Chapter 2
Some Single- and Multiobjective Optimization Techniques

2.1 Introduction

Optimization deals with the study of those kinds of problems in which one has to minimize or maximize one or more objectives that are functions of some real or integer variables. This is executed in a systematic way by choosing the proper values of real or integer variables within an allowed set. Given a defined domain, the main goal of optimization is to study the means of obtaining the best value of some objective function. An optimization problem [210] can be defined as follows:

Given a function $f : S \rightarrow R$ from some set S to the set of real numbers, the aim is to determine an element $\overline{x}_0$ in S such that $f(\overline{x}_0) \leq f(\overline{x}), \forall \overline{x} \in S$ ("minimization") or such that $f(\overline{x}_0) \geq f(\overline{x}), \forall \overline{x} \in S$ ("maximization").

Here, S denotes a subset of the Euclidean space R^n which is a collection of entities such as constraints, equalities or inequalities. The members of S should satisfy these entities. S, the domain of f, is called the search space, and the elements of S are called candidate or feasible solutions. The function f is called an objective function/cost function/energy function. A feasible solution that optimizes the objective function is called an optimal solution.

Multiobjective optimization (MOO) [77] (multicriteria or multiattribute optimization) deals with the task of simultaneously optimizing two or more conflicting objectives with respect to a set of certain constraints. If the optimization of one objective leads to the automatic optimization of the other, it should not be considered as MOO problem. However, in many real-life situations we come across problems where an attempt to improve one objective leads to degradation of the other. Such problems belong to the class of MOO problems and appear in several fields including product and process design, network analysis, finance, aircraft design, bioinformatics, the oil and gas industry, automobile design, etc.

In this chapter, some existing single- and multiobjective optimization techniques are first described. Thereafter, a newly developed simulated annealing-based multiobjective optimization technique named the archived multiobjective simulated annealing-based optimization technique (AMOSA) [29] is elaborately presented.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_2,

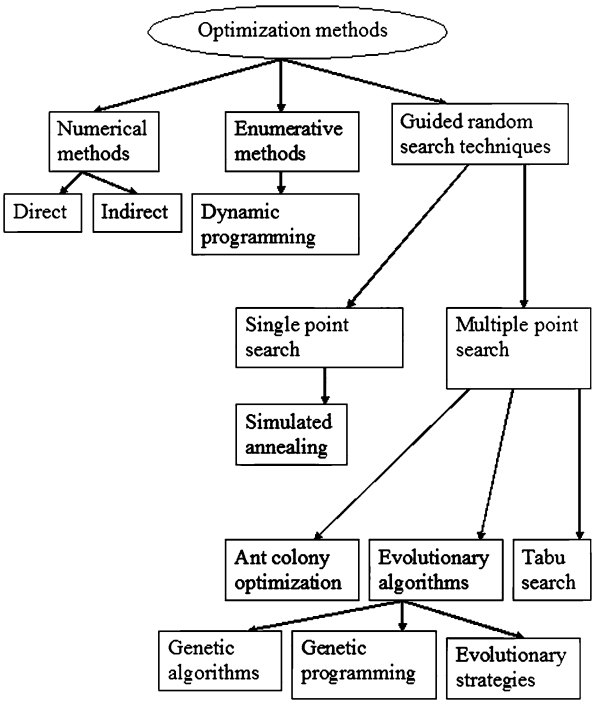

Fig. 2.1 The different search and optimization techniques

2.2 Single-Objective Optimization Techniques

Different optimization techniques that are found in the literature can be broadly classified into three categories (Fig. 2.1) [24, 112]:

- Calculus-based techniques.
- Enumerative techniques.
- Random techniques.

Numerical methods, also called calculus-based methods, use a set of necessary and sufficient conditions that must be satisfied by the solution of the optimization problem [24, 112]. They can be further subdivided into two categories, viz. direct and indirect methods. Direct search methods perform hill climbing in the function space by moving in a direction related to the local gradient. In indirect methods, the solution is sought by solving a set of equations resulting from setting the gradient of the objective function to zero. The calculus-based methods are local in scope and also assume the existence of derivatives. These constraints severely restrict their application to many real-life problems, although they can be very efficient in a small class of unimodal problems.

Enumerative techniques involve evaluating each and every point of the finite, or discretized infinite, search space in order to arrive at the optimal solution [24, 112]. Dynamic programming is a well-known example of enumerative search. It is obvious that enumerative techniques will break down on problems of even moderate

size and complexity because it may become simply impossible to search all the points in the space.

Guided random search techniques are based on enumerative methods, but they use additional information about the search space to guide the search to potential regions of the search space [24, 112]. These can be further divided into two categories, namely single-point search and multiple-point search, depending on whether it is searching just with one point or with several points at a time. Simulated annealing is a popular example of a single-point search technique that uses thermodynamic evolution to search for the minimum-energy states. Evolutionary algorithms such as genetic algorithms are popular examples of multiple-point search, where random choice is used as a tool to guide a highly explorative search through a coding of the parameter space. The guided random search methods are useful in problems where the search space is huge, multimodal, and discontinuous, and where a near-optimal solution is acceptable. These are robust schemes, and they usually provide near-optimal solutions across a wide spectrum of problems. In the remaining part of this chapter, we focus on such methods of optimization for both single and multiple objectives.

2.2.1 Overview of Genetic Algorithms

Genetic algorithms (GAs), which are efficient, adaptive, and robust search and optimization processes, use guided random choice as a tool for guiding the search in very large, complex, and multimodal search spaces. GAs are modeled on the principles of natural genetic systems, where the genetic information of each individual or potential solution is encoded in structures called *chromosomes*. They use some domain- or problem-dependent knowledge to direct the search to more promising areas; this is known as the *fitness function*. Each individual or chromosome has an associated fitness function, which indicates its degree of goodness with respect to the solution it represents. Various biologically inspired operators such as *selection, crossover*, and *mutation* are applied to the chromosomes to yield potentially better solutions.

Note that the classical gradient search techniques perform efficiently when the problems under consideration satisfy tight constraints. However, when the search space is discontinuous and/or huge in size, noisy, high dimensional, and multimodal, GAs have been found to consistently outperform both the gradient descent method and various forms of random search [24, 116]. The below discussion of GA is taken from [24].

2.2.1.1 Genetic Algorithms: Basic Principles and Features

Genetic algorithms (GAs) [75, 112, 197] are adaptive computational procedures modeled on the mechanics of natural genetic systems. They efficiently exploit historical information to speculate on new offspring with improved performance [112].

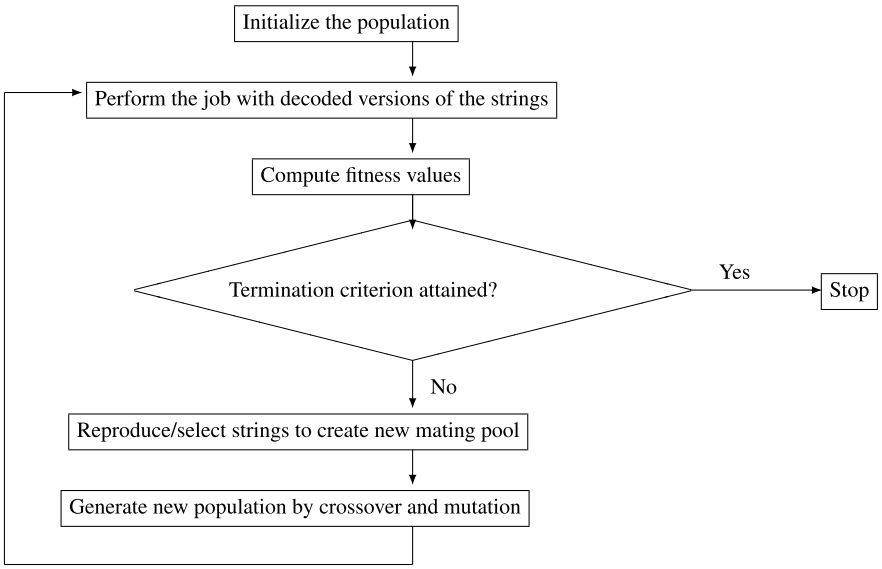

Fig. 2.2 Basic steps of a genetic algorithm

As mentioned before, GAs encode the parameters of the search space in structures called a *chromosomes* (or *strings*). They execute iteratively on a set of chromosomes, called *population*, with three basic operators: *selection/reproduction*, *crossover*, and *mutation*. GAs are different from most of the normal optimization and search procedures in four ways [112]:

- GAs work with a coding of the parameter set, not with the parameters themselves.
- GAs work simultaneously with multiple points, and not with a single point.
- GAs search via sampling (blind search) using only the payoff information.
- GAs search using stochastic operators, not deterministic rules, to generate new solutions.

Since a GA works simultaneously on a set of coded solutions, it has very little chance of getting stuck at a local optimum when used as an optimization technique. Again, the search space need not be continuous, and no auxiliary information, such as derivatives of the optimizing function, is required. Moreover, the resolution of the possible search space is increased by operating on coded (possible) solutions and not on the solutions themselves.

A schematic diagram of the basic structure of a genetic algorithm is shown in Fig. 2.2. The evolution starts from a set of chromosomes (representing a potential solution set for the function to be optimized) and proceeds from generation to generation through genetic operations. Replacement of an old population with a new one is known as a generation (or iteration) when the *generational replacement technique* (where all the members of the old population are replaced with the new ones) is used. Another population replacement technique, called *steady-state reproduc-*

tion, may be used, where one or more individuals are replaced at a time, instead of the whole population [75]. GAs require only a suitable objective function, which is a mapping from the chromosomal space to the solution space, in order to evaluate the suitability or *fitness* of the derived solutions.

A GA typically consists of the following components:

- A population of binary strings or coded possible solutions (biologically referred to as *chromosomes*).
- A mechanism to encode a possible solution (mostly as a binary string).
- An objective function and associated fitness evaluation techniques.
- A selection/reproduction procedure.
- Genetic operators (*crossover* and *mutation*).
- Probabilities to perform genetic operations.

These components are now briefly described.

Population To solve an optimization problem, GAs start with the chromosomal representation of a parameter set. The parameter set is to be coded as a finite-length string over an alphabet of finite length. Usually, the chromosomes are strings of 0s and 1s. For example, let $\{a_1, a_2, \ldots, a_p\}$ be a realization of the set of p parameters, and the binary representation of $a_1, a_2, \ldots, a_p$ be $10110, 00100, \ldots, 11001$, respectively. Then the string

$$10110\,00100\ \ldots\ 11001$$

is a chromosomal representation of the parameter set. It is evident that the number of different chromosomes (or strings) is 2^l, where l is the string length. Each chromosome actually refers to a coded possible solution. A set of such chromosomes in a generation is called a *population*, the size of which may be constant or may vary from one generation to another. A common practice is to choose the initial population randomly.

Encoding/Decoding Mechanism This is one of the primary tasks in GAs. It is the mechanism of converting the parameter values of a possible solution into strings, resulting in the chromosomal representation. If the solution of a problem depends on p parameters and if we want to encode each parameter with a string of length q, then the length, l, of each chromosome will be

$$l = p * q.$$

Decoding is the task of retrieving the parameter values from the chromosomes. It proceeds in a manner that is just the reverse of the encoding process.

One commonly used principle for coding is known as the *principle of minimum alphabet* [112]. It states that, for efficient coding, the smallest alphabet set that permits a natural expression of the problem should be chosen. In general, it has been found that the binary alphabet offers the maximum number of schemata per bit of information of any coding [112]. Hence, binary encoding is one of the commonly used strategies, although other techniques such as floating-point coding [79, 197] are also popular.

Objective Function and Associated Fitness Evaluation Techniques The fitness/objective function is chosen depending on the problem to be solved, in such a way that the strings (possible solutions) representing good points in the search space have high fitness values. This is the only information (also known as the payoff information) that GAs use while searching for possible solutions.

Selection/Reproduction Procedure The selection/reproduction process copies individual strings (called parent chromosomes) into a tentative new population (known as a mating pool) for genetic operations. The number of copies of an individual included in the next generation is usually taken to be directly proportional to its fitness value; thereby mimicking the natural selection procedure to some extent. This scheme is commonly called the *proportional selection scheme*. *Roulette wheel parent selection* [112] and *linear selection* [75] are two of the most frequently used selection procedures.

As proved in [236], one problem with *proportional selection* is that this procedure cannot guarantee asymptotic convergence to the global optima. There is no assurance that any improvement made up to a given generation will be retained in future generations. To overcome this, a commonly used strategy known as *elitist selection* [116] is adopted, thereby providing an *elitist GA* (EGA), where the best chromosome of the current generation is retained in the next generation.

Genetic operators are applied to parent chromosomes, and new chromosomes (also called offspring) are generated. The frequently used genetic operators are described below.

