Journal of the Operational Research Society (2006) 57, 1143–1160

www.palgrave-journals.com/jors

A survey of simulated annealing as a tool for single and multiobjective optimization

B Suman[1]* and P Kumar[2]

[1]*University of Minnesota, Minneapolis, MN, USA; and* [2]*North Carolina State University, Raleigh, NC, USA*

This paper presents a comprehensive review of simulated annealing (SA)-based optimization algorithms. SA-based algorithms solve single and multiobjective optimization problems, where a desired global minimum/maximum is hidden among many local minima/maxima. Three single objective optimization algorithms (SA, SA with tabu search and CSA) and five multiobjective optimization algorithms (SMOSA, UMOSA, PSA, WDMOSA and PDMOSA) based on SA have been presented. The algorithms are briefly discussed and are compared. The key step of SA is probability calculation, which involves building the annealing schedule. Annealing schedule is discussed briefly. Computational results and suggestions to improve the performance of SA-based multiobjective algorithms are presented. Finally, future research in the area of SA is suggested.

Journal of the Operational Research Society (2006) **57,** 1143–1160. doi:10.1057/palgrave.jors.2602068
Published online 19 October 2005

Keywords: simulated annealing; metaheuristic; multiobjective optimization; annealing schedule

Introduction

Simulated annealing (SA) is a compact and robust technique, which provides excellent solutions to single and multiple objective optimization problems with a substantial reduction in computation time. It is a method to obtain an optimal solution of a single objective optimization problem and to obtain a Pareto set of solutions for a multiobjective optimization problem. It is based on an analogy of thermodynamics with the way metals cool and anneal. If a liquid metal is cooled slowly, its atoms form a pure crystal corresponding to the state of minimum energy for the metal. The metal reaches a state with higher energy if it is cooled quickly. SA has received significant attention in the last two decades to solve optimization problems, where a desired global minimum/maximum is hidden among many poorer local minima/maxima. Kirkpatrick *et al* (1983) and Černy (1985) showed that a model for simulating the annealing of solids, proposed by Metropolis *et al* (1953), could be used for optimization of problems, where the objective function to be minimized corresponds to the energy of states of the metal. These days SA has become one of the many heuristic approaches designed to give a good, not necessarily optimal solution. It is simple to formulate and it can handle mixed discrete and continuous problem with ease. It is also efficient and has low memory requirement. SA takes less CPU time than genetic algorithm (GA) when used to solve optimization problems, because it finds the optimal solution using point-by-point iteration rather than a search over a population of individuals.

Initially, SA has been used with combinatorial optimization problems. Many combinatorial problems belong to a class known as NP-hard problems, which means that the computation time, giving an optimal solution, increases with N as exp(constant $\times N$). Maffioli (1987) showed that SA can be considered as one type of randomized heuristic approaches for combinatorial optimization problems. The well-known travelling salesman problem belongs to this class. The salesman visits N cities (with given positions) only once and returns to his city of origin. The objective is to make the route as short as possible. Afterwards, SA has been extended to the single and multiobjective optimization problems with continuous N-dimensional control spaces. A summary of these approaches is given by Van Laarhoven and Aarts (1987).

Glover and Greenberg (1989) summarized the approaches offered by GA, neural networks, tabu search, targeted analysis and SA. Surveys of the literature on different evolutionary and metaheuristic-based methods and their applications are compiled by Coello Coello (1996, 1999) and van Veldhuizen and Lamont (1998a, 1998b). Despite the considerable volume of research in single and multiobjective algorithms based on SA in the last two decades, no survey has been published in the literature that includes the multiobjective framework. Surveys on single objective SA have been performed (Collins *et al*, 1988; Rutenbar, 1989; Eglese, 1990) but few multiobjective algorithms to improve the performance of the SA have been proposed in the recent

**Correspondence: B Suman, Department of Chemical Engineering and Materials Science, Mailbox # 30, 151 Amundson Hall, 421 Washington Avenue SE, University of Minnesota, Minneapolis, MN 55455, USA.*
E-mail: suman@cems.umn.edu

years. This paper presents a survey on single and multiobjective algorithms based on SA to give an updated detailed knowledge to the researchers in the area of SA. The following section presents the application of SA in different optimization problems. In the section on SA for optimization, some single as well as multiobjective algorithms are discussed briefly. Then, the multiobjective algorithms are compared. Annealing schedule is an essential part of SA since it determines the performance of SA algorithm, it is discussed in the succeeding section. Section on computational results and performance environment, presents computational results related to SA along with suggestions to further improve the performance of the SA algorithm. Finally, the scope for future research is presented in the succeeding section. The last section presents importance of this work and future scope briefly. Acronyms and notations are listed in the Appendix.

Applications of SA

SA was started as a method or tool for solving single objective combinatorial problems, these days it has been applied to solve single as well as multiple objective optimization problems in various fields. The problems may have continuous or discrete variables. SA has been used in the area of integrated circuits (IC) layout. The problem in IC layout is placing components on the surface of an IC so as to optimize subsequent wirability. The IC itself is modelled as a grid, where each grid point can hold one module. Each set of module terminals to be wired together forms a net whose wirability is to be optimized. The chosen cost function is the estimated wire length. SA terminates when the cost improvement across three temperatures is very small (Rutenbar, 1989). It has been applied to the combined problems of the synthesis of structures/controls and the actuator-location problem for the design of intelligent structures. Multiple and conflicting design objectives such as vibration reduction, dissipated energy, power and a performance index are included by utilizing an efficient multiobjective optimization formulation. Piezoelectric materials are used as actuators in the control system. It can be used for optimization and an approximation technique is used to reduce computational effort (Chattopadhyay and Seeley, 1994). Lucic and Teodorovic (1999) have used it in airlines. Gradient methods are ineffective in dealing with many signal processing applications involving optimization problems with multimodal and non smooth cost functions. The adaptive simulated annealing (ASA) offers a viable optimization tool for tackling these difficult nonlinear optimization problems (Chen and Luk, 1999). Optimization of batch distillation processes, widely used in chemical industry can be solved using SA. Hanke and Li (2000) have showed the potential of SA for developing optimal operation strategies for batch chemical processes. Recently, it has been applied in antenna array synthesis (Girard *et al*, 2001), multimedia data placement (Terzi *et al*, 2004) molecular physics. Suman (2002, 2004a, 2005) has used SA based multiobjective algorithms to optimize the profit and its sensitivity of a refinery model problem. Suman (2003) has applied five SA-based multiobjective algorithms—SMOSA, UMOSA, PSA, PDMOSA and WMOSA to find a Pareto set of solutions of a system reliability multiobjective optimization problem in a short time. In each of these algorithms, the solution vector explores its neighbourhood in a way similar to that of Classical SA. Kumral (2003) has applied chance-constrained programming based on multiobjective SA to optimize blending of different available ores in a way that expected value and standard deviation of the cost of buying ores is minimized while satisfying the quality specifications.

SA has been greatly used in operational research problems. Chen *et al* (1988) reported a new approach to setup planning of prismatic parts using Hopfield neural net coupled with SA. Sridhar and Rajendran (1993) have described three perturbation schemes to generate new sequences for solving the scheduling problem in cellular manufacturing system. Suresh and Sahu (1994) have used SA for assembly line balancing. They only considered single objective problems. They have found that SA performed at least as well as the other approaches. Meller and Bozer (1996) have applied SA to facility layout problems with single and multiple floors. The facility layout problem is highly combinatorial in nature and generally exhibits many local minima. SA achieves low-cost solutions that are much less dependent on the initial layout than other approaches. Mukhopadhyay *et al* (1998) have used SA to solve the problem of Flexible Manufacturing system (FMS) machine loading with the objective of minimizing the system imbalance. Kim *et al* (2002) have considered a multi-period multi-stop transportation planning problem in a one-warehouse multi-retailer distribution system to determine the routes of vehicles and delivery quantities for each retailer. They have suggested a two-stage heuristic algorithm based on SA as an alternative for large problems that cannot be solved by the column generation algorithm in a reasonable computation time to minimize the total transportation distance for product delivery over the planning horizon while satisfying demands of the retailers. Golenko-Ginzburg and Sims (1992) have defined a priority list to be any permutation of a set of symbols, where the symbol for each job appears the same number of times as its operations. Every priority list can be associated in a natural way with a feasible schedule and every feasible schedule arises in the same way. Therefore, priority lists are a representation of feasible schedules that avoid the problems normally associated with schedule infeasibility. Shutler (2003) has presented priority list based Monte Carlo implementation of SA, which was competitive with the current leading schedule based SA and tabu search heuristics. New job sequences have been

generated with a proposed perturbation scheme named the 'modified insertion scheme' (MIS), which have been used in the proposed SA algorithm to arrive at a near global optimum solution. The SA algorithm using the proposed MIS gave substantial improvement in system imbalance. Its other applications have been presented by machine loading problem of FMS (Swarnkar and Tiwari, 2004), part classification (Tiwari and Roy, 2003), resource constrained project scheduling (Cho and Kim, 1997), etc. McCormick and Powell (2004) described a two stage SA algorithm to derive pump schedules for water distribution in a time short enough for routine operational use. They have built the model based on automatic interaction with a hydraulic simulator, which deals with nonlinear effects from reservoir-level variations.

Application of SA does not restrict to optimization of nonlinear objective function, these days it has been applied for many other purposes. Bell *et al* (1987) have used it to cluster tuples in databases. They have attempted to use SA in circuit board layout design and it suggests that it would be advantageously applied to clustering tuples in database in order to enhance responsiveness to queries. Recently, it has not only been applied for optimization but also for recognition of patterns and object classification (Liu and Huang, 1998; Yip and Pao, 1995; Starink and Backer, 1995). Liu and Huang (1998) have proposed hybrid pattern recognition based on the evolutionary algorithms with fast SA that can recognize patterns deformed by transformation caused by rotation, scaling or translation, singly or in combination. Object recognition problem as matching of a global model graph with an input scene graph representing either a single object or several overlapping objects has been formulated. Chu *et al* (1996) have used SA to analyse the network of interacting genes that control embryonic development and other biological processes. Suman (2004b) has proposed a new method for online optimization of multiobjective optimization algorithm parameters. The parameters have been optimized using SA-based algorithm. Generalizational distance and measure C as objective functions have been considered. They have been optimized to obtain the optimum values of algorithmic parameters. The developed technique can be implemented with any multiobjective optimization algorithm.

