and they have same value in terms of objective function f_2. In this case also, we may write solution 5 *dominates* solution 1. Intuitively, we can say that if a solution *a* dominates another solution *b*, then the solution *a* is better than *b* in the parlance of multiobjective optimization. Thus the concept of *domination* allows us to compare different solutions with multiple objectives.

Properties of Dominance Relation

The different binary relation properties of the dominance operator are as follows:

Reflexive: The dominance relation is not reflexive, since any solution p does not dominate itself. The second condition of the definition is not satisfied in this case.
Symmetric: The dominance relation is also not symmetric, because if *a* dominates *b*, it does not imply that *b* dominates *a*. Actually the opposite is true. Thus, the dominance relation is asymmetric.
Antisymmetric: Since the dominance relation is not symmetric, it cannot be antisymmetric.
Transitive: The dominance relation is transitive. This is because if *p* dominates *q* and *q* dominates *r*, then *p* dominates *r*.
Another interesting property that the dominance relation possess is that if a solution *p* doest not dominate solution *b*, this does not imply that *b* dominates *a*.

2.4 Nondominated Set

All decision vectors that are not dominated by any other decision vector of a given set are called nondominated with regard to this set. Therefore the set of non-dominated solutions with respect to the entire parameter space constitute the Pareto-optimal front or the Pareto-optimal set. As stated earlier, the goal of a multiobjective optimization technique should be to find this Pareto front *efficiently and effectively*.

We focus on the previous figure (Figure 1) once again. If we compare the solutions 3 and 5, we find that solution 5 is better than solution 3 in terms of objective function f_1, but solution 3 is better than solution 5 in terms of the objective function f_2. Thus we cannot conclude from the definition of *dominance*, which solution is the better one. Thus they are nondominated with respect to each other. The set of all such points constitute the pareto optimal set. In Figure 1, solution 3 and solution 5 constitute the pareto optimal set.

2.5 Strong Dominance and Weak Pareto-Optimality

Strong Dominance

The dominance relation previously defined is sometimes referred to as *weak* dominance relation. This definition can be modified and a strong relation can be defined as follows:
A solution *a* strongly dominates a solution *b* if solution *a* is strictly better than solution *b* in all the objectives. Now if we again refer back to the previous figure (Figure 1), we may find that solution 5 does not strongly dominate solution 1, as it is not *strictly better* than solution 1 in terms of objective function f_1, though it weakly dominates solution 1. Thus we may write that if a solution *a* strongly dominates another solution *b*, then it also holds weak domination relation, but not vice versa. The definition of strong dominance may be used to define a *weakly nondominated set*.

Weakly nondominated set

Among the set of solutions P, the weakly nondominated set of solutions P' are those that are not strongly dominated by any other member of the set P. The above definition suggests that a weakly nondominated set found from a set of P solutions contains all members of the nondominated ser obtained by using the definition of nondominated set from the same set P.

2.6 Rank

The rank of a solution 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.

2.7 Conclusions

The concepts discussed in this chapter are the very basic ideas that are used throughout this work. To follow the flow of any work on MOOP, these definitions are essential as all the works on MOOP evolves around these basic ideas. The following chapter discusses some existing MOOP solving algorithms. All of them require the concepts discussed in this chapter.

Chapter 3

Simulated Annealing

3.1 Introduction

The tool that has been used in this work for solving MOOP is simulated annealing. Many researchers have done work in the area of multiobjective optimization, using evolutionary algorithms. Though simulated annealing has a strong theoretical background and also been in the literature for a longer time than evolutionary algorithms, it has been rarely used in the study of MOOP. We have made an attempt in this direction by proposing an algorithm using simulated annealing as its underlying tool. The basic concepts of simulated annealing have been discussed in this chapter and also the reasons behind the selection of this particular tool, in spite of evolutionary algorithm, have been explained. In Section 3.2, we introduce the basic principles of simulated annealing. Then in Section 3.3 a pseudo code of simulated annealing has been provided. Section 3.4 explains about the reasons behind selecting simulated annealing in solving MOOP.

3.2 Basic Principles of Simulated Annealing

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution [37]. Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximation is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Evolutionary algorithms work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

The structure an evolutionary algorithm is shown in the following figure.

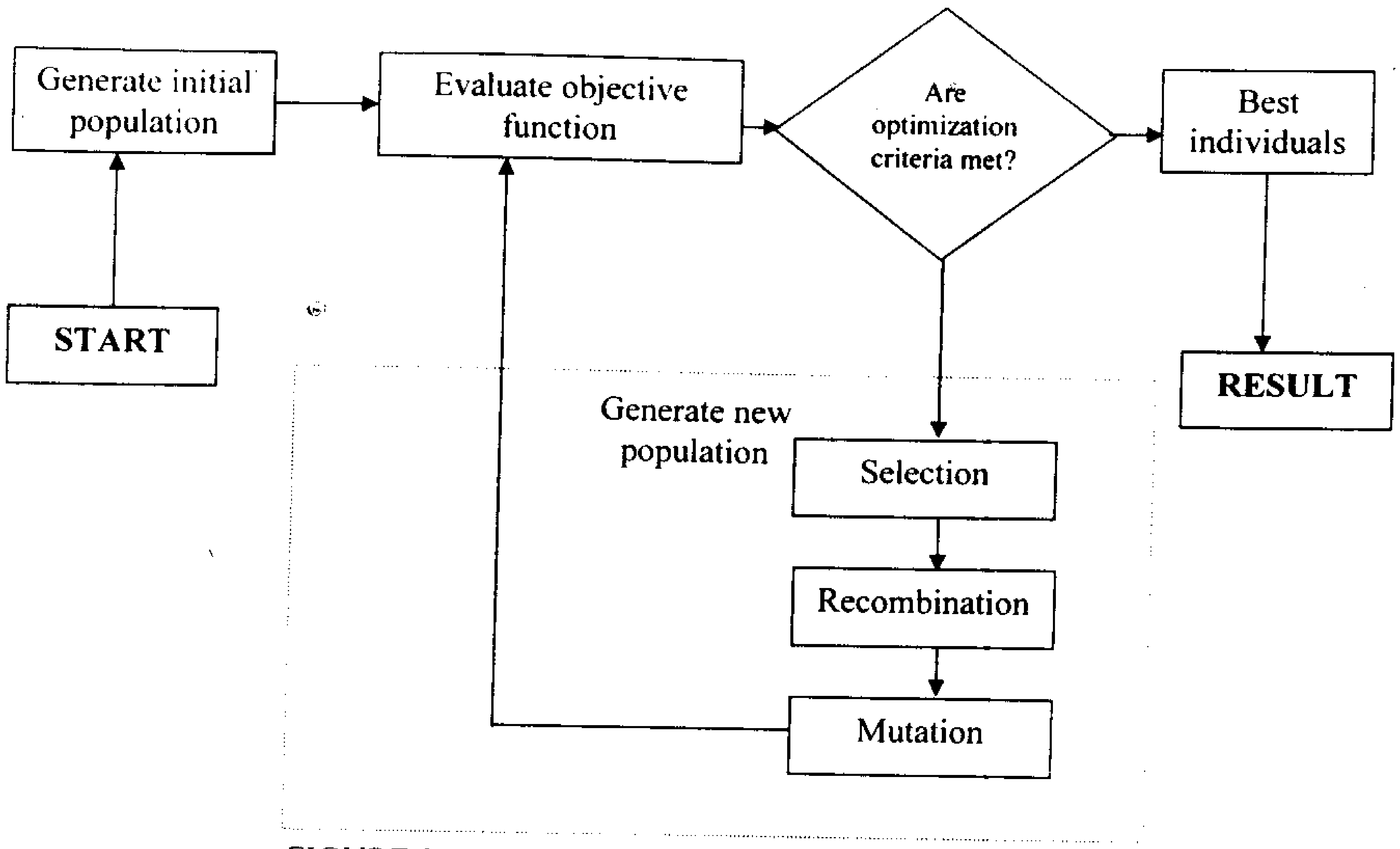
Generate initial population
Evaluate objective function
Are optimization criteria met?
Best individuals
START
RESULT
Generate new population
Selection
Recombination
Mutation

FIGURE 2: Structure of an Evolutionary Algorithm

The EA is started with an initial population set. Then it is evaluated according the objectives of the problem. If the optimization criteria are already met by this set of population, then we select the best individuals. Otherwise new population is generated by using different operators like selection, recombination or mutation.

Simulated Annealing (SA) is another technique for finding good solutions to a wide variety of combinatorial optimization problems. It mimics the principles of the annealing procedure, which is a physical process where a crystal is cooled down from the liquid to the solid phase. If the cooling is done slowly enough, the energy state of the crystal at the end will be very close to its minimum value. Simulation of this cooling may be done with the Metropolis algorithm. Simulated Annealing generates sequences of configurations in the following way: given a current configuration C_i with energy E_i, the next configuration C_j with energy E_j is generated by applying a small perturbation in C_i. If $(E_j - E_i)$ is less than or equal to zero, then C_j is accepted as the current configuration. Otherwise, it is accepted with a probability $\exp(-(E_j - E_i)/k_\beta T)$, where T and k_β represent the temperature and Boltzmann's constant respectively. If the lowering of temperature is done slowly enough, the crystal reaches thermal equilibrium at each temperature.

The SA process may be viewed as a graph with an energy E assigned to each node. Here, the nodes are called states, the arcs represent moves from one state to a neighboring state and the energy is equivalent to cost. The algorithm starts from a random initial configuration at high temperature. It then proceeds by generating new candidate states and accepting/rejecting them according to a probability, which is a function of the current temperature and energy difference. The temperature is gradually reduced to a minimum value, while the system settles down to a stable low energy state [38].

The correspondence between the physical aspect of SA and an optimization problem has been nicely defined in [39]. The parameters of the search space are encoded in strings (usually binary) and these represent the different states; low energy states correspond to near optimal solutions; the energy corresponds to objective function, and the temperature is a controlling parameter of the system. The important tasks are to establish a way of representing and generating different configurations (or states) of the problem and an annealing schedule. The primary objective of SA is to find the global minima of a cost function that characterizes large and complex systems. The main motivating idea behind this powerful tool is while optimizing a very large and complex system, i.e., a system with many degrees of freedom, instead of always going downhill, try to go downhill most of the time [27]. The major difficulty in implementation of the algorithm is that there is no obvious analogy for the temperature T with respect to a free parameter in the combinatorial problem. Furthermore, avoidance of entrapment in local minima (quenching) is dependent on the "annealing schedule". The annealing schedule is determined by the choice of the initial temperature, number of iterations to be performed at each temperature, and the rate of temperature decrement at each step.

3.3 Algorithm of SA

The most general algorithm of simulated annealing may be written as shown below: [39]

```
1.  Begin
2.      Generate the initial string randomly = $q$
3.      $T = T_{max}$
4.      Let $E(q,T)$ be the associated energy
5.      while ($T \geq T_{min}$)
6.          for $i = 1$ to $k$
7.              Mutate (flip) a random position in $q$ to yield $s$
8.              Let $E(s,T)$ be the associated energy
9.              Set $q \leftarrow s$ with probability $\frac{1}{1+e^{-(E(q,T)-E(s,T))/T}}$
10.             end for
11.             $T = rT$
12.         end while
13.         Decode the string $q$ to provide the solution of the problem.
14. End
```

In this annealing process, a new state is chosen with a probability

$$p_{qs} = \frac{1}{1+e^{-(E(q,T)-E(s,T))/T}}$$

The parameters of the search space are encoded in the form of a bit string of a fixed length. The objective value associated with the string is computed and mapped to its energy. The string with the minimum energy value provides the solution to the problem. The initial string (say q) of 0s and 1s is generated randomly and its energy value is computed. Keeping the initial temperature high (say $T = T_{max}$), a neighbor of the string (say s) is generated by randomly flipping one bit. The energy of the new string is computed and it is accepted in favor of q with a probability p_{qs}, mentioned earlier. This process is repeated a number of times (say k) keeping the temperature constant. Then the temperature is decreased using the equation $T = rT$, where $0 < r < 1$, and the k loops, as earlier, are executed. This process is continued till a minimum temperature (say T_{min}) is attained. The temperature-reducing schedule may be chosen different from the one mentioned above. Various cooling schedules are available that can be used with a simulated annealing optimization.

Simulated Annealing has been successfully applied in various domains [40]. The domains include computer design [41] [42], image restoration and segmentation [43][44], combinatorial problems such as traveling salesman problem [45] and artificial intelligence [46]. It is, however, not always trivial to map an optimization problem into the simulated annealing framework. The difficulties come from constructing an objective function that encapsulates the essential properties of the problem and that can be efficiently evaluated, determining a concise description of the parameter configurations and an efficient method for generating configurations and selecting an effective and efficient annealing schedule.

3.4 MOOP and SA

As MOOPs have many solutions, evolutionary algorithm (EA) has been widely used [2] for complex MOOPs instead of Simulated Annealing (SA). Up to now there are few, if any, alternative to EA-based multiobjective optimization. But we need to modify them to suite our purpose. When we consider the case of finding a set of nondominated solutions rather than a single-point solution, MultiObjective Evolutionary Algorithms (MOEA) have to perform a multimodal search that samples the Pareto-optimal front uniformly, which cannot be done by a simple elitist EA that tends to converge towards a single point. To overcome this problem, several methods have been proposed. The success of EA approaches in multiobjective optimizations is mainly based on the population concept with the ability of finding multiple optima simultaneously, which matches the idea of MOOPs.

However SA, which reports good performance in single-objective optimization problems, has been seldom used for multiobjective problems. The main reason is that SA usually finds one solution instead of a set of solutions and this is a critical handicap in MOOP, although SA has some favorable characteristics for multimodal search. The powerfulness of SA stems from its good selection technique and annealing scheme. Another reason why SA performs well is annealing, that is, the gradual temperature reduction technique. As the temperature and the cost difference mainly determine the amount of mutation in generating the next searching point, SA can do local fine-tuning towards the end of search to give finer results. However the disadvantage of SA is the long annealing time.

There are some algorithms to take care of this issue like Fast Simulated Annealing (FSA), Very Fast Simulated Re-annealing (VFSR), New Simulated Annealing (NSA) etc. [30][31][32].

But there has been little research into using the SA technique for multiobjective optimization. The first and most significant problem is that SA uses only one search agent. As solving multiobjective problem generally requires finding all the solutions at the same time, some researchers have thought of using multiple search agents at the same time, though SA was originally designed to use only one single search agent.

3.5 Conclusions

SA algorithm has been hardly used for multiobjective optimization because of these initial ideas. This is known to be a critical weakness of SA as it betrays the philosophy of multiobjective optimization – searching for all the Pareto solutions instead of only one solution. As the result of this weakness, SA has remained as one of the improper or not favorable algorithms for multiobjective optimization.

In our work, attempt has been made to use this powerful tool (i.e., Simulated Annealing) in solving MOOPs. Among the works studied in course of this dissertation, multiobjective simulated annealing [5] is an important one, as it is one of those very few algorithms that use simulated annealing in solving MOOP. But our proposed algorithm uses concepts of *clustering* and *archive*, which are not there in [5]. However, some previous works exists that have explored these concepts. The relevant works are discussed in the following chapter.

Chapter 4

Some Existing MultiObjective Optimization Techniques

4.1 Introduction

In the process of working out our algorithm, we studied most of the recent relevant works in the field of MOOP. It is not possible to discuss all of them in this report. We provide brief descriptions of the three works that were most motivating in nature and encouraged us to think in our way. They are Strength Pareto Evolutionary Approach [4], Multiobjective simulated annealing [5] and Pareto Archived Evolution Strategy [10]. Section 4.2 deals with Strength Pareto Evolutionary Approach. Multiobjective simulated annealing is discussed in the next section. Section 4.4 explores Pareto Archived Evolution Strategy. The discussion on these present researches helps identify the motivation behind the present work.

4.2 Strength Pareto Evolutionary Approach (SPEA)

We first study the Strength Pareto Evolutionary Approach (SPEA), proposed in [4]. It is similar to other multiobjective evolutionary approaches in the following respects:

• Uses the concept of Pareto dominance in order to assign scalar fitness value to individuals.

• Performs clustering to reduce the number of nondominated solutions stored without destroying the characteristics of the trade – off front.

• Stores the nondominated solutions found so far externally.

But it is unique in the following respects

• It combines the above 3 techniques in a single algorithm

• The fitness of an individual is determined only from the solutions stored in the external nondominated set.

• All solutions in the external set participate in the selection process.

• A new niching technique is provided in order to preserve diversity. This method is Pareto based and does not require any distance parameter like the niche radius for sharing.

Algorithm of SPEA:

The algorithm of SPEA deals with a population. The detailed *pseudo code of SPEA* has been given below.

```
1.Generate an initial population P and create an empty external nondominated set P'.
2.Copy nondominated members of P to P'
3.Remove solutions within P' which are covered by any other member of P'
```

4.If the of externally stored nondominated solutions exceed a given maximum N', prune P' by means of clustering.
5.Calculate the fitness of each individuals in P and P'.
6. Select the individuals in P+P' (multiset union), until the mating pool is filled. In this study binary tournament selection with replacement is used.
7.Apply problem specific crossover and mutation operators as usual.
8.If the maximum number of generations is reached, then stop, else go to step 2.

Fitness Assignment Procedure

The idea behind the *Fitness Assignment* in SPEA is a novel one. It is a two-stage process. First, the individuals in the external nondominated set P' are ranked. Afterwards, the individuals in the population P are evaluated. The steps are:

Step 1:Each solution $i \in P'$ is assigned a real value s_i in [0, 1), called strength; s_i is proportional to the number of population members j in P for which i covers j. Let n denote the number of individuals in P that are covered by i and let us assume N to be the size of P. Then s_i is defined as $s_i = n / (N + 1)$. The fitness f_i of i is equal to its strength: $f_i = s_i$.

Step 2:The fitness of an individual j in P is calculated by summing he strengths of all external non-dominated solutions i in P' that cover j. One is added to the total in order to guarantee that members of P' have better fitness than members of P.

