Journal of Global Optimization **19:** 27–50, 2001.

Parallel Simulated Annealing Algorithms in Global Optimization

ESIN ONBAŞOĞLU[1] and LINET ÖZDAMAR[2]
[1]*Yeditepe University, Department of Computer Engineering;* [2]*Yeditepe University, Department of Systems Engineering, Gayrettepe Emekli Subay Evleri 23/5, Istanbul, Turkey (e-mail: lozdamar@hotmail.com)*

(Received 17 August 1999; accepted in revised form 21 June 2000)

Abstract. Global optimization involves the difficult task of the identification of global extremities of mathematical functions. Such problems are often encountered in practice in various fields, e.g., molecular biology, physics, industrial chemistry. In this work, we develop five different parallel Simulated Annealing (SA) algorithms and compare them on an extensive test bed used previously for the assessment of various solution approaches in global optimization. The parallel SA algorithms consist of various categories: the asynchronous approach where no information is exchanged among parallel runs and the synchronous approaches where solutions are exchanged using genetic operators, or where solutions are transmitted only occasionally, or where highly coupled synchronization is achieved at every iteration. One of these approaches, which occasionally applies partial information exchanges (controlled in terms of solution quality), provides particularly notable results for functions with vast search spaces of up to 400 dimensions. Previous attempts with other approaches, such as sequential SA, adaptive partitioning algorithms and clustering algorithms, to identify the global optima of these functions have failed without exception.

Key words: Global optimization, Parallel simulated annealing

1. Introduction

We consider the optimization problem min f(x) subject to only lower and upper bound constraints on the variables, namely, $x \in \alpha_0 \subset \mathbb{R}^n$, where α_0 is a hypercube defined by boundaries $lb_j \leq x_j \leq ub_j$, $j = 1...n$, and lb_j, ub_j are lower and upper bounds on variable x_j. The problems of locating the global optima of such functions arise in diverse disciplines such as physics, chemistry and molecular biology.

Here, we do not aim to provide an exhaustive literature survey of previously developed approaches, but classify solution approaches under the following categories: probabilistic search methods, and deterministic methods including adaptive partitioning algorithms (interval estimation methods).

Probabilistic search methods such as random search integrated with local search, e.g., Density Clustering (Törn, 1978), Controlled Random Search (Price, 1978), Multistart and Multi Level Single Linkage (Rinnooy Kan and Timmer, 1984) are proposed for global optimization. In such methods, α_0 is subjected to uniform

random search and then, a number of steepest descent iterations are applied locally to every selected seed. In other approaches, promising seeds subjected to local search may be selected according to a geometric criterion incorporating topological knowledge gained from the functional values obtained in uniform Random Search (Törn and Viitanen, 1994; Ali and Storey, 1994). Furthermore, metaheuristics such as simulated annealing (SA) (Kirkpatrick et al., 1983), and genetic algorithms (GA) (Michalewicz, 1994) are also among probabilistic solution methods.

Adaptive partitioning algorithms partition and re-partition the search space into smaller sections with the aim of locating a small interval where the global optimum lies (e.g., Pinter, 1992; Horst and Tuy, 1996). In interval estimation approaches based on a model supposition of $f(x)$ (e.g., Wingo, 1983; Zilinskas, 1981), it is assumed that $f(x)$ belongs to a prespecified class of functions such as Lipshitzian functions or it is assumed that $f(x)$ is a realization of a stochastic process (e.g., Wiener (for unary $f(x)$), or Gaussian). The prior model of $f(x)$ is integrated with the information gained from evaluating the functional values of points at given locations such as the boundaries or the diagonal of the sub-region. Using this information an adaptive posterior model of $f(x)$ is obtained in each iteration and sub-regions are evaluated accordingly. In the interval estimation approach based on inclusion functions (Moore, 1979; Moore and Ratschek, 1988), the bounds on $f(x)$ within a specific interval are calculated by replacing the real operations in $f(x)$ by their pre-determined interval operations resulting in an interval on $f(x)$ for each operation. Then, when all the terms are aggregated, an enclosure of $f(x)$ is obtained. Thus, each partition is evaluated with the aid of an inclusion function defined on $f(x)$ and only the boundaries on each variable need to be known in every interval. Naturally, the strength of the bounds on $f(x)$ depends on the inclusion function as well as the behaviour of $f(x)$ within the interval.

An adaptive partitioning algorithm developed by Demirhan et al. (1999) combines the deterministic approach with the probabilistic one. Namely, in each partition samples are collected randomly and the evaluation of a partition is conducted by using a fuzzy measure which aggregates sample values.

2. The Goal of This Study

Özdamar and Demirhan (2000a, b) provide extensive computational surveys reflecting the performance of both types of approaches (deterministic adaptive partitioning approaches and probabilistic approaches including many SA versions and clustering methods) where a large number of test functions are used. The authors reach the following empirical conclusion: SA (SA with local search (Özdamar and Demirhan, 2000a), and Adaptive SA (ASA, Ingber, 1995)) and the fuzzy adaptive partitioning scheme (Özdamar and Demirhan, 2000b) are best performing ones among the tested algorithms. However, when the number of variables increase (above ten), the performance of both ASA and the fuzzy partitioning scheme deteri-

orate considerably and the SA with local search scheme becomes best performing. Yet, the results of the best approach is far from satisfactory.