SA for optimization

SA for single objective optimization

Suppose that the solution space, S, is a finite set of all solutions and the objective function, f, is a real valued function defined for the members of S. The minimization problem can be formulated to find a solution or state, $i \in S$, which minimizes f over S.

A simple form of local search, say a descent method, starts with an initial solution. In the neighbourhood of this solution a new solution is generated using suitable algorithms and the objective function is calculated. If reduction in the objective function is observed, the current solution is updated. Otherwise, the current solution is retained and the process is repeated until no further reduction in the objective function is obtained. Thus, the search terminates with a local minimum, which may or may not be the true global minimum. Owing to this disadvantage, we do not rely on this algorithm though this is simple and easy to execute. In SA, instead of this strategy, the algorithm attempts to avoid being trapped in a local minimum by sometimes accepting even the worse move. The acceptance and rejection of the worse move is controlled by a probability function. The probability of accepting a move, which causes an increase δ in f, is called the acceptance function. It is normally set to $\exp(-\delta/T)$, where T is a control parameter, which corresponds to the temperature in analogy with the physical annealing. This acceptance function implies that the small increase in f is more likely to be accepted than a large increase in f. When T is high most uphill moves are accepted, but as T approaches to zero, most uphill moves will be rejected. Therefore, SA starts with a high temperature to avoid being trapped in local minimum. The algorithm proceeds by attempting a certain number of moves at each temperature and decreasing the temperature. Thus, the configuration decisions in SA proceed in a logical order. The SA based algorithm for single objective optimization is illustrated in Table 1.

Like other heuristic optimization techniques, there is a chance of revisiting a solution multiple times in SA as well. It leads to extra computational time without any improvement in the optimal solution. Recently, researchers Zolfaghari and Liang (1999), Swarnkar and Tiwari (2004) and Jeon and Kim (2004) have attempted to use tabu search with SA to

Table 1 SA for single objective

1. Initialize the temperature.
2. Start with a randomly generated initial solution vector, X, and generate the objective function.
3. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of current solution vector, X, revaluate the objective function and apply penalty function approach to the objective function, if necessary.
4. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$. Update the existing optimal solution and go to Step 6.
5. Else accept Y with the probability:

$$P = \exp(-\Delta s/T) \tag{1}$$

   where $\Delta s = Z(Y) - Z(X)$.
   If the solution is accepted, replace X with Y.
6. Decrease the temperature periodically.
7. Repeat Steps 2–6 until stopping criterion is met.

avoid revisiting in single objective optimization. A pseudo-code for SA with tabu search for single objective optimization is illustrated in Table 2.

In SA, there is a slow convergence rate for some optimization problems. Mingjun and Huanwen (2004) have proposed chaos simulated annealing (CSA) by introducing chaotic systems to SA. The CSA is different from SA as chaotic initialization and chaotic sequences replace the Gaussian distribution. CSA is more likely to converge to the global optimum solution because it is random, stochastic and sensitive on the initial conditions. It has been shown that CSA improves the convergence and is efficient, applicable and easy to implement. The CSA-based algorithm for single objective optimization is illustrated in Table 3.

Table 2 SA with tabu search for single objective

1. Initialize the temperature and tabulist.
2. Start with a randomly generated initial solution vector, X, and generate the objective function.
3. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of current solution vector, X revaluate the objective function and apply penalty function approach to the objective function, if necessary.
4. If Y belongs to the tabu list, go to Step 5 else go to Step 6.
5. If Y does not qualify the aspiration criterion go to Step 2.
6. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$ and put Y into tabu list. Update the existing optimal solution and go to Step 8.
7. Else accept the Y with the probability:

$$P = \exp(-\Delta s/T) \tag{2}$$

where $\Delta s = Z(Y) - Z(X)$.
If the solution is accepted, put Y in the tabu list and replace X with Y.
8. Decrease the temperature periodically.
9. Repeat Steps 2–8 until stopping criterion is met.

Multiobjective optimization

Nowadays, multiobjective optimization has become an important research topic for scientists and researchers. This is due to the multiobjective nature of real world problems and few ambiguities related to multiobjective optimization. It is difficult to compare results of one multiobjective method to another, as there is not a unique optimum in multiobjective optimization as in single objective optimization. So, the best solution in multiobjective terms is decided by the decision maker. Researchers have developed many multiobjective optimization procedures. They have a number of disadvantages and pitfalls. Recently, multiobjective metaheuristic and evolutionary procedures have become very popular for multiobjective optimization. The increasing acceptance of SA and other heuristic algorithms is due to their ability to: (1) find multiple solutions in a single run, (2) work without derivatives, (3) converge speedily to Pareto-

Table 3 CSA for single objective

1. Initialize the temperature.
2. Start with a randomly generated initial solution vector, X, and generate the different chaotic variables (Z_{ki}), $i = 1, 2, \ldots\ldots, n$ by using the chaotic systems:

$$Z_{k+1} = f(\mu, Z_k) = \mu \times Z_k(1 - Z_k) \tag{3}$$

where $Z_k \in [0, 1]$, $k = 0, 1, \ldots.$
Z_k is the value of the variable Z at the kth iteration, μ is called the bifurcation parameter of the system.
3. Generate a new solution vector, Y, in the neighbourhood of current solution vector, X, by using the chaotic variables:

$$Y = X + \alpha(\text{upper limit of variable} - \text{lower limit of variable})Z_k \tag{4}$$

where i and k are the random integers, Z_k is a chaotic variable calculated from Equation (2) and α is a variable which is decreased by the formula $\alpha = \alpha \exp(-\beta)$ in each iteration.
4. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$. Update the existing optimal solution and go to Step 6.
5. Else accept the Y with the probability:

$$P = \exp(-\Delta s/T) \tag{5}$$

where $\Delta s = Z(Y) - Z(X)$
If the solution is accepted, replace X with Y.
6. Decrease the temperature periodically.
7. Repeat Steps 2–6 until stopping criterion is met.

optimal solutions with a high degree of accuracy, (4) handle both continuous function and combinatorial optimization problems with ease, (5) be less susceptible to the shape or continuity of the Pareto front. These issues are a real concern for the techniques of mathematical programming.

Definition of the multiobjective optimization problem. Multiobjective optimization problem (also called multicriteria, multiperformance or vector optimization) can be defined mathematically as:

To find the vector $X = [x_1, x_2, \ldots, x_k]^{\mathrm{T}}$ which will satisfy the following m inequality:

$$g_i(X) \geqslant 0, \quad i = 1, 2, 3, \ldots\ldots\ldots, m \tag{6}$$

the l equality constraints:

$$h_i(X) = 0, \quad i = 1, 2, 3, \ldots\ldots\ldots, l \tag{7}$$

and optimize the objective function vector

$$F(X) = [f_1(X), f_2(X), \ldots\ldots\ldots\ldots\ldots\ldots, f_N(X)]^{\mathrm{T}} \tag{8}$$

where $X = [x_1, x_2, \ldots, x_k]^{\mathrm{T}}$ is the vector of decision variable vector.

Pareto-optimal solutions. The concept of Pareto-optimal solutions was formulated by Vilfredo Pareto in the 19th century (Pareto, 1896). Real-life problems require simultaneous optimization of several incommensurable and often conflicting objectives. Usually, there is no single optimal solution, but there is a set of alternative solutions. These solutions are optimal in the wider sense that no other solutions in the search space are superior to each other when all the objectives are considered. They are known as Pareto-optimal solutions. To define the concept of Pareto optimality, we take the example of a minimization problem with two decision vectors $a, b \in X$. a is said to dominate b *if*

$$\begin{aligned} &\forall i = \{1, 2, \ldots.., N\} : f_i(a) \leqslant f_i(b) \\ &\text{and} \\ &\exists j = \{1, 2, \ldots\ldots, N\} : f_j(a) < f_j(b) \end{aligned} \tag{9}$$

When the objectives associated with any pair of non-dominated solutions are compared, it is found that each solution is superior with respect to at least one objective. The set of non-dominated solutions to a multiobjective optimization problem is known as the Pareto-optimal set (Zitzler and Thiele, 1998).

SA for multiobjective optimization. Initially, SA has been used as an optimization tool for combinatorial optimization problems. Recently, it has been adapted in a multiobjective framework because of its simplicity and capability of producing a Pareto set of solutions in single run with very little computational cost. Additionally, it is not susceptible to the shape of the Pareto set, whereas these two issues are real concerns for mathematical programming techniques. The first multiobjective version of SA has been proposed by Serafini (1985, 1992). The algorithm of the method is almost the same as the algorithm of single objective SA. The method uses a modification of the acceptance criteria of solutions in the original algorithm. Various alternative criteria have been investigated in order to increase the probability of accepting non-dominated solutions. A special rule given by the combination of several criteria has been proposed in order to concentrate the search almost exclusively on the non-dominated solutions. Thereafter, this method has been used by Ulungu and Teghem (1994). They have only used the notion of the probability in the multiobjective framework. Serafini (1994) has used a SA algorithm on the multiobjective framework. A target-vector approach to solve a bi-objective optimization problem has been used. Ulungu *et al* (1999) have proposed a complete MOSA algorithm which they had tested on a multiobjective combinatorial optimization problem. A weighted aggregating function to evaluate the fitness of solutions has been used. The algorithm worked with only one current solution but maintained a population with the non-dominated solutions found during the search. Tuyttens *et al* (2000) have used the MOSA method for the bicriteria assignment problem. This method has been further improved and extensively tested by Ulungu *et al* (1998) and an interactive version of MOSA has been used to solve an industrial problem (UMOSA method). Recently, Suppapitnarm and Parks (1999) have proposed a different SA based approach to tackle multiobjective problems (SMOSA method). The algorithm uses only one solution and the annealing process adjusts each temperature independently according to the performance of the solution in each criterion during the search. An archive set stores all the non-dominated solutions between each of the multiple objectives. A new acceptance probability formulation based on an annealing schedule with multiple temperatures (one for each objective) has also been proposed. The acceptance probability of a new solution depends on whether or not it is added to the set of potentially Pareto-optimal solutions. If it is added to this set, it is accepted to be the current solution with probability equal to one. Otherwise, a multiobjective acceptance rule is used. Czyżak *et al* (1994) have proposed another way to adopt SA to a multiobjective framework, the PSA method. Czyżak and Jaszkiewicz (1998) have combined unicriterion SA and a GA to provide efficient solutions of multicriteria shortest path problem. A population-based extension of SA proposed for multiobjective combinatorial optimization problems has been used. The population of solutions explored their neighbourhood similarly to the classical SA, but weights for each objective tuned in each iteration. The weights for each solution are adjusted in order to increase the probability of moving away from its closest neighbour-

hood in a similar way as in the multiobjective tabu search. Suman (2002, 2003) has proposed two different SA-based approaches (WMOSA and PDMOSA) to tackle the multiobjective optimization of constrained problems. Suman (2003) has also tested five SA algorithms for the system reliability optimization problem. The goal of these methods was to generate a set of solutions that are a good approximation to the whole set of efficient (non-dominated or Pareto-optimal) solutions in a relatively short time.