Here we have to keep in mind that fitness is to be minimized, i.e., small fitness values correspond to high reproduction probabilities.

$$f_j = 1 + \sum_{i, i \geq j} s_i \quad \text{where } f_j \in [1, N)$$

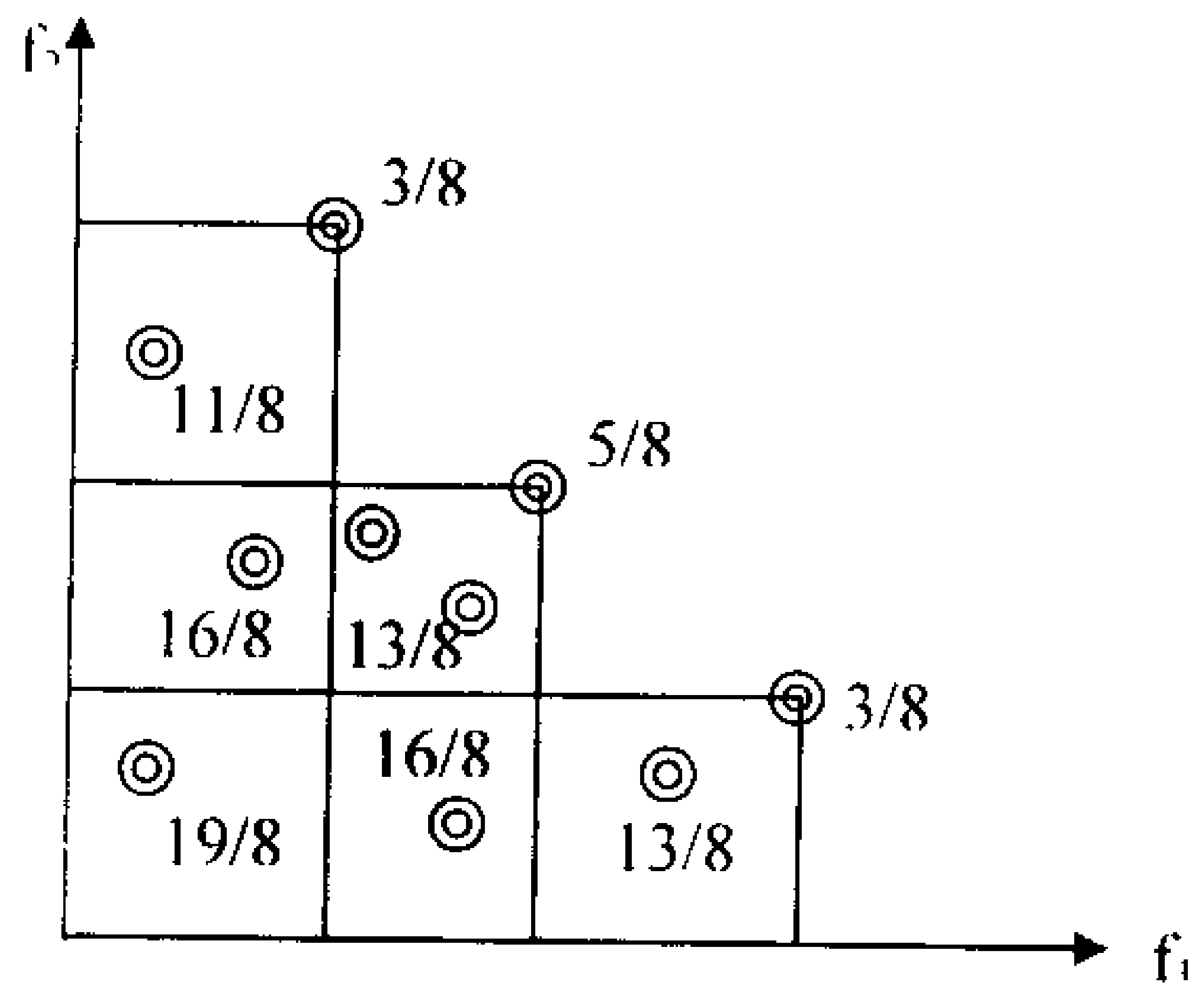

FIGURE 3: First example of fitness assignment by SPEA

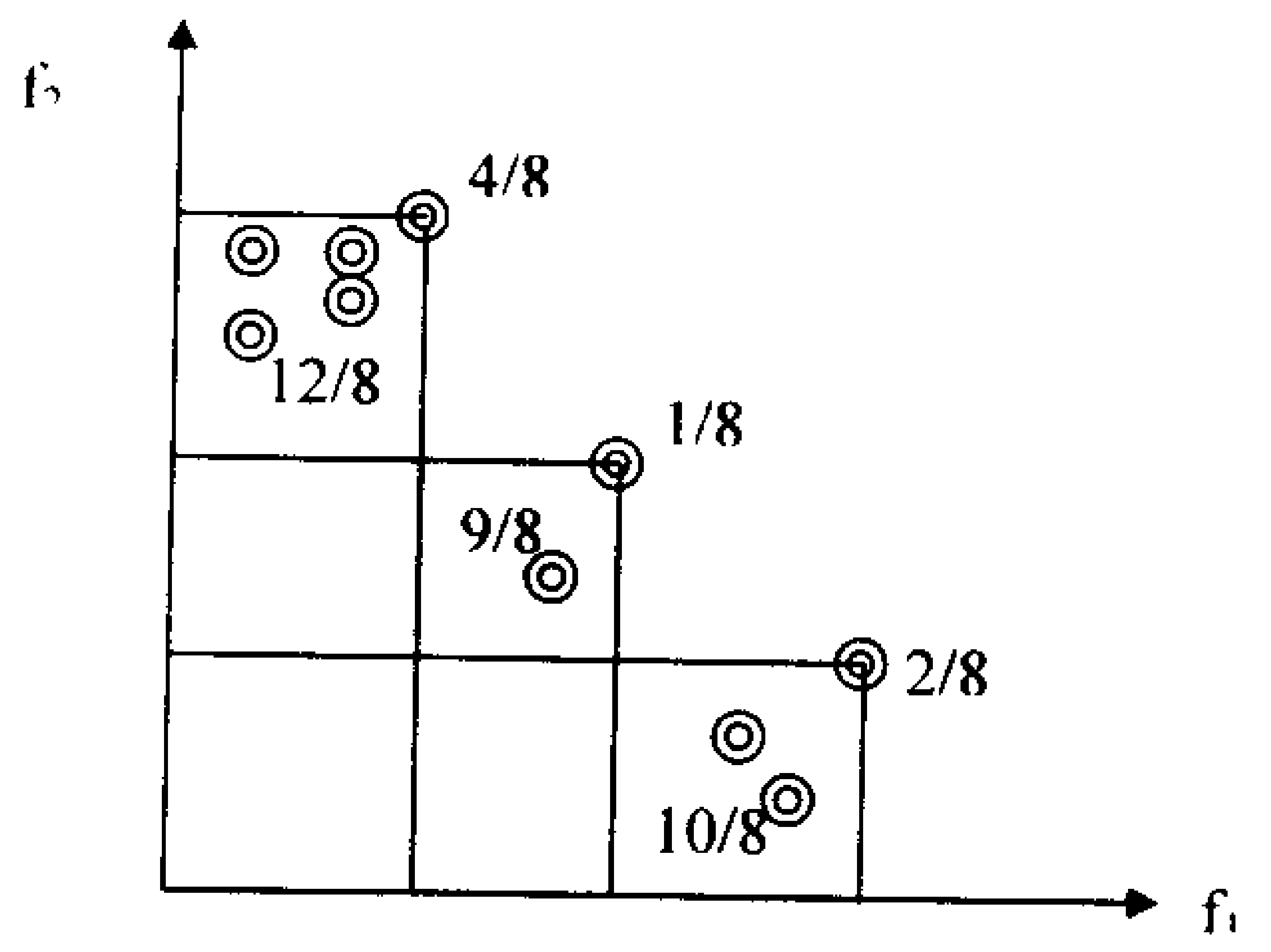

FIGURE 4: Second example of fitness assignment by SPEA

In Figure 3, we find that individuals located near the Pareto front achieve better fitness values than the remaining members. Figure 4 provides an example of the fact that individuals having many neighbors in their niche are penalized due to the high strength value and the associated nondominated point. Thus this algorithm more or less reflects the idea of preferring individuals near the Pareto-optimal front and distributing them at the same time along the trade-off surface.

Reducing the Pareto Set by Clustering

In certain problems, the Pareto-optimal set can be extremely large or even contain an infinite number of solutions. However, from the decision maker's point of view, presenting all nondominated solutions found is useless when their number exceeds reasonable bounds. A method that has been applied to this problem successfully and studied extensively in the same context is cluster analysis [33][34].
The basic problem that is handled in clustering analysis is that of dividing the given data set into K clusters, where the value of K may be known or unknown, in such a manner that the divisions provides the natural clusters in the data set. We note that for a given data set, there may exist several meaningful clusterings. Clustering is very much an experiment oriented art because the performance of an algorithm depends mainly on 1) type of data being analyzed, 2) chosen measures of similarity and 3) method of identifying clusters in the data. In this work, Average Linkage method has been chosen. In our algorithm, we took the concept of clustering from this work. However, we used Single Linkage clustering technique.

4.3 MultiObjective Simulated Annealing (MOSA)

In [5], Nam. D and Park. C.H [5] proposed this algorithm, where a possible way to use simulated annealing schemes in solving MOOP has been discussed. MOSA uses the domination concept and the annealing scheme for efficient search. The main obstacle for SA in multiobjective optimization is its inability to find multiple solutions. However SA can do the same work by repeating the trials as it converges to the global optima with a uniform probability distribution in the single objective optimization. When there are two global optima, it is proved that SA can find each optimum with probability 0.5 [29]. When this fact is also true in multiobjective optimization, SA has advantages over EAs because it does not need large memory to keep the population; nor does it use additional algorithms to spread the solution over the Pareto frontier. Additionally MOSA can find a small group of Pareto solutions in a short time with the demand of urgent simulation and then find more solutions by repeating the trials for detailed information about the Pareto frontier.

Algorithm of MOSA:

The general SA algorithm involves basic three steps. First, the objective function corresponding to the energy function need to be identified. Second, we need to select a proper annealing scheme consisting of decreasing temperature with increasing of iterations. Third, a method of generating a neighbor near the current search position is needed. In single objective optimization problems, the transition probability scheme is generally selected by the Metropolis and logistic algorithm [28], [29]. However the situation is different in MOOP. Choosing a proper transition probability is difficult in this case. The algorithm proposed by the authors in [5] to solve this MOOP is outlined below:

```
$s = s_0$
$T = T_0$
**repeat**
        Generate a neighbor s' = N (s)
        **if** c (s') dominates c (s)
                Move to s'
        **else if** c (s) dominates c (s')
                Move to s' with transition probability
        **else if** c (s) and c (s') do not dominate each other
                Move to s'
        **end if**
        T = annealing (T)
**end repeat** (until the termination status is satisfied)
```

Here s represents the current search position and T is the temperature parameter, which is gradually decreased as time goes on. A new search position s' is generated by the N(s) function, its cost is evaluated and compared with the previous cost. When it is determined to be a good solution by the domination test, the new state is accepted. Even when the new position is not proper (meaning the new state is dominated by the current state), it is accepted with some acceptance probability. When there is no superiority between the current state and the next state, the new state is accepted instead of the current one because moving in the non-dominated situation help increase the spread performance and evade the local optima. In their paper they followed geometric cooling as follows: $T_k = \alpha^k \cdot T_0$ where $0<\alpha<1$ is the cooling rate. Regarding transition probability, general transition rules such as the Metropolis or logistic method cannot be applied directly to the multiobjective problems as they support only a scalar cost criterion for the multiobjective cost function. The transition probability from state i to state j is,

$p_t(i, j) = \min\{\exp(-c(i, j)/T), 0\}$

Where c(i, j) is the cost criterion for transition from state i to state j and T is the annealing temperature. Six criterions for MOSA have been suggested: they are as follows

1. $c(i,j) = \min(c_k(j) - c_k(i))$
2. $c(i,j) = \max(c_k(j) - c_k(i))$
3. $c(i,j) = \sum(\alpha k(c_k(j) - c_k(i)))$
4. $c(i,j) = \sum c_k(i)$
5. $c(i,j) = \dfrac{\sum(c_k(j) - c_k(i))}{D}$
6. $c(i,j) =$ fixed value

When the new state is at the same level as the current state, there can exist two schemes – move to the new state or stay in the current state. It can be easily understood that the move scheme is better than the stay scheme as the search will be continued into the middle part of the frontier, move freely between non-dominated states like a random walk when the temperature is low and eventually will be distributed uniformly over the Pareto frontier as time goes on.

4.4 Pareto Archived Evolution Strategy (PAES)

Knowles J. D. and Corne D. W have introduced a simple evolution scheme for MOOPs, called PAES. The algorithm in its simplest form is a (1+1) evolution strategy employing local search but using a reference archive of previously found solutions in order to identify the approximate dominance ranking of the current and candidate solution vectors. PAES was initially developed as a multiobjective local search for finding solutions to the off-line routing problem, which is an important problem in the area of telecommunications routing optimization.

Algorithm of PAES:

The (1+1)-PAES algorithm is outlined below:

```
1. Generate initial random solution c and add it to the archive
2. Mutate c to produce m and evaluate m
3.      if (c dominates m)      discard m
4.      else if (m dominates c)
5.          replace c with m and add m to the archive
6.      else if (m is dominated by any member of the archive)      discard m
7.      else apply test (c, m, archive) to determine which becomes the new
                                  current solution and whether to add m to the archive
8.      until a termination criterion has been reached, return to line 2
```

PAES is comprised of basic three parts: the candidate solution generator, the candidate solution acceptance function and the nondominated-solutions (NDS) archive. The candidate solution generator is akin to simple random mutation hillclimbing; it maintains a single current solution and at each iteration, produces a single new candidate via random mutation. Since the aim of multiobjective search is to find a spread of nondominated solutions, PAES needs to provide an NDS list to explicitly maintain a limited number of these, as and when they are found by the hillclimber. The design of the acceptance function is obvious in the case of the mutant dominating the current solution or vice versa but is troublesome in the nondominated case. In their approach, they have used a comparison set to help decide between the mutant and the current solution in the latter case. The NDS archive provides a natural and convenient source from which to obtain comparison sets. Pseudocode indicating the procedure for determining whether to accept or reject the mutant solution and for deciding whether it is archived or not is given below.

```
1. if the archive is not full
2.         add m to the archive
3.         if (m is in a less crowded region of the archive than c)
4.                 accept m as the new current solution
5.         else maintain c as the current solution
6. else
```

```
7.          if (m is in a less crowded region of the archive than x for some member x
                 on the archive)
8.          add m to the archive, and remove a member of the archive from the most
        crowded region
9.               if (m is in a less crowded region of the archive than c)
10.                    accept m as the new current solution
11.              else maintain c as the current solution
12.        else
13.              if (m is in a less crowded region of the archive than c)
14.                    accept m as the new current solution
15.              else maintain c as the current solution
```

However, it has been acknowledged by the authors of PAES that the idea of maintaining a list of nondominated solution is not new. Parks and Miller [35] recently described a MOGA that also maintains an archive of nondominated solutions.

4.5 Conclusions

Our work has been influenced by all of these three works to some extent or the other. The concept of using simulated annealing in our work was influenced by [5]. The concept of *clustering* was used in [4] also. The work on PAES [10] provided a good algorithm to compare our algorithm with. Also it used the concept of *archive* in MOOP. The following chapter introduces the new proposed algorithm. We recognize the influence of these algorithms on our way of thinking. However, the novelty of the algorithm would also be evident from the discussion provided in the following chapter.

Chapter 5

The proposed Elitist Multiobjective Simulated Annealing

5.1 Introduction

In our approach, as already mentioned, we have used SA to solve MOOP. This was a challenge as not much work has been done yet in this direction. Only a few works [5] are available in the literature, which have solved MOOP using simulated annealing. In this chapter, we describe the proposed EMOSA algorithm for multiobjective optimization. The algorithm also uses the concepts of archive and clustering which, as already explained, has been used in existing algorithms. Section 5.2 introduces the basic principles of the Multiobjective Simulated Annealing. The proposed algorithm uses single linkage clustering as the clustering tool. Section 5.3 is devoted to this clustering technique. The main algorithm behind selection is explained in Section 5.4 and Section 5.5 concludes the discussion.

5.2 Basic Principles of Elitist Multiobjective Simulated Annealing (EMOSA)

This algorithm is elitist in nature. Whenever we get a solution that is nondominated, we keep the solution, thus following the principle of elitism. We have used an archive in our algorithm to store the nondominated solutions found so far. It is specified by two limits – one is the hard limit and the other is the soft limit. We go on selecting new points according to our algorithm. When the number of points exceeds the soft limit, clustering is done to reduce the number of points to match the hard limit. Note that we have used two limits, while PAES works with a single limit. We have done this way, to make the clustering effective. We have adopted single linkage clustering technique for our work. Below we briefly describe the principles of single linkage clustering.

5.3 Single Linkage Clustering

Single linkage clustering is an example of agglomerative clustering technique. In general agglomerative clustering technique, if there are *t* points, then *t* clusters are assumed to be present at level 0. The number of clusters at the ith level is (t-i) for i=0,1,2, ...(t-1). Ultimately there would be one cluster at the (t-1)th level. If the number of clusters K is known, then the process will be stopped when K clusters result in. Single linkage clustering is a kind of this agglomerative clustering technique. Here the distance between any two clusters is given by the value of the length of the shortest link between the two clusters [48].

The mechanism of single linkage clustering is explained by the following figure (Figure 5).

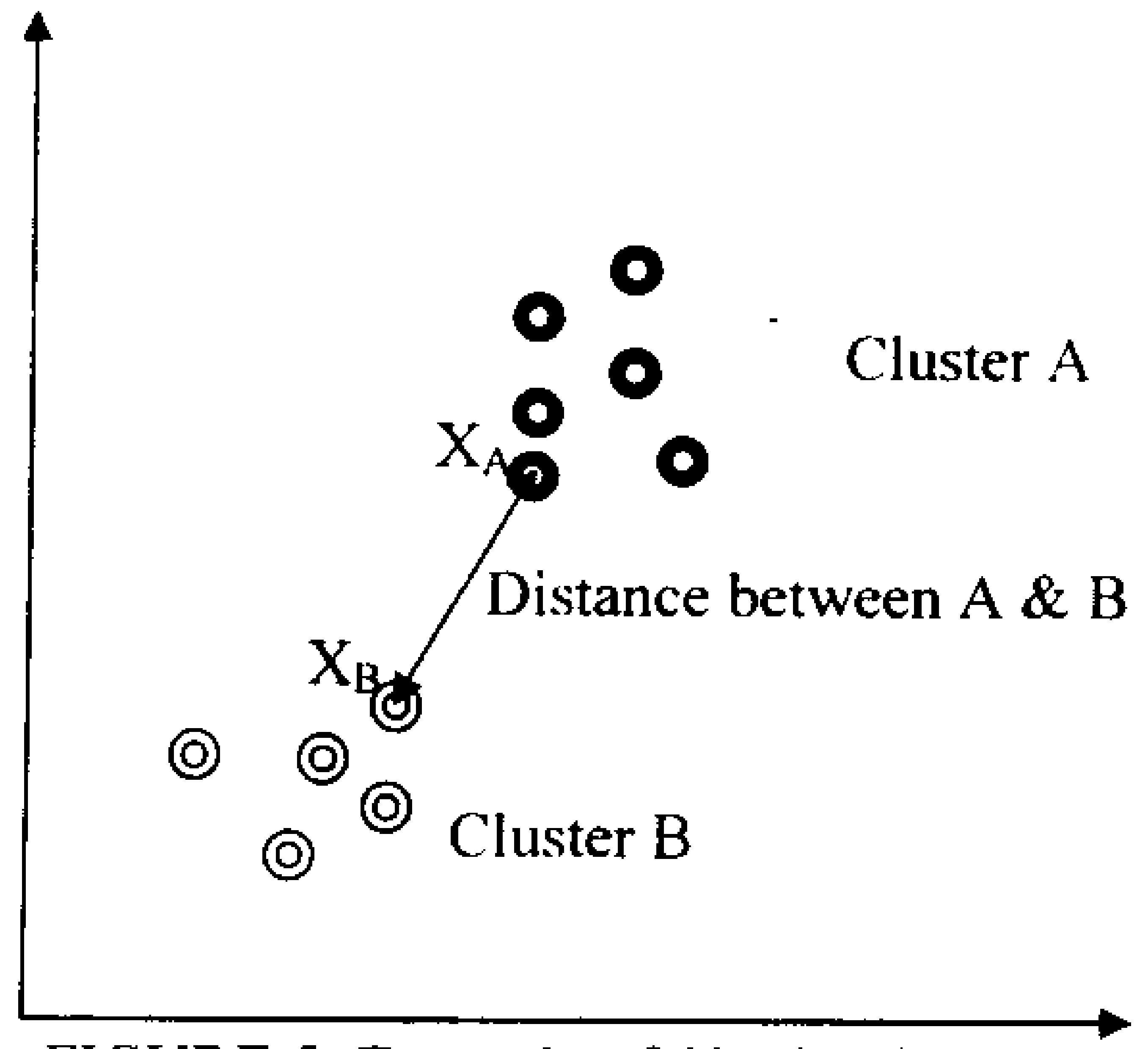

FIGURE 5: Example of Single Linkage Clustering

Here the darker points constitute one cluster, say cluster A and the other points constitute another cluster, cluster B. The shortest distance between them gives the distance between these two clusters. Thus the distance between clusters A and B is same as the distance between X_A and X_B. At each stage, the pair of clusters for which this distance is minimum, are merged. The distance is shown in the figure.