Thus, in this paper, in order to achieve better solution quality, we carry out further enhancements on SA by different parallelization schemes. In the following sections, we discuss several parallelization approaches proposed in the literature, and then, briefly describe the parallel algorithms developed here for implementation.

3. Sequential and Parallel Simulated Annealing

The performance of SA depends on the number of variables of the function under investigation, because, as a single point search technique, SA converges rather slowly in order to provide sufficient moves carried out in every direction (variable). A convergence proof for SA in the real domain is provided by Dekkers and Aarts (1991). Various SA implementations exist in the literature (e.g., Corana et al., 1987; Ingberg, 1994; Zacharias et al., 1998). The strong points of SA and some pitfalls for potential SA users are indicated in an extensive review given by Ingber (1994) where a wide range of application areas from finance to combat analysis are described.

SA is a single point stochastic search technique where in each iteration, a neighbour is created from a current solution by changing a part of the solution and if the neighbour's objective function value is better than that of the existing solution, the neighbour replaces the current solution. Else, the neighbour is accepted probabilistically. The probability of acceptance depends on the current annealing temperature which is controlled by the cooling scheme.

The quality of SA's performance is usually affected by the configuration of the starting solution. Researchers often compensate for the latter deficiency by rerunning SA with a new seed each time and reporting the best solution achieved among all processes. Hence, the diversity of the search space is achieved to some degree. Since each SA process (starting from a possibly different solution) is executed independent of the other SA processes, this approach can be implemented on parallel processors in a completely asynchronous (no communication between processors) manner.

Some researchers develop hybrid meta-heuristics which result in a natural diversification of SA. For instance, Kurbel et al. (1998) discuss the parallel recombinative SA (a GA/SA hybrid) where mutated chromosomes are accepted into the population by Boltzman annealing. Some others employ SA as a pre-optimizer or post-optimizer of some chromosomes in GA's population (e.g., Lin et al., 1991; Özdamar and Birbil, 1998).

Parallelization of SA as a multi-start search technique is discussed in Frost et al. (1993) and also in Ingber (1994). Due to the fact that SA is a naturally sequential algorithm, it is difficult to parallelize SA without changing its serial nature (Chen et al., 1998). One method to classify parallel SA implementations is to distinguish

between Single Markov Chain and Multiple Markov Chain implementations (Lee and Lee, 1996). In the literature, numerous methods of parallelization exist, ranging from asynchronous to various categories of synchronous (e.g., Aarts et al., 1986; Casotto et al., 1987; Kravitz and Rutenbar, 1987; Greening, 1990; Aarts and Korst, 1991; Balardi and Orlando, 1989; Wendt and König, 1998; Lee and Lee, 1996).

Kliewer and Tschöke (1998) discuss in detail the synchronization alternatives in parallel SA. First, parallel approaches are classified under the two following categories:

i. Application-dependent parallelization where a problem instance is divided among several processors which communicate only when dependencies make it necessary;
ii. Problem independent parallelization whose sub-categories are the following:
a. asynchronous parallelization where independent SA chains (workers) with differing starting solutions are generated and run for a given number of iterations, and the best solution among them is reported,
b. synchronous parallelization where only cost values are transmitted among workers,
c. synchronous parallelization based on the occasional exchange of solutions among workers,
d. clustered parallelization based on the intensive exchange of solutions,
e. highly coupled synchronization where each worker generates and evaluates a neighbour solution.
Here, a worker is defined generically as a SA chain or a SA procedure. Hence, there could be several workers on one processor.

Kliewer and Tschöke (1998) develop a hybrid algorithm based on ii.a, ii.d and ii.e where each worker starts making moves independently on its own configuration, and when the acceptance ratio becomes low, workers are grouped into a cluster, working together in the search of a neighbour on a common SA chain. Thus, algorithms of categories ii.a, ii.c, ii.d define Multiple Markov Chains SA, because they enable independent sequences of moves on each worker (SA chain) during the intervals between consecutive communications. On the other hand, category ii.e defines a Single Markov Chain SA, since a single move is selected from a number of moves (carried out by each worker on the same SA chain) and it is then transmitted to the workers as the global configuration. Similarly, algorithms of category ii.b run a Single Markov Chain, since workers share the computational burden of cost calculations.

In the family of algorithms classified under category ii.c, several options are available. One option would be to conduct several SA chains independently, and after a number of successful moves are executed by each worker, to replace the configuration (solution) on each worker by the best solution among all chains. It is also possible to choose a configuration by Boltzmann annealing or randomly.

Then, independent execution of each SA chain starts again. Another option is to run independent SA chains, and then, after each worker reaches down to a given target temperature, one configuration is selected and replaces the one on every worker which starts to work independently again until the next target temperature is reached. In category ii.c, one can also devise an algorithm which exchanges parts of each worker's solution occasionally (similar to the crossover operator in Genetic Algorithms) after a number of independent moves made by each worker (Wendt and König, 1998).

The degree of coupling in the synchronization of parallel processes depends on the number of communications realized by the algorithms. For instance, in category ii.e all workers work on the same master configuration each in parallel by generating independently one move. One of the accepted moves (on a worker) replaces the master configuration and the process starts again (Diekmann et al., 1993). It is possible to refine the last algorithm by enabling each accepted move to modify the master configuration provided that their joint effects do not lead to a deterioration in the objective function value. Namely, perturbations which are beneficial to the objective on their own, become detrimental when applied to the solution jointly. For instance, an objective function value to be minimized decreases when a positive variable with a positive coefficient becomes negative. However, if the product of two positive variables exists in the objective, it is harmful to turn the values of both variables into negative numbers. The Highly Coupled Synchronous (HCS) algorithm proposed here is based on the latter idea.