Suman (2005) has further improved the SA-based multiobjective algorithm so that the user does not need to give a predefined number of maximum iterations. All SA multiobjective algorithms have the advantage that they allow the full exploration of the solution space; since, the starting temperature is high, any move is accepted. The move becomes selective as temperature decreases with increase in iteration number and by the end it accepts only the improving moves.

SA-based multiobjective optimization algorithms

The method of Suppapitnarm and Parks (SMOSA). The concept of archiving the Pareto-optimal solutions for solving multiobjective problems with SA has been used by Suppapitnarm *et al* (2000). The method enables the search to restart from an archived solution in a solution region, where each of the pair of non-dominated solutions is superior with respect to at least one objective. This is called a return-to-base strategy. A new acceptance probability formulation based on an annealing schedule with multiple temperatures (one for each objective) is also proposed. The method does not use a composite objective function, the changes in each objective are compared to each other directly before archiving. This ensures that the moves to a non-dominated solution are accepted. It does not use any weight vector in the acceptance criteria. Hence, the key probability step is given as

$$P = \min\left(1, \prod_{i=1}^{N} \exp\left\{\frac{-\Delta s_i}{T_i}\right\}\right) \tag{10}$$

where $\Delta s_i = (z_i(Y) - z_i(X))$, X is the current solution, Y is the generated solution, z_i is the objective function, T_i is the annealing temperature and P is the probability. Thus, the overall acceptance probability is the product of individual acceptance probabilities for each objective associated with a temperature T_i. All temperatures are set to a large value at the start of the search. A statistical record of the values of each of the objective functions, f_i, is maintained. The temperatures are first lowered after N_{T1} iterations by setting each temperature to the standard deviation of the accepted values of f_i. Thereafter, the temperatures are lowered after every N_{T2} iterations or N_A acceptances. The maximum step change in the control variables is monitored and is varied to reduce violation of the constraints.

The SMOSA algorithm. The basic steps involved in the SMOSA algorithm for a problem having N objective functions and n decision variables are as follows:

1. Start with a randomly generated initial solution vector, X (an $n \times 1$ vector whose elements are decision variables) and evaluate all objective functions and put it into a Pareto set of solutions.
2. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of current solution vector, X, re-evaluate the objective functions and apply a penalty function approach to the corresponding objective functions, if necessary.
3. Compare the generated solution vector with all the solutions in the Pareto set and update the Pareto set, if necessary.
4. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$ and go to Step 7.
5. If the generated solution vector is not archived, accept it with the probability:

$$P = \min\left(1, \prod_{i=1}^{N} \exp\left\{\frac{-\Delta s_i}{T_i}\right\}\right) \tag{11}$$

   where $\Delta s_i = (z_i(Y) - z_i(X))$
   If the generated solution is accepted, make it the current solution vector by putting $X = Y$ and go to Step 7.
6. If the generated solution vector is not accepted, retain the earlier solution vector as the current solution vector and go to Step 7.
7. Periodically, restart with a randomly selected solution from the Pareto set. While periodically restarting with the archived solutions, Suppapitnarm *et al* (2000) have recommended biasing towards the extreme ends of the trade-off surface.
8. Reduce the temperature periodically using a problem-dependent annealing schedule.
9. Repeat Steps 2–8, until a predefined number of iterations is carried out.

The method of Ulungu and Teghem (UMOSA). For a multiobjective problem, a move from the present position to a new position can result in three different possibilities: (a) Improving moves with respect to all objectives is always accepted with probability one. (b) Simultaneous improvement and deterioration with respect to different objectives. In this case neither the new move nor the current solution dominates each other. Therefore, the strategy devised must be sound enough to discriminate between both the solutions. (c) Deterioration with respect to all objectives is accepted with a calculated probability. Here, a strategy has been adapted to handle these situations. The UMOSA algorithm of Ulungu *et al* (1998, 1999) uses a strategy called the criterion scalarizing approach since probability to accept the new solution must take into account the distance

between the old and the new move. The multidimensional criteria space (Δf) is projected into a one-dimensional space (Δs) by using scalarizing functions. Scalarizing functions aggregate the multi-criteria into a unicriterion one using a weight vector. In this approach, the new move is accepted with the probability:

$$P = 1, \quad \text{if } \Delta s \leqslant 0 \tag{12}$$

$$= \exp\left(\frac{-\Delta s}{T}\right), \quad \text{if } \Delta s > 0 \tag{13}$$

where Δs can be defined by:

$$\Delta s = s(Z(Y), \lambda) - s(Z(X), \lambda), \quad s(Z, \lambda) = \sum_{l=1}^{N} \lambda_l z_l \tag{14}$$

where λ is a weight vector.

This method works with predefined diversified weight vector. This set of weights is uniformly generated. Two scalarizing functions have been used: the easiest function, the weighted sum and the Chebysev norm, L. The effect of using different scalarizing functions is small due to the stochastic character of the method. These issues are discussed in detail in Ulungu *et al* (1995, 1998) and Teghem *et al* (2000).

The UMOSA Algorithm. The basic steps involved in the UMOSA algorithm for a problem having N objective functions and n decision variables are as follows:

1. Generate a wide diversified set L of uniform random weight vectors, where $\lambda^l = (\lambda_i^l, i = 1, 2, \ldots\ldots, N)$ and normalize them between 0 and 1, such that $\sum_{i=1}^{i=N} \lambda_i^l = 1$.
2. Start with a randomly generated initial solution vector, X (an $n \times 1$ vector whose elements are decision variables), evaluate all objective functions and apply a penalty function approach to the corresponding objective functions, if necessary and put them into a Pareto set of solutions.
3. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of current solution vector, X, revaluate the objective functions and apply a penalty function approach to the corresponding objective functions, if necessary.
4. Compare the generated solution vector with all solutions in the Pareto set and update the Pareto set, if necessary.
5. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$ and go to Step 8.
6. Accept the generated solution vector as the current solution vector, if it is not archived with the probability

$$P = 1, \quad \text{if } \Delta s \leqslant 0$$

$$= \exp\left(\frac{-\Delta s}{T}\right), \quad \text{if } \Delta s > 0 \tag{15}$$

where $\Delta s = s(Z(Y), \lambda) - s(Z(X), \lambda)$, $s(Z, \lambda) = \sum_{l=1}^{N} \lambda_l z_l$.

If the generated solution is accepted, make it the current solution vector by putting $X = Y$ and go to Step 8.

7. If the generated solution vector is not accepted, retain the earlier solution vector as current solution vector and go to Step 8.
8. Periodically, restart with a randomly selected solution from the Pareto set. While periodically restarting with the archived solutions, Suppapitnarm *et al* (2000) have recommended biasing towards the extreme ends of the trade-off surface.
9. Reduce the temperature periodically using a problem-dependent annealing schedule.
10. Repeat Steps 3–9, until a predefined number of iterations are carried out.
11. Repeat Steps 1–10 for each weight vector λ^l of the set L.

Pareto simulated annealing. Czyżak and Jaszkiewicz (1996, 1997a, 1997b, 1998) have modified the procedure of Ulungu *et al* (1995) by combining unicriterion SA with a GA to provide efficient solutions. This method, known as Pareto simulated annealing (PSA), generates a good approximation of the efficient solution set in a relatively short time. PSA uses the concept of neighbourhood, acceptance of new solutions with some probability and annealing schedule from SA and the concept of using a sample population of interacting solutions from GA. PSA uses scalarizing functions based on probabilities for accepting new solutions. In each iteration of the procedure, a set of solutions called generating samples controls the objective weights used in the acceptance probability. This assures that the generating solutions cover the whole set of efficient solutions. One can increase or decrease the probability of improving values of a particular objective by controlling the weights. The higher the weight associated with a given objective, the lower the probability of accepting moves that decrease the value of this objective and the greater the probability of improving the value of this objective.

The PSA algorithm. The basic steps involved in the PSA algorithm for a problem having N objective functions and n decision variables are as follows:

1. Choose a starting sample of generating solutions, $X \in G$ and update the non-dominated set, for each solution of generating solution set.
2. Generate a random solution, Y, in the neighbourhood of each solution $X \in G$ in the generating sample, evaluate all the objectives and add penalty values to the corresponding objective functions, if necessary.
3. Update the archived set of efficient solutions with Y if it is non-dominated.

4. Select. a non-dominated solution, X', from set G closest to X.
5. If there exists no such X' or it is the first iteration with X then set random weights such that

$$\forall j, \quad \lambda_j \geqslant 0 \text{ and } \sum_j \lambda_j = 1 \tag{16}$$

6. Else for each objective function z_j

$$\lambda_j = \begin{cases} \alpha\lambda_j^x, & z_j(X) \geqslant z_j(X') \\ \lambda_j^x/\alpha, & z_j(X) < z_j(X') \end{cases} \tag{17}$$

where, α is greater than one.
Normalize the weight such that $\sum_j \lambda_j = 1$.
7. Accept solution with the probability

$$P = \min\left(1, \prod_{i=1}^{N} \exp\left\{\frac{-\Delta s_i}{T_i}\right\}\right) \tag{18}$$

where $\Delta s_i = \lambda_i(z_i(Y) - z_i(X))$.
This is known as Rule SL in which all the objectives are aggregated with a weighted sum of the objectives. If the solution is accepted, make it the current solution vector and go to Step 9.
8. If the solution is not accepted, retain the earlier solution as a current solution and go to Step 9.
9. Reduce the temperature periodically using a problem-dependent annealing schedule.
10. Repeat Steps 1–8, until a predefined number of iterations are carried out.