5.4 Algorithm of Elitist MOSA

We first do hill-climbing to get some initial solutions. They are generated randomly. The solutions, which are not dominated by any other solutions, are stored in the archive. Next one of the points in the archive is randomly selected and perturbed. The perturbation is done by changing a randomly chosen bit of the solution string, i.e., one position of the string is chosen randomly and if the value of the string at that position is one it is changed to zero and vice versa. Then the new perturbed string is evaluated and compared with the original string, from which we got this string. Depending upon their dominance relation we decide on our future course of action. For every new point, we actually check the domination status with respect to all other individuals in the archive as well as the current point.

We have used the concept of coverage in the present work. It is defined in the following way.

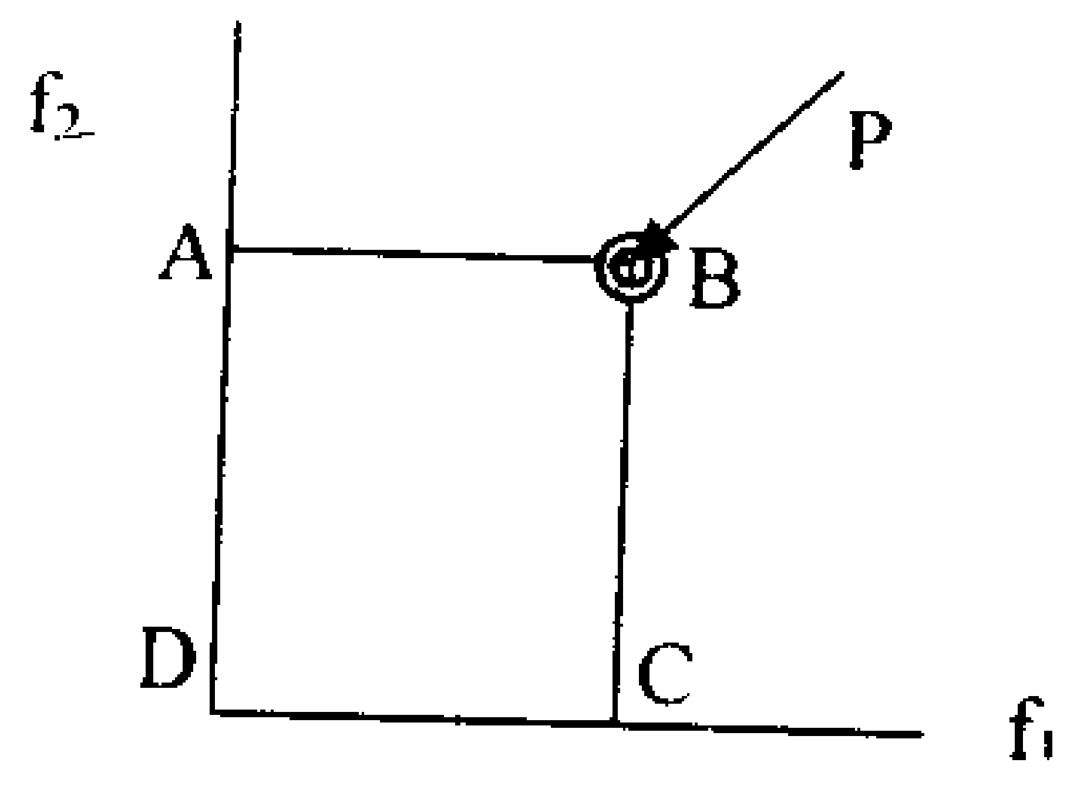

FIGURE 6: Example of coverage

The coverage is defined by the area covered by the point in the objective space. So in the figure above, the area of the rectangle ABCD gives the coverage of the point P for a two objective problem, when both f_1 and f_2 are to be maximized.

All the possible cases are discussed below one by one and also the course of action associated with each of them.

1. The current point dominates new point, but no other point dominates the new point. In this case, we find the difference in coverage of the new point and the current point. $\Delta cov = cov_{old} - cov_{new}$.
   The new point is selected as the current point with the probability inversely proportional to the difference in coverage. The situation would be as shown in Figure 7; the new point and the current point are shown. Also shown are the points in the archive.

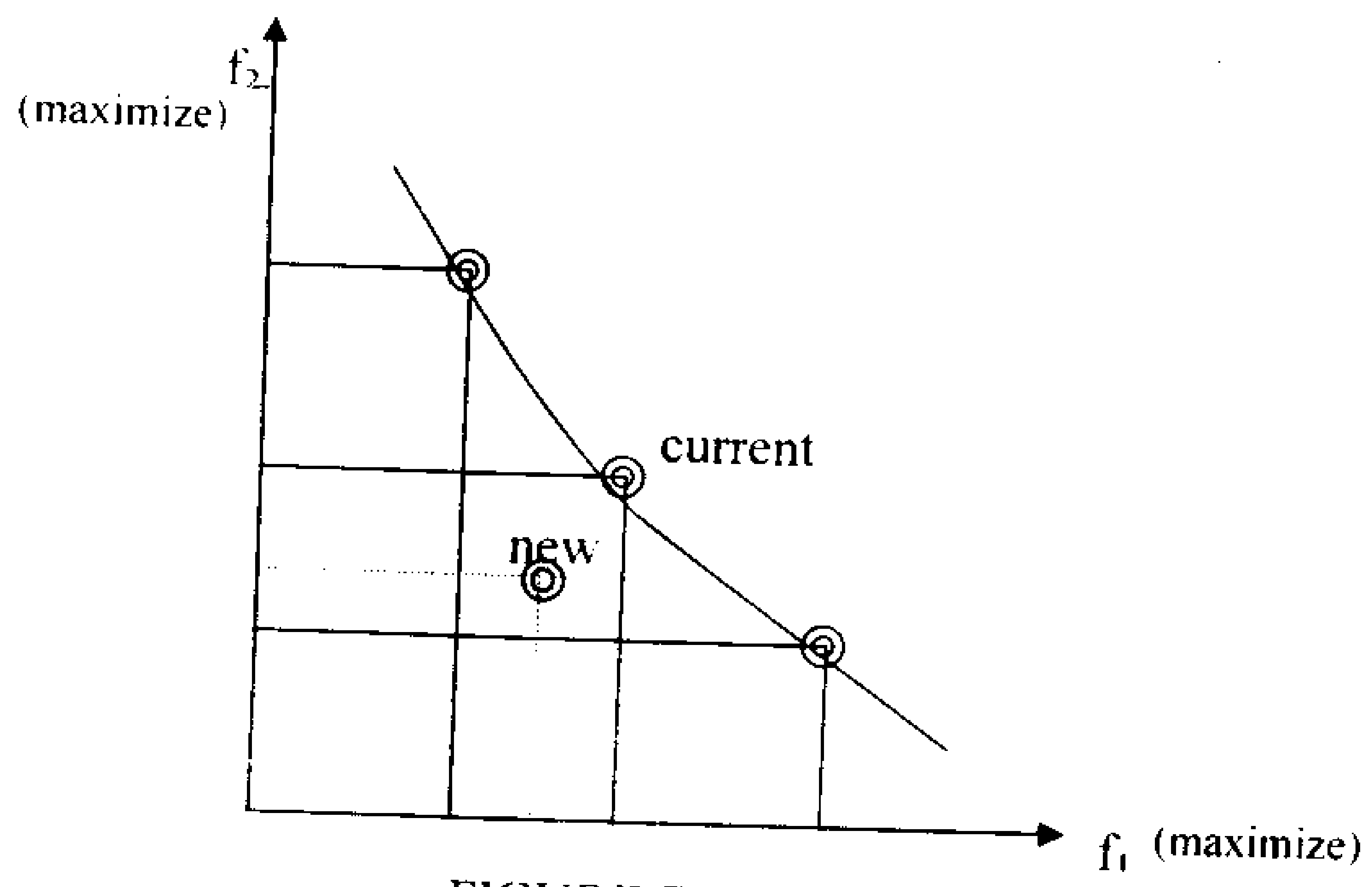

FIGURE 7: Elitist MOSA

2. The new point is dominated by not only the current point, but also by k other points in the archive as shown in Figure 8. In this case, we find the difference in coverage of each such point from that of the new point and take the sum of it.

$$\Delta \text{cov} = \sum_{i=1}^{k} (\text{cov}_i - \text{cov}_{new}) + (\text{cov}_{current} - \text{cov}_{new})$$

The probability of selecting the new point as the current point is taken to be inversely proportional to this Δcov value.

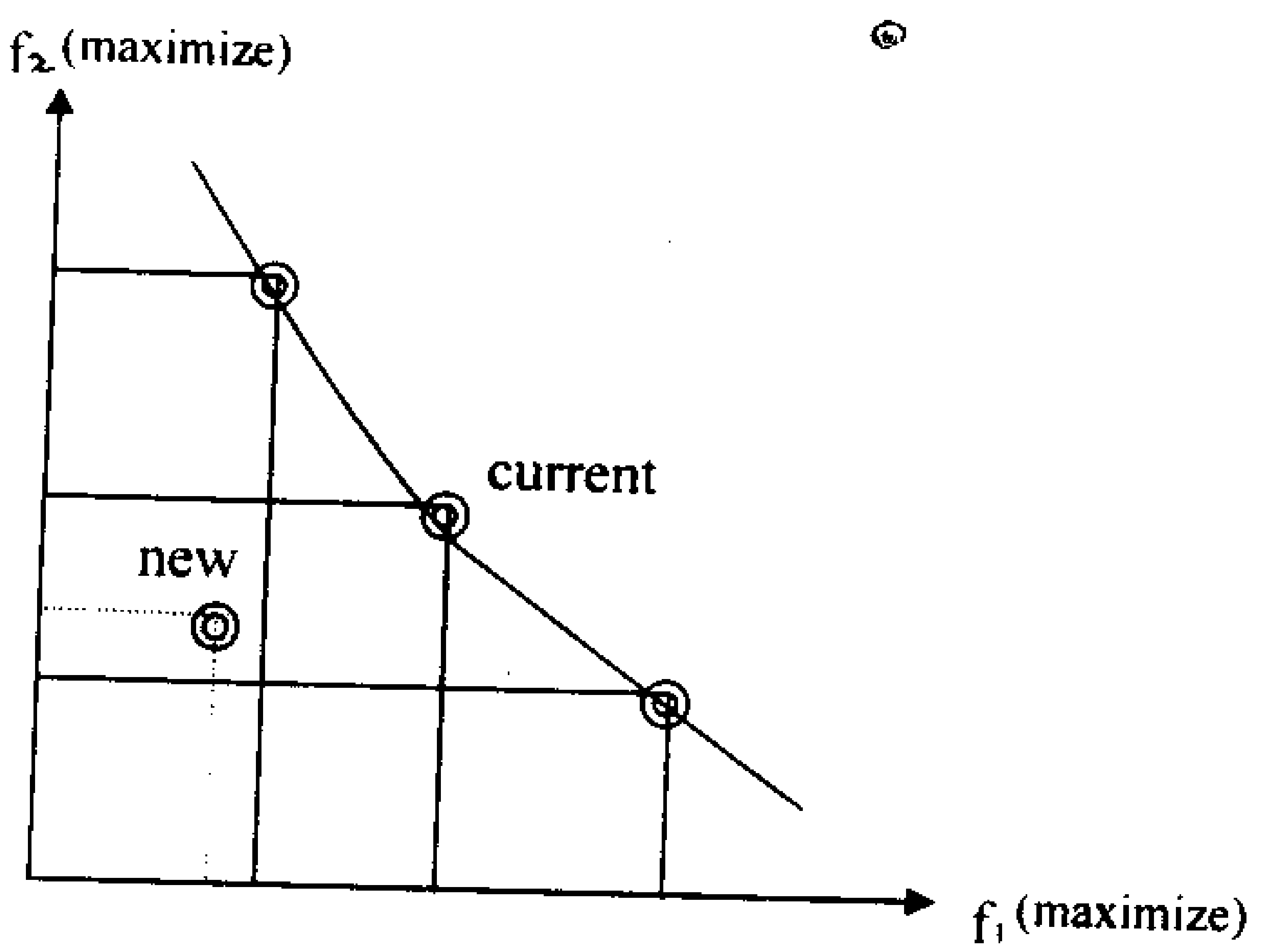

FIGURE 8: Elitist MOSA

3. New point is not dominated by the current point, but it is dominated by k points in the archive where k greater than or equal to 1. This situation is shown in Figure 9. In that case we find the sum of the Δcov values with respect to all those k points. The probability of selection of the new point as the current point is made to be inversely proportional to this sum S.

$$\Delta \text{cov} = \Sigma(\text{cov}_{archive} - \text{cov}_{new})$$

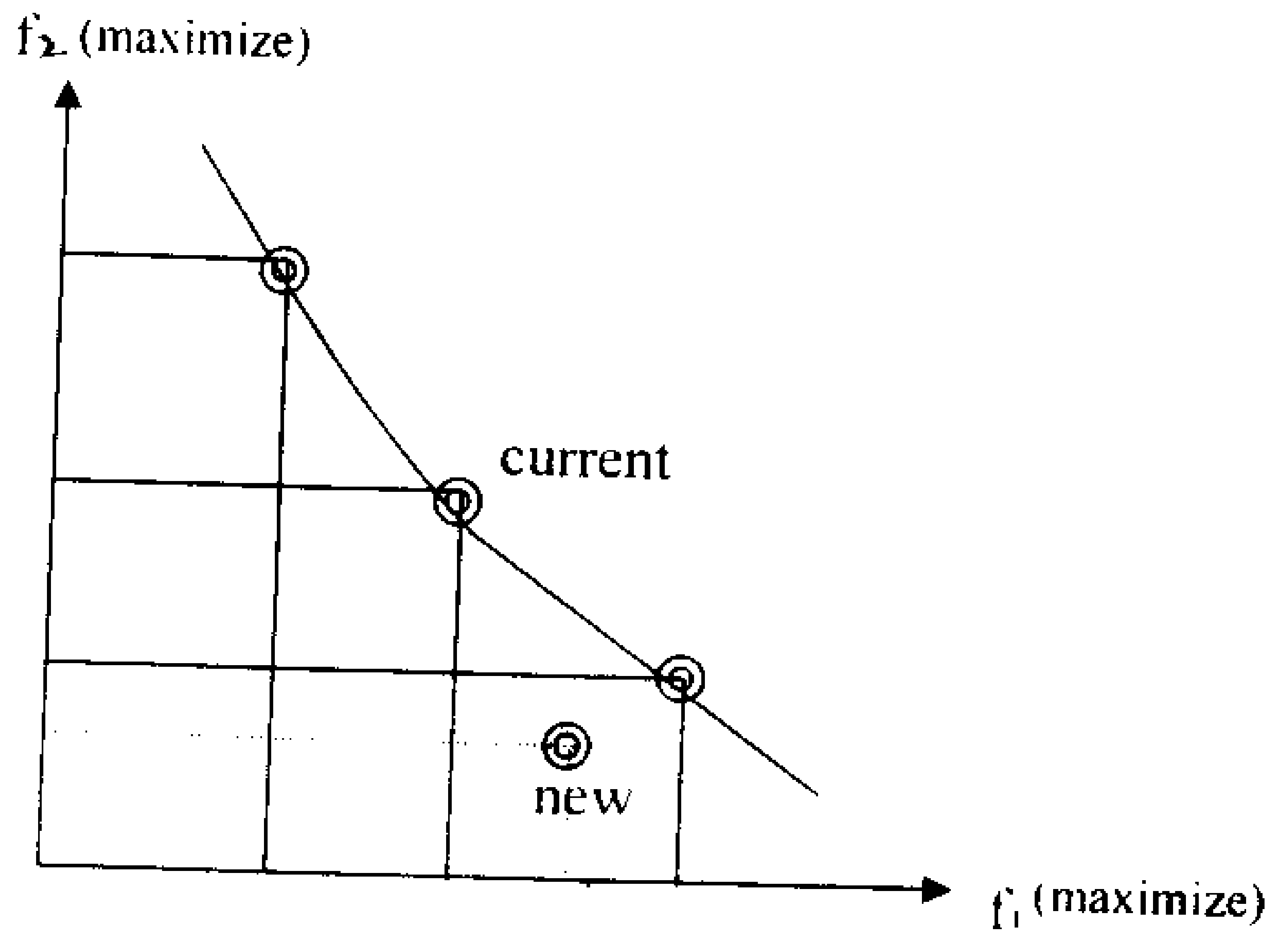

FIGURE 9: Elitist MOSA

4. New point is not dominated by either the current point, or by any other point in the archive. In that case the new point is on the same front as the archive as shown in Figure 10. So we should select the new point as the current point and add to the archive. Here we have to keep in mind that if the number of points in the archive exceeds the soft bound, then we perform clustering to reduce the number of points to the hard limit.