4. Implementation of Parallel SA in Global Optimization

4.1. THE CORE SA APPROACH

The basic SA approach (Özdamar and Demirhan, 2000a) which is the core of all parallel implementations proposed here is briefly described as follows.

4.1.1. *Initial solution.* The initial solution is generated randomly by assigning a value to each dimension i, i=1...N, in the interval [lb_i,ub_i] where lb_i and ub_i are the lower and upper bounds for variable i, respectively.

4.1.2. *Neighbour generation scheme.* The neighbour solution, $\mathbf{x}^{k+1}$, to the current solution $\mathbf{x}^k$ in iteration k is created as follows. Select a random dimension i^* and compute the variable values (over all dimensions, i) of the neighbour by the following equations. If i=i^*, then

$$x_i^{k+1} = x_i^k + u(ub_i - x_i^k),\ \text{sgn}(u - 0.5) < 0,$$

and

$$x_i^{k+1} = x_i^k - u(x_i^k - lb_i),\ \text{sgn}(u - 0.5) \geq 0.$$

Else,

$$x_i^{k+1} = x_i^k, \text{ where } u \in U[0, 1].$$

Here, u is a random variable uniformly distributed between zero and one. In this scheme, a variable i^* is selected randomly and a decision is made arbitrarily if the value of the variable in the selected dimension is to be decreased or increased. Then, a random amount is added/subtracted from the current variable value without violating the corresponding upper and lower bound while the values of the remaining variables stay intact.

4.1.3. *Acceptance of a neighbour solution.* The new function value is calculated and the solution is accepted if the move improves the current solution. Else, a non-improving move is accepted according to the geometric cooling scheme (described as Simulated Quenching by Ingber, 1994) used with success previously in the global optimization context (Özdamar and Demirhan, 2000a):

$$PA(\Delta, t^k) = exp(-(\Delta/(f(\mathbf{x}^k)t^k)))$$

where, PA is the probability of acceptance, Δ is the difference between $f(\mathbf{x}^{k+1})$ and $f(\mathbf{x}^k)$, and t^k is the temperature in iteration k. If a randomly generated number between zero and one turns out to be less than PA, then the deteriorating move is carried out. t^k depends on the number of times a deteriorated solution has been accepted. Initially t^k is equal to one, but after each accepted non-improving move, it is reduced as follows.

$$t^{k+1} \leftarrow t^k/(1 + \beta t^k),$$

where β is a nonnegative small constant less than one. The aim in reducing the temperature at every accepted non-improving move is to direct the search away from unfavourable regions. Re-annealing takes place if t^k drops below 0.01 during the search and t^k is then re-assigned the value of one.

The number of iterations allowed (moves or perturbations), MaxMove, is proportional to the dimension N of the function. Here, we set MaxMove to 1000^*N.

4.2. PARALLEL IMPLEMENTATIONS

In this work, we adopt the problem-independent parallelization approach, because mathematical functions whose global optima are to be identified have no restrictions such as the separability of terms. Five parallel SA algorithms are developed here for distributed memory Multiple Instruction Multiple Data (MIMD) systems. In all algorithms, there are P workers, which are responsible for processing the iterations of SA chains, and a master process gathers the solutions from workers

to trigger the next round of parallel chains. The five approaches proposed here are described as follows.

4.2.1. *Asynchronous Approach (AS).* The asynchronous approach is of category ii.a described in the previous section. The master generates *P* independent SA chains which execute concurrently MaxMove perturbations of the core SA algorithm and reports the best solution obtained among all workers. This approach does not carry communications overhead except for identifying the best configuration when all SA chains terminate.

4.2.2. *Synchronous Approach with Occasional Solution Exchanges (SOS).* The second approach is of category ii.c where independent SA chains exchange solutions by occasional crossover operations similar to Genetic Algorithms (GAs). SOS is a hybrid SA/GA. The basic idea of SOS is suggested by Wendt and König (1998) among other authors who have proposed SA/GA hybrids for zero-one integer programming (e.g., Chen et al., 1998; Özdamar and Birbil, (1998). However, the SOS implemented here has its own particular properties: the temperature reduction policy is the one explained in the core SA algorithm; the probability of selecting a crossover mate is adapted from Özdamar (1999), and the crossover operator is developed here. The aim of SOS is to merge the best solutions obtained independently by each worker with those of other SA chains via a crossover operation. In this manner, parallel search efforts resulting in good quality solution segments are utilized more efficiently.

Initially, the user specifies the number of communications (NCom) to take place between the master and workers. The number of communications is counted by *s* where $1 \leq s \leq$ NCom.

Each worker generates its own initial solution and is allowed to run independently for MaxMove/NCom moves. However, the temperature reduction scheme is not realized by the workers. Rather, the master imposes a given temperature, t^s, which is updated only by the master after set of communications. (Initially, t^s is set to one.)

After MaxMove/NCom moves, each worker *p* reports the best solution $r_p{}^s$, and its value $f_p{}^s$. The sum of $f_p{}^s$ over p=1,...*P*, is stored as E^s. The master then carries out the following crossover operation among solutions $r_p{}^s$, p=1,...*P*. The offspring and the solutions which have not been replaced by the offspring constitute the new population.