The detailed discussion on PSA is described by Czyżak and Jaszkiewicz (1996, 1997a, 1997b), Hapke *et al* (1996, 1997, 1998a, 1998b), Jaszkiewicz (1997) and Jaszkiewicz and Ferhat (1999).

Multiobjective simulated annealing using constraints violation in acceptance criterion (WMOSA)

Constrained multiobjective optimization is important for practical problem solving. Constraint handling is a crucial part of a real-world problem. Most of the SA-based algorithms handle constraints by using a separate technique like a penalty function approach. WMOSA algorithm (Suman, 2002, 2003, 2004a, 2004b) has attempted to handle constraints with its main algorithm by using a weight vector in the acceptance criterion by directing the move towards the feasible solutions. It does not use any extra technique to handle constraints. It has been shown that the substantial reduction in computational time can be achieved without worsening the quality of solution with WMOSA. Weight vector, W, depends on the number of constraints violated by the solution vector and the objective function vector. W is an N dimensional column vector. Its element is given by

W_j =(Number of constraints satisfied by the solution vector and the objective function vector + number of constraints satisfied by the jth element of the objective function vector (if there are constraints on the jth element of the objective functionspecifically))/ (Number of constraints on the solution vector and the objective function vector + number of constraints onthe jth element of the objective function vector (if there are constraints on the jth element of the objective function vector specifically)) (19)

where N is the number of objective functions.

The WMOSA Algorithm. The basic steps involved in the WMOSA algorithm for a problem having N objective functions and n decision variables are as follows:

1. Start with randomly generated initial solution vector, X (an $n \times 1$ vector whose elements are decision variables), evaluate all objective functions and put it into a Pareto set of solutions.
2. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of current solution vector, X, and revaluate the objective functions.
3. Compare the generated solution vector with all solutions in the Pareto set and update the Pareto set, if necessary.
4. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$ and go to Step 7.
5. Accept the generated solution vector, if it is not archived, with the probability

$$P = \exp\left(-\frac{\Delta s}{T}\right) \tag{20}$$

where $\Delta s = W(Y)Z(Y) - W(X)Z(X)$ and W vector is defined in Equation (16).

If the generated solution vector is accepted make it the current solution vector by putting $X = Y$ and go to Step 7.
6. If the generated solution vector is not accepted, retain the earlier solution vector and go to Step 7.
7. Periodically, restart with a randomly selected solution from the Pareto set. While periodically restarting with the archived solutions, Suppapitnarm *et al* (2000) has recommended biasing towards the extreme ends of the trade-off surface.
8. Reduce the temperature periodically using a problem-dependent annealing schedule.
9. Repeat Steps 2–8, until a predefined number of iterations is carried out.

Multiobjective simulated annealing using pareto-domination-based acceptance criterion (PDMOSA). A strategy of Pareto-domination based fitness can easily be adapted to simulate annealing in the acceptance criterion. This novel idea is used to formulate the PDMOSA algorithm (Suman, 2003, 2004a, 2004b, 2005). Here, fitness of a generated solution in the acceptance criteria is used. Fitness of a solution is defined as one plus the number of dominating solutions in Pareto-optimal set (containing both feasible as well as infeasible solutions). The larger the value of fitness, the worse is the solution. Initially, fitness difference between the current and the generated solution is less and the temperature is high so any move is accepted due to both of them. This gives a way to explore the full solution space. As the number of iterations increases, temperature decreases and fitness difference between the current and generated solutions may increase. Both make the acceptance move more selective and it results in a well-diversified solution in true Pareto-optimal solutions. In this algorithm, the penalty function approach is used to deal with infeasible solutions and the weight vector is not used in acceptance criterion.

The major difference between other SA-based algorithms and PDMOSA is that PDMOSA need not use the objective function value in the acceptance criterion. PDMOSA uses a fitness value, which can be obtained from the Pareto set of solutions (containing both feasible as well as infeasible solutions). This makes the calculation of the probability step simpler and the move tends towards the non-dominated solutions, which is of interest in multiobjective problems. In this way, it has the advantages of diversification of Pareto-optimal solutions over all other algorithms and computational cost over algorithms that use a penalty function approach, that is, SMOSA, UMOSA and PSA.

The PDMOSA algorithm. The basic steps involved in the PDMOSA algorithm for a problem having N objective functions and n decision variables are as follows:

1. Start with randomly generated initial solution vector, X (an $n \times 1$ vector whose elements are decision variables), evaluate all objective functions and put them into a set of potentially Pareto-optimal solutions.
2. Give a random perturbation and generate a new solution vector, Y, in the neighbourhood of the current solution vector, X. Re-evaluate the objective functions and apply a penalty function approach to the corresponding objective functions, if necessary.
3. Compare the generated solution vector with all potentially Pareto-optimal solutions (feasible as well as infeasible solutions) and update the set of potentially Pareto-optimal solutions, if necessary. This solution set is used for the fitness calculation of a solution and go to Step 6.
4. Accept the generated solution vector as the current solution vector with probability, which is given by

$$P = \exp\left(\frac{-\Delta s'}{T}\right) \tag{21}$$

where $\Delta s' = S'_{i-1,\text{current}} - S'_{i,\text{generated}}$ and where $S'_{i,\text{generated}}$ is the Pareto-domination-based fitness of generated solution at iteration number i. $S'_{i-1,\text{current}}$ is the Pareto-domination-based fitness of current solution at iteration number i–1.

The fitness of the current solution at iteration number i, $S'_{i,\text{current}} = 1 +$ the number of solutions from the potentially Pareto-optimal solutions set generated so far dominating the current solution at iteration number i. This means that the better solutions have lower fitness values and fitness has to be minimized. If the generated solution is accepted take it as the current solution vector by putting $X = Y$ and go to Step 6.
5. If the generated solution vector is not accepted, retain the earlier solution vector as the current solution vector and go to Step 6.
6. Periodically, restart with a randomly selected solution from the set of potentially Pareto-optimal solutions.
7. Reduce the temperature periodically using a problem-dependent annealing schedule.
8. Repeat Steps 2–7 until a predefined number of iterations is carried out and take out a set of feasible Pareto-optimal solutions from the Pareto-optimal solution set that has feasible as well as infeasible solutions.

Comparison of multiobjective algorithms

The SMOSA method does not form a composite objective function between each of the objectives (Suppapitnarm *et al*, 2000). This method tests a solution for acceptance by a new probability test based on the product of changes in all objectives only if the solution is not archived. Thus, this method improves the performance of the algorithm by reducing the computation time. This method gives the outcome of a multiobjective optimization as a number of optimal solutions from which a user may choose a particular solution in a number of ways. The method has been successfully applied to problems with two or three objectives. This method was applied to a two objective problem with a simple equation of constraint in the objectives. SMOSA gave performance comparable to that of a well-developed GA. This method can be further improved by applying it to a wide range of problems with more than three objectives.

UMOSA method uses the criterion scalarizing approach to project the multidimensional criteria space (Δf) into a one-dimensional space (Δs) (Ulungu *et al*, 1999). This method builds a list of potentially efficient solutions that are not dominated by any other generated solution. UMOSA method can be easily adapted in an interactive

way to provide satisfying results to the decision maker according to his choices. Ulungu *et al* (1999) have applied this method to the knapsack problem, which is a classical combinatorial optimization problem. Experiments in both single objective and multiobjective cases have been conducted. It is found that the most critical parameter is the cooling factor and it is stressed that its value should be close to 0.975. The same value of the cooling factor for the bi-objective case has been used and it is shown that this method provides a good approximation of the potentially efficient set.

The PSA method uses the concepts of SA and GA for the optimization of multiobjective combinatorial problems (Czyżak and Jaszkiewicz, 1998). The use of a population of generating solutions, each of them exploring the search space according to SA, assures that the generating solutions cover the whole set of efficient solutions. This method can have parallel implementation as the calculations required for each solution may be done on different processors. Czyżak and Jaszkiewicz (1998) have used this method to solve a multiobjective knapsack problem. PSA gave better results than the method of Serafini even in the case of sequential implementation if the size of the generating sample is appropriately set. The method of Serafini, similar to single objective SA, gives a set of potentially efficient solutions for a multiobjective optimization problem. PSA can be further improved by adaptive setting of the size of generating sample and using concepts of other metaheuristics procedures.

The WMOSA method attempts to handle constraints with its main algorithm by using a weight vector in the acceptance criterion (Suman, 2002). This method has been applied on three multiobjective optimization problems and it has been shown that WMOSA takes the least computation time since it does not need to use the penalty function approach to handle the constraints. Suman (2003) has shown that the WMOSA algorithm performs reasonably well for continuous functions as well as for combinatorial problems with a large number of constraints.

The PDMOSA method uses Pareto-domination based fitness to simulate annealing in the acceptance criterion (Suman, 2004a, 2004b) because a good approximation of true Pareto set is needed in multiobjective. Pareto set instead of the value of the objective function at the probability step is used. Thus, this method diversifies the Pareto-optimal solutions and reduces the computation time. Suman (2004a, 2004b) has shown that this method is next to the WMOSA method in terms of computation time. It is also shown that PDMOSA generates well-diversified Pareto-optimal solution and performs reasonably well for continuous function problems with fewer variables.

Annealing schedule

The setting of the parameters for the SA-based algorithm determines the generation of the new solution. The precise rate of cooling is an essential part of SA as it determines the performance of the SA-based algorithm. A high cooling rate leads to poor results because of lack of the representative states, while a low cooling rate requires high computation time to get the result. The following choices must be made for any implementation of SA and they constitute the annealing schedule: initial value of temperature (T), cooling schedule, number of iterations to be performed at each temperature and stopping criterion to terminate the algorithm.