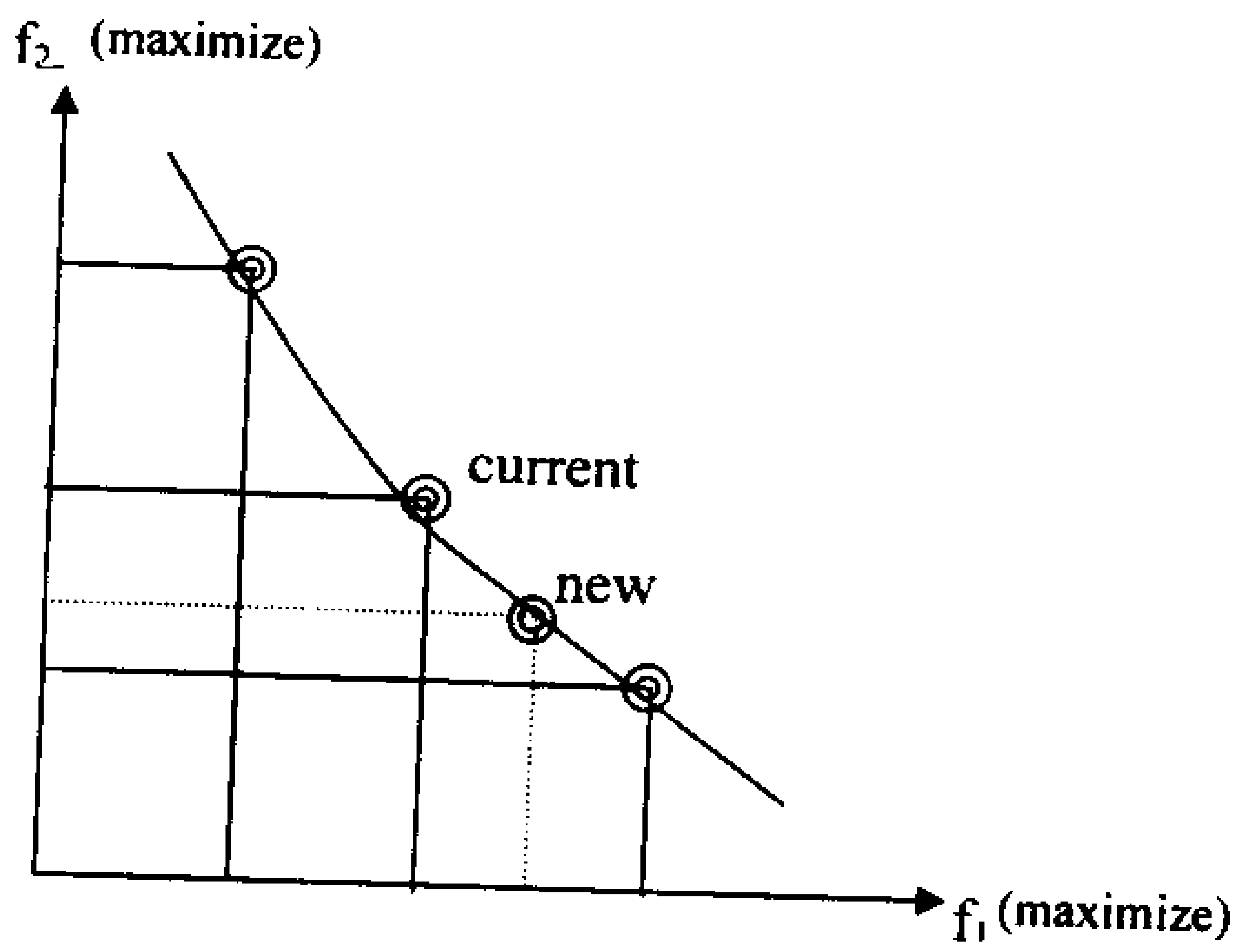

FIGURE 10: Elitist MOSA

In our case, we considered single linkage clustering, which is explained latter.

5. New point dominates the current point but k points in the archive dominate this new point. This situation (shown in Figure 11) would arise if the current point was not a member of the archive. We calculate the difference of coverage between the new point and the k points, and select the point from the archive as the current point, which corresponds to the minimum difference. This selection is done with probability of selection proportional to the Δcov.

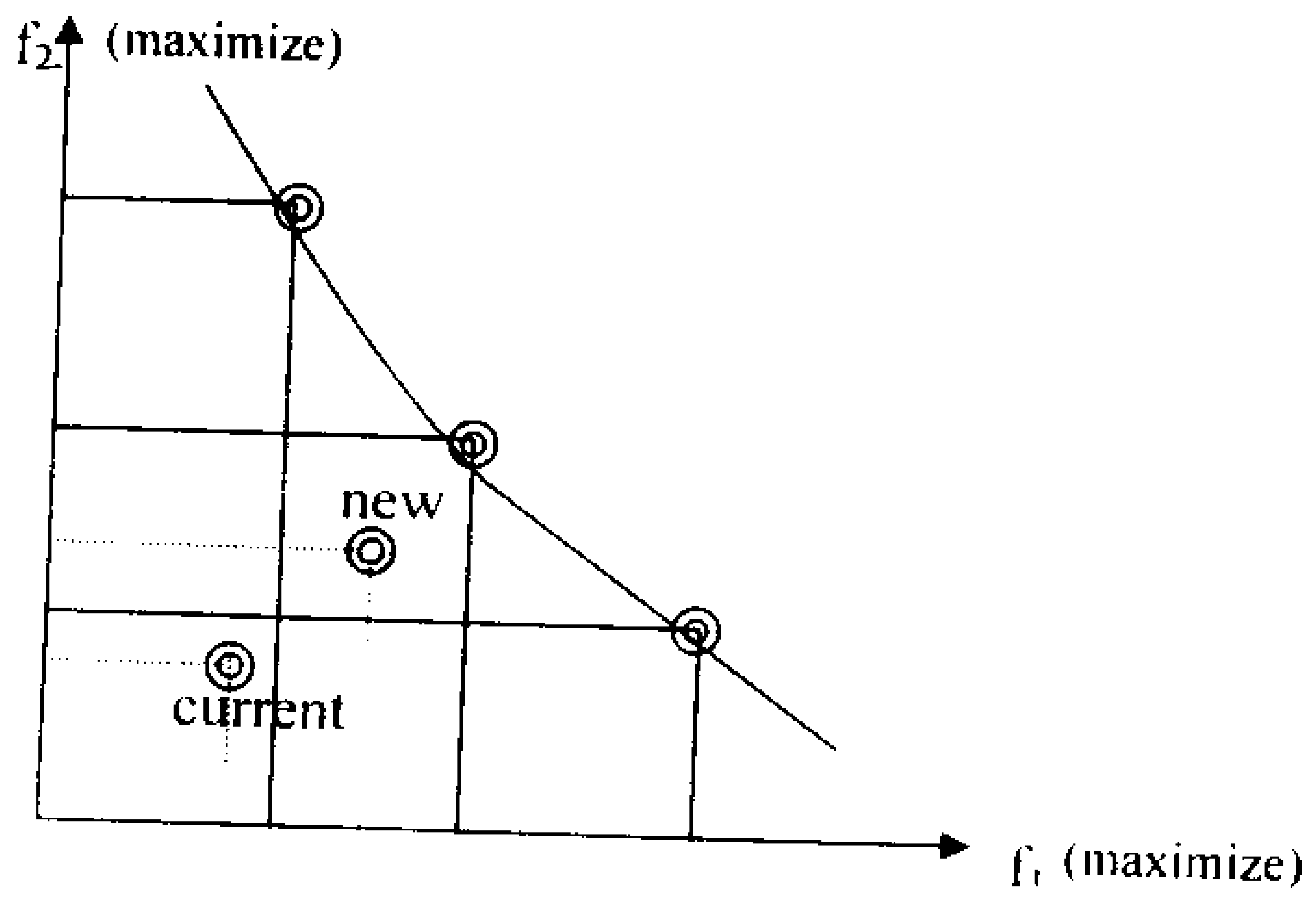

FIGURE 11: Elitist MOSA

6. New point dominates the current point and it is non-dominating with respect to the points in the archive. In this case, we should select the new point and add it to the archive keeping in mind that if the number of points in the archive becomes more than the soft bound, we have to perform clustering.

7. New point dominates the current point and also k other points (see Figure 12) in the archive. Now this new point clearly is better than the current point as well as the k other points of the archive. So apart from selecting the new point as the current point, we also remove the k dominated points of the archive.

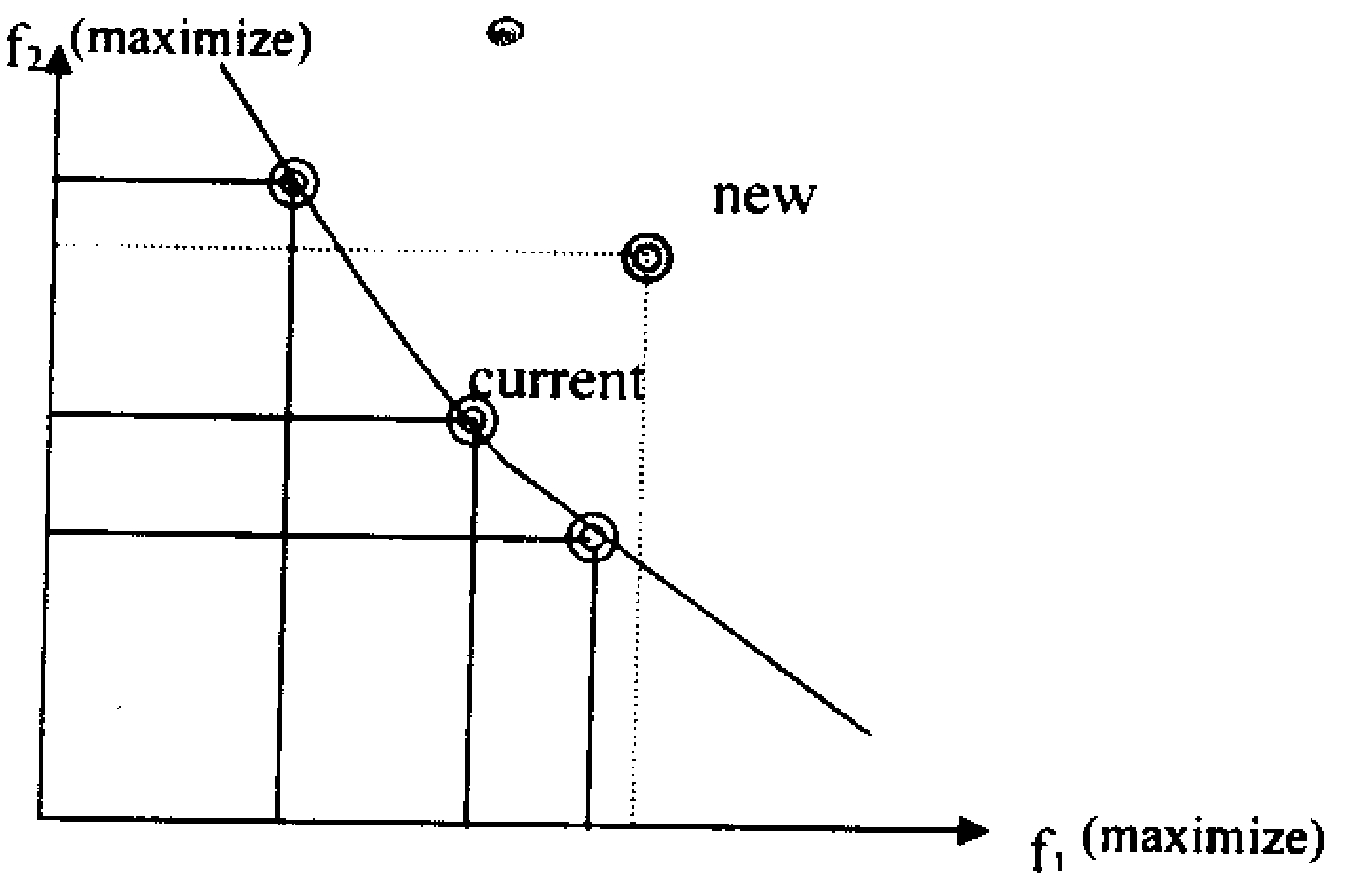

FIGURE 12: Elitist MOSA

In the next page, we provide a flowchart of the proposed EMOSA algorithm.

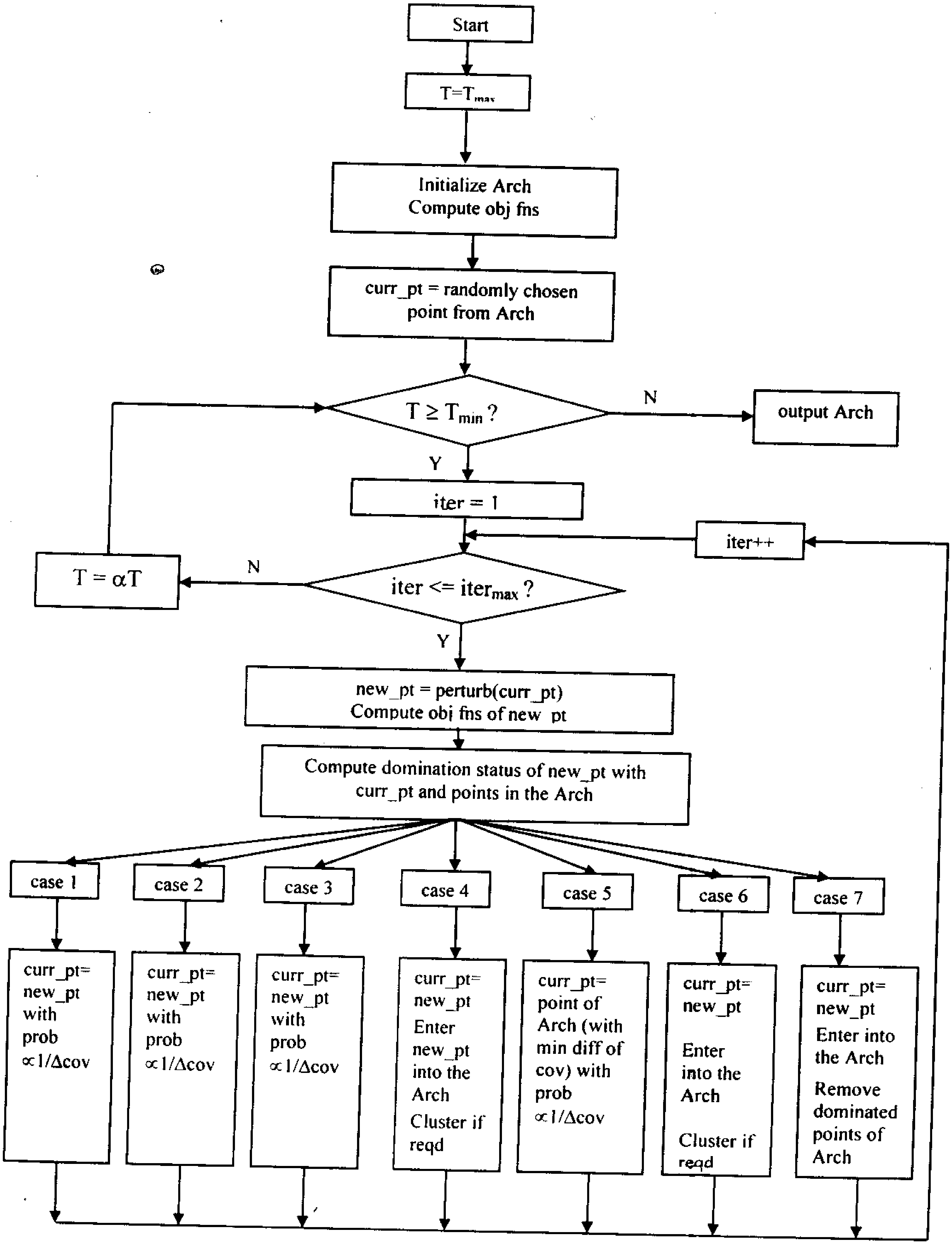
Start
T=T_max
Initialize Arch
Compute obj fns
curr_pt = randomly chosen point from Arch
T ≥ T_min ?
N
output Arch
Y
iter = 1
iter++
T = αT
N
iter <= iter_max ?
Y
new_pt = perturb(curr_pt)
Compute obj fns of new_pt
Compute domination status of new_pt with curr_pt and points in the Arch
case 1
case 2
case 3
case 4
case 5
case 6
case 7
curr_pt= new_pt with prob ∝1/Δcov
curr_pt= new_pt with prob ∝1/Δcov
curr_pt= new_pt with prob ∝1/Δcov
curr_pt= new_pt Enter new_pt into the Arch Cluster if reqd
curr_pt= point of Arch (with min diff of cov) with prob ∝1/Δcov
curr_pt= new_pt Enter into the Arch Cluster if reqd
curr_pt= new_pt Enter into the Arch Remove dominated points of Arch

FIGURE 13: Flowchart of EMOSA

5.5 Conclusions

In this chapter we have proposed a new elitist multiobjective simulated annealing algorithm. Analogous to PAES, an archive is maintained that contains the non-dominated solutions. As in SPEA, clustering is performed to reduce the size of the archive whenever it becomes overfull. The proposed algorithm is tested on several benchmark problems, and the results are compared with some existing point based search techniques in Chapter 7. As already mentioned, measuring the quality of solutions provided by an MOO algorithm is itself a problem. In the next chapter, we review some such existing measures. We also propose a modified measure in this regard.

Chapter 6

Measures of Comparison

6.1 Introduction

As discussed earlier, *Measure* is one of the areas that have recently got high focus in the researches on multiobjective optimization problem areas. No universally accepted *standard* exist till date for measuring the quality of the solution set produced by an algorithm. Though there are à wide variety of such measures in the literature. This chapter discusses some such measures. Also a new way has been proposed in this chapter, which tries to quantify the effectiveness of an algorithm in solving MOOP in a better way.

6.2 Basic Purpose of Measure

As suggested in [19], in multiobjective optimization problem, there are two primary functionalities that an MOO strategy must achieve regarding the obtained solution set.

1. Converge as close to the true pareto optimal front as possible;
2. Maintain as diverse a solution set as possible.

It is clear, that the first condition ensures that the obtained solutions are near optimal and the second one ensures that a wide range of trade-off solutions is obtained.
Thus for a comparison based on the attainment of each of the two functionalities of multiobjective optimization, it may be possible to define two performance metrics, one for measuring each functionality exclusively, even for more than two objectives. In [19] the metrics have been classified into three classes: those evaluating closeness to the Pareto-Optimal front. those evaluating diversity among non-dominated solutions and those which try to achieve both. We may think of maximizing the number of points obtained to be another functionality. Attempt has been made in this direction to cover this criterion also, i.e., preferring an algorithm, which produces more points.

6.3 Some Existing Quality Measures

As explained earlier, two kinds of measures exist in the literature. We first discuss some of those well-known metrics, which evaluate *closeness* to the Pareto-Optimal front. These metrics give a measure of the closeness of a nondominated set, also known as approximation set, obtained by an algorithm from a known Pareto front. We consider a set Q of N solutions and a known set of pareto optimal points P^*.

1. Error Ratio: Veldhuizen proposed this measure [15]. It counts the number of solutions of Q that are not members of the pareto optimal set P^*.

$$ER = \frac{\sum_{i=1}^{|Q|} e_i}{|Q|}$$

Here $e_i = 1$ if $i \notin P^*$ and e_i is zero otherwise. But this metric has some inherent problems. As pointed out by Knowles & Corne in [12], say an algorithm returns two points, one of which is on the Pareto front and the other one away from the front. In that case, the error ratio is 0.5. But if an algorithm returns 99 points evenly distributed *just away* from the front and one on the front, then the error ratio is 0.99, thus violating the intuitive belief that the second one is a better solution. Also we can get the value only if the Pareto front is known a priori.