Crossover The main purpose of crossover is to exchange information between randomly selected parent chromosomes by recombining parts of their corresponding strings. It recombines genetic material of two parent chromosomes to produce offspring for the next generation. *Single-point crossover* is one of the most commonly used schemes. Here, first of all, the members of the reproduced strings in the mating pool are paired at random. Then an integer position k (known as the crossover point) is selected uniformly at random between 1 and $l-1$, where l is the string length greater than 1. Two new strings are created by swapping all characters from position $(k+1)$ to l. For example, let

$$a = 11000\ 10101\ 01000\ \ldots\ 01111\ 10001,$$
$$b = 10001\ 01110\ 11101\ \ldots\ 00110\ 10100$$

be two strings (parents) selected from the mating pool for crossover. Let the randomly generated crossover point be 11 (eleven). Then the newly produced offspring (swapping all characters after position 11) will be

$$a' = 11000\ 10101\ 01101\ \ldots\ 00110\ 10100,$$
$$b' = 10001\ 01110\ 11000\ \ldots\ 01111\ 10001.$$

Some other common crossover techniques are multiple-point crossover, shuffle-exchange crossover, and uniform crossover [75].

Mutation The main aim of mutation is to introduce genetic diversity into the population. Sometimes, this helps to regain information lost in earlier generations. In case of binary representation, it negates the bit value and is known as bit mutation. Like natural genetic systems, mutation in GAs is usually performed occasionally. A random bit position of a randomly selected string is replaced by another character from the alphabet. For example, let the third bit of string a, given above, be selected for mutation. Then the transformed string after mutation will be

$$11100\ 10101\ 01000\ \ldots\ 01111\ 10001.$$

High mutation rate can lead the genetic search to a random one. It may change the value of an important bit and thereby affect the fast convergence to a good solution. Moreover, it may slow down the process of convergence at the final stage of GAs.

Probabilities to Perform Genetic Operations Both the crossover and mutation operations are performed stochastically. The probability of crossover is chosen in a way so that recombination of potential strings (highly fit chromosomes) increases without any disruption. Generally, the crossover probability lies between 0.6 and 0.9 [75, 112]. Since mutation occurs occasionally, it is clear that the probability of performing a mutation operation will be very low. Typically the value lies between $1/l$ and 0.1 [75, 112].

As shown in Fig. 2.2, the cycle of selection, crossover, and mutation is repeated a number of times till one of the following occurs:

1. The average fitness value of a population becomes more or less constant over a specified number of generations.
2. A desired objective function value is attained by at least one string in the population.
3. The number of generations (or iterations) is greater than some threshold.

2.2.2 Simulated Annealing

Simulated annealing (SA) [159] is another popular search algorithm which utilizes the principles of statistical mechanics regarding the behavior of a large number of atoms at low temperature for finding minimal cost solutions to large optimization problems by minimizing the associated energy. In statistical mechanics, investigating the ground states or low-energy states of matter is of fundamental importance. These states are achieved at very low temperatures. However, it is not sufficient to lower the temperature alone, since this results in unstable states. In the annealing process, the temperature is first raised, then decreased gradually to a very low value (*Tmin*), while ensuring that one spends sufficient time at each temperature value. This process yields stable low-energy states. SA has been applied in diverse areas [22, 52, 191] by optimizing a single criterion. The below discussion of SA is taken from [26].

2.2.2.1 Basic Principle

In statistical mechanics, if a system is in thermal equilibrium, the probability $\pi_T(s)$ that the system is in state s, $s \in S$, S being the state space, at temperature T, is given by

$$\pi_T(s) = \frac{e^{\frac{-E(s)}{kT}}}{\sum_{w \in S} e^{\frac{-E(w)}{kT}}}, \tag{2.1}$$

where k is Boltzmann's constant and $E(s)$ is the energy of the system in state s.

Metropolis [195] developed a technique to simulate the behavior of a system in thermal equilibrium at temperature T as follows: Let the system be in state q at time t. Then the probability p that it will be in state s at time $t+1$ is given by the equation

$$p = \frac{\pi_T(s)}{\pi_T(q)} = e^{\frac{-(E(s)-E(q))}{kT}}. \tag{2.2}$$

If the energy of the system in state s is less than that in state q, then $p > 1$ and the state s is automatically accepted. Otherwise, it is accepted with probability p. Thus, it is also possible to attain higher energy values. It can be shown that, for $T \to \infty$, the probability that the system is in state s is given by $\pi_T(s)$ irrespective of the starting configuration [111].

2.2.2.2 Annealing Schedule

When dealing with a system of particles, it is important to investigate very low-energy states, which predominate at extremely low temperatures. To achieve such states, it is not sufficient to lower the temperature. An annealing schedule is used, where the temperature is first increased and then decreased gradually, spending enough time at each temperature in order to reach thermal equilibrium.

The annealing process of the Boltzmann machine is often used for this purpose, being a variant of the Metropolis algorithm. Here, at a given temperature T, the new state is chosen with probability

$$p_{qs} = \frac{1}{1 + e^{\frac{-(E(q,T)-E(s,T))}{T}}}. \tag{2.3}$$

2.2.2.3 Algorithm

In SAs, a configuration or a state represents a potential solution of the problem in hand. The objective value associated with the state is computed, which is analogous to the fitness computation in GA, and mapped to its energy. The state with the minimum energy value provides the solution to the problem. The initial state (say q) is generated randomly, and its energy value is computed. Keeping the initial

Fig. 2.3 Steps of simulated annealing

```
Begin
  Generate the initial state q
  T = T_max
  Let E(q, T) be the associated energy
  while (T ≥ T_min)
    for i = 1 to k
      Perturb q to yield s
      Let E(s, T) be the associated energy
      Set q ← s with probability 1/(1+e^{-(E(q,T)-E(s,T))/T})
    end for
    T = rT
  end while
Decode q to provide the solution of the problem.
End
```

temperature high (say $T = T_{max}$), a neighbor of the current state (say s) is generated. As an example, consider the states to be represented by binary strings. Then the operation of flipping a randomly selected bit can be used to generate a possible neighbor. The energy of the new state is computed, and it is accepted in favor of q with the 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 simulated annealing steps are shown in Fig. 2.3.

Simulated annealing has been successfully applied in various domains [74] including computer design [159, 160], image restoration and segmentation [259], contour detection [40, 52, 191], edge detection [171], combinatorial problems such as the traveling salesman problem [158], and artificial intelligence [22, 26]. 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. It is necessary to determine a concise description of the parameter configurations as well as an efficient method for generating configurations. Moreover, it is important to select an effective and efficient annealing schedule.

2.3 Multiobjective Optimization

In our everyday life, we make decisions consciously or unconsciously. These decisions can be very simple, such as selecting the color of a dress or deciding the menu for lunch, or may be as difficult as those involved in designing a missile or in selecting a career. The former decisions are easy to make, while the latter might take several years due to the level of complexity involved. The main goal of most kinds of decision-making is to optimize one or more criteria in order to achieve the desired result. In other words, problems related to optimization abound in real life.

The development of optimization algorithms has therefore been a 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 specific problems are known as multiobjective optimization problems (MOOPs). In this regard, a multitude of metaheuristic single-objective optimization techniques such as genetic algorithms, simulated annealing, differential evolution, and their multiobjective versions have been developed.

2.3.1 Multiobjective Optimization Problems

We encounter numerous real-life scenarios where multiple objectives need to be satisfied in the course of optimization. Finding a single solution in such cases is very difficult, if not impossible. In such problems, referred to as multiobjective optimization problems (MOOPs), it may also happen that optimizing one objective leads to some unacceptably low value of the other objective(s). For an in-depth discussion on MOOPs, the reader is referred to [62, 77]. Some definitions and basic concepts related to MOOPs are given below.

2.3.1.1 Formal Definition of MOOPs

Multiobjective optimization (MOO) can be formally stated as follows[62, 77]: Find the vector $\overline{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ of decision variables which will satisfy the m inequality constraints

$$g_i(\overline{x}) \geq 0, \quad i = 1, 2, \ldots, m, \tag{2.4}$$

the p equality constraints

$$h_i(\overline{x}) = 0, \quad i = 1, 2, \ldots, p, \tag{2.5}$$

and simultaneously optimize the M objective values

$$f_1(\overline{x}), \quad f_2(\overline{x}), \quad \ldots, \quad f_M(\overline{x}). \tag{2.6}$$

The constraints given in Eqs. 2.4 and 2.5 define the feasible region $\mathscr{F}$ which contains all the admissible solutions. Any solution outside this region is inadmissible since it violates one or more constraints. The vector $\overline{x}^*$ denotes an optimal solution in $\mathscr{F}$. In the context of multiobjective optimization, the difficulty lies in the definition of optimality, since it is only rarely that a situation can be found where a single vector $\overline{x}^*$ represents the optimum solution to all the M objective functions.

2.3.1.2 Dominance Relation and Pareto Optimality

An important concept of multiobjective optimization is that of domination. Below, a formal definition of domination is given in the context of maximization problems [77]. The definition is easily extended to minimization problems.

A solution $\overline{x}_i$ is said to dominate $\overline{x}_j$ if

$$\forall k \in 1, 2, \ldots, M, \quad f_k(\overline{x}_i) \geq f_k(\overline{x}_j) \quad \text{and}$$
$$\exists k \in 1, 2, \ldots, M \text{ such that } \quad f_k(\overline{x}_i) > f_k(\overline{x}_j).$$

This concept can be explained using a two-objective optimization problem that has five different solutions, as shown in Fig. 2.4 (example taken from [77]). Let us assume that the objective function f1 needs to be maximized while f2 needs to be minimized. Five solutions having different values of the objective functions are shown. Evidently, solution 1 dominates solution 2 since the former is better than the latter on both objectives. Again solution 5 dominates 1, but 5 and 3 do not dominate each other. 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. It may be noted that the dominance relation is irreflexive, asymmetric, and transitive in nature.

Assume a set of solutions P. The solutions of P that are not dominated by any other solution in P form the nondominated set [77]. The rank of a solution $\overline{x}$ in P is defined as the number of solutions in P that dominate $\overline{x}$ [77]. In Fig. 2.4, solutions 3 and 5 are in the nondominated set, and their ranks are 0. The non-dominated set of the entire search space S is the globally Pareto-optimal set [77]. Below, some properties of the dominance relation are mentioned.

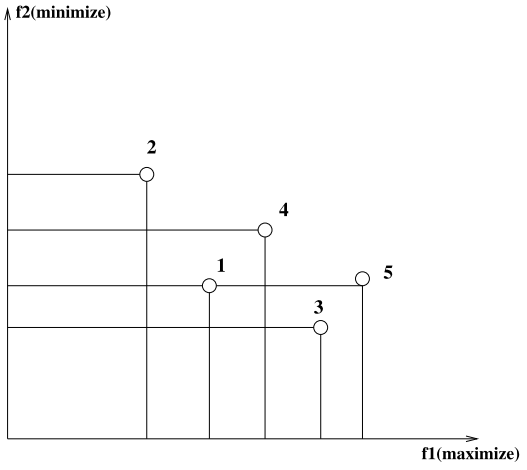

Fig. 2.4 Example of dominance and Pareto optimality

Properties of Dominance Relation [77] *Reflexive*: *The dominance relation is not reflexive, since a solution p does not dominate itself. The second condition of the definition is not satisfied in this case.*

Symmetric: *The dominance relation is not symmetric, because if a dominates b, this 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 possesses is that, if a solution p does not dominate solution q, this does not imply that q dominates p.*

2.3.1.3 Performance Measures

In order to evaluate the performance of different MOO algorithms, several measures have been proposed in the literature. Some such measures are the convergence measure γ [78], *Purity* [25], *Spacing* [250], and *Minimal Spacing* [25]. The convergence measure γ indicates how close an obtained nondominated front is from the true Pareto-optimal front. The *Purity* of a MOO algorithm measures the fraction of the nondominated solutions that remain nondominated with respect to the solutions obtained using several other MOO techniques. *Spacing* and *Minimal Spacing* measure the diversity of the solutions in the final nondominated set. More details about these measures appear later on in this chapter while reporting the comparative performance of different MOO algorithms. Apart from these, there are many other measures in the literature. Details can be found in [63, 77].

2.3.2 Various Methods to Solve MOOPs

A large number of approaches exist in the literature to solve multiobjective optimization problems [62, 77, 193]. These are aggregating, population-based non-Pareto- and Pareto-based techniques. In case of aggregating techniques, different objectives are generally combined into one using weighting or a goal-based method. One of the techniques in the population-based non-Pareto approach is the vector evaluated genetic algorithm (VEGA). Here, different subpopulations are used for the different objectives. Pareto-based approaches include multiple objective GA (MOGA), non-dominated sorting GA (NSGA), and niched Pareto GA. Note that all these techniques are essentially nonelitist in nature. Some recent elitist techniques are NSGA-II [77], the strength Pareto evolutionary algorithm (SPEA) [309], and SPEA2 [308].