The probability of each solution $r_p{}^s$ for being selected as a parent is calculated as $\mathrm{Min}^s/f_p{}^s$, where Min^s is the minimum function value over $f_p{}^s$, p=1,...*P*. Hence, a solution whose function value is closer to Min^s has a higher chance of being selected as a mate for solution $r_p{}^s$. Next, these selection probabilities are normalized over all *p*.

To each solution $r_p{}^s$, p=1,...*P*, the following crossover operator is applied if its selection probability permits it to be a parent. After crossover is realized, the

offspring replaces r_p^s to obtain r_p^{s+1}. First, a solution r_q^s ($q \neq p$) is selected randomly as a mate for crossover with r_p^s according to its own cumulative selection probability. The crossover operator then assigns a weight to solution r_p^s where $w_p^s = f_p^s/(f_p^s + f_q^s)$. Then, the value of one randomly selected variable i pertaining to solution r_p^s is assigned the following new value: $x_{pi}^{s+1} = (1 - w_p^s)\, x_{pi}^s + w_p^s\, x_{qi}^s$. Thus, a weighted combination of both solutions' corresponding variable values is assigned to variable i in solution r_p^{s+1}, the weights depending on the function values of each solution. The crossover operation results in a single offspring stored as r_p^{s+1}.

The sum of the new population's function values is stored as E^{s+1}. If after the crossover operations, E^{s+1} is less than E^s, then the temperature t^{s+1}, is not reduced, because the population performance has improved. Otherwise, t^{s+1} is reduced to $t^{s+1} \leftarrow t^s / (1+\beta\, E^{s+1}/ E^s)$. Finally, the master redistributes the new population, r_p^{s+1}, p=1,...P, to workers to execute another MaxMove/Ncom moves with the updated temperature t^{s+1}. The communication index s is then incremented by one. This procedure is repeated until each worker executes MaxMove moves in NCom communication sets.

Hence, *SOS* is a SA/GA hybrid where partial exchanges of solutions take place among independent SA threads realized concurrently. Since the crossover operator favors good quality solutions, the overall population's solution quality is likely to improve after MaxMove/NCom moves. It should be noted that unlike *AS* where each worker controls its own cooling rate, in *SOS*, the global temperature is updated according to the overall solution quality of the population.

4.2.3. *Synchronous Approach with Occasional Enforcement of Best Solution-Fixed Intervals (SOEB-F).* This approach is also of category ii.c. Similar to SOS, the user specifies the number of communications (NCom) that are to take place among parallel SA chains and the master. In *SOEB-F*, the master generates P random initial solutions, selects the best one and sends it to each worker as the starting solution. Then, each worker is allowed to run independently for a fixed number (MaxMove/NCom) of moves. After MaxMove/NCom moves, the workers report their corresponding f_p^s , p=1,...P, to the master who selects Min^s among them. The master then retrieves the solution whose f_p^s is equal to Min^s, and sends $r_{p^*}^s$ (p^* is the worker which reports Min^s) to the other workers as the new starting solution.

The temperature t^s is also controlled by the master and communicated along with the best solution. Initially, t^s is set to one. Then, after MaxMove/NCom moves, the master checks the values of Min^s and Min^{s-1}. If Min^s is greater than Min^{s-1}, t^s is reduced by $t^{s+1} \leftarrow t^s / (1+\beta\, t^s)$. Otherwise, t^{s+1} assumes the value of t^s. The above scheme of imposing the best solution on the workers and adjusting the global annealing temperature is repeated NCom times.

The communication policy of *SOEB-F* is described as 'periodic exchange scheme' by Lee and Lee (1996). The Markov Chains between two successive communications are called segments. Since the best global solution becomes the

seed after every communication, the multiple Markov Chains are not completely independent. When viewed in this respect, SOS is also a periodic exchange scheme.

4.2.4. *Synchronous Approach with Occasional Enforcement of Best Solution-Varying Intervals (SOEB-V).* Similar to SOEB-F, in SOEB-V (also of category ii.c), the master generates P random initial solutions, selects the best one and sends it to each worker as the starting solution. However, the user does not specify NCom. Rather, the master sends each worker, the current temperature, t^s and the target temperature, Tar^s. Tar^s is set to t^s minus ϵ, at the beginning of each set of communications. ϵ is a user-specified small real number larger than or equal to β and less than one. In order to prevent negative temperatures, the master resets t^s to one as soon as it becomes less than ϵ. Each time there is communication, besides t^s and Tar^s, workers also receive $r_{p^*}{}^s$ from the master. Starting from $r_{p^*}{}^s$ each worker conducts an independent search until t^s is reduced to Tar^s according to the cooling policy described in the core SA algorithm. Thus, as the value of ϵ increases, the number of moves carried out during independent search increases and the number of communications decreases. However, the number of moves carried out by each worker is not constant, but variable depending on the successive states visited by the Markov Chain.

SOEB-V's communication structure is called the "dynamic exchange scheme" in Lee and Lee (1996). In other words, in order to enable communication, the master waits for each worker to reach an equivalently favourable state in terms of the convergence of its Markov Chain. Hence, independent Markov Chains may guide each other more efficiently.

4.2.5. *Highly Coupled Synchronous Approach (HCS).* HCS is an approach of category ii.e. A single Markov Chain is followed by every worker. The algorithm is briefly described as follows.