2. Set Coverage Metric: This metric was proposed by Zitzler [20]. It calculates the proportion of solutions in an approximation set B which are weakly dominated by another approximation set A, denoted by *C(A,B)*.

$$C(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \leq b\}|}{|B|}.$$

This metric can also be used to get an idea of the relative spread of solutions between the two sets A and B. The metric value *C(A,B) = 1* means all members of B are weakly dominated by A. If it is zero, then it signifies that no member of B is weakly dominated by A. It is not necessarily true that *C(A,B)* is equal to *1-C(B,A)*, i.e., we need to calculate both of them. But if two sets are of different cardinality and/or the distributions of the sets are non-uniform, then it gives unreliable results. It cannot determine the degree of outperformance if one set completely outperforms the other [12].

3. Generational Distance: This measure was also proposed by Veldhuizen [15]. Instead of finding whether a solution of Q belongs to the set P^* or not, it finds an average distance of the solutions of Q from P^* as follows:

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|}$$

For p = 2, the parameter d_i is the Euclidean distance between the solution $i \in Q$ and the nearest member of P^*.

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*(k)})^2}$$

Here $f_m^{*(k)}$ is the mth objective function value of the kth member of P^*. Clearly lower values of this metric represent better sets. GD measures general progress towards P. But to calculate this metric, knowledge of Pareto front is required. So when Pareto front is not known, this metric cannot be used. It estimates how far is our current Pareto front from the true Pareto front of a problem using Euclidean distance (measured in objective space) between each vector and the nearest member of the true pareto front. The problem with this approach is that uniform spread is not considered along the Pareto front.

Next we discuss some metrics, which evaluate diversity among non-dominated solutions.

1. Spacing: Schott [21] proposed this metric, which is calculated with a relative distance measure between consecutive solutions in the obtained nondominated set, as follows:

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(di - \overline{d})^2}$$

where $d_i = \min_{k \in Q \wedge k \neq i} \sum_{m=1}^{M} | f_m^i - f_m^k |$ and $\overline{d}$ is the mean value of the above distance measure $\overline{d} = \sum_{i=1}^{|Q|} d_i / |Q|$.

The distance measure is the minimum value of the sum of the absolute difference in objective function values between the i^{th} solution and any other solution in the obtained nondominated set. This metric measures the standard deviation of the different d_i values. When the solutions are near uniformly spaced, the corresponding distance measure will be small. Thus evidently an algorithm that finds an approximation set, having smaller spacing value is the better one. But this metric does not take care of the extent of the spread.

2. Spread: This metric was proposed in [22] to overcome the difficulty associated with Spacing. This is defined as:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} | d_i - \overline{d} |}{\sum_{m=1}^{M} d_m^e + |Q| \overline{d}}$$

Here d_i can be any distance measure between neighboring solutions and $\overline{d}$ is the mean value of these distance measures. The parameter d_m^e is the distance between the extreme solutions of the pareto optimal front and the nondominated solution set corresponding to mth objective function. Though it addresses the problem associated with spacing, still it does not make any comparisons regarding the number of solutions obtained by an algorithm.

In other words, it does not give any preference to an algorithm which produces more points than another having distribution and width of points same as that of the latter one. To take care of this effect also, we can define a new measure, called Combined Measure, *(CM)* in the following way.

$$CM = \frac{|Q| + \min_m \sum (d_{it} - d_m^e)^2}{\sum (d_{it} - \overline{d_t})^2 + 1}$$

This metric produces higher values for better solution sets. Here we consider only those points, which are on the true pareto front, indicated by the suffix t. In the denominator, we find the uniformity of the points using the standard deviation of different d_{it} values, which are computed in a way similar to that in *spacing*, except for the fact that we consider only the *true* points here. One is added in the denominator to take care of the ideal situation when $\forall i,\ d_{it} = \overline{d_t}$.

The first term in the numerator computes the number of true points obtained and the second term computes the sum of distances of all the points from one of the extreme points. The higher the value of the second term, farther the points are away from one of the end points. But this may have the effect of preferring a set having a concentrated set of points at the central region, which is taken care of by the denominator. Thus all of the requirements of an MOOP are captured in this single measure.

3. Maximum spread: Zitzler [20] introduced this measure also. It measures the length of the diagonal of the hyperbox formed by the extreme function values observed in the non-dominated set. It is given as

$$D = \sqrt{\sum_{m=1}^{M} (\max_{i=1}^{|Q|} f_m^i - \min_{i=1}^{|Q|} f_m^i)^2}$$

For a two objective problem, this metric refers to the Euclidean distance between the two extreme solutions in the objective space.

4. Chi-square-like deviation measure: This was proposed in [19], [23]; a neighborhood parameter *e* is used to count the number of solutions, within each chosen Pareto optimal solution. The distance calculation can be made in either the objective space or in the decision variable space.

Also some measures exists to take into consideration both of these tasks:

1. Hypervolume: It calculates the volume covered by members of Q for problems where all objectives are to be minimized. Mathematically, for each solution i $\in$ Q, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube [15], [24]. Therefore a union of all hypercubes is found and its Hypervolume (HV) is calculated.

$$HV = \text{volume}(\bigcup_{i=1}^{|Q|} v_i)$$

An algorithm is considered more worthy if it gives a larger value of Hypervolume.

2. Weighted Metric: This metric is formed by combining one of the convergence metrics and one of the diversity measuring metric together [19] as follows. W = w_1*GD + w_2 *Δ, with $w_1 + w_2 = 1$, being two weight values. Here GD is the generational distance measure for evaluating the converging ability and Δ is a metric that measures the diversity-preserving ability of an algorithm.

So basically we have metrics, which measure distance between the Pareto front and the front resulting from one run, or the distribution of the non-dominated solutions along the front or the diversity of the computed solutions. But when the Pareto front is unknown, then we cannot use the first two metrics. Again the third approach alone is not adequate. In order to overcome these problems, concepts of *Pareto Like Front (PLF)* have been proposed in [8], which is built up with solutions resulting from simulations with different sets of parameters. This approach was an attempt to use the distance from each single front to the *PLF*. In another approach introduced in the same paper [8], distance of individual fronts was not measured from Pareto front, but from a single point. The point coordinates may correspond to the best value in each objective found in the results of all simulations, which is called *"floating optimal point"* *(FOP)*. It may be *"meta optimal point" (MOP)* also, whose coordinates are pre-established by the decision maker, based on his/her preferences and according to the results of the optimization problem. When the pareto front and the optimal point are unknown, FOP appears to be an attractive way to deal with the need for ranking populations, as FOP is easy to find in each simulation. The MOP on the other hand requires some expertise on the part of the decision maker. In case of MOP, we usually find an approximation of the Pareto optimal front, which is denoted as *approximation set* [9]. So the main issue is how to evaluate the approximation set.

6.4 Unary, Binary and Attainment Surface Based Measures

Alternatively the measures may be classified into 3 categories [9] - unary, binary and attainment surface based measures. In case of unary measures, a number is assigned to each approximation set, which reflects a certain quality aspect, and usually a combination of them is used [25][22]. In case of binary quality measures, a number is assigned to pairs of approximation sets. In contrast to unary measures, there are very few binary measures found in the literature. Zitzler and Thiele proposed one of them in [26]. Attainment surface is conceptually bit different [26]. If a boundary is drawn in the Objective Space that separates those points, which are dominated from those that are not, then this boundary is called Attainment Surface.

But the discussion on Measure is going to be in a different way nowadays. Giving a new dimension to the whole issue, the authors in [9] have shown the following points, which are going to be very important in further studies of designing quality metrics.

1. There exists no unary quality measure that is able to indicate whether an approximation is better than the other.
2. The above statement holds if we consider a finite combination of unary measures.
3. Unary quality measures being able to detect that approximation A is better than approximation B exists, but their use in general is restricted.
4. Binary quality measures overcome the limitations of unary measures and if properly designed, are capable of indicating whether A is better than B. Sometimes combinations of unary measures are used. But the authors [9] claim that although an approximation set A may be evaluated better than an approximation set B with respect to all of the indicators, B can actually be superior to A with respect to dominance relations. This holds especially for various diversity measures and also for some of the distance indicators proposed in the literature.

In their work they have given the sketch of two measures. Following the direction, we introduce one quality measure, as follows. Suppose we are handling a two objective minimization problem. We construct the smallest hyper rectangle that completely covers the approximation set. If the rectangles of the two approximation sets are not intersecting, then we can easily decide which one is better as shown in the following figures.

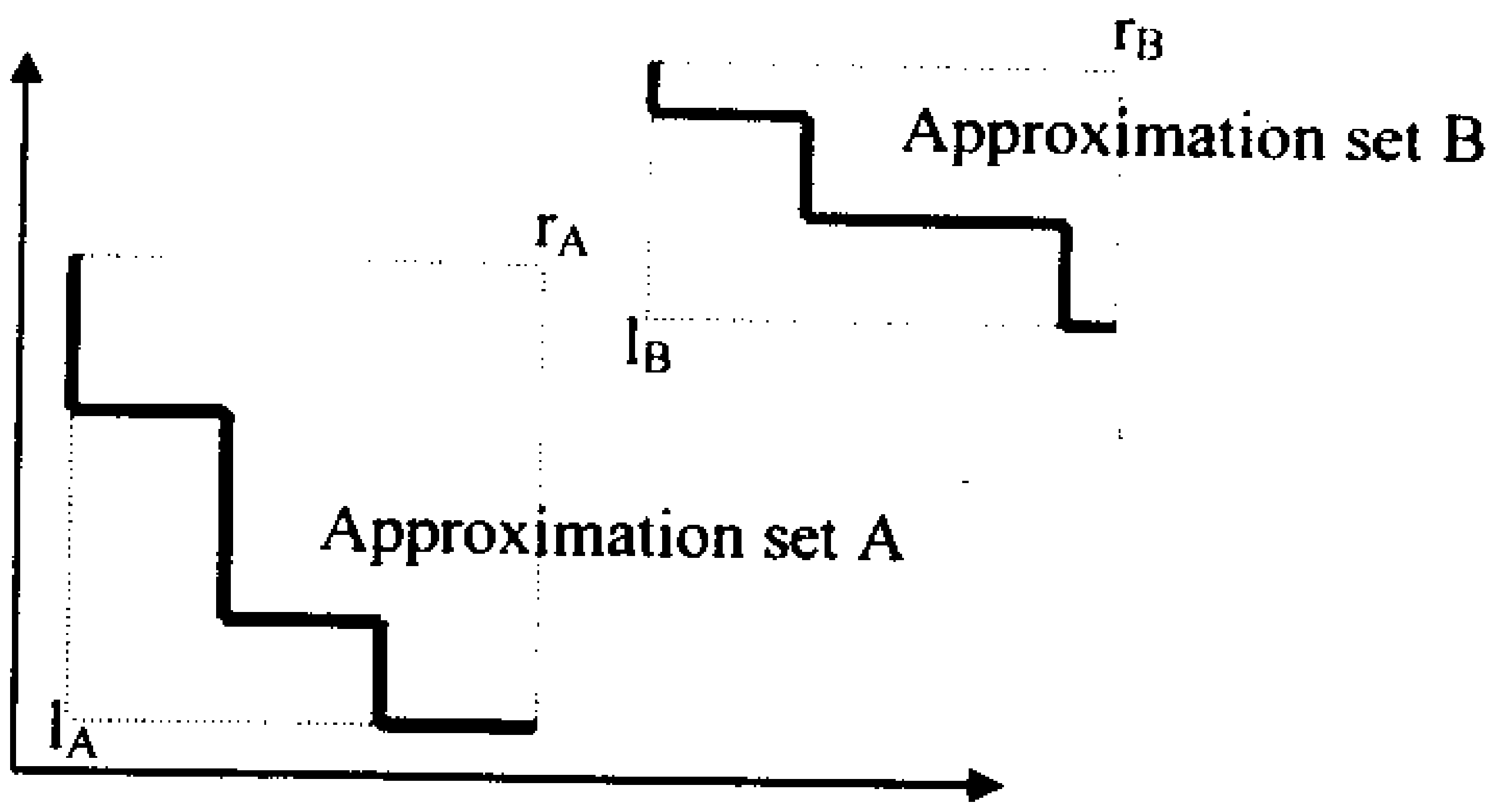

FIGURE 14: Measure (case I)

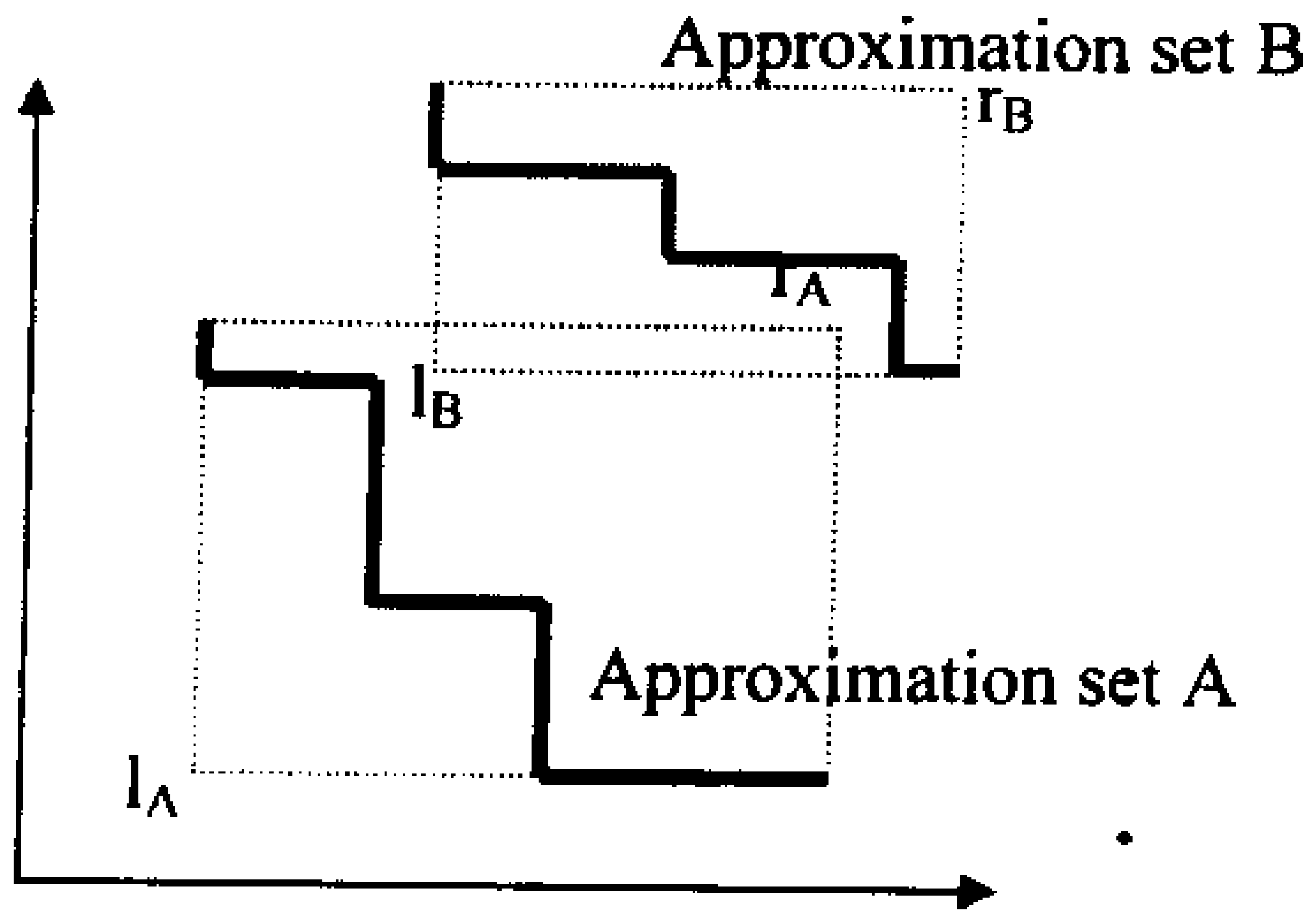

FIGURE 15: Measure (case II)

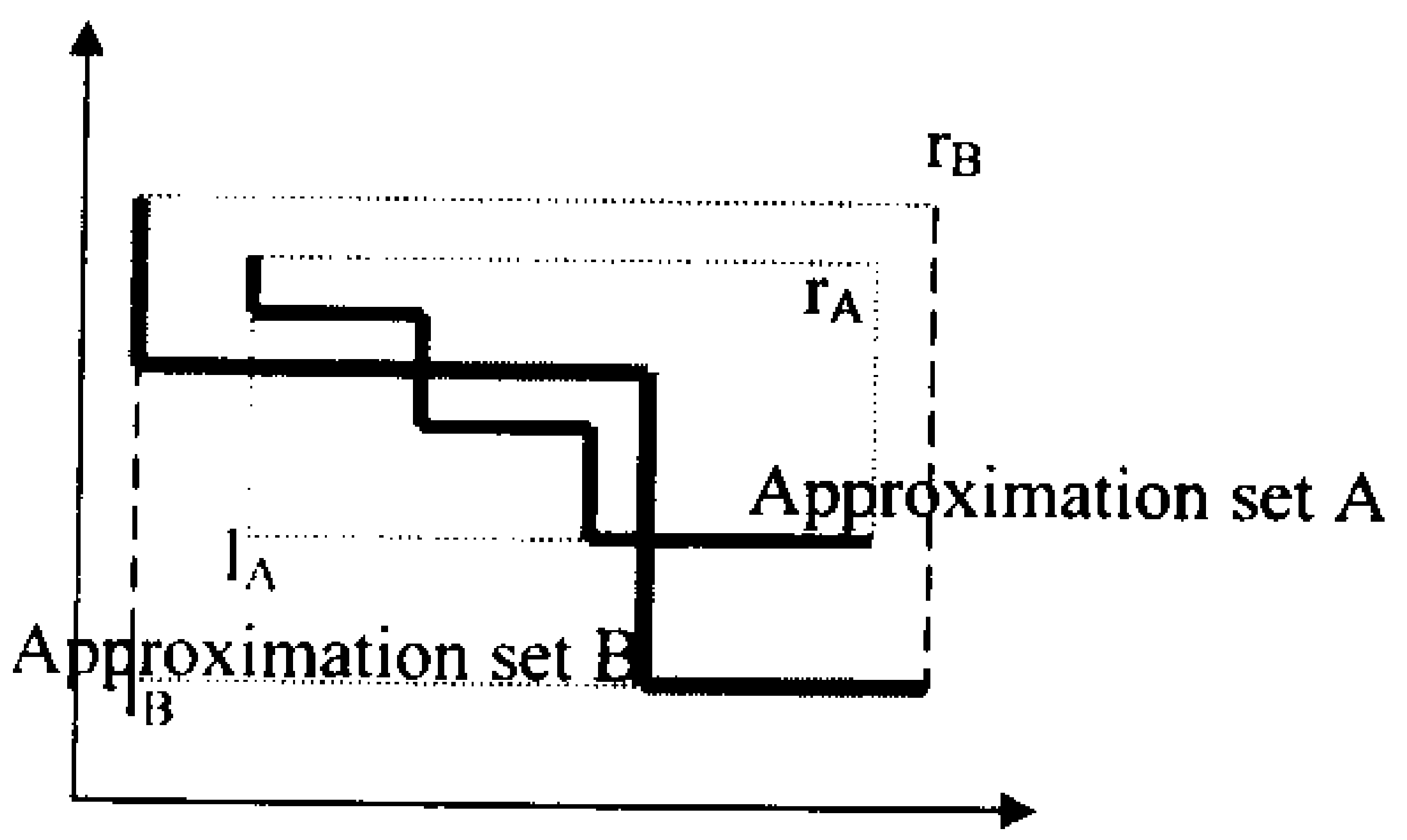

FIGURE 16: Measure (case III)

In case I, we find that the hyper rectangles are not intersecting. As we are talking of minimization problem, it is easy to find that the approximation set A is better than B. Mathematically, if the top right corner point of one rectangle dominates the same of the other, AND the lower left corner point of the previous rectangle dominates that of the other, then the later one corresponds to a better approximation set in case of minimization problem. We can extend this concept for cases where more than two objectives are dealt with. But if one of the two conditions fails, then we cannot give a definite answer on which one dominates the other, as shown in Figure 15 and Figure 16. So if r_B dominates r_A AND l_B dominates l_A, then approximation set A is better in a minimization problem. However in [7], Deb has argued that it may be possible to compare two or more approximation sets functionally, as is often followed in understanding behaviors of complex systems, although it may seem to be mathematically incorrect from the discussions in [9]. In the same work [7], concept of running metric has been brought into the arena of MOOP. This concept had its inspiration from their use in single-objective EAs, where such generation-wise performance exists from early days, showing how the average or best fitness or some other performance metric is varying with generation. So it is not surprising that if appropriate performance measures of MOOP solving strategies' populations are also recorded and analyzed, important inferences can be made about their working, like how an MOO strategy arrives at the final solution. The main reason of their absence in MOOP literature is the cardinality of the necessary performance metrics to properly evaluate an MOOP solving strategy and the complexities involved in computing them. Two running metrics have been introduced in [7]. The first metric is a distance measure of a population from a reference set and is used to measure the convergence ability of an MOO strategy. The second metric uses a local template based evaluation technique to estimate the diversity of one set compared to a reference set.