Simulated annealing (SA) performs reasonably well in solving single-objective optimization problems. However, its application for solving multiobjective problems has been limited, mainly because it finds a single solution in a single run

instead of a set of solutions. This appears to be a critical bottleneck in MOOPs. However, SA has been found to have some favorable characteristics for multimodal search. The advantage of SA stems from its good selection technique. Another reason behind the good performance of SA is the annealing (the gradual temperature reduction technique). However, the disadvantage of SA is the long annealing time. There are some algorithms, namely fast simulated annealing (FSA), very fast simulated re-annealing (VFSR), new simulated annealing (NSA), etc. [138, 276, 298], that take care of this crucial issue very effectively [206]. In this chapter a newly developed multiobjective version of the standard simulated annealing algorithm [29] is elaborately described. In this context the next section reviews some existing multiobjective simulated annealing-based techniques in detail.

2.3.3 Recent Multiobjective Evolutionary Algorithms

Population-based methods such as genetic algorithms can be easily extended to solve multiobjective optimization problems. There are different approaches for solving multiobjective optimization problems [24, 62, 77, 164]. (The below discussion is taken from [24].) They are categorized as follows:

- Aggregating approaches
  1. Weighted sum approach: Here, different objectives are combined using some weights w_i, $i = 1, \ldots, M$ (where M is the number of objectives). Then using these weights the objective functions are merged into a single function. Thus, the objective to be optimized is $\sum_{i=1}^{M} w_i \times f_i(\overline{x})$.
  2. Goal programming-based approach: Here, apart from the target vector, users are asked to fix targets T_i, $i = 1, \ldots, M$, or goals for each objective f_i. The purpose is then to minimize the deviation from the targets to the objectives, i.e., $\sum_{i=1}^{M} |f_i(\overline{x}) - T_i|$.
  3. Goal attainment-based approach: Here, the users are required to provide, along with the target vector, a weight vector w_i, $i = 1, \ldots, M$, relating the relative under- or overattainment of the desired goals.
  4. ε-Constraint approach: Here, the primary objective function is to be optimized considering the other objective functions as constraints bounded by some allowable levels ε_i.
- Population-based non-Pareto approaches
  1. Vector evaluated genetic algorithm (VEGA) [249]: This algorithm incorporates a special selection operator where a number of subpopulations are generated by applying proportional selection according to each objective function in turn. This was the first multiobjective genetic algorithm (MOGA) to handle multiple objective functions.
  2. Lexicographic ordering [114]: In this approach the objectives are ranked in order of importance by the user. The optimization is performed on these objectives according to this order.

3. Game theory-based approach: This approach considers that a player is associated with each objective.
4. Use of gender for identifying the objectives: In this approach, palmitic reproduction where several parents combine to produce a single offspring is allowed.
5. Use of the contact theorem: Here the fitness of an individual is set according to its relative distance with respect to the Pareto front.
6. Use of nongenerational GA: In this strategy, a multiobjective problem is transformed into a single-objective one through a set of appropriate transformations. The fitness of an individual is calculated incrementally. Genetic operators are applied to yield a single individual that replaces the worst individual in the population.

- Pareto-based nonelitist approaches

1. Multiple objective GA (MOGA) [101]: In this approach, an individual is assigned a rank corresponding to the number of individuals in the current population by which it is dominated plus 1. All nondominated individuals are ranked 1. The fitness of individuals with the same rank is averaged so that all of them are sampled at the same rate. A niche formation method is used to distribute the population over the Pareto-optimal region. This method has a very slow convergence rate, and there are some problems related to niche size parameters.
2. Niched Pareto GA (NPGA) [91]: Here, Pareto dominance-based tournament selection with a sample of the population is used to determine the winner between two candidate solutions. Around ten individuals are used to determine dominance, and the nondominated individual is selected. If both individuals are either dominated or nondominated, then the result of the tournament is decided through fitness sharing. This method again suffers from the problem of selecting an appropriate value of the niche size parameter.
3. Nondominated sorting GA (NSGA) [261]: In this approach, all nondominated individuals are classified into one category, with a dummy fitness value proportional to the population size. This group is then removed, and the remaining population is reclassified. The process is repeated until all the individuals in the entire population are classified. Stochastic remainder proportionate selection is used here. This method has a very high convergence rate, but it also suffers from problems related to the niche size parameter.

- Pareto-based elitist approaches

1. Strength Pareto evolutionary algorithm (SPEA) [309]: This algorithm implements elitism explicitly by maintaining an external population called an archive. Hence, at any generation t both the main population P_t of constant size N and the archive population P'_t (external) of maximum size N' exist. All nondominated solutions of P_t are stored in P'_t. For each individual in the archive P'_t, a strength value similar to the ranking value in MOGA is computed. The strength of an individual is proportional to the number of individuals dominated by it. The fitness of each individual in the current population P_t

is computed according to the strength of all the archived nondominated solutions that dominate it. Moreover, average linkage clustering is used to limit the size of the external population and also to increase the diversity in the population. This algorithm is well tested, and for diversity preservation no parameter is required. Its most limiting aspect is the use of clustering.

2. Strength Pareto evolutionary algorithm 2 (SPEA2) [308]: The SPEA algorithm has two potential weaknesses. Firstly, the fitness assignment is entirely based on the strength of the archive members. This results in individuals having the same fitness value in P_t, if the corresponding set of nondominated members in the archive P'_t is the same. In the worst case, if the archive contains a single member, then all the members of P_t will have same rank. In SPEA2, a technique is developed to avoid the situation where individuals dominated by the same archive members have the same fitness values. Here, both the main population and the archive are considered to determine the fitness values of the individuals. Secondly, the clustering technique used in SPEA to maintain diversity may loose outer and boundary solutions, which should be kept in the archive in order to obtain a good spread of nondominated solutions. In SPEA2, a different scheme has been adopted that prevents the loss of the boundary solutions during the updating of the archive. Diversity is maintained by using a density-based approach on the kth nearest neighbor. This method suffers from computationally expensive fitness and density calculations.
3. Pareto archived evolutionary strategy (PAES) [161]: Knowles and Corne [161] suggested a simple MOEA using a single-parent, single-child evolutionary algorithm which is similar to a $(1 + 1)$ evolutionary strategy. Here a binary representation and bitwise mutation are used while creating offspring. First the child is created, then the objective functions are computed. Thereafter, it is compared with respect to the parent. A child is accepted as the next parent if it dominates the parent, and the iteration continues. Otherwise, the child is discarded and a new mutated solution (a new solution) is generated from the parent if the parent dominates the child. However, it is also possible that the parent and the child are nondominating to each other. In such cases, both child and the parent are compared with an archive of best solutions found so far in order to find an appropriate choice. The child is compared with all members of the archive to check if it dominates any member of the archive. If it does, the child is accepted as the new parent and all the dominated solutions are eliminated from the archive. If the archive does not have any member that is dominated by the child, then both the parent and the child are checked for their nearness to the solutions of the archive. The child is accepted as a parent if it resides in a less crowded region in the parameter space. A copy of the child is also added to the archive. In order to implement the concept of crowding, the whole solution space is divided into d^M subspaces, where d is the depth parameter and M is the number of objective functions. The subspaces are updated dynamically.
4. Pareto envelope-based selection algorithm (PESA) [66]: In this approach, a smaller internal (or primary) population and a larger external (or secondary)

population are used. PESA uses the same hypergrid division of objective space adopted by PAES to maintain diversity. Its selection mechanism is based on the crowding measure used by the hypergrid. This same crowding measure is used to decide which solutions to introduce into the external population (i.e., the archive of nondominated individuals found along the evolutionary process).

5. Pareto envelope-based selection algorithm-II (PESA-II) [65]: PESA-II is the revised version of PESA in which region-based selection is proposed. In case of region-based selection, the unit of selection is a hyperbox instead of an individual. The procedure of selection is to select (using any of the traditional selection techniques) a hyperbox and then randomly select an individual within such hyperbox. The main advantage of this algorithm is that it is easy to implement and computationally very efficient. The performance of this algorithm depends on the gridsize. In order to draw grids in the objective space, prior information is needed for the objective space.
6. Elitist non-dominated sorting GA (NSGA-II) [78]: NSGA-II was proposed to eliminate the weaknesses of NSGA, especially its nonelitist nature and specification of the sharing parameter. Here, the individuals in a population undergo nondominated sorting as in NSGA, and individuals are given ranks based on this. A new selection technique, called crowded tournament selection, is proposed where selection is done based on crowding distance (representing the neighborhood density of a solution). To implement elitism, the parent and child population are combined and the nondominated individuals from the combined population are propagated to the next generation. NSGA-II is one of the widely used MOO algorithms, and is therefore described in more detail below [24].

   The non-dominated sorting method is an important characteristic of NSGA-II. This is performed as follows: Given a set of solutions S, the nondominated set of solutions $N \subseteq S$ is composed of those solutions of S which are not dominated by any other solution in S. To find the nondominated set, the following steps are carried out [24, 78]:
   - Step 1: Set $i = 1$ and initialize the nondominated set N to empty.
   - Step 2: For each solution $j \in S$ ($j \neq i$), if solution j dominates solution i then go to step 4.
   - Step 3: If $j < \|S\|$, set $j = j + 1$ and go to step 2. Otherwise, set $N = N \cup i$.
   - Step 4: Set $i = i + 1$. If $i \leq \|S\|$ then go to step 2. Otherwise, output N as the nondominated set.

   The nondominated sorting procedure first finds the nondominated set N from the given set of solutions S. Each solution belonging to N is given the rank 1. Next the same process is repeated on the set $S = S - N$ and the next set of nondominated solutions N' is found. Each solution of the set N' is given the rank 2. This procedure goes on until all the solutions in the initial set are given some rank, i.e., S becomes empty.

   A measure called crowding distance has been defined on the solutions of the nondominated front for diversity maintenance. The crowding distances for

the boundary solutions are set to maximum values (logically infinite). For each solution i among the remaining solutions, the crowding distance is computed as the average distance of the $(i+1)$th and $(i-1)$th solutions along all the objectives. The following are the steps for computing the crowding distance d_i of each point i in the nondominated front N [78]:

- For $i = 1, \ldots, N$, initialize $d_i = 0$.
- For each objective function f_k, $k = 1, \ldots, M$, do the following:
  - Sort the set N according to f_k in ascending order.
  - Set $d_1 = d_{\|N\|} = \infty$.
  - For $j = 2$ to $(\|N\| - 1)$, set $d_j = d_j + (f_{k(j+1)} - f_{k(j-1)})$.

In NSGA-II, a binary tournament selection operator is used based on the crowding distance. If two solutions a and b are compared during a tournament, then solution a wins the tournament if either:

- The rank of a is better (less) than the rank of b, i.e., a and b belong to two different nondominated fronts, or
- The ranks of a and b are the same (i.e., they belong to the same nondominated front) and a has higher crowding distance than b. This means that, if two solutions belong to the same nondominated front, the solution situated in the lesser crowded region is selected.

The following are the main steps of the NSGA-II algorithm [24, 78]:

- Initialize the population.
- While the termination criterion is not met, repeat the following:
  - Evaluate each solution in the population by computing m objective function values.
  - Rank the solutions in the population using nondominated sorting.
  - Perform selection using the crowding binary tournament selection operator.
  - Perform crossover and mutation (as in conventional GA) to generate the offspring population.
  - Combine the parent and child populations.
  - Replace the parent population by the best members (selected using nondominated sorting and the crowded comparison operator) of the combined population.
- Output the first nondominated front of the final population.

2.3.4 MOOPs and SA

As already mentioned, one of the major obstacles to use SA for MOOPs is that it produces only a single solution at a time. Since solving a 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. Good reviews of multiobjective simulated annealing techniques can be found in Chap. 9 of [63] and in [274]. A part of the following discussion is based on [63] and [274].

In most of the earlier attempts, a single objective function was constructed by combining the different objectives into one [68, 90, 123, 207, 252, 275, 285, 286]. In general, a weighted sum approach is used, where the objectives are combined as $\sum_{i=1}^{M} w_i f_i(\overline{x})$. Here $f_i(\overline{x})$, $1 \leq i \leq M$, are the M different objective functions defined on the solution vector $\overline{x}$, and the w_is are the corresponding weights. This composite objective is then used as the energy to be minimized in a scalar SA optimization method. Evidently, a major problem here is the appropriate selection of the weights. Some alternative approaches have also been used in this regard. For example, in [90], the sum of $\log f_i(\overline{x})$ is taken. Additionally, the weighted summation essentially provides a convex combination of the different objectives. Solutions in the concave regions of the Pareto surface will never be considered at all.

Multiobjective simulated annealing with a composite energy clearly converges to the true Pareto front if the objectives have ratios given by w_i^{-1}. In [69], it is proved that part of the front will be inaccessible with fixed weights. In [147], several different schemes are explored for adapting the w_i during the annealing process to encourage exploration along the front. However, a proper choice of the w_i remains a challenging task.