*Initial value of temperature (*T*)*

Initial temperature is chosen such that it can capture the entire solution space. One choice is a very high initial temperature as it increases the solution space. However, at a high initial temperature, SA performs a large number of iterations, which may be without giving better results. Therefore, the initial temperature is chosen by experimentation depending upon the nature of the problem. The range of change, Δf_0 in the value of the objective function with different moves is determined. The initial value of temperature should be considerably larger than the largest Δf_0 encountered. van Laarhoven *et al* (1988) have proposed a method to select the initial temperature based on the initial acceptance ratio χ_0, and the average increase in the objective function, Δf_0:

$$T = -\frac{\Delta f_0}{\ln(\chi_0)} \tag{22}$$

where χ_0 is defined as the number of accepted bad moves divided by the number of attempted bad moves. A similar formula has been proposed by Sait and Youssef (1999) with the only difference being in the definition of χ_0. They have defined χ_0 as the number of accepted moves divided by the number of attempted moves. A simple way of selecting initial temperature has been proposed by Kouvelis and Chiang (1992). They have proposed to select the initial temperature by the formula

$$P = \exp\left(\frac{-\Delta s}{T}\right) \tag{23}$$

where P is the initial average probability of acceptance and is taken in the range of 0.50–0.95. Still no conclusion has been made about the method of selection of initial temperature.

Cooling schedule

Cooling schedule determines functional form of the change in temperature required in SA. The earliest annealing schedules have been based on the analogy with physical annealing. Therefore, they set initial temperature high enough to accept all transitions, which means heating up substances till all the molecules are randomly arranged

in liquid. A proportional temperature is used, that is, $T(i+1)=\alpha T(i)$, where α is a constant known as the cooling factor and it varies from 0.80 to 0.99. Finally, temperature becomes very small and it does not search any smaller energy level. It is called the frozen state.

Three important cooling schedules are logarithmic, Cauchy and exponential (Azencott, 1992). SA converges to the global minimum of the cost function if temperature change is governed by a logarithmic schedule in which the temperature $T(i)$ at step i is given by $T(i)=T_0/\log i$ (Geman and Geman, 1984). This schedule requires the move to be drawn from a Gaussian distribution. A faster schedule is the Cauchy schedule in which $T(i)=T_0/i$ converges to the global minimum when moves are drawn from a Cauchy distribution (Szu and Hartley, 1987). It is sometimes called 'fast simulated annealing'. The fastest is exponential or geometric schedule in which $T(i)=T_0\exp(-C_i)$ where C is a constant. There is no rigorous proof of the convergence of this schedule to the global optimum although good heuristic arguments for its convergence have been made for a system in which annealing state variables are bounded (Ingber, 1989).

A proportional temperature cooling schedule does not lead to equilibrium at low temperature. Therefore, there is a need for a small number of transitions to be sufficient to reach the thermal equilibrium. However, recently a serious attempt has been made with ASA (Gong *et al*, 2001). Few annealing schedules use information about the cost function obtained during the annealing run itself. Such a schedule is called an adaptive cooling schedule (Ingber, 1989; Azizi and Zolfaghari, 2004). An adaptive cooling schedule tries to keep the annealing temperature close to the equilibrium as well as reducing the number of transitions to reach equilibrium. It adjusts the rate of temperature decrease based on the past history of the run. Otten and van Ginneken (1984) have proposed the following cooling schedule:

$$T_{i+1}=T_i-\frac{1}{M_k}\frac{T_k^3}{\sigma^2(T_i)} \tag{24}$$

where σ^2 is the variance of the objective function at equilibrium and M_k is given by

$$M_k=\frac{f_{\max}+T_i\ln(1+\delta)}{\sigma^2(T_i)\ln(1+\delta)}T_i \tag{25}$$

where $f_{\max}$ is an estimated maximum value of the objective function.

Similar to Equation (3), van Laarhoven *et al* (1988) have proposed the following cooling schedule:

$$T_{i+1}=\frac{T_i}{1+(\ln(1+\delta)T_i/3\sigma T_i)} \tag{26}$$

where δ is a small real number.

One of the adaptive cooling schedules is the adaptive schedule of Lam. The Lam schedule (Lam and Delosme, 1988a, 1988b) has been derived by optimizing the rate at which temperature can be decreased subject to the constraint of maintaining quasi-equilibrium. It is given as

$$S_{k+1}=S_k+\lambda\left(\frac{1}{\sigma(S_k)}\right)\left(\frac{1}{S_k^2\sigma^2(S_k)}\right)\times\left(\frac{4\rho_0(S_k)(1-\rho_0(S_k))^2}{(2-\rho_0(S_k))^2}\right) \tag{27}$$

where $S_i=1/T_i$ and T_i is the temperature at ith iteration of the cost function E. $\sigma(S_k)$ is the standard deviation of E at this step and $\rho_0(S_k)$ is the acceptance ratio, that is, the ratio of accepted to attempted moves. The following four factors play important roles:

(a) λ is a quality factor. Making smaller λ improves the quality of the solution, but it also increases the computation time.
(b) $1/(\sigma(S_k))$ measures the distance of the system from quasi-equilibrium.
(c) $(1/(S_k^2\sigma^2(S_k)))$ is the inverse of the statistical specific heat which depends on the variance.
(d) $((4\rho_0(S_k)(1-\rho_0(S_k))^2)/((2-\rho_0(S_k))^2))$ is equal to $\rho_2/2$ where ρ_2 is the variance of the average energy change during a move. It is a measure of how effectively the state space is sampled and was found to be at a maximum value when $\rho_0\approx 0.44$.

Azizi and Zolfahgari (2004) have used an adaptive annealing schedule that adjusts the temperature dynamically based on the profile of the search path. Such adjustments could be in any direction including the possibility of reheating. In their first proposed method, an adaptive temperature control scheme has been used that changes temperature based on the number of consecutive improving moves. In the second method, a tabulist has been added to the ASA algorithm in order to avoid revisits to the solutions.

Triki *et al* (2004) have studied annealing schedules. They have performed experiments to construct an optimum annealing schedule that showed that there was no clearly better annealing schedule than the logarithmic schedule to ensure convergence towards the set of optima with probability one. They have developed software to calculate a practical and optimum annealing schedule for a given objective function. They have also conducted experiments on adaptive annealing schedules to compare classical annealing schedules. They proposed the following cooling schedules:

$$T_{i+1}=T_i\exp\left(-\frac{\lambda T_i}{\sigma(T_i)}\right) \tag{28}$$

$$T_{i+1}=T_i\left(1-T_i\frac{\Delta(T_i)}{\sigma^2(T_i)}\right) \tag{29}$$

They have showed that several classical adaptive temperature decrement rules proposed in the literature, that had different theoretical foundations and different mathematical equations, were in fact the same practical rule. They have calculated a new adaptive decrement rule for controlling and tuning the SA algorithm.

Other cooling schedules make a more direct appeal to the theoretical results on asymptotic convergence. Lundy and Mees (1986) have proposed an annealing schedule where there is only a single iteration at each temperature. They have used heuristic arguments to derive a temperature function of form

$$T_{i+1} = \frac{T_i}{1 + BT_i} \tag{30}$$

where B is a constant. Equation (9) is equivalent to

$$T_i = \frac{C_1}{1 + iC_2} \tag{31}$$

where C_1 and C_2 are constants. SA proposed by Connolly (1987, 1988) suggested that the majority of the iterations should be conducted at a suitably fixed temperature.

The choice of decreasing the temperature is an important issue as there has been a conflict, since early days of SA, between theory and practice. There is no universally valid conclusion in the literature. However, a general choice is to cool the system slowly at the stage where the objective function is rapidly improving.

Number of iterations

The number of iterations at each temperature is chosen so that the system is sufficiently close to the stationary distribution at that temperature. Aarts and Korst (1989) and van Laarhoven and Aarts (1987) refer this as 'quasi-equilibrium'. Enough number of iterations at each temperature should be performed if temperature is decreased periodically. If less number of iterations is performed, all represented states will not be searched and the solution will not be able to reach the global optimum. The value of the number of iterations depends on the nature of the problem. Again, there is no general agreement about it.

Stopping criterion

Various stopping criteria have been developed with time. (i) Total number of iterations and number of iterations to move at each temperature have been given. This criterion leads to higher computation time without much update in f and sometimes it may lead to local minimum due to less number of iterations. The number of iterations used by an algorithm depends on the complexity of a problem, which may not be known beforehand. (ii) A minimum value of temperature and number of iterations to move at each temperature has been given. This idea is generated with the fact that the chance of improvement in a solution is rare once the temperature is close to zero. At very low temperature, moves will be trapped in the neighbourhood of the current solution. (iii) Number of iterations to move at each temperature and a predefined number of iterations to get a better solution (called FROZEN) has been given.

Computational results and performance improvement

SA is a stochastic algorithm, which requires a neighbourhood as well as a number of parameters to specify a cooling schedule. Many variants of the basic algorithms have been proposed. Therefore, it requires considerable testing of the algorithms to give sound conclusions. Johnson *et al* (1989), van Laarhoven (1988) and Lundy and Mees (1986) have shown that SA works better than the descent algorithm. Hertz and de Werra (1987) have shown that tabu search works better than SA in graph colouring problem. On the other hand, Bland and Dawson (1989) have obtained better results with SA for layout optimization problems. It is a premature stage to make any judgment about the performance of SA. SA has been applied to many problems where no problem specific algorithms were available. They include problems of VLSI design and molecular structure. SA provides good solutions for a football pool problem (Wille, 1987; van Laarhoven, 1988). Johnson *et al* (1989, 1991) have shown that SA outperforms the classical Kernighan and Lin algorithm in both quality and speed for certain type of random graphs. In 1992, Ingber and Rosen (1992) have proposed a very fast simulated reannealing (VFSR) method that is efficient in its search strategy and which statistically guarantees to find the global optima. Their results reveal that the VSFR method is orders of magnitude more efficient than a GA. Pirlot (1996) has showed that tabu search works well then SA, though he was not sure if he has used the correct annealing schedule. However, Johnson *et al* (1989) and van Laarhoven (1988) have shown that there was not much difference in the solution with different annealing schedule. Recently, Youssef *et al* (2003) have studied SA, tabu and evolutionary algorithm together. They have shown that tabu and evolutionary algorithms outperform over SA.