The master generates a random initial solution and distributes it among the workers. The master also sends to each worker the specific variable, *d*, on which the worker should carry out a move. (*d* is selected randomly for each worker *p*.) The master stores the global solution as R^s. (Initially, the counter *s* is set to one.) Each worker *p* performs one perturbation on the value of variable *d*, $x_{pd}{}^s$, in solution R^s, and evaluates the move based on t^s which is also sent by the master to the worker along with R^s. The master then receives the changed or unchanged $x_{pd}{}^s$ and $f_p{}^s$, *p*=1,...*P*, from the workers and sorts the moves in ascending order of $f_p{}^s$. The move $x_{p*,d}{}^s$ resulting in Min^s is accepted and R^s is updated, replacing the value of variable d by $x_{p*,d}{}^s$ to result in R^{s+1}. Next, the master re-configures R^{s+1} by trying the next move in the sorted list. The variable's value corresponding to the variable of the next sorted move is now replaced. Taking into consideration the joint effect of the two moves, the master re-calculates the new $f_R{}^{s+1}$ and if the new value is better than the previous value, the next move is accepted. This process of replacing the values of variables as indicated in the sorted array and

re-calculating the corresponding $f_R{}^{s+1}$ goes on until all moves achieved by the workers are tested consecutively, taking into account their joint effects. The final configuration provides R^{s+1}. At that point, the master adjusts t^{s+1}after comparing $f_R{}^{s+1}$ with $f_R{}^s$ according to the cooling strategy described in the core SA algorithm. Next, R^{s+1} is re-distributed among workers for the next round of moves and the index *s* is incremented by one.

Thus, in HCS a perturbation improving $f_R{}^s$ updates R^s after having been first screened by a worker. Although the screening carried out by the worker involves Boltzman acceptance rule, that of the master is purely hill-climbing.

Obviously, unlike SOEB-F and SOEB-V, the master has a larger CPU load due to sorting and *P* times re-evaluating $f_R{}^s$. Yet, the communication overhead is more essential. In HCS, the master conducts a very tight surveillance on the search process. Each move invokes a set of communications resulting in a heavy communication overhead. Due to this reason it may be desirable to reduce the number of communications by using the periodic exchange scheme. Namely, the master's dominant control over the workers is relaxed by letting each worker execute its own moves starting from R^s at the temperature t^s which is also received from the master. Similar to *SOEB-F*, the master communicates with the workers at every (MaxMove/NCom) moves. However, unlike *SOEB-F*, the worker is allowed to perform MaxMove/NCom moves only on the selected variable, *d*, received from the master. Then, the master receives $x_{pd}{}^s$ and the corresponding $f_p{}^s$ from every worker and starts the hill-climbing procedure to decide on R^{s+1}. We denote this procedure the Modified HCS (MHCS). Thus, MHCS becomes a more efficient approach (a combination of categories ii.c and ii.e) where search is conducted in parallel by the workers along (possibly) different directions in space. Namely, the SA chain is split into parallel chains in a relatively more controlled manner than *SOEB-F*, since moves are executed along pre-determined paths. The end products of these paths (the best solution reached) are then collected back and build up the solution of the common SA chain by implementing the hill-climbing procedure. The temperature reduction scheme controlled by the master is preserved as applied in the original *HCS* version. However, temperature is reduced after MaxMove/NCom moves at the earliest.

5. Computational Experiments with Parallel SA

The performance of the parallel approaches described here is investigated on 77 test functions with up to 10 variables and 29 test functions up to 400 variables. The first set of 77 problems and 19 test functions of the second set are also used by Özdamar and Demirhan (2000a). Thus, here, the second set of problems is extended by 10 test functions. This set of 29 functions are composed of 11 different test functions with variants of increasing numbers of variables. (Performance with respect to solution quality deteriorates as the number of variables increase, because

the search space expands.) We cite both the first and the second set of test problems in the Appendix.

At this point, several remarks about the difficulty of the test functions should be made. In a recent article by Törn et al. (1999), test problems are assigned several complexity grades and they are classified as unimodal (U), easy (E1, E2), moderate (M1, M2), difficult (D1, D2). This classification is based on properties such as number of local minimizers, embedded or isolated global minimum and the size of the region of attraction of the global minimum. For instance, E1 and E2 (and similarly, M1/M2 and D1/D2) are differentiated by the number of local minima. M1/M2 type problems have embedded (local optima are near the global optimum) global optima whereas D1/D2 type problems have isolated global optima which are harder to detect. In both M1/M2 and D1/D2 types the region of attraction is small.

In the test problems provided in the Appendix, almost all types of problems exist. One can deduce the complexity of a test function by its properties indicated in the Appendix. For instance, in the first set of problems, the initial 7 problems are multimodal but well-behaved functions (E1/E2 types). The next 6 functions are of moderate complexity, e.g., the region of attraction in the 13th function is very small but embedded among a very large number of local minima. It is not possible to provide graphical illustrations of all these functions here, but they are found in their respective references. Similarly, the functions 20–22 are of D2 type (isolated global optimum in a small interval and local minima far from it) whereas the 23rd is a convex function which is solvable using gradient information. The 24th test function is of D1 type. The 28th function (Griewank) has M1 type complexity. Most of the Levy problems fall into the category E2. As indicated in Törn et al. (1999) popular test functions such as Branin, Shekel5, etc. fall into the E1 category. The 63rd function is of D2 category due to the abundant number of local optima and an isolated global minimum. The 75th test function is of D1 category due the singular global optimum located in a very small interval (no local optima exist). As for the second set of functions the first four are of easy type, the fifth one (the 13th in the first set) has increased M2 complexity, the Powersum (22nd in the first set) and Schewel's sine root functions (24th in the first set) have increased D2/D1 complexity due to the number of variables involved. The Griewank function has increased M2 complexity. Ackley's function is of easy type. Consequently, the collection of test functions provided in the Appendix represents a group composed of all complexity categories described in the recent work of Törn et al. (1999).