An integration of the weighted sum approach and the concept of Pareto optimality was introduced in [68]. The algorithm, called Pareto simulated annealing (PSA), uses a population instead of a single solution at each iteration. The non-dominated vectors are stored in an external file, and quadtrees are used to store and retrieve them efficiently. Another new approach taken into consideration in PSA is that, when a new solution $\overline{x}'$ is generated in the neighborhood of $\overline{x}$ (where $\overline{x}'$ denotes the closest neighborhood solution of $\overline{x}$), the weights are incremented or decremented for those objectives depending upon whether $f(\overline{x})$ dominates $f(\overline{x}')$ or $f(\overline{x}')$ dominates $f(\overline{x})$. The main goal is to increase the probability of moving far from $f(\overline{x})$ as much as possible. The concept of parallelism is applicable to this approach as the computations required at each step can be parallelized. Experimental studies demonstrate that PSA generates more solutions on the true Pareto-optimal front and the solutions are also well distributed. PSA has been applied to solve various real-world problems.

SA is used with an energy function that transforms the MOO problem into a single-objective min-max problem in [56]. Here, the problem is to minimize the maximum deviations of each objective with respect to a set of user-defined goals. Suppapitnarm et al. [275] have used an approach where the non-dominated solutions are stored in an external file. This algorithm basically employs a single-point search. In order to be added to the external file, a new solution has to be nondominated with respect to all the solutions of the external file. This external population plays the role of a population. If the newly generated solution is archived, then it is selected as the new search starting point. Otherwise, the acceptance probability is given by $p = \prod_{i=1}^{k} \exp\{-\frac{f_i(\overline{x}) - f_i(\overline{x}')}{T_i}\}$, where k is the number of objective functions, T_i is the temperature associated with objective $f_i(\overline{x})$, and $\overline{x}$ and $\overline{x}'$ denote the current and new solution, respectively. A potential solution will be evaluated based on this acceptance criterion. It is treated as the new starting point of search once accepted, else the previous solution is considered again as the starting point. There

is also a strategy called "return to base" by which the currently accepted solution is replaced by some randomly chosen solution from the external file. This helps the SA to maintain diversity and avoid convergence to a local optimal front. However, authors have shown that their approach does not provide good results as compared with MOGA; its only advantage is its simplicity.

In [207] and [286] different nonlinear and stochastic composite energy functions have been investigated. In [207] six different criteria for energy difference calculation for MOSA are suggested and evaluated. These are (i) minimum cost criterion, (ii) maximum cost criterion, (iii) random cost criterion, (iv) self cost criterion, (v) average cost criterion, and (vi) fixed cost criterion. Since each run of the SA provides just a single solution, the algorithm attempts to evolve the set of Pareto-optimal (PO) solutions by using multiple SA runs. As a result of the independent runs, the diversity of the set of solutions suffer. After performing some comparative study, the authors conclude that the criteria that work best are the random, average, and fixed criteria [63]. For comparison with respect to some existing multiobjective evolutionary algorithms (MOEAs), the average criterion is considered. Authors have compared their approach with respect to the NPGA [91]. The proposed approach presents competitive performance but has some diversity problems. The use of niches is suggested to deal with this problem.

A modified multiobjective simulated annealing method named Pareto cost simulated annealing (PCSA) is proposed in [206]. Here, the cost of a state is computed by sampling either the neighborhood or the whole population. These techniques are very similar to the tournament selection of the NPGA with a small tournament size in the first case and the whole population size in the second case. PCSA was compared with the existing MOEA techniques, MOGA [101], NPGA [91], and NSGA [261] for 18 test problems. These problems are basically two-objective problems, having two to four variables. Authors have shown that, in 67 % of the cases, PCSA performs better than the MOEAs.

A multiobjective version of simulated annealing based on Pareto dominance is proposed by Suman [270–273]. An external archive and a scheme to handle constraints within the expression used to determine the probability of moving to a different state are proposed. A weight vector is calculated for the acceptance criterion. Each weight vector considers the number of constraints satisfied by a particular solution. In another work, five multiobjective extensions of SA, Suppapitnarm multiobjective simulated annealing (SMOSA) [271], Ulungu multiobjective simulated annealing (UMOSA) [272], Pareto simulated annealing (PSA) [273], weight-based multiobjective simulated annealing (WMOSA) [270] and Pareto dominant based multiobjective simulated annealing (PDMOSA) are compared for several constrained multiobjective optimization problems. The selection criterion adopted by SPEA [309] is used in PDMOSA. Here, solutions stored in the external archive take part in the selection process. Constraints are handled through the penalty function approach. Results show that PSA [273] provides the best qualitative results, whereas PDMOSA provides the best results with respect to diversity. Some more recent Pareto-dominance based multiobjective SA methods are proposed in [257, 258, 270].

In the Pareto domination-based multiobjective SAs developed relatively recently [257, 258, 270], the acceptance criterion between the current and a new solution has often been formulated in terms of the difference in the number of solutions that they dominate. One of the recently developed MOSA algorithms is by Smith et al. [257, 258]. Here, a dominance-based energy function is used. If the true Pareto front is available, then the energy of a particular solution $\overline{x}$ is calculated as the total number of solutions that dominate $\overline{x}$. However, as the true Pareto front is not available all the time, a proposal has been made to estimate the energy based on the current estimate of the Pareto front, F', which is the set of mutually nondominating solutions found thus far in the process. Then the energy of the current solution $\overline{x}$ is the total number of solutions in the estimated front which dominate $\overline{x}$. If $\|F'_{\overline{x}'}\|$ is the energy of the new solution $\overline{x}'$ and $\|F'_{\overline{x}}\|$ is the energy of the current solution $\overline{x}$, then the energy difference between the current and the proposed solution is calculated as $\delta E(\overline{x}', \overline{x}) = \frac{(\|F'_{\overline{x}'}\| - \|F'_{\overline{x}}\|)}{\|F'\|}$. Division by $\|F'\|$ ensures that δE is always less than unity and provides some robustness against fluctuations in the number of solutions in F'. If the size of F' is less than some threshold, then the attainment surface sampling method is adopted to increase the number of solutions on the final Pareto front. Authors have perturbed a decision variable with a random number generated from a Laplacian distribution. Two different sets of scaling factors, namely traversal scaling which generates moves to a nondominated proposal within a front and location scaling which locates a front closer to the original front, are kept. These scaling factors are updated with the iterations. A newly developed multiobjective SA termed archived multiobjective simulated annealing (AMOSA), which incorporates a concept of amount of dominance in order to determine the acceptance of a new solution as well as situation-specific acceptance probabilities, is described in detail in Sect. 2.4.

2.4 An Archive-Based Multiobjective Simulated Annealing Technique: AMOSA

2.4.1 Introduction

In Pareto domination-based multiobjective SAs developed so far, the acceptance criterion between the current and a new solution has been formulated in terms of the difference in the number of solutions that they dominate [257, 270]. Here, a newly developed multiobjective SA, referred to as archived multiobjective simulated annealing (AMOSA), is presented which incorporates a concept of amount of dominance in order to determine the acceptance of a new solution [29]. The Pareto-optimal (PO) solutions are stored in an archive. A complexity analysis of AMOSA is also provided. The performance of AMOSA has been compared with the two other well-known MOEA's, namely NSGA-II [78] and PAES [161] (described earlier) for several function optimization problems when binary encoding is used. The

comparison has been made in terms of several performance measures, namely *Convergence* [78], *Purity* [25, 140], *Spacing* [250], and *Minimal Spacing* [25]. Another measure called *displacement* [68, 141], which reflects both the proximity to and the coverage of the true PO front, is also used here for the purpose of comparison. This measure is especially useful for discontinuous fronts in order to estimate if the solution set is able to approximate all the subfronts. Many existing measures are unable to achieve this.

It may be noted that the multiobjective SA methods developed in [257, 270] are on lines similar to AMOSA. The concept of an archive or set of potentially PO solutions is also utilized in [257, 270] for storing the nondominated solutions. Instead of scalarizing the multiple objectives, a domination-based energy function is defined. However, there are notable differences. Firstly, while the number of solutions that dominate the new solution x determined the acceptance probability of x in the earlier attempts, in AMOSA this is based on the amount of domination of x with respect to the solutions in the archive and the current solution. In contrast to the works in [257, 270], where a single form of acceptance probability is considered, AMOSA deals with different forms of acceptance probabilities depending on the domination status, the choice of which are explained intuitively later on.

In [257] an unconstrained archive is maintained. Note that, theoretically, the number of Pareto-optimal solutions can be infinite. Since the ultimate purpose of a MOO algorithm is to provide the user with a set of solutions to choose from, it is necessary to limit the size of this set for it to be usable by the user. Though maintaining unconstrained archives as in [257] is novel and interesting, it is still necessary to finally reduce it to a manageable set. Limiting the size of the population (as in NSGA-II) or the archive (as in AMOSA) is an approach in this direction. Clustering appears to be a natural choice for reducing the loss of diversity, and this is incorporated in AMOSA. Clustering has also been used earlier in [309].

For comparing the performance of real-coded AMOSA with that of multiobjective SA (MOSA) [257], six three-objective test problems, namely, DTLZ1-DTLZ6 are used. Results demonstrate that the performance of AMOSA is comparable to, and often better than, that of MOSA in terms of *Purity*, *Convergence*, and *Minimal Spacing*. Comparison is also made with real-coded NSGA-II for the above-mentioned six problems, as well as for some 4-, 5-, 10-, and 15-objective test problems. Results show that the performance of AMOSA is superior to that of NSGA-II, especially for the test problems with many objective functions. This is an interesting and the most desirable feature of AMOSA, since Pareto ranking-based MOEAs, such as NSGA-II, [78] do not work well on many-objective optimization problems as pointed out in some recent studies [137, 139].

2.4.2 Archived Multiobjective Simulated Annealing (AMOSA)

AMOSA incorporates the concept of an *Archive* where the nondominated solutions seen so far are stored. In [99], the use of an unconstrained *Archive* size to reduce the

loss of diversity is discussed in detail. In this approach the archive size is kept limited, since finally only a small number of well-distributed Pareto-optimal solutions are needed. Two limits are kept on the size of the *Archive*: a hard or strict limit denoted by *HL*, and a soft limit denoted by *SL*. During the process, the nondominated solutions are stored in the *Archive* as and when they are generated until the size of the *Archive* increases to *SL*. Thereafter, if more nondominated solutions are generated, these are added to the *Archive*, the size of which is thereafter reduced to *HL* by applying clustering. The structure of the simulated annealing-based AMOSA is shown in Fig. 2.5. The parameters that need to be set a priori are mentioned below:

- *HL*: The maximum size of the *Archive* on termination. This set is equal to the maximum number of nondominated solutions required by the user.
- *SL*: The maximum size to which the *Archive* may be filled before clustering is used to reduce its size to *HL*.
- *Tmax*: Maximum (initial) temperature.
- *Tmin*: Minimal (final) temperature.
- *iter*: Number of iterations at each temperature.
- α: The cooling rate in SA.

The different steps of the algorithm are now explained in detail.

2.4.3 Archive Initialization

The algorithm begins with the initialization of a number $\gamma \times SL$ ($\gamma > 1$) of solutions. Each of these solutions is refined by using a simple hill-climbing technique, accepting a new solution only if it dominates the previous one. This is continued for a number of iterations. Thereafter, the nondominated solutions (*ND*) that are obtained are stored in the *Archive*, up to a maximum of *HL*. In case the number of nondominated solutions exceeds *HL*, clustering is applied to reduce the size to *HL* (the clustering procedure is explained below). This means that initially *Archive* contains a maximum of *HL* number of solutions.

In the initialization phase it is possible to get an *Archive* of size one. In MOSA [257], in such cases, other newly generated solutions which are dominated by the archival solution will be indistinguishable. In contrast, the amount of domination as incorporated in AMOSA will distinguish between "more dominated" and "less dominated" solutions. However, in the future, it is intended to use a more sophisticated scheme, in line with that adopted in MOSA.