The SA algorithm for single objective problems sometimes accepts solutions, which are worse than the current solution. Therefore, it is possible that the final solution can be worse than the best solution. It is suggested to store the best solution to improve the performance of SA. It is supported by Glover and Greenberg (1989) and Connolly (1988). Anily and Federguen (1987), Faigle and Schrader (1988), Romeo and Sangiovanni-Vincentelli (1985) and Matsuo *et al* (1988) have considered to replace the standard probability function $\exp(-\delta/T)$. Grover (1986), Greene and Supowit (1986) and Tovey (1988) have suggested using an approximate value of

δ rather than the exact value. A very important way of improving a heuristic technique is to mix two algorithms. Attempts have been made to improve the performance of SA by combining it with another algorithm. There are two ways of using SA with another algorithm. In the first way, a good initial guess is provided to the SA algorithm which is improved by SA (Chams *et al*, 1987; and Johnson, 1989). In the second way, SA is implemented by using a parallel version of the algorithm. A nice discussion is found in the book by Aarts and Korst (1989). Lam and Delsome (1988a, b) have attempted to improve SA algorithm by adaptively controlling the rate of temperature decrease and the size of moves so as to maximize the optimization efficiency. Chu *et al* (1996) proposed a parallelized Lam's version of SA so that SA can be parallelized. SA is difficult to parallelize due to its intrinsic serial nature and direct parallelization of the Metropolis algorithm. Their approach involves strategies of pooling statistics of both state and algorithm parameters while stirring the system. Rosen and Harmonosky (2003) have improved SA by using discrete decision variable space. Tiwari and Roy (2003) have proposed an SA-hybrid algorithm. Jeon and Kim (2004) have proposed an efficient SA that uses tabu search to get an optimal solution fast. Nwana *et al* (2004) have mixed SA with branch and bound.

Although the SA method can find the optimal solution to most single and multiobjective optimization problems, the algorithm always requires numerous numerical iterations to yield a good solution. Suman *et al* (2005) has proposed orthogonal SA, which combines SA with fractional factorial analysis and enhances the convergence and accuracy of the solution. Fractional factorial analysis, which involved several factorial experiments based on orthogonal tables to determine the best combination of factors, has been incorporated. The performance of the orthogonal SA method has been evaluated by solving several global optimization problems. It has been shown that the orthogonal SA method outperforms the SA in solving global optimization problems with a linear or nonlinear objective function. CSA uses SA with chaotic systems (Mingjun and Huanwen, 2004). The CSA is different from SA as chaotic initialization and chaotic sequences replace the Gaussian distribution. However, the theory of the algorithm of CSA and the method of combining SA to chaotic system need to be further improved.

Complete physical analogy and theoretical studies on SA are well documented in the literature. Apart from the discussed references, they are described by Aarts and van Laarhoven (1985), Alspector and Allen (1987), Casotto *et al* (1987), Chams *et al* (1987), Chu *et al* (1996), Eglese and Rand (1987), Farhat (1987), Hejak (1988), Jerrum and Sinclair (1988), van Laarhoven *et al* (1988, 1992), Mitra *et al* (1986), Sasaki and Hajek (1988) and Wright (1989).

Direction of future research

SA-based optimization algorithms can be used to optimize single as well as multiobjective optimization problems. It has been applied in various fields like process system engineering, operational research, smart materials etc. Though in the last two decades a lot of work has been done in this area, most of it has been concentrated on application of some conventional or *ad-hoc* techniques to certain difficult problems. Therefore, there are still many open problems. Some of them have been discussed herein.

Annealing schedule has been a topic of research in SA since it has been used as an optimization technique. It is used at the probability step in SA and therefore, it governs the move. A wise choice of annealing schedule can save computational time and can improve the quality of solution. However, it has not been explored properly though few attempts have been made in past. A sincere effort is required to choose an optimal annealing schedule for a problem. An attention is needed to select the initial temperature and to decide the number of iterations to be performed at each temperature.

The search space in SA is explored one by one and in a sequential manner. Therefore, there is a chance of revisiting a solution multiple times, which leads to extra computation time without improving the quality of the solution. Recently, researchers have attempted to use tabu search with SA to avoid revisiting. However, avoidance of revisiting a solution by tabu search depends on the size of the tabu list. A large tabu list is also computationally intensive. Additionally, it has been used only in single objective optimization problems. SA with tabu search should be more effective in multiobjective since the management of a Pareto set is computationally intensive.

SA generates the new solution vector randomly; it is inefficient for searching for the optimal solution of large parameter optimization problems. Suman *et al* (2005) performed factorial experiments according to orthogonal tables to generate a better new solution vector. This orthogonal SA method should be further investigated as it has the potential to optimize complex optimization problems. The basis for the selection of the initial solution vector should also be investigated.

In SA, the selection of an initial solution vector and the next move require the lower and upper bound of the solution vector, the neighbourhood of all solution vectors, to be specified. In real-life optimization problem, the neighbourhood of all solution vectors may not be known. SA can be extended to search the neighbourhood of the solution vector in a manner it has been extended to estimate the algorithmic parameters.

SA-based optimization algorithms use few parameters in their algorithms as in other optimization methods. It will be advantageous if the effect of each parameter on the performance of the algorithms has been studied. Then, the

optimal values of the parameters to obtain the best results can be obtained. SA-based multiobjective algorithms give a Pareto set of solutions. It is difficult for a decision maker to decide on the solution due to the considerable large size of Pareto set. It would be better if we could reduce the Pareto set to a less (desired) number of solutions based on the choice of the decision maker. Such an attempt has been made by Kunha *et al* (1997). They clustered together points that were within a certain distance of each other in the Pareto front.

Multiobjective optimization has multiple goals unlike single objective and it makes it difficult to measure the quality of the solution. Several attempts have been made to measure the quality of a solution (van Veldhuizen and Lamont, 1998b; Zitzler and Thiele, 1998; Srinivas and Deb, 1994; Suman, 2005). Some of them are based on visualization, whereas some use matrices. The method of matrices is the most promising so far. A general measurement technique to measure the performance of an algorithm should be developed. Methods need to be developed to handle a situation when a particular objective function is more desirable. Bentley and Wakefield (1997) proposed the use of weights to estimate the importance of an objective, which may not be always true.

Researchers have started to solve problems of single objective with constraint as a multiobjective optimization by considering constraints as separate objectives (Fonseca and Fleming, 1995; Surry and Mudge, 1995). Similarly, a multiobjective optimization problem can be solved as a single objective problem with one objective function and the rest of the objective functions can be treated as the constraints. Improvement in this front is required since it may give promising results in term of saving in computational cost and quality of solution. There should be a set of benchmark problems (constraints and unconstraint) that can be used to test the SA-based old and new multiobjective algorithms. Therefore, a few benchmark problems should be formulated.

Conclusions

SA-based algorithms solve single objective and multiobjective optimization problems, where a desired global minimum/maximum is hidden among many local minima/maxima. These methods have attractive and unique features when compared with other optimization technique. Firstly, a solution does not get trapped in a local minimum or maximum by sometimes accepting even the worse move. Secondly, configuration decisions proceed in a logical manner in SA. The paper provides a comprehensive review of SA-based optimization algorithms. Few single and multiobjective algorithms have been presented briefly and their applications have been discussed. Each of the algorithms has their own advantages and requires reasonable computation time. Therefore, all algorithms should be suggested to use to generate a larger set of optimal solutions giving a wider choice to the decision maker. Annealing schedule is an essential part of SA as it determines the performance of the SA algorithm. Computational results show some conflicting results when SA is compared with other algorithms. Therefore, it is premature to judge the performance of SA. The performance of SA-based multiobjective algorithms can be improved by using optimal annealing schedule, other algorithmic parameters and by using SA with another algorithm. The areas of future research in SA have been suggested and they are: choosing an optimal annealing schedule, using tabu search with SA, selecting new solution vector efficiently, studying effect of algorithmic parameters on the performance of SA, reducing the Pareto set to a smaller number of solutions and measuring the quality of a solution.

Acknowledgements—The correspondence of Balram Suman with Kiran Mishra, graduate student, The State University of New Jersey, Rutgers and her valuable suggestions are gratefully acknowledged.

References

Aarts EHL and van Laarhoven PJM (1985). Statistical cooling: a general approach to combinatorial optimization problems. *Philips J Res* **40**: 193–226.

Aarts EHL and Korst JHM (1989). *Simulated Annealing and Boltzmann Machine*. Wiley: Chichester.

Alspector J and Allen RB (1987). A neuromorphic VLSI learning system. In: Losleben P (ed). *Advanced Research in VLSI*. MIT Press, Cambridge, MA, pp 313–349.

Anily S and Federguen A (1987). Simulated Annealing method with general acceptance probablisties. *J App Prob* **24**: 657–667.

Azencott R (1992). Sequential simulated annealing: speed of convergence and acceleration techniques. In: *Simulated Annealing: Penalization Techniques*. Wiley, New York, p 1.

Azizi N and Zolfaghari S (2004). Adaptive temperature control for simulated annealing: a comparative study. *Comput Opns Res* **31**: 2439–2451.

Bell DA, McErlean FJ, Stewart PM and McClean S (1987). Application of simulated annealing to clustering tuples in database. *J Am Soc Inform Sci* **41**: 98–110.

Bentley PJ and Wakefield JP (1997). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In: Chawdhry PK, Roy R and Pant RK (eds). *Proceedings of the 2nd On-Line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2)*, pp 23–27.

Bland JA and Dawson GP (1989). *Tabu Search Applied to Layout Optimization*. Report, Department of Maths, Stats and OR, Trent Polytechnic, UK. Presented at CO89, Leeds.

Casotto A, Romeo F and Sangiovanni-Vincentelli AL (1987). A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Trans Computer-Aided Design* **6**: 838–847.

Černy V (1985). Thermodynamics approach to the traveling salesman problem: an efficient simulation algorithm. *J Optimization Theory Appl* **45**: 41–51.