6.5 Entropy Based Measures

In [14] an entropy-based metric is presented that can be used for assessing the quality of a solution set as obtained from multiobjective optimization techniques. This metric quantifies the *goodness* of a set of solutions in terms of distribution quality of solutions over the Pareto frontier. In the same paper, a new metric has been presented to address the diversity of solution points. The new metric, hereafter referred to as the entropy of a solution set, is based on the notion of information-theoretic entropy and encapsulates into a single scalar quantity different aspects of the distribution quality such as uniformity of distribution, coverage (i.e., portion of the Pareto frontier covered by the observed solution set), number of solution points and clustering. The basic idea behind this metric is that each solution point provides some information about its neighborhood in the feasible space that can be modeled as a function, called an influence function. Let us assume a stochastic process with n possible outcomes where the probability of the ith outcome is p_i. The probability distribution of this process can be shown as:

$P = [p_1, p_1, \ldots, p_1, \ldots, p_1]$

$\sum p_i = 1 \qquad p_i \geq 0$

The Shannon's entropy measures the flatness of P, i.e., if the values of the entries in the vector are approximately the same then the entropy is high, but if the values are very different (uneven probability distribution), the corresponding entropy is low. A desirable solution set must have a 'flat' density surface within the feasible domain. To quantify this flatness, one may take advantage of the formal similarities between this problem and the Shannon's entropy, which also measures the flatness of a distribution. Thus here entropy is used to measure the flatness of the information distribution provided by a set of solution points, and hence it is desirable to be maximized (i.e., that corresponds to a uniform distribution of solution points over the feasible domain).

6.6 Conclusions

This discussion leads us to the conclusion that till now there does not exist any single *universally accepted* measure, thus having a huge scope of future work in this area. More concerted effort is required to evolve better measures that would be able to properly judge how nice an algorithm works in solving an MOOP. Moreover, extensive experimental results are required to establish the newly defined measures.

Chapter 7

Test Problems and Performance Issues

7.1 Introduction

The Elitist MOSA algorithm, proposed in this work, has been compared with two existing algorithms; first one is the Multiobjective Simulated Annealing (MOSA), proposed by Nam. D and Park. C.H [5] and the second one is Pareto Archived Evolution Strategy (PAES), proposed by Knowles J. D. and Corne D. W. The reason behind the choice of MOSA is that it also uses the concept of simulated annealing in solving MOOP. PAES, on the other hand, has a lot of similarity, in the way of thinking with Elitist MOSA. For this reason, it was also compared with the proposed algorithm.

7.2 Choice of MOSA and PAES for comparison

As already discussed, Multiobjective Simulated Annealing algorithm is one of the very few algorithms that use simulated annealing in solving MOOP. Our proposed algorithm also works in the same direction. In other words, it also uses simulated annealing in solving MOOP. We have used four variants of MOSA, as proposed in [5]. They differ in the way the transition probability from one state to another is calculated. They are indicated as MOSA(avg), MOSA(max), MOSA(min), MOSA(self) to indicate the different cost criterions in calculating transition probability as discussed previously.

PAES, though independently worked out, works in certain aspects in a manner similar to our algorithm, though having some major differences. One of the commonalities of these two algorithms is the use of archive in storing the solution set. However, this concept is not unique. Previously work was done in this direction. PAES, as well as our algorithm, first does hill-climbing to get an initial solution set. And later, while *new points* are generated from the *current point*, they use some *criterion* to decide whether to select the *current point* or not. Obviously the novelty of an algorithm would lie in designing these *criterions*. And this is where our algorithm differs from PAES. The selection criterions of both of these two algorithms have been discussed previously. Clearly our proposed algorithm handles the cases in a more detailed fashion. For this reason, it also works in a much better manner than PAES in most of the problems, as evident from the following discussions.

To compare these three algorithms, we have used three existing metrics and one new metric proposed by us, i.e., Combined Measure *(CM)*. The first measure is the *Error Ratio*, which finds the ratio of the points that are not on the pareto front to the total number of points. Second metric is *Spacing*, which measures the uniformity in spacing of the obtained points. The third one is *Spread*, which measures not only the uniformity, but it also takes care of the extent of the spread of solutions. Detailed discussions are provided in Chapter 3.

Four problems have been solved using these three algorithms. They are discussed and also performance in solving them is analyzed below. The parameters that have been used in solving the following problems are as follows: maximum temperature is 50, minimum temperature is 0.1, and temperature is educed geometrically with coefficient of cooling as 0.85. We have performed 100 iterations at each temperature. The archive has a hard limit of 10 and soft limit of 20. In case of PAES, archive has a size of 10 and 2000 iterations are performed to make them equal with EMOSA and MOSA. The results reported in the following tables correspond to the average values obtained over 10 simulations of each of the algorithms.

7.3 Results

Problem I

The first problem that we handled is Schaffer's problem [19]. Though it is simple, it has been the most studied single-variable test problem.

It is as follows:

$$\text{SCH1:} \begin{cases} \text{Minimize } f_1(x) = x^2 \\ \text{Minimize } f_2(x) = (x-2)^2 \end{cases}$$
$$-A \leq x \leq A$$

This problem has Pareto-optimal solutions $x^* \in [0,2]$ and the Pareto set is a convex set: $f_2^* = (\sqrt{f^*_1} - 2)^2$ in the range $0 \leq f^*_1 \leq 4$.

The figure below shows the Pareto optimal front for this problem. Different values of the bound-parameter A are used in different studies. Values as low as A = 10 to values as high as A = 10^5 have been used.

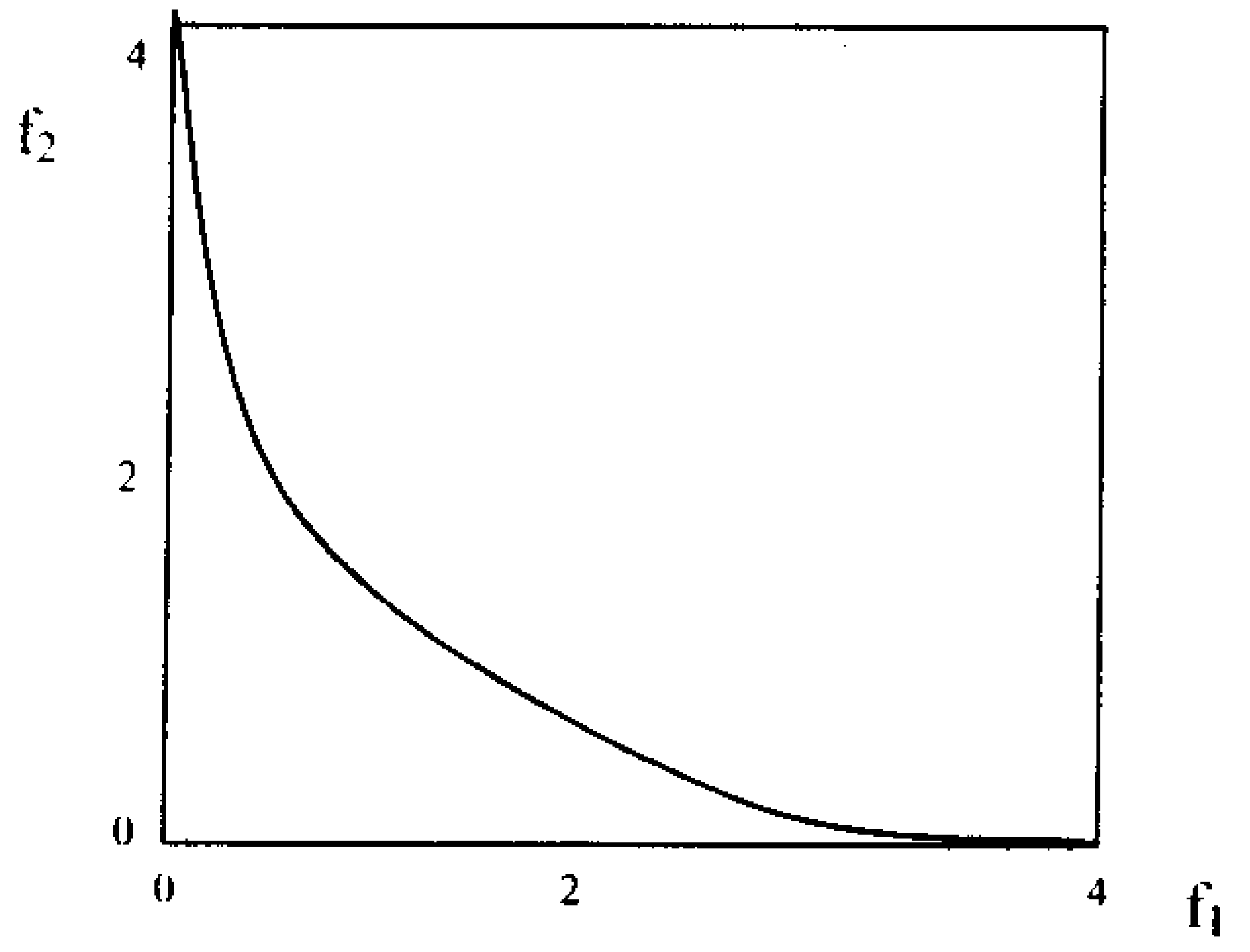

FIGURE 17: Pareto optimal front of Test problem I

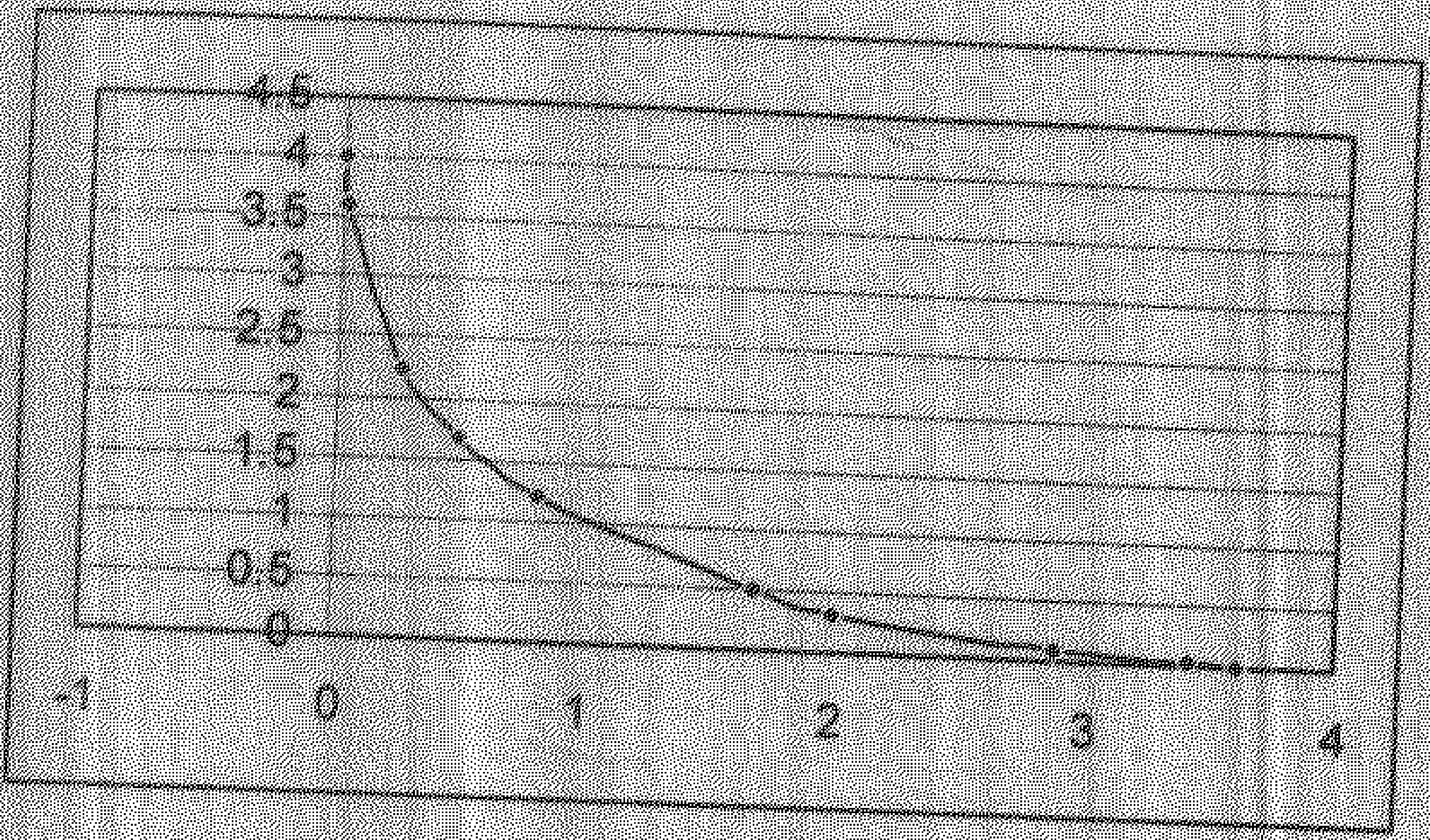

FIGURE 17a: Archive obtained from EMOSA for Test Problem I

TABLE 1: Comparison of results on test problem I

	ElitistMOSA	PAES	MOSA(avg)	MOSA(max)	MOSA(min)	MOSA(self)
ER	0.0	0.06	0.4	0.3	0.3	0.2
SPACING	0.523	1.9694	1.712	1.1215	1.72	1.13
SPREAD	0.531	0.5967	0.584	0.6085	0.63	0.58
CM	27.871	22.358	3.258	9.744	3.568	3.798

As evident from this table (Table 1), the Error Ratio is minimum for our case. Also it gives a good spacing and spread characteristics compared different MOSAs as well as PAES. Figure 17a gives the distribution of points as obtained from EMOSA.

Problem II

The second problem that we tested is Schaffer's second function, SCH2. It is also used in many studies. The problem is as below:

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

$$-5 \le x \le 10$$

The Pareto-optimal set consists of two discontinuous regions: $x^* \in \{[1, 2] \cup [4, 5]\}$
The figure below shows the objective space. The Pareto optimal regions are marked separately. The main difficulty an algorithm may face in solving this problem is that a stable subpopulation on each of the two disconnected Pareto-optimal regions may be difficult to maintain.