2.4.4 Clustering the Archive Solutions

Clustering of the solutions in the *Archive* has been incorporated in AMOSA in order to explicitly enforce diversity of the nondominated solutions. In general, the size of

Set *Tmax*, *Tmin*, *HL*, *SL*, *iter*, α, temp=*Tmax*.
Initialize the *Archive*.
current-pt = random(*Archive*). /* randomly chosen solution from *Archive**/
while (temp > *Tmin*)
  for (i=0; i< iter; i++)
    new-pt=perturb(*current-pt*).
    Check the domination status of *new-pt* and *current-pt*.
    /* Code for different cases */
    if (*current-pt* dominates *new-pt*) /* Case 1*/
      $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k}(\Delta dom_{i,new\text{-}pt}) + \Delta dom_{current\text{-}pt,new\text{-}pt}\right)}{(k+1)}$.
      /* k=total-no-of points in the *Archive* which dominate *new-pt*, $k \geq 0$. */
      Set *new-pt* as *current-pt* with probability calculated using Eq. 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg}.

    if (*current-pt* and *new-pt* are nondominating to each other) /* Case 2*/
      Check the domination status of *new-pt* and points in the *Archive*.
      if (*new-pt* is dominated by k ($k \geq 1$) points in the *Archive*) /* Case 2(a)*/
        $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k} \Delta dom_{i,new\text{-}pt}\right)}{k}$.
        Set *new-pt* as *current-pt* with probability calculated using Eq. 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg}.
      if (*new-pt* is non-dominating w.r.t all the points in the *Archive*) /* Case 2(b)*/
        Set *new-pt* as *current-pt* and add *new-pt* to the *Archive*.
        if *Archive-size* > *SL*
          Cluster *Archive* to *HL* number of clusters.
      if (*new-pt* dominates k, ($k \geq 1$) points of the *Archive*) /* Case 2(c)*/
        Set *new-pt* as *current-pt* and add it to *Archive*.
        Remove all the k dominated points from the *Archive*.
    if (*new-pt* dominates *current-pt*) /* Case 3 */
      Check the domination status of *new-pt* and points in the *Archive*.
      if (*new-pt* is dominated by k ($k \geq 1$) points in the *Archive*) /* Case 3(a)*/
        Δdom_{min} = minimum of the difference of domination amounts between the *new-pt* and the k points
        $prob = \frac{1}{1+exp(-\Delta dom_{min})}$.
        Set point of the archive which corresponds to Δdom_{min} as *current-pt*
        with probability=*prob*.
        else set *new-pt* as *current-pt*
      if (*new-pt* is nondominating with respect to the points in the *Archive*) /* Case 3(b) */
        select the *new-pt* as the *current-pt* and add it to the *Archive*.
        if *current-pt* is in the *Archive*, remove it from *Archive*.
        else if *Archive-size*> *SL*.
          Cluster *Archive* to *HL* number of clusters.
      if (*new-pt* dominates k other points in the *Archive*) /* Case 3(c)*/
        Set *new-pt* as *current-pt* and add it to the *Archive*.
        Remove all the k dominated points from the *Archive*.
  End for
$temp = \alpha * temp$.
End while
if *Archive-size* > *SL*
  Cluster *Archive* to *HL* number of clusters.

Fig. 2.5 The AMOSA algorithm [29] (source code is available at: www.isical.ac.in/~sriparna_r)

the *Archive* is allowed to increase up to *SL* (>*HL*), after which the solutions are clustered for grouping the solutions into *HL* clusters. Allowing the *Archive* size to increase up to *SL* not only reduces excessive calls to clustering, but also enables the formation of more spread-out clusters and hence better diversity. Note that, in the initialization phase, clustering is executed once even if the number of solutions in the *Archive* is less than *SL*, as long as it is greater than *HL*. This enables it to start with at most *HL* nondominated solutions and reduces excessive calls to clustering in the initial stages of the AMOSA process.

For clustering, the well-known single linkage algorithm [143] has been used. Here, the distance between any two clusters corresponds to the length of the shortest link between them. This is similar to the clustering algorithm used in SPEA [309], except that they used the average linkage method [143]. After *HL* clusters are obtained, the member within each cluster whose average distance to the other members is the minimum is considered as the representative member of the cluster. A tie is resolved arbitrarily. The representative points of all the *HL* clusters are thereafter stored in the *Archive*.

2.4.5 Amount of Domination

As already mentioned, AMOSA uses the concept of amount of domination in computing the acceptance probability of a new solution. Given two solutions a and b, the amount of domination is defined as $\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{M} \frac{|f_i(a) - f_i(b)|}{R_i}$ where M, is the number of objectives and R_i is the range of the ith objective. Note that, in several cases, R_i may not be known a priori. In such situations, the solutions present in the *Archive* along with the new and the current solutions are used for computing it. The concept of $\Delta dom_{a,b}$ is illustrated pictorially in Fig. 2.6 for the two-objective case. $\Delta dom_{a,b}$ is used in AMOSA while computing the probability of acceptance of a newly generated solution.

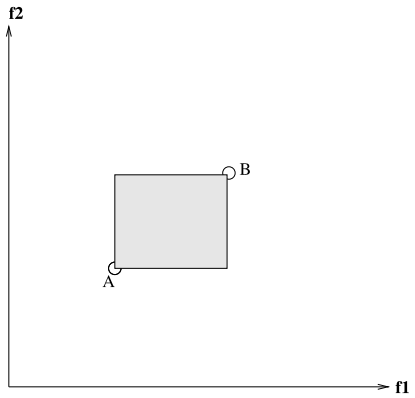

Fig. 2.6 Total amount of domination between the two solutions A and B = the area of the shaded rectangle

2.4.6 The Main AMOSA Process

One of the points, called *current-pt*, is randomly selected from *Archive* as the initial solution at temperature $temp = Tmax$. *current-pt* is perturbed to generate a new solution called *new-pt*. The domination status of *new-pt* is checked with respect to *current-pt* and solutions in the *Archive*.

Based on the domination status between *current-pt* and *new-pt*, three different cases may arise. These are enumerated below:

- Case 1: *current-pt* dominates *new-pt*, and k ($k \geq 0$) points from the *Archive* dominate *new-pt*.

  This situation is shown in Fig. 2.7. Figures 2.7(a) and 2.7(b) denote the situations where $k = 0$ and $k \geq 1$, respectively. (Note that all the figures correspond to a two-objective maximization problem.) In this case, *new-pt* is selected as *current-pt* with a probability calculated using the following equation (as in Eq. 2.3):

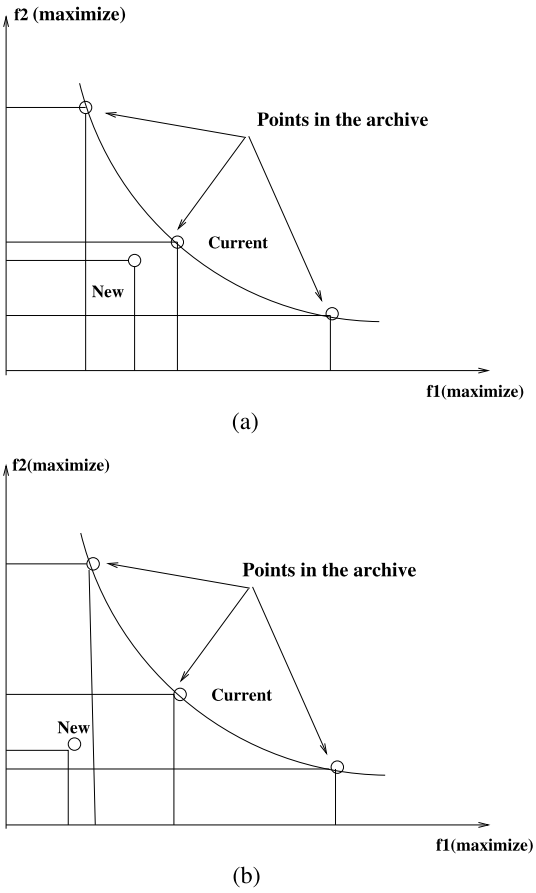

Fig. 2.7 Different cases when *New* is dominated by *Current*: (**a**) *New* is nondominating with respect to the solutions of *Archive* except *Current* if it is in the *Archive*; (**b**) Some solutions in the *Archive* dominate *New*

$$p_{qs} = \frac{1}{1 + e^{\frac{\Delta dom_{avg}}{T}}}, \tag{2.7}$$

where

$$\Delta dom_{avg} = \left(\sum_{i=1}^{k} (\Delta dom_{i,new\text{-}pt}) + \Delta dom_{current\text{-}pt,new\text{-}pt} \right) / (k+1).$$

Note that Δdom_{avg} denotes the average amount of domination of *new-pt* by $(k+1)$ points, namely, *current-pt* and k points of the *Archive*. Also, as k increases, Δdom_{avg} will increase, since here dominating points that are further away from the *new-pt* are contributing to its value.

Lemma *When $k = 0$, current-pt is on the archival front.*

Proof If this is not the case, then some point, say A, in the *Archive* dominates it. Since *current-pt* dominates *new-pt*, by transitivity, A will also dominate *new-pt*. However, it is considered that no other point in the *Archive* dominates *new-pt*, as $k = 0$. Hence proved. □

However, if $k \geq 1$, this may or may not be true.

- Case 2: *current-pt* and *new-pt* are non-dominating with respect to each other.
  Now, based on the domination status of *new-pt* and members of *Archive*, the following three situations may arise:

  1. *new-pt* is dominated by k points in the *Archive*, where $k \geq 1$. This situation is shown in Fig. 2.8(a). *new-pt* is selected as *current-pt* with a probability calculated using the following equation (as in Eq. 2.3):

  $$p_{qs} = \frac{1}{1 + e^{\frac{\Delta dom_{avg}}{T}}}, \tag{2.8}$$

  where

  $$\Delta dom_{avg} = \sum_{i=1}^{k} (\Delta dom_{i,new\text{-}pt}) / k.$$

  Note that here *current-pt* may or may not be on the archival front.
  2. *new-pt* is nondominating with respect to the other points in the *Archive* as well. In this case *new-pt* is on the same front as the *Archive*, as shown in Fig. 2.8(b). So *new-pt* is selected as *current-pt* and added to the *Archive*. In case the *Archive* becomes overfull (i.e., *SL* is exceeded), clustering is performed to reduce the number of points to *HL*.
  3. *new-pt* dominates k ($k \geq 1$) points of the *Archive*. This situation is shown in Fig. 2.8(c). Again, *new-pt* is selected as *current-pt* and added to the *Archive*. All the k dominated points are removed from the *Archive*. Note that here too *current-pt* may or may not be on the archival front.

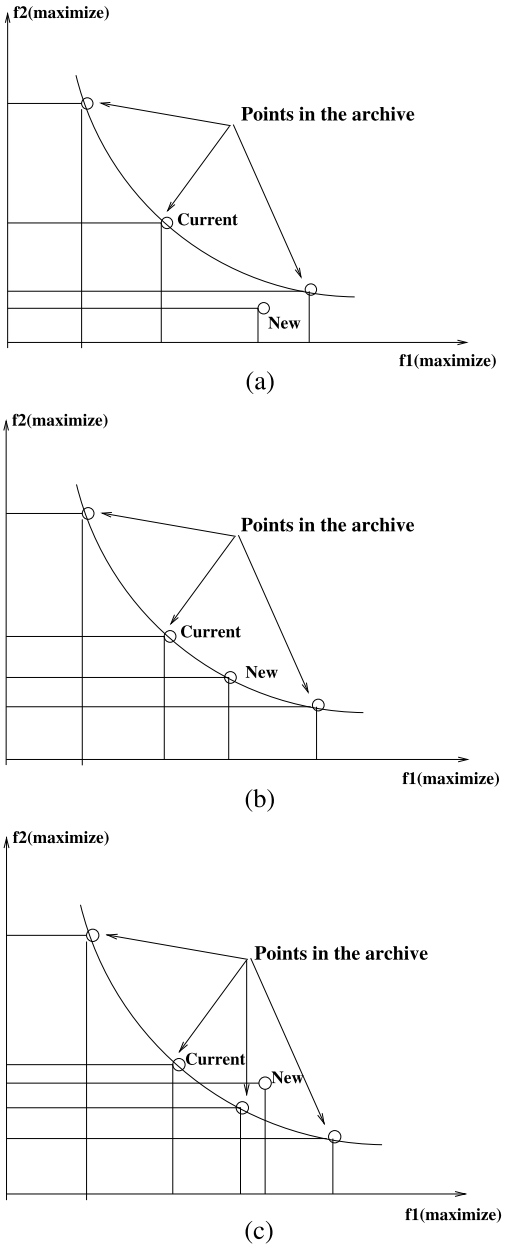

Fig. 2.8 Different cases when *New* and *Current* are nondominating: (**a**) Some solutions in *Archive* dominate *New*; (**b**) *New* is nondominating with respect to all the solutions of *Archive*; (**c**) *New* dominates k ($k \geq 1$) solutions in the *Archive*

- Case 3: *new-pt* dominates *current-pt*

  Now, based on the domination status of *new-pt* and members of *Archive*, the following three situations may arise:

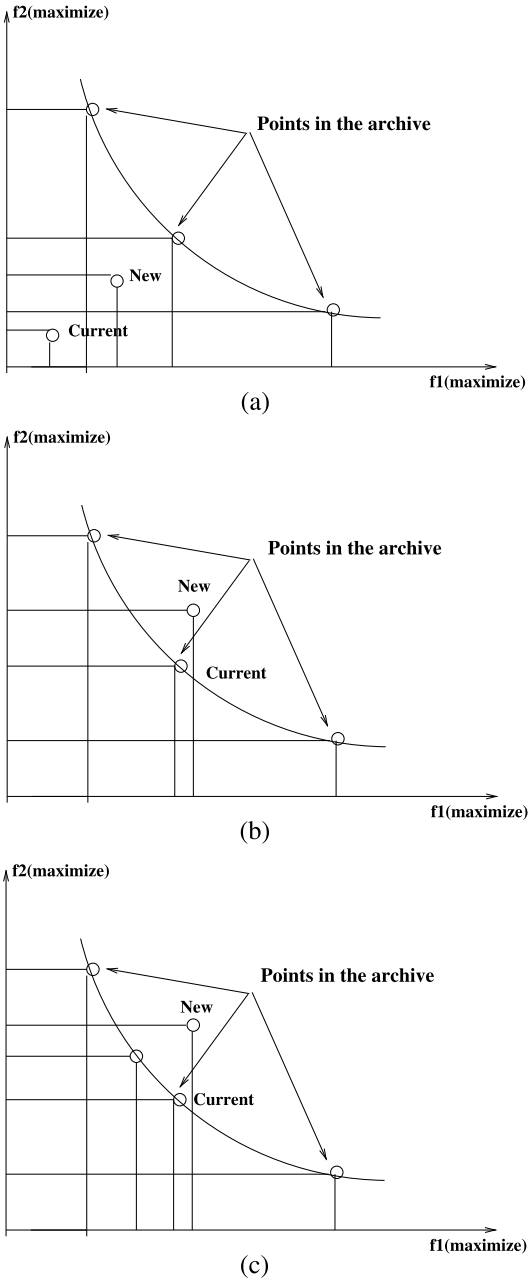

Fig. 2.9 Different cases when *New* dominates *Current*: (**a**) *New* is dominated by some solutions in *Archive*; (**b**) *New* is nondominating with respect to the solutions in the *Archive* except *Current*, if it is in the *Archive*; (**c**) *New* dominates some solutions of *Archive* other than *Current*

1. *new-pt* dominates *current-pt*, but k ($k \geq 1$) points in the *Archive* dominate this *new-pt*. Note that this situation [shown in Fig. 2.9(a)] may arise only if the *current-pt* is not a member of the *Archive*. Here, the minimum of

the difference of domination amounts between *new-pt* and the k points, denoted by Δdom_{min}, of the *Archive* is computed. The point from the *Archive* which corresponds to the minimum difference is selected as *current-pt* with $prob = \frac{1}{1+exp(-\Delta dom_{min})}$. Otherwise, *new-pt* is selected as *current-pt*. Note that, according to the SA paradigm, *new-pt* should have been selected with probability 1. However, due to the presence of *Archive*, it is evident that solutions still better than *new-pt* exist. Therefore, *new-pt* and the dominating points in the *Archive* that are closest to the *new-pt* (corresponding to Δdom_{min}) compete for acceptance. This may be considered a form of informed reseeding of the annealer only if the *Archive* point is accepted, but with a solution closest to the one which would otherwise have survived in the normal SA paradigm.

2. *new-pt* is nondominating with respect to the points in the *Archive* except *current-pt* if it belongs to the *Archive*. This situation is shown in Fig. 2.9(b). Thus, *new-pt*, which is now accepted as *current-pt*, can be considered as a new nondominated solution that must be stored in the *Archive*. Hence, *new-pt* is added to the *Archive*. If *current-pt* is in the *Archive*, then it is removed. Otherwise, if the number of points in the *Archive* becomes more than *SL*, clustering is performed to reduce the number of points to *HL*. Note that here *current-pt* may or may not be on the archival front.
3. *new-pt* also dominates k ($k \geq 1$) other points in the *Archive* [see Fig. 2.9(c)]. Hence, *new-pt* is selected as *current-pt* and added to the *Archive*, while all the dominated points of the *Archive* are removed. Note that here *current-pt* may or may not be on the archival front.

The above process is repeated *iter* times for each temperature (*temp*). The temperature is reduced to $\alpha \times temp$, using the cooling rate of α, till the minimum temperature, *Tmin*, is attained. The process thereafter stops, and the *Archive* contains the final nondominated solutions.

Note that in AMOSA, as in other versions of multiobjective SA algorithms, there is a possibility that a new solution worse than the current solution may be selected. In most other MOEAs, e.g., NSGA-II and PAES, if a choice needs to be made between two solutions $\overline{x}$ and $\overline{y}$, and if $\overline{x}$ dominates $\overline{y}$, then $\overline{x}$ is always selected. It may be noted that, in single-objective EAs or SA, usually a worse solution also has a nonzero chance of surviving in subsequent generations; this leads to a reduced possibility of getting stuck at suboptimal regions. However, most of the MOEAs have been so designed that this characteristic is lost. The present simulated annealing-based algorithm provides a way of incorporating this feature.

2.4.7 Complexity Analysis

The complexity analysis of AMOSA is provided in this section. The basic operations and their worst-case complexities are as follows:

1. Archive initialization: $O(SL)$.
2. Procedure to check the domination status of any two solutions: $O(M)$, $M =$ number of objectives.
3. Procedure to check the domination status between a particular solution and the *Archive* members: $O(M \times SL)$.
4. Single linkage clustering: $O(SL^2 \times \log(SL))$ [283].
5. Clustering procedure is executed

   - Once after initialization if $|ND| > HL$
   - After each $(SL - HL)$ number of iterations.
   - At the end if final $|Archive| > HL$

   So the maximum number of times the clustering procedure is called is $(TotalIter/(SL - HL)) + 2$.

   Therefore, the total complexity due to the clustering procedure is $O((TotalIter/(SL - HL)) \times SL^2 \times \log(SL))$.

The total complexity of AMOSA becomes

$$(SL + M + M \times SL) \times (TotalIter) + \frac{TotalIter}{SL - HL} \times SL^2 \times \log(SL). \tag{2.9}$$

Let $SL = \beta \times HL$, where $\beta \geq 2$, and $HL = N$, where N is the population size in NSGA-II and archive size in PAES. Therefore, overall complexity of AMOSA becomes

$$(TotalIter) \times \left(\beta \times N + M + M \times \beta \times N + \left(\beta^2/(\beta - 1)\right) \times N \times \log(\beta N)\right) \tag{2.10}$$

or

$$O\left(TotalIter \times N \times \left(M + \log(N)\right)\right). \tag{2.11}$$

Note that the total complexity of NSGA-II is $O(TotalIter \times M \times N^2)$ and that of PAES is $O(TotalIter \times M \times N)$. The NSGA-II complexity depends on the complexity of non-dominated procedure. With the best procedure, the complexity is $O(TotalIter \times M \times N \times \log(N))$.

2.5 Simulation Results

The experimental results of AMOSA as compared with several other methods are reported in detail in [29]. Some of these are described in this section. The comparison metrics used for the experiments are described first. The performance analysis of both the binary-coded AMOSA and the real-coded AMOSA are provided thereafter.

2.5.1 Comparison Measures

In multiobjective optimization, there are basically two functionalities that an MOO strategy must achieve regarding the obtained solution set [77]: It should converge as close to the true PO front as possible, and it should maintain as diverse a solution set as possible.

The first condition clearly ensures that the obtained solutions are near optimal, and the second condition ensures that a wide range of trade-off solutions is obtained. Clearly, these two tasks cannot be measured with one performance measure adequately. A number of performance measures have been suggested in the past. Here mainly three such performance measures are used. The first measure is the *Convergence* measure γ [78]. It measures the extent of convergence of the obtained solution set to a known set of PO solutions.The lower the value of γ, the better the convergence of the obtained solution set to the true PO front. The second measure, called *Purity* [25, 140], is used to compare the solutions obtained using different MOO strategies. It calculates the fraction of solutions from a particular method that remains nondominating when the final front solutions obtained from all the algorithms are combined. A value near to 1 (0) indicates better (poorer) performance. The third measure, named *Spacing*, was first proposed by Schott [250]. It reflects the uniformity of the solutions over the nondominated front. It is shown in [25] that this measure will fail to give an adequate result in some situations. In order to overcome the above limitations, a modified measure, named *Minimal Spacing*, is proposed in [25]. Smaller values of *Spacing* and *Minimal Spacing* indicate better performance.

It may be noted that, if an algorithm is able to approximate only a portion of the true PO front, not its full extent, none of the existing measures will be able to reflect this. In case of a discontinuous PO front, this problem becomes severe when an algorithm totally misses a subfront. Here, a performance measure which is very similar to the measure used in [68] and [141], named *displacement*, is used to overcome this limitation. It measures how far the obtained solution set is from a known set of PO solutions. In order to compute the *displacement* measure, a set P^* consisting of uniformly spaced solutions from the true PO front in the objective space is found (as is done while calculating γ). Then, *displacement* is calculated as

$$displacement = \frac{1}{|P^*|} \times \sum_{i=1}^{|P^*|} \Big(min_{j=1}^{|Q|} d(i, j) \Big), \tag{2.12}$$

where Q is the obtained set of final solutions, and $d(i, j)$ is the Euclidean distance between the ith solution of P^* and jth solution of Q. The lower the value of this measure, the better the convergence to and extent of coverage of the true PO front.

2.5.2 Comparison of Binary-Encoded AMOSA with NSGA-II and PAES

Firstly, the binary-encoded AMOSA is compared with the binary-coded NSGA-II and PAES algorithms. For AMOSA, binary mutation is used. Seven test problems have been considered for the comparison. These are SCH1 and SCH2 [77], Deb1 and Deb4 [76], ZDT1, ZDT2, and ZDT6 [77]. All the algorithms were executed ten times per problem, and the results reported are the average values obtained for the ten runs. In NSGA-II, the crossover probability (p_c) is kept equal to 0.9. For PAES, the depth value d is set equal to 5. For AMOSA, the cooling rate α is kept equal to 0.8. The number of bits assigned to encode each decision variable depends on the problem. For example, in ZDT1, ZDT2, and ZDT6, which all are 30-variable problems, 10 bits are used to encode each variable; for SCH1 and SCH2, which are single-variable problems, and for Deb1 and Deb4, which are two-variable problems, 20 bits are used to encode each decision variable. In all the approaches, binary mutation applied with a probability of $p_m = 1/l$, where l is the string length, is used as the perturbation operation. The values of *Tmax* (maximum value of the temperature), *Tmin* (minimum value of the temperature), and *iter* (number of iterations at each temperature) are chosen so that the total number of fitness evaluations of the three algorithms becomes approximately equal. For PAES and NSGA-II, identical parameter settings as suggested in the original studies have been used. Here the population size in NSGA-II, and archive sizes in AMOSA and PAES, are set to 100. The maximum number of iterations for NSGA-II and PAES are 500 and 50,000, respectively. For AMOSA, $Tmax = 200$, $Tmin = 10^{-7}$ and, $iter = 500$. The parameter values were determined after extensive sensitivity studies, which are omitted here to restrict the size of the chapter.

2.5.2.1 Discussion of the Results

The *Convergence* and *Purity* values obtained using the three algorithms are presented in Table 2.1. AMOSA performs best for ZDT1, ZDT2, ZDT6, and Deb1 in terms of γ. For SCH1, all three perform equally well. NSGA-II performs well for SCH2 and Deb4. Interestingly, for all the functions, AMOSA is found to provide a greater number of overall nondominated solutions than NSGA-II. (This is evident from the quantities in parentheses in Table 2.1, where x/y indicates that on average the algorithm produced y solutions of which x remained good even when solutions from other MOO strategies are combined.) AMOSA took 10 seconds to provide the first PO solution, compared with 32 seconds for NSGA-II in case of ZDT1. From Table 2.1 it is also clear that AMOSA and PAES provide a greater number of distinct solutions than NSGA-II.

Table 2.2 shows the *Spacing* and *Minimal Spacing* measurements. AMOSA provides the best values of *Spacing* in most cases, while PAES performs the worst. This

Table 2.1 *Convergence* and *Purity* measures on the test functions for binary encoding

Test problem	Convergence			Purity		
	AMOSA	PAES	NSGA-II	AMOSA	PAES	NSGA-II
SCH1	0.0016	0.0016	0.0016	0.9950 (99.5/100)	0.9850 (98.5/100)	1 (94/94)
SCH2	0.0031	0.0015	0.0025	0.9950 (99.5/100)	0.9670 (96.7/100)	0.9974 (97/97.3)
ZDT1	0.0019	0.0025	0.0046	0.8350 (83.5/100)	0.6535 (65.4/100)	0.970 (68.64/70.6)
ZDT2	0.0028	0.0048	0.0390	0.8845 (88.5/100)	0.4050 (38.5/94.9)	0.7421 (56.4/76)
ZDT6	0.0026	0.0053	0.0036	1 (100/100)	0.9949 (98.8/99.3)	0.9880 (66.5/67.3)
Deb1	0.0046	0.0539	0.0432	0.91 (91/100)	0.718 (71.8/100)	0.77 (71/92)
Deb4	0.0026	0.0025	0.0022	0.98 (98/100)	0.9522 (95.2/100)	0.985 (88.7/90)

Table 2.2 *Spacing* and *Minimal spacing* measures on the test functions for binary encoding

Test problem	Spacing			Minimal spacing		
	AMOSA	PAES	NSGA-II	AMOSA	PAES	NSGA-II
SCH1	0.0167	0.0519	0.0235	0.0078	0.0530	0.0125
SCH2	0.0239	0.5289	0.0495	*N.A.*	*N.A.*	*N.A.*
ZDT1	0.0097	0.0264	0.0084	0.0156	0.0265	0.0147
ZDT2	0.0083	0.0205	0.0079	0.0151	0.0370	0.0130
ZDT6	0.0051	0.0399	0.0089	0.0130	0.0340	0.0162
Deb1	0.0166	0.0848	0.0475	0.0159	0.0424	0.0116
Deb4	0.0053	0.0253	0.0089	*N.A.*	*N.A.*	*N.A.*

Table 2.3 New measure *displacement* on the test functions for binary encoding

Algorithm	SCH2	Deb4	ZDT1	ZDT2	ZDT6
AMOSA	0.0230	0.0047	0.0057	0.0058	0.0029
PAES	0.6660	0.0153	0.0082	0.0176	0.0048
NSGA-II	0.0240	0.0050	0.0157	0.0096	0.0046

is also evident from Figs. 2.10 and 2.11, which show the final PO fronts of SCH2 and Deb4 obtained by the three methods for the purpose of illustration. With respect to *Minimal Spacing* the performance of AMOSA and NSGA-II is comparable.