Chams M, Hertz A and de Werra D (1987). Some experiments with simulated annealing for coloring graphs. *Eur J Opl Res* **32**: 260–266.

Chattopadhyay A and Seeley CE (1994). A simulated annealing technique for multiobjective optimization of intelligent structures. *Smart Mater Struct* **3**: 98–106.

Chen J, Zhang YF and Nee AYC (1988). Setup planning using Hopfield net and simulated annealing. *Int J Product Res* **36**: 981–1000.

Chen S and Luk BL (1999). Adaptive simulated annealing for optimization in signal processing applications. *Signal Process* **79**: 117–128.

Cho J-H and Kim Y-D (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *J Opl Res Soc* **48**: 736–744.

Chu KW, Deng Y and Reinitz J (1996). Parallel simulated annealing by mixing of states. *J Comput Phys* **148**: 646–662.

Coello Coello CA (1996). *An empirical study of evolutionary techniques for multiobjective optimization in engineering design*. PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA.

Coello Coello CA (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge Inform Sys* **1**: 269–308.

Collins NE, Eglese RW and Golden BL (1988). Simulated annealing—an annotated bibliography. *Am J Math Mngt Sci* **8**: 209–307.

Connolly DT (1987). *Combinatorial optimization using simulated annealing*. Report, London School of Economics, London, WC2A 2AE. Presented at the Martin Beale Memorial Symposium, London, July, 1987.

Connolly DT (1988). An improved annealing scheme for the QAP. *Eur J Opl Res* **46**: 93–100.

Czyżak P, Hapke M and Jaszkiewicz A (1994). *Application of the Pareto-simulated annealing to the multiple criteria shortest path problem*. Technical Report, Politechnika Poznanska Instytut Informatyki, Poland.

Czyżak P and Jaszkiewicz A (1996). A multiobjective metaheuristic approach to the localization of a chain of petrol stations by the capital budgeting model. *Control Cybernet* **25**: 177–187.

Czyżak P and Jaszkiewicz A (1997a). Pareto simulated annealing. In: Fandel G, Gal T (eds). *Multiple Criteria Decision Making. Proceedings of the XIIth International Conference, Hagen (Germany)*. Springer-Verlag, Berlin-Heidelberg, pp 297–307.

Czyżak P and Jaszkiewicz A (1997b). The multiobjective metheuristic approach for optimization of complex manufacturing systems. In: Fandel G, Gal T (eds). *Multiple Criteria Decision Making. Proceedings of the XIIth International Conference, Hagen (Germany)*. Springer-Verlag, Berlin-Heidelberg, pp 591–592.

Czyżak P and Jaszkiewicz A (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *J Multi-Criteria Decision Anal* **7**: 34–47.

Eglese RW (1990). Simulated annealing: a tool for operational research. *Eur J Opl Res*. **46**: 271–281.

Eglese RW and Rand GK (1987). Conference seminar timetabling. *J Opl Res Soc* **38**: 591–598.

Faigle U and Schrader R (1988). On the convergence of stationary distributions in simulated annealing algorithms. *Inform Process Lett* **27**: 189–194.

Farhat NH (1987). Optoelectric analogs of self-programming neural nets: architecture and methodologies for implementing fast stochastic learning by simulated annealing. *Appl Optics* **26**: 5093–5103.

Fonseca MC and Fleming PJ (1995). *Multiobjective optimization and multiple constraints handling with evolutionary algorithms II: application example*. Technical Report 565, University of Sheffeld, Sheffeld, UK.

Geman S and Geman D (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans Pattern Anal Mach Intell* **6**: 721.

Girard T, Staraj R, Cambiaggio E and Muller F (2001). A simulated annealing algorithm for planner or conformal antenna array synthesis with optimized polarization. *Microwave Opt Technol Lett* **28**: 86–89.

Golenko-Ginzburg D and Sims JA (1992). Using permutation spaces in job-shop scheduling. *Asia Pacific J Opn Res* **9**: 183–193.

Gong G, Lin Y and Qian M (2001). An adaptive simulated annealing algorithm. *Stoch Process Appl* **94**: 95–103.

Glover F and Greenberg HJ (1989). New approaches for heuristic search: a bilateral linkage with artificial intelligence. *Eur J Opl Res* **39**: 119–130.

Greene JW and Supowit KJ (1986). Simulated annealing without rejected move. *IEEE Trans Comput Aided Design* **5**: 221–228.

Grover LK (1986). A new simulated annealing algorithm for standard cell placement. *Proceedings of IEEE International Conference on Computer-Aided Design*. Santa Clara, pp 378–380.

Hanke M and Li P (2000). Simulated annealing for the optimization of batch distillation process. *Comput Chem Engin* **24**: 1–8.

Hapke M, Jaszkiewicz A and Słowiński R (1996). Interactive analysis of multiple-criteria project scheduling problems. *Proceedings of the Fifth International Workshop on Project Management and Scheduling—EURO PMS'96*. Poznań, Poland, 11–13.04.96, pp 107–110.

Hapke M, Jaszkiewicz A and Słowiński R (1997). Fuzzy project scheduling with multiple criteria. *Proceedings of Sixth IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'97*, July 1–5, Barcelona: Spain, pp 1277–1282.

Hapke M, Jaszkiewicz A and Słowiński R (1998a). Fuzzy multi-mode resource-constrained project scheduling with multiple objectives. In: Wêglarz J (ed). *Recent Advances in Project Scheduling*, Chapter 16. Kluwer Academic Publishers, Dordrecht, pp 355–382.

Hapke M, Jaszkiewicz A and Słowiński R (1998b). Interactive analysis of multiple-criteria project scheduling problems. *Eur J Opl Res* **107**: 315–324.

Hejak B (1988). Cooling schedule for optimal annealing. *Math Opns Res* **13**: 311–329.

Hertz A and de Werra D (1987). Using tabu search techniques for graph coloring. *Computing* **39**: 345–351.

Ingber L (1989). Very fast simulated annealing. *Math Comput Model* **12**: 967.

Ingber L and Rosen B (1992). Genetic algorithms and very fast simulated annealing: a comparison. *Math Comput Model* **16**: 87–100.

Jaszkiewicz A (1997). A metaheuristic approach to multiple objective nurses scheduling. *Foundations of Comput Decision Sci* **22**: 169–184.

Jaszkiewicz A and Ferhat AB (1999). Solving multiple criteria choice problems by interactive trichotomy segmentation. *Eur J Opl Res* **113**: 271–280.

Jeon YJ and Kim JC (2004). Application of simulated annealing and tabu search for loss minimization in distributed systems. *Int J Electrical Power Energy Sys* **26**: 9–18.

Jerrum M and Sinclair A (1988). *Approximating the permanent*. Internal Report CSR-275-88, Department of Computer Science, University of Edinburgh.

Johnson DS, Aragon CR, Mcgeogh LA and Schevon C (1989). Optimization by simulated annealing: An experimental evolution part I (graph partitioning). *Opns Res* **37**: 865–892.

Johnson DS, Aragon CR, Mcgeogh LA and Schevon C (1991). Optimization by simulated annealing: An experimental evolu-