In Table 1, we report the results provided by Özdamar and Demirhan (2000a) both for the first and second set of test functions. The sequential algorithms that are reported as best performing by the authors are SA-III and ASA. SA-III is basically the same algorithm as the Asynchronous Approach (AS) described above. The main difference of SA-III is an occasionally implemented local search procedure which is activated after a given number of SA moves. The local search procedure starts searching the neighbourhood of the current solution within a given distance

Table I. Results by SA-III and ASA (Run times are measured on a Pentium II 350 MHz PC with LINUX operating system). Relative deviations should be multiplied by 100

First Set of Functions (77 functions)	Absolute Deviation (46 test functions)		Relative Deviation (31 test functions)	
Method	SA-III	ASA	SA-III	ASA
Average	0.53	0.62	0.33	0.24
Standard Deviation	1.84	1.95	1.03	0.37
Average Run Time (20 runs)	2.11	3.30	1.20	1.93
Second Set of Functions (29 functions)	(25 test functions)		(4 test functions)	
Method	SA-III	ASA	SA-III	ASA
Average	14523.8	16985.28	6759.54	5754.40
Standard Deviation	34855.6	36042.18	15113.03	12865.72
Average Run Time(20 runs)	712.79	654.8	1811.6	1813.6

in the feasible space and always accepts solutions superior to the incumbent one. The second method, namely, the Adaptive Simulated Annealing procedure (ASA) is described in Ingber (1995) and its major contribution is the formula used for generating neighbours. A neighbour to the current reference solution is generated via an expression which depends on the temperature corresponding to the considered variable on which the move is to be carried out. A move is considered completed after a perturbation is carried out in every direction.

In Table 1 we provide the summary of the results obtained by SA-III and ASA which are the best reported performing methods in both sets of test functions. SA-III and ASA are re-run including the extension of 10 test functions introduced here. For each test function 20 runs are taken independently and the best result obtained is stored. In Table 1, we report the average absolute deviation of these best results from the global optimum when the global optimum is in the interval [–1,1], and the relative deviation of the best results from the global optimum, i.e., (best-opt)/abs(opt), when the absolute global optimum is greater than one. 46 test functions are compared in terms of absolute deviation and 31 test functions are compared in terms of relative deviation, making up the 77 test functions of the first set. In the second set, 25 functions and 4 functions are compared in terms of absolute and relative deviation, respectively. Standard deviations are also provided. All results in Table 1 are obtained by the corresponding sequential algorithms which carry out 20 independent runs (asynchronously) each executing MaxMove=1000$*$N moves (starting from a random solution).

All parallel SA algorithms are developed for Distributed Memory (DM) MIMD systems and the following hardware and software are used: 8 Pentium II 350 MHz PCs connected by 10Mbit/second Ethernet. The parallel programs are written in C and run using Parallel Virtual Machine (PVM) programming environment (LINUX

Table II. Results obtained by parallel SA implementations. (Run Times are in secs.).

Relative Deviation	(31 test functions)											
Method	HCS	MHCS	MHCS	AS	SOEB-F	SOEB-F	SOEB-F	SOEB-V	SOEB-V	SOS	SOS	SOS
First Set of Functions	Every move	NCom=100	NCom=300		NCom=5	NCom=10	NCom=20	Eps=0.05	Eps=0.10	NCom=5	NCom=10	NCom=20
Average	0.22	0.51	0.24	4.09	0.62	0.56	0.37	0.22	0.38	1.58	0.76	0.64
Standard Deviation	0.38	1.64	0.40	18.58	1.80	1.97	0.99	0.41	1.05	6.54	2.25	1.75
Average Parallel Run Time	69.90	1.90	7.33	0.33	0.47	0.53	0.63	1.40	1.43	2.47	2.41	2.44
Speedup Factor	0.09	1.29	0.31	4.03	3.47	3.26	2.79	1.74	1.91	0.69	0.69	0.70
Second Set of Functions	(4 test functions)											
Average	0.26	1.10	1.03	60.17	2580.13	1941.65	2037.30	1917.83	1746.31	2807.85	1421.45	1.63
Standard Deviation	0.43	1.12	1.23	132.99	5767.68	4339.99	4553.87	4286.81	3903.25	6276.86	3176.79	2.00
Average Parallel Run Time	19750.90	336.80	342.80	252.80	251.80	252.00	252.20	426.17	426.10	247.40	248.20	471.02
Speedup Factor	0.37	4.79	4.69	6.86	6.39	6.38	6.60	3.86	3.87	6.50	6.48	3.41
Absolute Deviation	(46 test functions)											
First Set of Functions												
Average	0.25	0.13	0.20	2.31	34.90	28.53	24.01	0.45	11.46	35.67	33.68	31.03
Standard Deviation	0.90	0.48	0.61	12.29	178.86	175.52	145.00	2.21	72.49	220.26	207.19	193.12
Average Parallel Run Time	98.61	2.51	7.02	0.42	0.51	0.57	0.76	5.36	5.12	0.57	0.57	0.76
Speedup Factor	0.08	1.20	0.44	5.50	5.41	4.74	3.63	2.55	2.49	4.65	4.70	3.66
Second Set of Functions	(25 test functions)											
Average	0.02	9274.55	0.42	13104.12	14708.37	14606.67	14696.27	12144.22	12496.62	14820.69	14568.06	14833.71
Standard Deviation	0.03	26676.53	1.00	32887.01	33443.22	33082.06	33481.06	28251.01	28495.36	33714.15	33028.26	33818.93
Average Parallel Run Time	31706.67	157.67	164.25	117.42	116.33	116.54	117.04	933.42	948.00	113.75	114.00	114.42
Speedup Factor	0.05	4.86	4.62	1.00	6.50	6.49	6.62	0.98	0.98	6.62	6.62	6.59