Table 2.3 presents the value of *displacement* for five problems, two with discontinuous and three with continuous PO fronts. AMOSA performs the best in almost

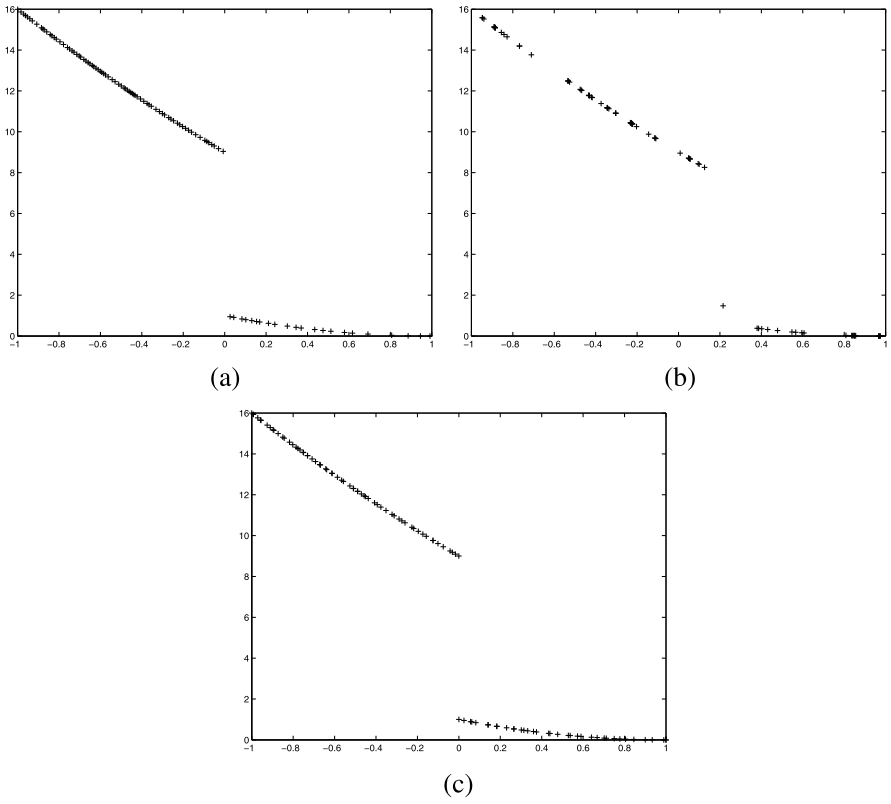

Fig. 2.10 The final nondominated front for SCH2 obtained by: (**a**) AMOSA, (**b**) PAES, and (**c**) NSGA-II

all the cases. The utility of the new measure is evident in particular for Deb4, where PAES performs quite poorly (see Fig. 2.11). Interestingly, the *Convergence* value for PAES (Table 2.1) is very good here, though the *displacement* correctly reflects that the PO front has been represented very poorly.

Table 2.4 presents the time taken by the algorithms for the different test functions. It is seen that PAES takes less time in six of the seven problems because of its lower complexity. AMOSA takes less time than NSGA-II in 30-variable problems such as ZDT1 and ZDT2, and the 10-variable problem ZDT6. But for the single- and two-variable problems SCH1, SCH2, Deb1 and Deb4, AMOSA takes more time than NSGA-II. This may be due to the complexity of its clustering procedure. Generally, for single- or two-variable problems, this procedure dominates the crossover and ranking procedures of NSGA-II. But for 30-variable problems the scenario is reversed. This is because of the increased complexity of ranking and crossover (due to increased string length) in NSGA-II.

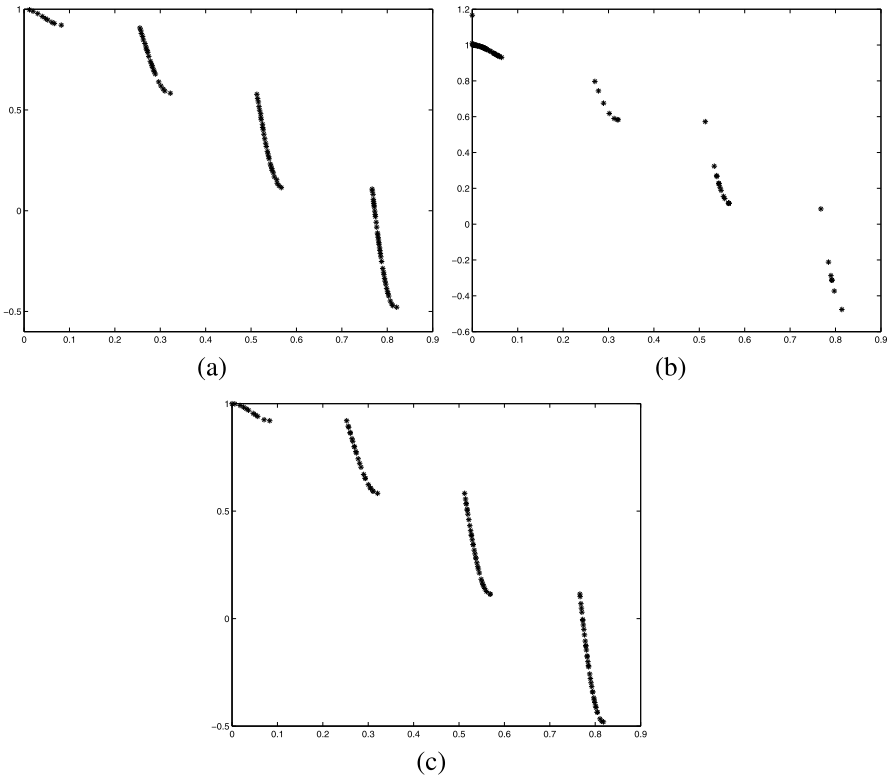

Fig. 2.11 The final nondominated front for Deb4 obtained by: (**a**) AMOSA, (**b**) PAES, and (**c**) NSGA-II

2.5.3 Comparison of Real-Coded AMOSA with the Algorithm of Smith et al. and Real-Coded NSGA-II

In the real-coded version of AMOSA [29] (source code available at: www.isical.ac.in/~sriparna_r), mutation is done as suggested in [257]. Here, a new string is generated from the old string $\overline{x}$ by perturbing only one parameter or decision variable of $\overline{x}$. The parameter to be perturbed is chosen at random and perturbed with a random variable ε drawn from a Laplacian distribution, $p(\varepsilon) \propto e^{-\frac{|\varepsilon-\mu|}{\delta}}$, where the scaling factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. A fixed scaling factor equal to 0.1 is used for mutation. The initial temperature is selected by the procedure mentioned in [257]; that is, the initial temperature, *Tmax*, is calculated by using a short 'burn-in' period during which all solutions are accepted and setting the temperature equal to the average positive change of energy divided by ln(2). Here, *Tmin* is kept equal to 10^{-5} and the temperature is adjusted according to $T_k = \alpha^k Tmax$, where α is set equal to 0.8. For NSGA-II, population size is kept equal to 100 and the total number of generations is set such

Table 2.4 Time taken by different programs (in seconds) for binary encoding

Algorithm	SCH1	SCH2	Deb1	Deb4	ZDT1	ZDT2	ZDT6
AMOSA	15	14.5	20	20	58	56	12
PAES	6	5	5	15	17	18	16
NSGA-II	11	11	14	14	77	60	21

that the total number of function evaluations of AMOSA and NSGA-II are the same. For AMOSA the archive size is set equal to 100. (However, in a part of the investigations, the archive size is kept unlimited as in [257]. The results are compared with those obtained by MOSA [257] and provided in [3].) AMOSA is executed for a total of 5,000, 1,000, 15,000, 5,000, 1,000, 5,000 and 9,000 run lengths, respectively, for DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ5, and DTLZ6. The total number of iterations, $iter$, per temperature is set accordingly. Real-coded NSGA-II (code obtained from KANGAL site: http://www.iitk.ac.in/kangal/codes.html) is executed. For NSGA-II the following parameter setting is used: probability of crossover = 0.99, probability of mutation = $(1/l)$, where l is the string length, distribution index for the crossover operation = 10, distribution index for the mutation operation = 100.

In MOSA [257], authors have used unconstrained archive size. Note that the archive size of AMOSA and the population size of NSGA-II are both 100. For the purpose of comparison with MOSA, which has an unlimited archive [257], the clustering procedure (adopted for AMOSA) is used to reduce the number of solutions of [3] to 100. Comparison is performed in terms of *Purity*, *Convergence*, and *Minimal Spacing*. Table 2.5 presents the *Purity*, *Convergence*, and *Minimal Spacing* measurements for problems DTLZ1-DTLZ6 obtained after application of AMOSA, MOSA, and NSGA-II. It can be seen from this table that AMOSA performs the best in terms of *Purity* and *Convergence* for DTLZ1, DTLZ3, DTLZ5, and DTLZ6. In DTLZ2 and DTLZ4, the performance of MOSA is marginally better than that of AMOSA. NSGA-II performs the worst of all. Table 2.5 presents the *Minimal Spacing* values obtained by the three algorithms for DTLZ1-DTLZ6. AMOSA performs the best in all the cases.

As mentioned earlier, for comparison with the performance of MOSA (by considering the results reported in [3]), a version of AMOSA without clustering and with unconstrained archive is executed. The results reported here are averages over 10 runs. Table 2.6 presents the corresponding *Purity*, *Convergence*, and *Minimal Spacing* values. Again AMOSA performs much better than MOSA for all the test problems, except DTLZ4. For DTLZ4, MOSA performs better than AMOSA in terms of both *Purity* and *Convergence* values. Figure 2.12 shows the final Pareto-optimal front obtained by AMOSA and MOSA for DTLZ1-DTLZ3, while Fig. 2.13 shows the same for DTLZ5 and DTLZ6. As can be seen from the figures, AMOSA appears to be able to better approximate the front with more dense-solutions as compared with MOSA.

It was mentioned in [307] that, for a particular test problem, almost 40 % of the solutions provided by an algorithm with truncation of the archive are domi-

Table 2.5 *Convergence*, *Purity*, and *Minimal Spacing* measures on the three-objective test functions while *Archive* is bounded to 100

Test problem	Convergence			Purity			Minimal spacing		
	AMOSA	MOSA	NSGA-II	AMOSA	MOSA	NSGA-II	AMOSA	MOSA	NSGA-II
DTLZ1	0.01235	0.159	13.695	0.857 (85.7/100)	0.56 (28.35/75)	0.378 (55.7/100)	0.0107	0.1529	0.2119
DTLZ2	0.014	0.01215	0.165	0.937 (93.37/100)	0.9637 (96.37/100)	0.23 (23.25/100)	0.0969	0.1069	0.1236
DTLZ3	0.0167	0.71	20.19	0.98 (93/95)	0.84 (84.99/100)	0.232 (23.2/70.6)	0.1015	0.152	0.14084
DTLZ4	0.28	0.21	0.45	0.833 (60/72)	0.97 (97/100)	0.7 (70/100)	0.20	0.242	0.318
DTLZ5	0.00044	0.0044	0.1036	1 (97/97)	0.638 (53.37/83.6)	0.05 (5/100)	0.0159	0.0579	0.128
DTLZ6	0.043	0.3722	0.329	0.9212 (92.12/100)	0.7175 (71.75/100)	0.505 (50.5/100)	0.1148	0.127	0.1266

Table 2.6 *Convergence*, *Purity*, and *Minimal Spacing* measures on the three-objective test functions by AMOSA and MOSA while *Archive* is unbounded

Test problem	Convergence		Purity		Minimal spacing	
	AMOSA	MOSA	AMOSA	MOSA	AMOSA	MOSA
DTLZ1	0.010	0.1275	0.99 (1253.87/1262.62)	0.189 (54.87/289.62)	0.064	0.083.84
DTLZ2	0.0073	0.0089	0.96 (1074.8/1116.3)	0.94 (225/239.2)	0.07598	0.09595
DTLZ3	0.013	0.025	0.858 (1212/1412.8)	0.81 (1719/2003.9)	0.0399	0.05
DTLZ4	0.032	0.024	0.8845 (88.5/100)	0.4050 (38.5/94.9)	0.1536	0.089
DTLZ5	0.0025	0.0047	0.92 (298/323.66)	0.684 (58.5/85.5)	0.018	0.05826
DTLZ6	0.0403	0.208	0.9979 (738.25/739.75)	0.287 (55.75/194.25)	0.0465	0.0111

nated by solutions provided by an algorithm without archive truncation. However, the experiments conducted in [29] did not adequately justify this finding. Let the set of solutions of AMOSA with and without clustering be denoted by S_c and S_{wc}, respectively. For DTLZ1 it was found that 12.6 % of S_c were dominated by S_{wc}, while 4 % of S_{wc} were dominated by S_c. For DTLZ2, 5.1 % of S_{wc} were dominated by S_c while 5.4 % of S_c were dominated by S_{wc}. For DTLZ3, 22.38 % of S_{wc} were dominated by S_c while 0.53 % of S_c were dominated by S_{wc}. For DTLZ4, all the members of S_{wc} and S_c were the same since AMOSA without clustering did not provide more than 100 solutions. For DTLZ5, 10.4 % of S_{wc} were dominated by S_c while 0.5 % of S_c were dominated by S_{wc}. For DTLZ6, all the members of S_{wc} and S_c were nondominating with respect to each other.