tion part II (graph coloring and number partitioning). *Opns Res* **39**: 378–406.
Kim JU, Kim YD and Shim SO (2002). Heuristic algorithms for a multi-period multi-stop transportation planning problem. *J Opl Res Soc* **53**: 1027–1037.
Kirkpatrick S, Gelatt Jr CD and Vecchi MP (1983). Optimization by simulated annealing. *Science* **220**: 671–680.
Kouvelis P and Chiang W (1992). A simulated annealing procedure for single row layout problems in flexible manufacturing systems. *Int J Product Res* **30**: 717–732.
Kumral M (2003). Application of chance-constrained programming based on multiobjective simulated annealing to solve a mineral blending problem. *Engin Optimi* **35**: 661–673.
Kunha AG, Oliveira P and Covas JA (1997). Use of genetic algorithms in multicriteria optimization to solve industrial problems. In: Back T (ed). *Proceedings of the Seventh International Conference on Genetic Algorithms*. pp 682–688.
Lam J and Delosme JM (1988a). *An efficient simulated annealing schedule: derivation*. Technical Report 8816, Electrical Engineering Department, Yale, New Havan, CT, September.
Lam J and Delosme JM (1988b). *An efficient simulated annealing schedule: implementation and evaluation*. Technical Report 8817, Electrical Engineering Department, Yale, New Havan, CT, September.
Liu HC and Huang JS (1998). Pattern recognition using evolution algorithms with fast simulated annealing. *Pattern Recognition Lett* **19**: 403–413.
Lucic P and Teodorovic D (1999). Simulated annealing for the multiobjective aircrew roistering problem. *Transportation Res Part A* **33**: 19–45.
Lundy M and Mees A (1986). Convergence of an annealing algorithm. *Mathematical Programming* **34**: 111–124.
Maffioli F (1987). Randomized heuristic for NP-hard problem. In: Andreatta G, Mason F and Serafini P (eds). *Advanced School on Stochastic in Combinatorial Optimization*. World Scientific, Singapore, pp 760–793.
Matsuo H, Suh CJ and Sullivan RS (1988). *A controlled search simulated annealing method for the general jobshop scheduling problem*. Working paper # 03-04-88, Department of Management, The University of Texas at Austin, Austin.
McCormick G and Powell RS (2004). Derivation of near-optimal pump schedules for water distribution by simulated annealing. *J Opl Res Soc* **55**: 728–736.
Meller RD and Bozer YA (1996). A new simulated annealing algorithm facility layout problem. *Int J Product Res* **34**: 1675–1692.
Metropolis N *et al* (1953). Equations of state calculations by fast computing machines. *J Chem Phys* **21**: 1087–1092.
Mingjun J and Huanwen T (2004). Application of chaos in simulated annealing. *Chaos, Solitons Fractals* **21**: 933–944.
Mitra D, Romeo F and Sangiovanni-Vincentelli AL (1986). Convergence of finite-time behavior of simulated annealing. *Adv Appl Prob* **18**: 747–771.
Mukhopadhyay SK, Singh MK and Srivastava R (1998). FMS machine loading: a simulated annealing approach. *Int J Product Res* **36**: 1529–1547.
Nwana V, Darby Dowman K and Mitra G (2004). A co-operative parallel heuristic for mixed zero-one linear programming: combining simulated annealing with branch and bound. *Eur J Opl Res* **164**: 12–23.
Otten RHJM and van Ginneken LPPP (1984). Floorpan design using simulated annealing. In: *Proceedings of the IEEE International Conference in Computer-Aided Design*. Santa Clara, pp 96–98.
Pareto V (1896). *Cours D'Economie Politique*, Vol. I and II. F. Rouge: Lausanne.
Pirlot M (1996). General local search methods. *Eur J Opl Res* **92**: 493–511.
Romeo F and Sangiovanni-Vincentelli AL (1985). Probablistic hill climbing algorithms: properties and applications. *Proceedings of Chapel Hill Conference on VLSI*, Chapel Hill, NC, pp 393–403.
Rosen LS and Harmonosky MC (2003). An improved simulated annealing simulation optimization method for discrete parameter stochastic systems. *Comput Opns Res* **32**: 343–358.
Rutenbar RA (1989). Simulated annealing algorithms: an overview. *IEEE Circuits Devices Mag* **5**: 1989.
Sait SM and Youssef H (1999). *Iterative Computer Algorithms with Applications in Engineering*. Press of IEEE Computer Society: Silver Spring MD.
Sasaki GH and Hajek B (1988). The time complexity of maximum matching by simulated annealing. *J ACM* **35**: 387–403.
Serafini P (1985). *Mathematics of Multiobjective Optimization*, Vol. 289, CISM Courses and Lectures. Springer-Verlag: Berlin.
Serafini P (1992). Simulated annealing for multiple objective optimization problems. In: *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*. Taipei, 19–24 July 1, pp 87–96.
Serafini P (1994). Simulated annealing for multiple objective optimization problems. In: Tzeng GH, Wang HF, Wen VP and Yu PL (eds). *Multiple Criteria Decision Making. Expand and Enrich the Domains of Thinking and Application*. Springer-Verlag, Berlin, pp 283–292.
Shutler PME (2003). A priority list based heuristic for the job shop problem. *J Opl Res Soc* **54**: 571–584.
Sridhar J and Rajendran C (1993). Scheduling in a cellular manufacturing system: a simulated annealing approach. *Int J Product Res* **31**: 2927–2945.
Srinivas N and Deb K (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol Comput* **2**: 221.
Starink JPP and Backer E (1995). Finding point correspondences using simulated annealing. *Pattern Recog* **28**: 231–240.
Suman B (2002). Multiobjective simulated annealing—a metaheuristic technique for multiobjective optimization of a constrained problem. *Foundations of Comput Decision Sci* **27**: 171–191.
Suman B (2003). Simulated annealing based multiobjective algorithm and their application for system reliability. *Engin Optim* **35**: 391–416.
Suman B (2004a). Study of Simulated annealing based multiobjective algorithm for multiobjective optimization of a constrained problem. *Comput Chem Engin* **28**: 1849–1871.
Suman B (2004b). On-line multiobjective optimization algorithmic parameter estimation. *Applied Soft Computing*, submitted.
Suman B (2005). Self-stopping PDMOSA and performance measure in simulated annealing based multiobjective optimization algorithms. *Comput Chem Engin* **29**: 1131–1147.
Suman B, Jha S and Hoda N (2005). Novel orthogonal simulated annealing for multiobjective optimization. *IEEE Trans Evol Opti*, submitted.
Suppapitnarm A and Parks T (1999). Simulated annealing: an alternative approach to true multiobjective optimization. In: *Genetic and Evolutionary Computation Conference*. Conference Workshop Program, Orlando, Florida, pp 406–407.
Suppapitnarm A, Seffen KA, Parks GT and Clarkson PJ (2000). Simulated annealing: an alternative approach to true multiobjective optimization. *Engin Optim* **33**: 59.
Suresh G and Sahu S (1994). Stochastic assembly line balancing using simulated annealing. *Int J Product Res* **32**: 1801–1810.
Surry PD and Mudge T (1995). A multiobjective approach to constrained optimization of gas supply networks: The COMO-

GA method. In: Fogarty TG (ed). *Evolutionary Computing*. AISB Workshop. Selected papers, Lecture Notes in Computer Science. Springer-Verlag, Sheffield, UK, pp 166–180.
Swarnkar R and Tiwari MK (2004). Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. *Robot Computer-Integrated Manufac* **20**: 199–209.
Szu H and Hartley R (1987). Fast simulated annealing. *Phys Lett A* **122**: 157–162.
Teghem J, Tuyttens D and Ulungu EL (2000). An intractive heuristic method for multiobjective combinatorial optimization. *Comput Opns Res* **27**: 621–634.
Terzi E, Vikiali A and Angelis L (2004). A simulated annealing approach for multimedia data placement. *J Sys Software* **73**: 467–480.
Tiwari MK and Roy D (2003). Solving a part classification problem using simulated annealing-like hybrid algorithm. *Robot Computer-Integrated Manufac* **19**: 415–424.
Tovey CA (1988). Simulated Annealing. *Am J Math Mngt Sci* **8**: 389–407.
Triki E, Collette Y and Siarry P (2004). A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *Eur J Opl Res* **166**: 77–92.
Tuyttens D, Teghem J, Fortemps PH and Nieuwenhuyze KV (2000). Performance of the MOSA method for the bicriteria assignment problem. *J Heuristics* **6**: 295.
Ulungu LE and Teghem J (1994). Multiobjective combinatorial optimization problems: a survey. *J Multicriteria Decision Anal* **3**: 83–104.
Ulungu LE, Teghem J and Fortemps P (1995). Heuristics for multiobjective combinatorial optimization problems by simulated annealing. In: Gu J, Chen G, Wei Q, and Wang S (eds). *MCDM: Theory and Applications*. Sci-Tech, Windsor, UK, pp 269–278.
Ulungu LE, Teghem J and Ost C (1998). Interactive simulated annealing in a multiobjective framework: application to an industrial problem. *J Opl Res Soc* **49**: 1044–1050.
Ulungu LE, Teghem J, Fortemps PH and Tuyttens D (1999). MOSA Method: a tool for solving multiobjective combinatorial optimization problems. *J Multicriteria Decision Anal* **8**: 221–236.
van Laarhoven PJM and Aarts EHL (1987). *Simulated Annealing: Theory and Practice*. Kluwer Academic Publishers: Dordrecht.
van Laarhoven PJM (1988). *Theoretical and computational aspects of simulated annealing*. PhD thesis, Erasmus University: Rotterdam.
van Laarhoven PJM, Aarts EHL, van Lint JH and Willie LT (1988). New upper bounds for the football pool problem for 6, 7 and 8 matches. *J Combinat Theory A* **52**: 304–312.
van Laarhoven PJM, Aarts EHL and Lenstra JK (1992). Jobshop scheduling by simulated annealing. *Opns Res* **40**: 113–125.
van Veldhuizen DA and Lamont GB (1998a). *Multiobjective evolutionary algorithm research: a history and analysis*. Technical Report TR-98-03, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Ptterson, AFB, Ohio.
van Veldhuizen DA and Lamont GB (1998b). Evolutionary computing conference to a Pareto front. In: Koza JR (ed). *Late Breaking Papers at the Genetic Programming*. Conference, Stanford University, California, July 1998, Stanford University Bookstore, Stanford, California, pp 221–228.
Wille LT (1987). The football pool problem for 6 matches: a new upper bound obtained by simulated annealing. *J Combinat Theory A* **45**: 171–177.
Wright MB (1989). Applying stochastic algorithms to a locomotive scheduling problem. *J Opl Res Soc* **40**: 187–192.
Yip PPC and Pao YH (1995). Combinatorial optimization with use of guided evolutionary simulated annealing. *IEEE Trans* **6**: 290–295.
Youssef H, Sait SM and Adiche H (2003). Evolutionary algorithm simulated annealing and tabu search: a comparative study. *Eng Appl Artif Intel* **14**: 167–181.
Zitzler E and Thiele L (1998). Multiobjective optimization using evolutionary algorithms: a comperative case study. In: Eiben AE, Back T, Schoenauer M and Schwefel HP (eds). *Parallel Problem Solving from Nature V*. Springer, Berlin, Germany, pp 292–301.
Zolfaghari S and Liang M (1999). Jointly solving the group scheduling and machine speed selection problems. *Int J Product Res* **37**: 2377–2397.

Appendix

Acronyms

ASA	Adaptive Simulated Annealing
GA	Genetic Algorithm
PDMOSA	Pareto Dominant based Multiobjective Simulated Annealing
PSA	Pareto Simulated Annealing
SA	Simulated Annealing
CSA	Chaotic Simulated Annealing
SMOSA	Suppapitnarm Multiobjective Simulated Annealing
UMOSA	Ulungu Multiobjective Simulated Annealing
WMOSA	Weight based Multiobjective Simulated Annealing

Notation

a, b	decision vector
B, C, C_1, C_2	constant
ΔE	change in energy
F	objective function vector
f	objective function value
$f_{\max}$	estimated maximum value of the objective function
Δf	multidimensional criteria space
Δf_0	range of change in the value of the objective function
g	inequality constraint violation
h	equality constraint violation
I	a column vector with all elements equal to 1.
i, j	index
J	number of constraints on the solution vector
K	a constant
K_B	Boltzmann's constant
L	set of uniform random weight vectors
L_∞	Chebysev norm
l	number of equality constraint
m	no of inequality constraint
M_k	objective function at eqilibrium
N	number of objective functions

n	number of decision variables
$Prob$	probability
$S(A)$	spacing of a Pareto set A
$\Delta S'$	change in fitness value
Δs	one-dimensional space
T	annealing temperature (controlling parameter in the algorithm)
W	weight vector
X	current solution vector
Y	generated solution vector
Z	vector whose elements are objective function value ie z_i after applying the penalty function approach
Z_k	Chaotic variable

Greek symbols

α	a constant
β	a constant
δ	parameter for probability calculation
λ	weight vector
σ^2	variance of the objective function at equilibrium
μ	bifurcation parameter of the system
χ_0	defined as the number of accepted bad moves divided by the number of attempted bad moves

Received December 2004;
accepted July 2005 after two revisions