version). We would like to mention that these algorithms could also be adapted to Shared Memory (SM) MIMD systems.

The following parameters are used in the implementation of the parallel procedures. In all of the SA implementations devised here, β is assigned a value of 0.1 after preliminary experimentation, except for SOEB-V where β is assigned a value of 0.05. All approaches are run using 20 workers each carrying out MaxMove (1000$*$N) moves. In the implementation of SOS and SOEB-F, NCom is set to 5, 10 and 20 and results are compared. For SOEB-V, ϵ is set to 0.05 and 0.1. HCS is implemented in three modes: communication takes place after each parallel move (HCS), it takes place 100 times (MHCS, NCom=100) and 300 times (MHCS, NCom=300). Note that we need a much larger number of communications with MHCS as compared to SOS and SOEB-F, because in MHCS each worker works on only one variable whereas in the latter two approaches, search is carried out in parallel through the whole domain.

We evaluate the performance of parallel SA approaches in terms of the quality of their solutions and their computational efficiency. The algorithms are first run sequentially and then, in parallel. In Table 2, the results of the parallel SA schemes are reported in terms of absolute and relative deviations. The average run times (in seconds) related to parallel applications and the speedup factor obtained by parallelization (Sequential run time/Parallel run time) are provided in Table 2.

In Table 2, we observe that when absolute deviation is the performance criterion in the first set of test functions, there is a significant difference among the results provided by MHCS (for all NCom values) and those of the sequential SA procedures developed previously (SA-III, ASA) as well as the remaining parallel algorithms. However, when the criterion of relative deviation is considered, ASA's performance is very close to those of HCS and MHCS (NCom=300) and to that of SOEB-V (Eps=0.05). Thus, the performance MHCS is best for the criterion of absolute deviation as compared to those parallel and sequential algorithms. Yet, the results obtained by MHCS seem somewhat surprising (for the criterion of absolute deviation, 46 test functions) in terms of the values of the parameter Ncom, since less frequent data transmission produces better results.However, the latter may be due to the effect of randomization. Such a contradiction is not observed in the remaining sets of test functions.

Performance in the second set of functions is crucial for all approaches, because in the experimentation conducted previously, even the best performing sequential methods are far from efficient both in terms of solution quality and computation time. Consequently, we now discuss performance on the second set of problems. In this set, all SA approaches stumble at Schwefel's Sine Root function with the exception of HCS and MHCS (NCom=300). The reason is that a local minimum which is far away from the global exists in this function and it traps the search. Nevertheless, the latter function is far from being an outlier, because performance in other test functions is also extremely inferior as compared to those of HCS and MHCS (NCom=300).

The best performing (in terms of solution quality) parallel SA approach is the HCS. HCS (where communication takes place at every move) provides outstanding results; yet, its sequential and parallel run times are excessive as compared to the other approaches. HCS results in speedup factors which are less than one due to the communication overhead. However, in the two MHCS implementations (NCom=100 and NCom=300), although we observe a deterioration in performance as compared to HCS, the quality of results are still highly superior when compared with those of other parallel and sequential approaches, including SA-III and ASA. An exception occurs when NCom=100 in the second set of functions where the performance criterion is the absolute deviation from the optimum. The number of communications in this case is far from sufficient. For the same set of functions, when NCom=300, we obtain exceptional results with computation times and speedup factors which are compatible with other approaches. A remark on MHCS is that its parallelized modes with 100 and 300 communications result in almost no speedup in the first set of test functions and almost five times speedup in the second set. The reason for the latter is that in test functions with less than 10 variables the CPU times of the workers are negligible as compared to the time required for communications. The situation becomes vice versa in the second set of functions with many variables, because the major load of the CPU consists of evaluating the functional value of the considered test function after each perturbation. Consequently, when MHCS is utilized, different values of NCom should be attempted, but the general tendency is to increase NCom since parallel computation times result in an insignificant increase specifically in solving test functions with many variables.

Again in the second set of test functions, among the approaches other than HCS and MHCS, SOEB-V and AS produce better results with respect to solution quality, but the differences are not statistically significant from other parallel SA approaches when the criterion is the absolute deviation (25 test functions). An exception occurs when the relative deviation criterion is considered in the second set of test functions (4 test functions); in that case, SOS provides relatively superior results as compared to other function sets.