To investigate the performance on a four-objective problem, AMOSA and NSGA-II were applied to the 13-variable DTLZ2 test problem [29]. This is referred to as DTLZ2_4. The problem has a spherical Pareto front in four dimensions given by the equation: $f_1^2 + f_2^2 + f_3^2 + f_4^2 = 1$ with $f_i \in [0, 1]$ for $i = 1$ to 4. Both algorithms were applied for a total of 30,000 function evaluations (for NSGA-II popsize 100 and number of generations 300) and the *Purity*, *Convergence*, and *Minimal Spacing* values are presented in Table 2.7. AMOSA performs much better than NSGA-II in terms of all three measures.

AMOSA and NSGA-II were also compared for DTLZ1_5 (9-variable 5-objective version of the test problem DTLZ1), DTLZ1_10 (14-variable 10-objective version of DTLZ1), and DTLZ1_15 (19-variable 15-objective version of DTLZ1). The three problems have a spherical Pareto front in their respective dimensions given by the equation $\sum_{i=1}^{M} f_i = 0.5$, where M is the total number of objective functions. Both algorithms were executed for a total of 100,000 function evaluations for these three

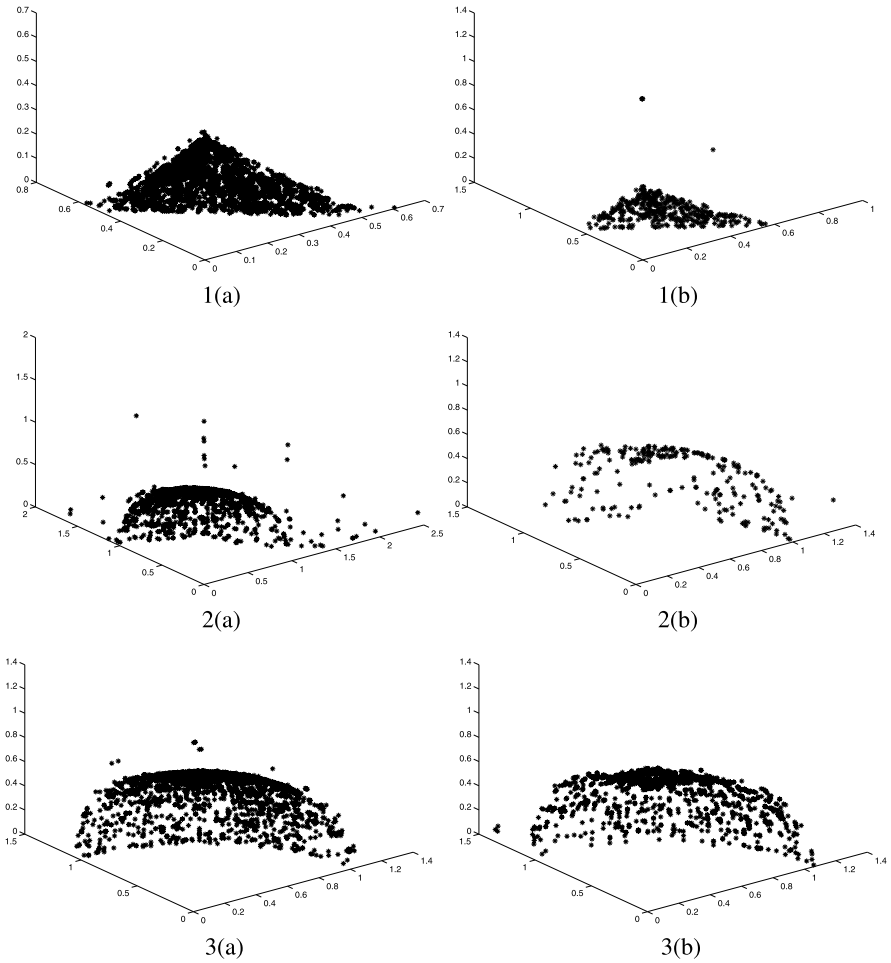

Fig. 2.12 The final nondominated front obtained by (**a**) AMOSA and (**b**) MOSA for the test problems (**1**) DTLZ1, (**2**) DTLZ2, and (**3**) DTLZ3

test problems (for NSGA-II popsize 200, number of generations 500), and the corresponding *Purity*, *Convergence*, and *Minimal Spacing* values are presented in Table 2.7. The *Convergence* values indicate that NSGA-II does not converge to the true PO front where as AMOSA reaches the true PO front for all three cases. The *Purity* measure also indicates this. The results on many-objective optimization problems show that AMOSA performs much better than NSGA-II. These results support the fact that Pareto ranking-based MOEAs such as NSGA-II do not work well on many-objective optimization problems, as pointed out in some recent studies [137, 139].

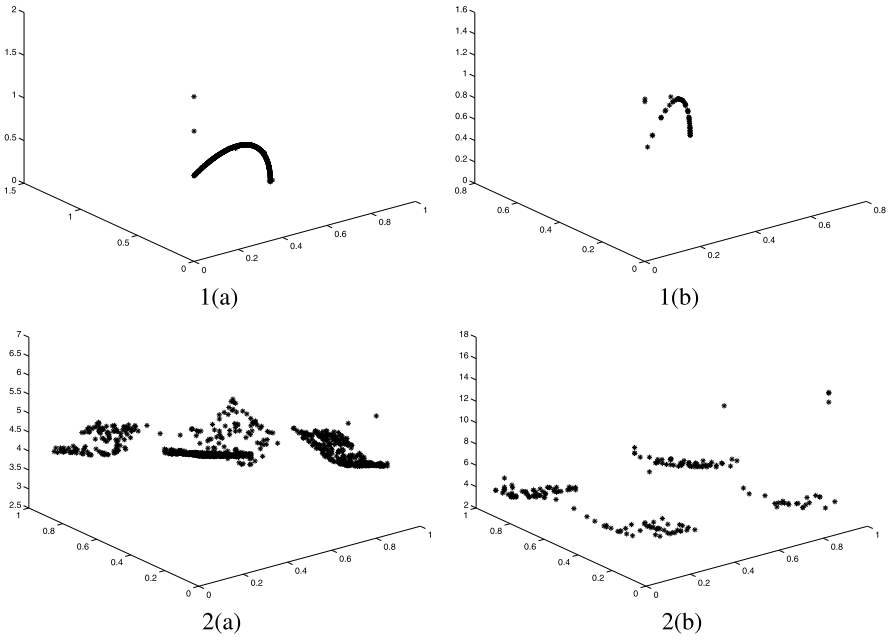

Fig. 2.13 The final nondominated front obtained by (**a**) AMOSA and (**b**) MOSA for the test problems (**1**) DTLZ5 and (**2**) DTLZ6

Table 2.7 *Convergence*, *Purity*, and *Minimal Spacing* measures on the DTLZ2_4, DTLZ1_5, DTLZ1_10, and DTLZ1_15 test functions by AMOSA and NSGA-II

Test problem	Convergence		Purity		Minimal spacing	
	AMOSA	NSGA-II	AMOSA	NSGA-II	AMOSA	NSGA-II
DTLZ2_4	0.2982	0.4563	0.9875 (98.75/100)	0.903 (90.3/100)	0.1876	0.22
DTLZ1_5	0.0234	306.917	1	0	0.1078	0.165
DTLZ1_10	0.0779	355.957	1	0	0.1056	0.2616
DTLZ1_15	0.193	357.77	1	0	0.1	0.271

2.5.4 Discussion on Annealing Schedule

The annealing schedule of an SA algorithm consists of (i) the initial value of temperature (*Tmax*), (ii) the cooling schedule, (iii) the number of iterations to be performed at each temperature, and (iv) the stopping criterion to terminate the algorithm. The initial value of the temperature should be so chosen that it allows the SA to perform a random walk over the landscape. Some methods to select the initial temperature

are given in detail in [274]. In AMOSA [29], as in [257], the initial temperature is set to achieve an initial acceptance rate of approximately 50 % on derogatory proposals. This is described in Sect. 2.5.3.

The functional form of the change in temperature required in SA is determined by the cooling schedule. The most frequently used decrement rule, also used in this chapter, is the geometric schedule given by $T_{k+1} = \alpha \times T_k$, where α $(0 < \alpha < 1)$ denotes the cooling factor. Typically the value of α is chosen in the range between 0.5 and 0.99. This cooling schedule has the advantage of being very simple. Some other cooling schedules available in the literature are logarithmic, Cauchy, and exponential. More details about these schedules are available in [274]. The cooling schedule should be so chosen that it is able to strike a good balance between exploration and exploitation of the search space.

The third component of an annealing schedule is the number of iterations performed at each temperature. It should be so chosen that the system is sufficiently close to the stationary distribution at that temperature. As suggested in [274], the value of the number of iterations should be chosen depending on the nature of the problem. Several criteria for termination of an SA process have been developed. In some of them, the total number of iterations that the SA procedure must execute is given, whereas in some others, the minimum value of the temperature is specified. Detailed discussion on this issue can be found in [274].

2.6 Discussion and Conclusions

Some existing single- and multiobjective optimization techniques are described in this chapter. This includes details of a newly developed simulated annealing-based multiobjective optimization technique, AMOSA [29]. In contrast to most other MOO algorithms, AMOSA selects dominated solutions with a probability that is dependent on the amount of domination, measured in terms of the hypervolume between the two solutions in the objective space. The results of binary-coded AMOSA are compared with those of two existing well-known multiobjective optimization algorithms, NSGA-II (binary-coded) [78] and PAES [161], for a suite of seven two-objective test problems having different complexity levels. In a part of the investigation, comparison of the real-coded version of AMOSA is conducted against a very recent multiobjective simulated annealing algorithm, MOSA [257], and real-coded NSGA-II for six three-objective test problems. Real-coded AMOSA is also compared with real-coded NSGA-II for some 4-, 5-, 10- and 15-objective test problems. Several different comparison measures such as *Convergence*, *Purity*, *Minimal Spacing*, and *Spacing*, and the time taken are used for the purpose of comparison. In this regard, a measure called *displacement* has also been used, which is able to reflect whether a front is close to the PO front as well as its extent of coverage. A complexity analysis of AMOSA is performed. It is found that its complexity is greater than that of PAES but less than that of NSGA-II.

It is seen from the given results that the performance of AMOSA is better than that of MOSA and NSGA-II in a majority of the cases, while PAES performs poorly

in general. AMOSA is found to provide more distinct solutions than NSGA-II in each run for all the problems; this is a desirable feature in MOO. AMOSA is less time consuming than NSGA-II for complex problems such as ZDT1, ZDT2, and ZDT6. Moreover, for problems with many objectives, the performance of AMOSA is found to be much better than that of NSGA-II. This is an interesting and appealing feature of AMOSA, since Pareto ranking-based MOEAs, such as NSGA-II [78], do not work well on many-objective optimization problems, as pointed out in some recent studies [137, 139]. An interesting feature of AMOSA, as in other versions of multiobjective SA algorithms, is that it has a nonzero probability of allowing a dominated solution to be chosen as the current solution in favor of a dominating solution. This makes the problem less greedy in nature, thereby leading to better performance for complex and/or deceptive problems. Note that it may be possible to incorporate this feature as well as the concept of amount of domination into other MOO algorithms in order to improve the performance.

In order to compute similarities/dissimilarities between different objects, several similarity/distance measurements are developed in the literature. These similarity measurements have wide application in data clustering. Depending on the similarity measurements, points can be grouped into different clusters. In the next chapter we discuss different similarity/distance measurements reported in the literature. These definitions of distance measures suitable for handling binary, categorical, ordinal, and quantitative variables are described in detail.

http://www.springer.com/978-3-642-32450-5

Unsupervised Classification
Similarity Measures, Classical and Metaheuristic Approaches, and Applications
Bandyopadhyay, S.; Saha, S.
2013, XVIII, 262 p., Hardcover
ISBN: 978-3-642-32450-5