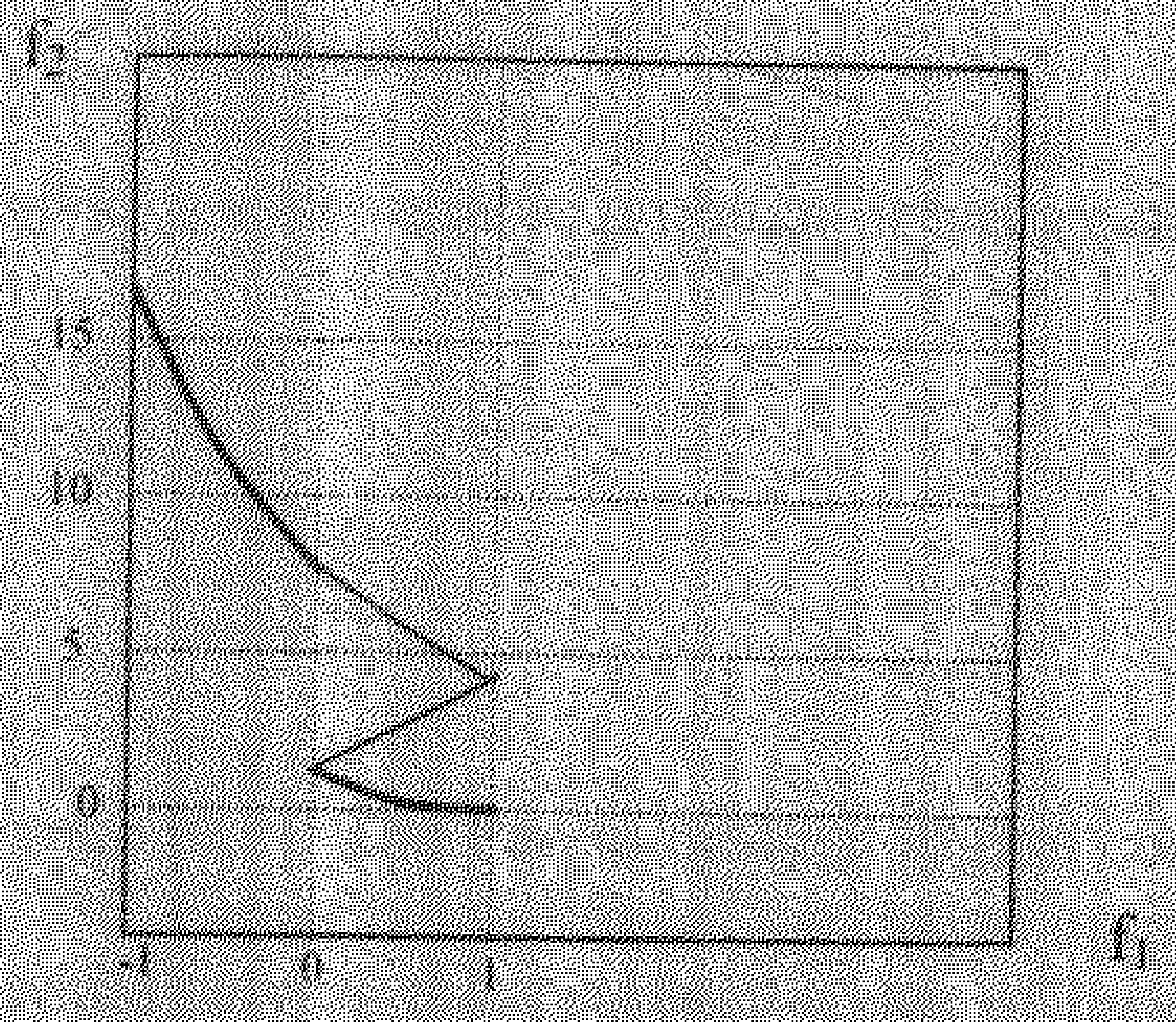

FIGURE 18: Pareto optimal front of Test problem II

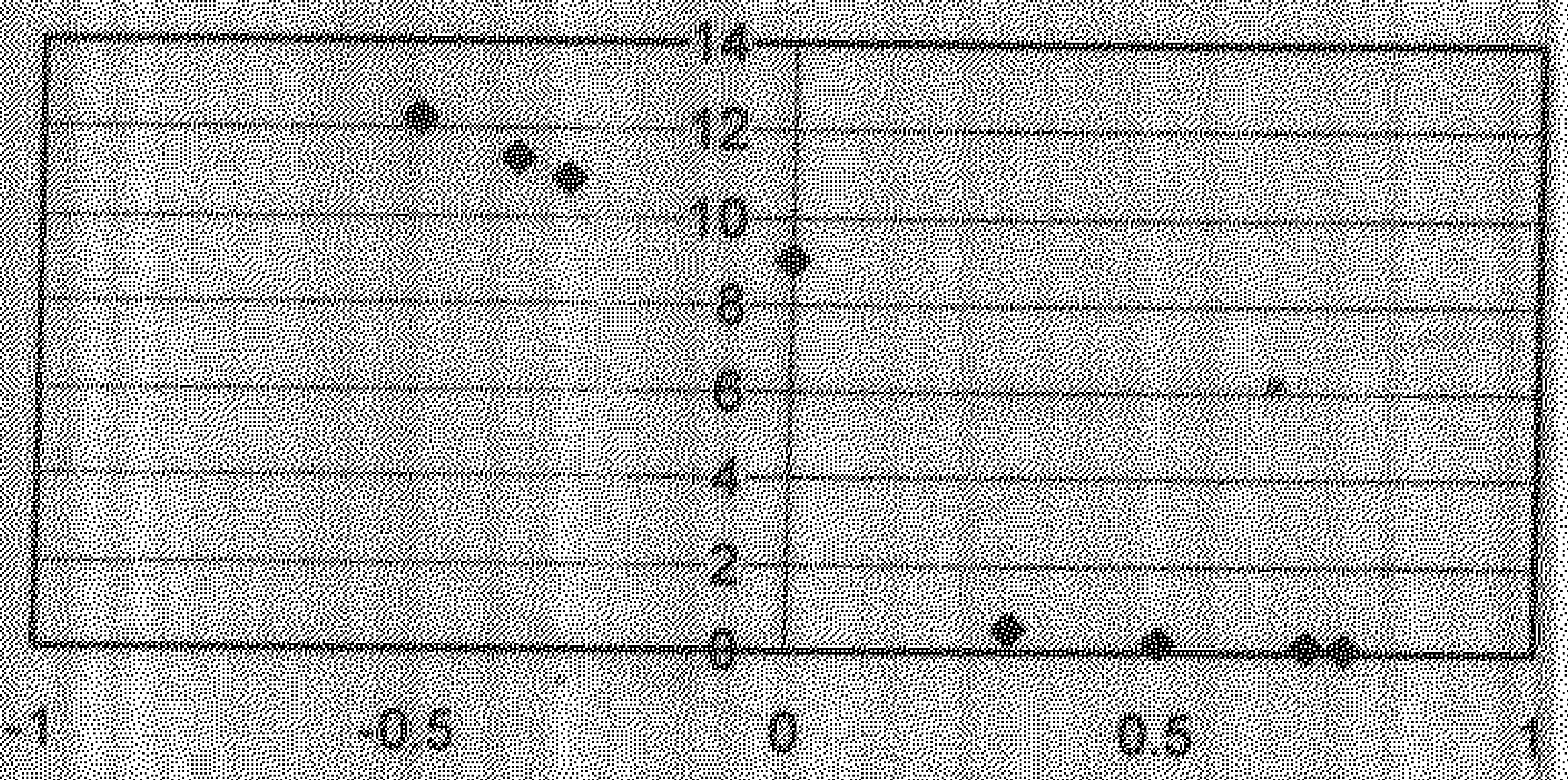

FIGURE 18a: Archive points obtained from EMOSA for Test problem II

TABLE 2: Comparison of results on test problem II

	ElitistMOSA	PAES	MOSA(avg)	MOSA(max)	MOSA(min)	MOSA(self)
ER	0.0	0.08	0.5	0.2	0.4	0.2
SPACING	0.784	2.181	3.65	3.64	2.58	3.42
SPREAD	0.612	0.537	0.689	0.59	0.72	0.66
CM	113.457	65.943	78.782	82.239	76.574	74.235

Table 2 clearly shows that in this problem also, our algorithm works in a better manner compared to the other five for all the measures. Figure 18a gives the distribution of points as obtained from EMOSA.

Problem III

The third problem is different in nature compared to first two. Zitzler et al. [19] proposed a set of problems, which have two objectives that are to be minimized.

The general construction of the problems is as follows:

$$\begin{cases} \text{Minimize } f_1(x) \\ \text{Minimize } f_2(x) = g(x)h(f_1(x), g(x)). \end{cases}$$

The problem that we studied is a 30 variable problem. It has a convex Pareto optimal set.

The functions used are as follows:

$$f_1(x) = x_1$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

All variables lie in the range [0,1]. The Pareto optimal region corresponds to $0 \le x_1^* \le 1$ and $x_i^* = 0$ for $i = 2, 3, \ldots, 30$. Figure below shows the Pareto optimal front. The difficulty that an algorithm faces while solving this problem is that of handling a large number of variables.

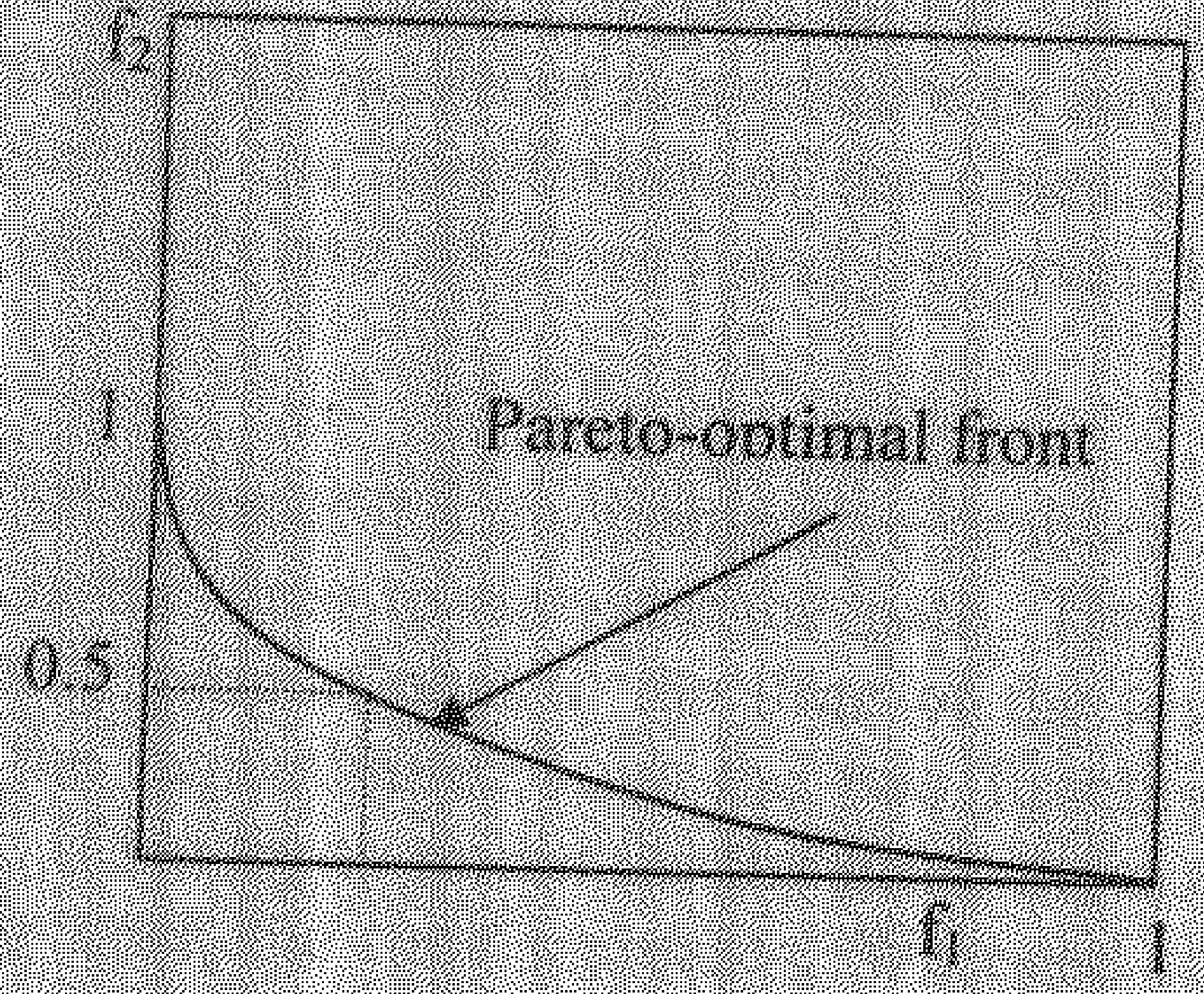

FIGURE 19: Pareto optimal front of Test problem III

TABLE 3: Comparison of results on test problem III

	ElitistMOSA	PAES	MOSA(avg)	MOSA(max)	MOSA(min)	MOSA(self)
ER	0.01	0.02	0.2	0.5	0.2	0.1
SPACING	0.382	1.436	1.629	1.376	1.429	1.376
SPREAD	0.641	0.646	0.562	0.582	0.5626	0.582
CM	76.474	35.142	12.733	19.846	13.943	16.212

This is one of the difficult problems. Our algorithm as well as PAES and MOSA fail to achieve perfect Error Ratio of 0.0 as can be seen from Table 3.

Problem IV

The next problem is a modified version of the Traveling Salesman Problem (TSP). According to this problem, a salesman must visit n cities. Modeling the problem as a complete graph with n vertices, we can say that the salesman wishes to make a tour, visiting each city exactly once and finishing the tour at the city he starts from. There is an integer cost c(i, j) attached with each pair of cities i, j. The salesman wishes to make a tour whose total cost is minimum, where the total cost is the sum of the individual costs along the edges of the tour.

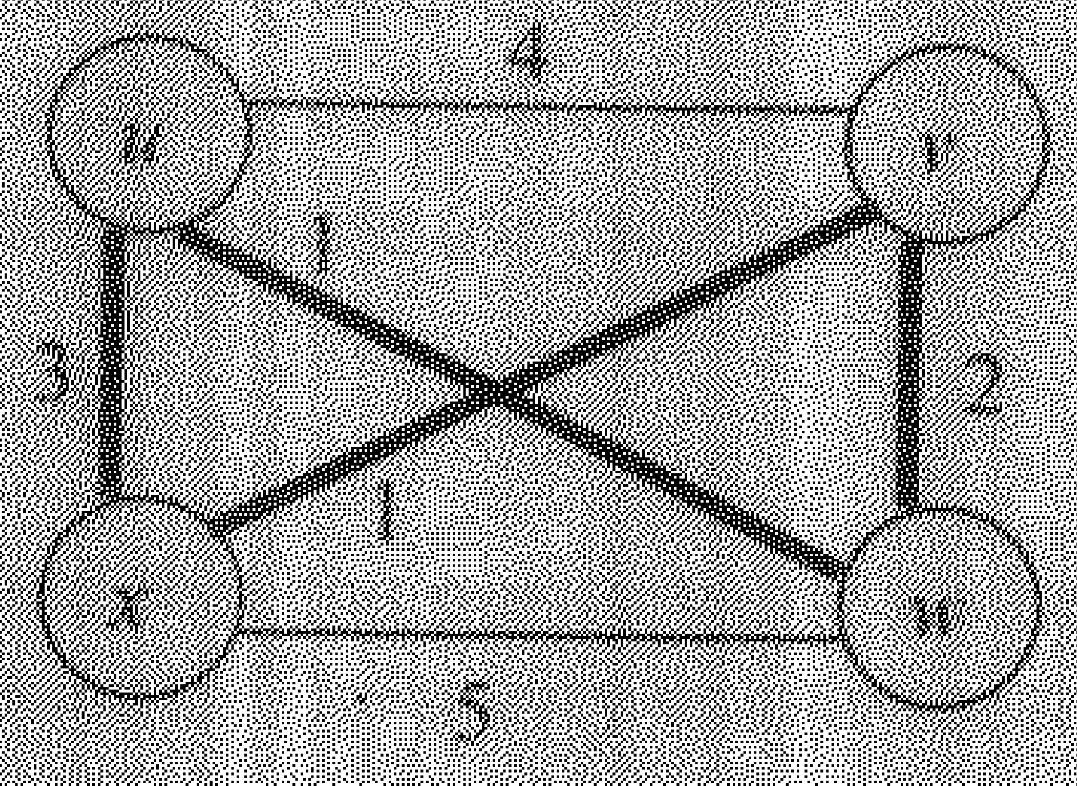

FIGURE 20: Example of Traveling Salesman Problem

For example, in the figure above, a minimum-cost tour is <*u, w, v, x, u*> with cost 7. This problem is an NP-complete problem. Proof is available in any standard literature [36]. In coding this problem, generally the coordinates of the cities are taken randomly and the cost c(i, j) associated with each pair of cities i and j is given by the Euclidean Distance between the two cities.

We have extended the TSP to make it a multiobjective problem. We have two conflicting functions. One of them is the time to traverse the path. The other function is expense function, which is inversely proportional to the time of the tour. The faster we go, i.e., smaller the time, more the expenditure. Thus the two functions are going in opposite direction. In other words, if we try to minimize the time, the cost involved increases and if we go for reducing the cost, the time increases. We need to minimize both of these conflicting functions, thus forming a true multiobjective optimization problem.

This problem is entirely different from the previous ones. All the earlier problems were purely mathematical functions. But this is a more or less real world problem. We have produced the result by taking average of ten readings each using 25 cities.

TABLE 4: Comparison of results on test problem IV

	ElitistMOSA	PAES	MOSA(avg)	MOSA(max)	MOSA(min)	MOSA(self)
SPACING	0.619	0.749	0.627	0.645	0.683	0.675
SPREAD	0.517	0.786	0.630	0.672	0.792	0.789

We have provided results based on the two measures, spacing and spread in Table 4. Here the pareto optimal front is not known. So Error Ratio has not been computed. However, we have computed the *Purity Measure* [47]. This is defined in the following manner:

Let there be N, N > 2, MOO strategies applied to a problem. Let $n_i = |R_1^i|, i = 1,2,...,N$ be the number of rank one solutions obtained from each MOO strategies. Next union of all these solution is computed as $R^* = \bigcup_{i=1}^{N} \{R_1^i\}$. Thereafter a ranking procedure is applied on R^* and the new rank one solutions, R_1^* are obtained. Let n_i^* be the number of rank one solutions which are present in R_1^*, i.e.,

$n_i^* = |\{\gamma \mid \gamma \in R_1^i and \gamma \in R_1^*\}|$

Then the purity measure for the ith MOO strategy, P_i, is defined as

$$P_i = \frac{n_i^*}{n_i} \quad i = 1,2, \ldots,N.$$

It may be noted that the purity value lie between [0, 1], where a value nearer 1 indicates better performance. The purity values for the six different algorithms are provided below.

TABLE 5: Purity Measures for TSP

ALGORITHM	**PURITY**
EMOSA	0.533
PAES	0.25
MOSAavg	0.075
MOSAmax	0.033
MOSAmin	0.1
MOSAself	0.225

Here from Table 5, we can notice that the probability of getting a rank zero solution is maximum for EMOSA and it is followed by PAES. All the variants of MOSA has a relatively lower probability of getting the same.

7.4 Minimal Deceptive Problems (MDP)

This is a different kind of problem that poses serious difficulty to any algorithm that intends to solve MOOP. In this problem, the Basin Of Attraction (BOA) of the best solution is very small whereas the second best solution or some other sub-optimal solution has a bigger BOA. This property deceives the algorithm to converge towards the non-optimal solution. Thus we do not get the correct solution, but some sub-optimal solution.

We can design a Minimal Deceptive Problem as explained below:

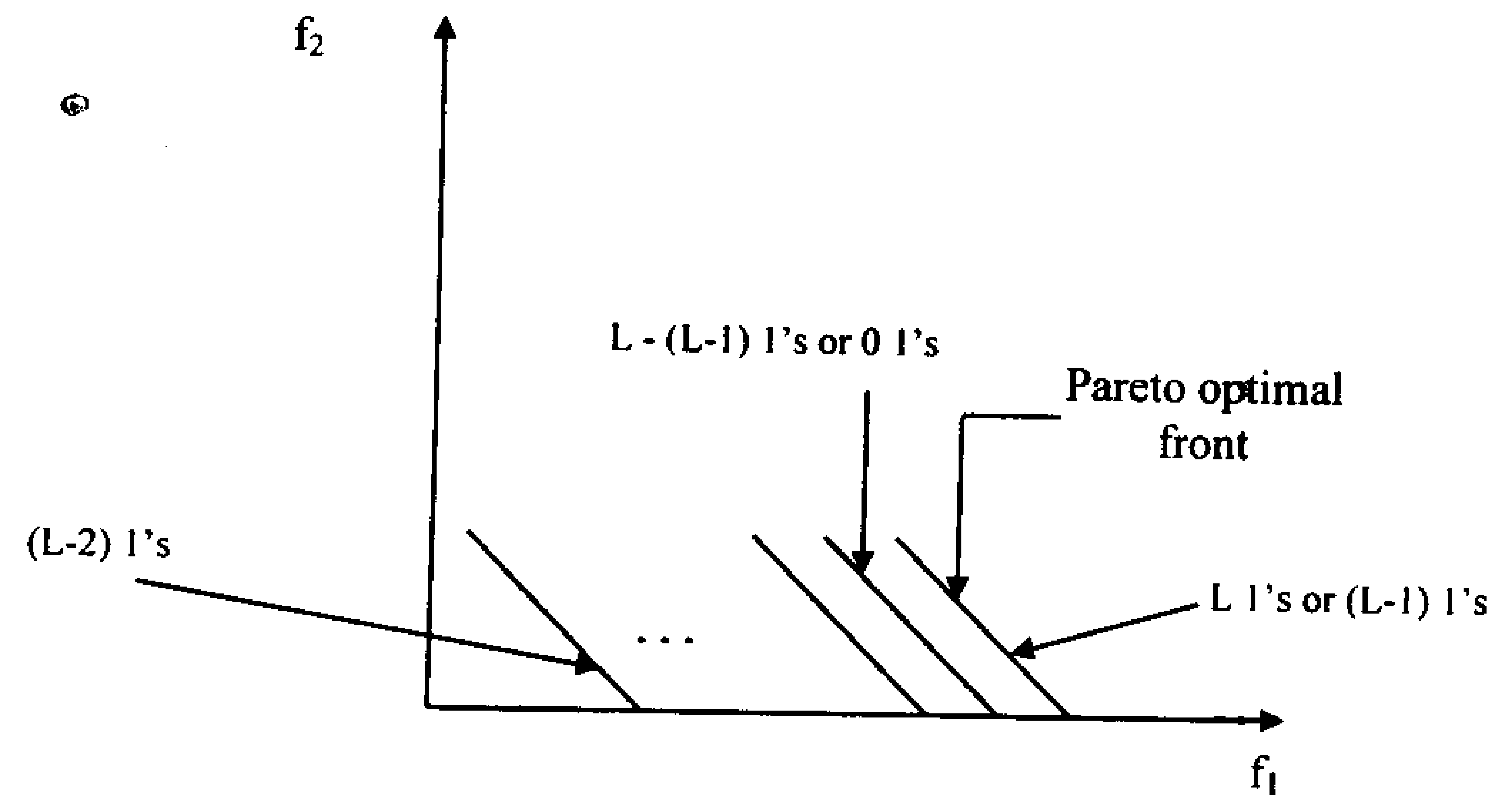

FIGURE 21: Minimal Deceptive Problem

In this problem, we handle a string of length L. The string is assumed to be binary coded. The entire problem rotates on the number of ones and number of zeros in the aforesaid string. We consider a string to be the best one when all of its positions are filled with ones, or at most one position is not filled with ones, i.e., a string with at least (L-1) 1's. The second best strings are those that have (L – (L – 1)) 1's or no 1's, as shown in Figure 21. The worst string consists of (L – 2) 1's. The following observations can be made from the above situation. First of all, all the fronts are distinct ones, without any intersections. Secondly, and most importantly, the best set of strings has a very small basin of attraction. Also the nearest string from the best string becomes the worst string. Thus this problem turns to be a minimal deceptive problem. To make the fitness values reflect the number of ones and zeros in the string in the abovementioned manner, we make use of weights corresponding to each position of the string. For the best strings, the weight value corresponding to position i is equal to $w_{ib} = (i + 1)*L$, where the positional values range from 0 to (L – 1). The fitness values are calculated in the following manner:

$f_1 = \sum w_{ib} * string[i]$ and $f_2 = \sum w_{ib} * (1 - string[i])$, for *string* containing L 1's or (L – 1) 1's.

For any other string, the weight value is $w_{io} = w_{ib} - n_l$, where n_l is the total number of ones in the string. For these non best strings, the fitness values are calculated in the following manner:

$f_1 = \sum w_{io} * (1 - string[i])$ and $f_2 = \sum w_{io} * (string[i])$

Note that for this problem both f_1 and f_2 must be maximized.

Thus in case of the best strings, the objective function value f_1 is equal to the sum of weight values of those positions, where the string has ones. The objective function value f_2 is the sum of the weight values of the positions, where the string has zeros.

For any other string, this assignment is just the opposite one. Note that the pareto optimal strings are those that contain L 1's or (L – 1) 1's whereas the second best front consists of strings having L 0's or (L – 1) 0's. In fact, excluding the pareto optimal strings, in general, both the objective functions increase as the number of 0's increases in the string. This indicates that the BOA for the second best strings is significantly larger as compared to that of pareto optimal strings.

We solved this problem using SPEA, MOSA and our proposed Elitist MOSA and the result obtained is presented below.

We have made a count of the number of points in the obtained solution set, which are best strings and also of those points that are second best strings. The count is provided below.

The result obtained from PAES is as shown in Table 6:

TABLE 6: Result of PAES on MDP

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
# Best strings	0	0	0	0	0
# 2nd Best strings	1	2	0	2	0

The result obtained from EMOSA is as shown in Table 7:

TABLE 7: Result of EMOSA on MDP

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
# Best strings	1	0	0	1	1
# 2nd Best strings	2	2	1	1	3

In case of MOSA, all the variants failed to get Best string ever, and in only one case tested (MOSAmin), it achieved one 2nd Best string in one run. These results are the reflection of the robustness of the algorithms. PAES, though performed better than MOSA, failed to achieve comparable result with respect to EMOSA.

Chapter 8

Conclusions, Discussion and Scope of Future Work

There have been many researches using evolutionary algorithms to solve MOOP. Many efficient algorithms exist in this area. But as simulated annealing uses only one search agent, it has seldom been used for solving MOOP, though it is a very powerful searching algorithm. Attempt has been made in this dissertation to make use of simulated annealing for performing multiobjective optimization. The output of any multiobjective optimization algorithm should be a set of nondominated solutions. Since simulated annealing, by its nature, deals with only a single candidate solution at a time, the concept of archive has been used for storing all the nondominated solutions that are obtained during the SA run. Moreover clustering has been used to spread out the solutions over the nondominated front as far as possible.

For the purpose of comparison, two existing algorithms have been used, both of which essentially use single point search. These two techniques are PAES, which is an (1+1) evolutionary strategy based search, and MOSA, which is based on standard simulated annealing with different cost criteria. Experimental results are provided for five multiobjective optimization problems. The EMOSA algorithm, proposed in this dissertation, is found to significantly outperform the other two methods.

Comparing solution sets obtained from different algorithms is itself a problem in multiobjective optimization. In this dissertation, we have first done a review of some of the existing measures and then a new measure has been proposed. But as more and more algorithms are developed in multiobjective optimization, some *standard* needs to be evolved regarding this issue. Otherwise it would be impossible to *effectively compare* one algorithm with another. The authors in [9] have introduced a completely new idea in this regard as already discussed. They state that there exists no unary quality measure that is able to indicate whether an approximation set is better than the other, even if we consider a finite combination of some unary measures. Most of the existing measures belong to three categories: those evaluating closeness to the Pareto-Optimal front, those evaluating diversity among non-dominated solutions and those which try to achieve both. Thus they do not follow this new idea [9]. Efforts should be made in this front to devise new measure(s) that would be capable of indicating whether an approximation set is better than another approximation set. Using most of the existing measures, although an approximation set A may be evaluated better than an approximation set B with respect to all of the indicators, B can actually be superior to A with respect to dominance relations. Work should be done in such a fashion that this kind of conflict does not arise.

Another interesting way of evaluating algorithms might be to observe the intergenerational behavior. Thus we may be able to find the way in which different algorithms converge. If we knew how is the population behaving and what issues are making it difficult to keep nondominated solutions, we could devise techniques in which the progress towards the global pareto front could be considerably faster than with the current approaches.

It is also important to define stopping criteria for a multiobjective optimization technique, because it is not obvious to know when the population has reached a point from which no further improvement can be reached. Currently, the main approaches used to stop GA based MOOP have been to either use a fixed number of generations, or to monitor the population at certain intervals and interpret visually the results to determine when to halt the evolution process. Further works are needed to investigate this area and effectively define some concrete stopping criteria both for GA based as well as SA based MOOP solving algorithms.

One big limitation with most of the problems used in the literature is that they are generally two-objective optimization problems. Two objectives make the objective space two dimensional, thereby making it easier to demonstrate the working of an algorithm. However, since most algorithms lack a rigorous mathematical proof of convergence and a proof of maximum diversity in the obtained solutions, the scalability issue must be kept in mind while developing a new algorithm solving MOOP. An algorithm should not have a serious limitation in handling more than two objectives. This can only be tested with proper test problems, having more than two objectives. Designing such test problems is another interesting area of future studies.

Another related point to be considered is the size of the random initial population required in solving problems having more than two objectives. An algorithm always works better with a bigger initial population size. But we need to find avenues to minimize this size. But an interesting topic for future research would be to check whether this size affects the final front achieved.

Any study involving development of algorithm is not complete unless they are applied to real world problem. In our case we applied our algorithm on multiobjective traveling salesman problem. But they can be applied to harder engineering problems also. Already several researchers have applied their MOOP solving algorithm in such diverse areas as VLSI circuit design, broadband microwave absorber design, marine vehicle design, gas turbine engine design to radio network optimization, medical image reconstruction and cancer chemotherapy. Some of the possible applications in the area of mechanical engineering have been explored in [19] by Deb. They include truss design, gear train design, spring design and low-thrust spacecraft trajectory optimization among others. Also several new application areas are now coming up, having a lot of scope of applying the theory of optimization, like bioinformatics. This would be a new challenge for future researchers to handle this new kind of data, having huge dimension, as well as extending the current MOOP solving techniques into these domains. The number of such applications of multiobjective optimization techniques to real-world problems is bound to increase over the years, and a probable trend in research could be to reformulate many problems that are currently considered to be single objective problems. This will constitute a more realistic approach to the solution of problems that frequently arise in areas such as engineering; because such problems are normally reduced to a single objective and the remaining objectives are treated as constraints instead of handling all (conflicting) objectives simultaneously, thus sacrificing the multiobjective flavour of the problems.

In real world, often we may be interested not in the complete solution set, rather a restricted Pareto optimal solution set, which is a subset of the complete solution set. For example, we can think of a prospective buyer of a car who is sure of emphasizing on cost *more* than comfort. Then the search should try to find a preferred set of Pareto optimal solutions with *more solutions near the minimum cost solution*. But we have to always keep in mind that we should not aim to get a single solution, because in that case we would not be in a position to present the customer any *choice* and mathematically, that would actually convert the problem to a single objective one [19]. The advantages of this kind of restricted search would be in two ways: first of all the computational time would get reduced as the search space gets restricted and secondly more trade-off solutions can be obtained in the *desired* region of interest, thus providing the user a wide range of choice. This may constitute another prospective area of future research in multiobjective optimization problems.

Bibliography

[1] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley, 1989.

[2] C. M. Fonseca and P. J. Flemming. *An overview of evolutionary algorithms in multiobjective optimization*, Evolutionary Computation, 3(1), 1995.

[3] D. E. Goldberg and J. Richardson. *Genetic algorithm with sharing for multimodal function optimization*, Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, J. J. Grefenstette, Ed., Hillsdale, NJ, Lawrence Erlbaum, 1987.

[4] E. Zitzler. L. Thiele. *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*, Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 1998.

[5] D. Nam and C. H. Park. *Multiobjective Simulated Annealing: A Comparative Study to Evolutionary Algorithms*, International Journal on Fuzzy Systems, vol. 2, no. 2, 2000.

[6] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithm*, PhD thesis, Vanderbilt University, 1984.

[7] Deb. K and Jain. S, *Running performance metrics for Evolutionary Multi-Objective Optimization*. KanGAL Report No. 2002004, 2002.

[8] Gomes. A, Antunes. C. H and Martins. A. G, *Performance Assessment of a GA for Multiobjective Problems*

[9] E. Zitzler, L. Thiele Laumanns. M, Fonseca. C. M and Fonseca V. G, *Performance Assessment of Multiobjective Optimizers: An Analysis and Review*, TIK Report No. 139, To appear in IEEE Transactions on Evolutionary Computation.

[10] Knowles. J. D and Corne. D. W, *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*, Evolutionary Computation, 8(2), 2000.

[11] Deb. K and Nain. P. K. S, *A Computationally Effective Multi-Objective Search and Optimization Technique Using Coarse-to-Fine Grain Modeling*, KanGAL Report No. 2002005, 2002.

[12] Knowles. J and Corne. D, *On Metrics for Comparing Nondominated Sets*. Congress on Evolutionary Computation (CEC 2002), Piscataway, IEEE press, 2002.

[13] Hansen. M. P and Jaszkiewicz. A, *Evaluating the quality of approximations to the non-dominated set*, Technical Report IMM-REP-1998-7 Technical University of Denmark, March 1998.

[14] Mehr. A. F and Azarm. S, *An Information-Theoretic Metric for Assessing Multi-Objective Optimization Solution Set Quality*, ASME Trans., Journal of Mechanical Design (under review).

[15] Veldhuizen. D. A. V, *Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations*, PhD Thesis, Dayton, OH: Air Force Institute of Technology, 1999.

[16] Coello C. A. C, *A Comprehensive Survey of Evolutionary-Based MultiObjective Optimization Techniques*, Knowledge and Information Systems, 1999.

[17] Knowles. J. D and Corne. D. W, *A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization*, Data Mining with Evolutionary Algorithms, Las Vegas, Nevada, USA, Ed. Alex A. Freitas, William Hart, Natalio Krasnogor and Jim Smith", 2000.

[18] Zitzler. E, Deb. K and Thiele. L, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation Journal, 8(2), 2000.

[19] Deb. K, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Ltd, 2001.

[20] Zitzler. E, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, PhD Thesis, Zurich, Switzerland, Swiss Federal Institute of Technology (ETH), Dissertation ETH No. 13398, 1999.

[21] Schott. J. R, *Fault Tolerant design using Single and Multi-Criteria Genetic Algorithms*. Master's Thesis, Boston, MA; Department of Aeronautics and Astronautics, MIT, 1995.

[22] Deb. K, Agarwal. S, Pratap. A and Meyarivan. T, *A fast and elitist multi-objective genetic algorithm: NSGA-II*, Technical report 200001, Indian Institute of Technology, Kanpur, KanGAL, 2000.

[23] Deb. K, *Genetic Algorithms in Multi-Modal Function Optimization*, Master's Thesis, Tusaloosa, AL: University of Alabama, 1989.

[24] E. Zitzler, L. Thiele, *Multiobjective Evolutionary Algorithms – A Comparative Case Study*, In Parallel Problem Solving from Nature V (PPSN-V), 1998.

[25] Veldhuizen. D. A and Lamont. G. B, *On measuring multiobjective evolutionary algorithm performance*, In Zalzala. A and Eberhart. R, editors, Congress on Evolutionary Computation (CEC 2000), volume 1, pages 204-211, Piscataway, NJ, IEEE Press, 2000.

[26] Fonseca. V. G, Fonseca. C. M and Hall. A. O, *Inferential performance assessment of stochastic optimizers and the attainment function*, In Zitzler. E, Deb. K, Thiele. L, Coello. C. A. C and Corne. D, editors, Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), volume 1993 of Lecture Notes in Computer Science, pages 213-225, Berlin, Springer-Verlag, 2001.

[27] Haykin. S, *Neural Networks, a comprehensive foundation*, second edition, Addison Wesley Longman, 2001.

[28] Metropolis. N, Rosenbluth. A. W, Rosenbloth. M. N, Teller A. H and Teller. E, *Equation of State Calculation by Fast Computing Machines*, Journal of Chemical Physics, vol. 21, 1953.

[29] Mitra. D, Romeo F and Sangiovanni-Vincentelli A, *Convergence and Finite time behavior of Simulated Annealing*, Advanced Applied Probability, vol. 18, 1986.

[30] Ingber. L, *Very Fast Simulated Re-annealing*, Mathematical and Computer Modelling, vol. 16, 1992.

[31] Szu. H. H and Hartley R. L, *Fast Simulated Annealing*, Physics Letters A, vol. 122, 1987.

[32] Yao. X, *A New Simulated Annealing Algorithm*, International Journal of Computer Mathematics, vol. 56, 1995.

[33] Morse. J. N, *Reducing the size of nondominated set: Pruning by clustering*, Computers and Operations Research, vol. 7, 1980.

[34] Rosenman. M. A and Gero. J. S, *Reducing the pareto optimal set in multicriteria optimization*, Engineering Optimization, vol. 8, 1985.

[35] Parks. G. T and Miller. I, *Selective breeding in a Multiobjective genetic algorithm*, Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Springer, Berlin, Germany, 1998.

[36] Cormen. T. H, Leiserson. C. E and Rivest. R. L, *Introduction to Algorithms*, Prentice Hall of India Private Limited, 2001.

[37] Tutorial on Evolutionary algorithms http://www.geatbx.com/docu/algindex.html

[38] Bandyopadhyay. S, Maulik. U and Pakhira. M. K, *Clustering using Simulated Annealing with probabilistic redistribution*, International Journal of Pattern Recognition and Artificial Intelligence, vol. 15, no. 2, 269-285, 2001.

[39] Bandyopadhyay. S, Pal. S. K and Murthy. C. A, *Simulated Annealing based pattern classification*, Journal of Information Sciences 109, 165-184, 1998.

[40] Davis. L, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers Inc., Los Altos, California, 1987.

[41] Kirkpatrick. S, Gelatt. S, C. D. & Vecchi. M. P, *Optimization by Simulated Annealing*, Science, 220, 671-680, 1983.

[42] Kirkpatrick. S and Vecchi. M, *Global Wiring by Simulated Annealing*, IEEE Transactions on Computer-Aided Design, vol. CAD-2, No. 4, October, 215-222, 1983.

[43] Geman. S and Geman D, *Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images*, IEEE transactions on Pattern Analysis and Machine Intelligence, PAMI-6, 721-741, 1984.

[44] Sontag. E and Sussman. H, *Image Restoration and Segmentation using the Annealing Algorithm*, In Proceedings of the 24th Conference on Decision and Control, Ft. Lauderdale, FL, December, 1985.

[45] Kirkpatrick. S, *Optimization by Simulated Annealing: Quantitative Studies*, Journal of Statistical Physics, 34, Nos. 5/6, 975-986, 1984.

[46] Hinton. G. E and Sejnowski. T. J, *Analyzing Cooperative Computation*, Proceedings of the Fifth Annual Conference of the Cognitive Science Society, Rochester, NY, May, 1983.

[47] Bandyopadhyay. S, Pal. S. K and Aruna. B, *Multi-objective GAs, Quantitative Indices and Pattern Classification*, Communicated to IEEE transactions on Systems, Man and Cybernetics Part B.

[48] Tau. J. T and Gonzalez. R. C, *Pattern Recognition Principles*, Addison Wesley, Reading, 1974.