At this point, one may deduce the reasons why SOEB-F, SOEB-V and SOS fail to produce sufficiently good quality results. The only difference between SOEB-F and AS is the occasional enforcement of the best solution as the seed for all parallel SA chains. Since the performance of AS is superior to that of SOEB-F, it seems that imposing the same seed with no regard to individual worker annealing temperature leads the search to unfavorable directions. At least in SOEB-V, the best solution is imposed when all Markov Chains reach the same medium of equilibrium (annealing temperature). The performance of SOS is a disappointment in our expectations of achieving a good rate of diversity and interaction through crossover. This operation also seems to disrupt the state arrived by each worker after so much effort.

As for the average parallel run times, in all approaches, as the number of communications increase, the parallel run time increases slightly. However, in the first set of functions, the number of communications affect run times more significantly as compared to the second set of functions due to reduced CPU time/Communication time ratio. For instance, the average parallel run time for AS is lower than that of SOEB-F in the first set of functions, but not in the second set. Similar observations can be made on SOEB-F, SOS and MHCS run times with different NCom values. It is sufficient to compute the increase ratio for sequential and parallel run times as NCom increases to deduce this fact.

As mentioned previously, HCS slows down considerably when parallelized (this is also true in the second set of problems where the number of moves and hence, communications, increase significantly). On the other hand, MHCS results in parallel run times which are compatible with other methods. In general, AS, SOEB-F and SOS result in very similar parallel run times when the second set of functions are considered. However, SOEB-V has the slowest parallel run times (except for HCS), because some workers are idle while one or more workers struggle towards the specified Tar^s.

6. Conclusion

We implement parallel SA procedures and evaluate them on large scale test functions with the aim of locating the global optima within efficient computation times. One such procedure, the Modified Highly Coupled Synchronous SA, (MHCS) which is based on checking the joint effects of different solution configurations achieved by parallel SA chains, outperforms previously introduced sequential SA algorithms as well as the other parallel SA implementations described here. The new MHCS conducts the search in parallel along possibly different directions in space and performs moves according to probabilistic acceptance rules. Communications with the workers take place at fixed intervals and at these times the master re-collects each incumbent variable value (pertaining to the workers) according to a hill-climbing procedure taking into account their joint effects on the function value. Extension of MHCS may involve features such as adaptive temperature reduction schemes and combinatorial treatment of joint effects.

7. Appendix (First Set of Test Functions)

No	Name	Source / Property	Reported by	No. of Vars.	Upper Bound, Lower Bound	Global Optimum
1	Complex	Press et al. (1992) (multi-modal, well-behaved)	Androulakis and Vrahatis (1996)	2	(–2, 2)	0
2	Stenger	Stenger (1975) (properties same as above)	"	2	(–1, 4)	0

No	Name	Source / Property	Reported by	No. of Vars.	Upper Bound, Lower Bound	Global Optimum
3	Himmelblau	Botsaris (1978) (properties same as above)	"	2	(–6, 6)	0
4	Helical Valley	More et al. (1981) (properties same as above)	"	3	(–2, 2)	0
5	Brown almost linear-3	More et al. (1981) (properties same as above)	"	3	(–2, 4)	1
. 6	Brown almost linear-4	More et al. (1981) (properties same as above)	"	4	(–2, 4)	1
7	Extended Kearfott	Kearfott (1979) (properties same as above)	"	4	(–3, 4)	0
8	Sine Envelope	Schaffer (1989) Increasing barrier between minima, multimodal, symmetric	Srinivas and Patnaik (1994)	2	(–100, 100)	0
9	Sine Envelope –	Srinivas, Patnaik (1994) Decreasing barrier between minima, multimodal	"	2	(–100, 100)	0
10	Epistacity- *k*=5	Srinivas, Patnaik (1994) Multimodal, 2^5 local optima	"	2	(-0.5, 0.5)	0
11	"	Srinivas, Patnaik (1994) 4^5 local optima	"	4	(–0.5, 0.5)	0
12	"	Srinivas, Patnaik (1994) 5^5 local optima	"	5	(–0.5, 0.5)	0
13	Spike	Michalewicz (1994) Innumerable local optima with a slight increasing tendency towards the global maximum located in small interval	Michalewicz (1994)	2	X_0:–3, 12.1 X_1: 4.1, 5.8	38.85 (max)
14	Sphere	http://iridia.ulb.ac.be/largerman/ICEO.html Unimodal	Bilchev and Parmee (1996)	5	(–5.12, 5.12)	0
15	Griewank	http://iridia.ulb.ac.be/langerman/ICEO.html	"	5	(–600, 600)	0
16	Shekel's Foxholes	http://iridia.ulb.ac.be/langerman/ICEO.html Very many local optima, *m*=30	"	5	(0,10)	–10, 40
17	Michalewicz	http://iridia.ulb.ac.be/langerman/ICEO.html	"	5	$(0, \pi)$	–4.687
18	Langerma	http://iridia.ulb.ac.be/langerman/ICEO.html valley between three local optima	"	5	(0, 10)	–1.5
19	Sphere	Same as 14	"	3	(–5.12, 5.12)	0
20	Permutation Beta = 0.005	Neumaier web site Many local minima, very hard problem , small discrepancy in variable values lead to large difference in function value	Neumaier solon.cma.univie.ac .at/~neum/glopt/my -problems	4	(–4, 4)	0
21	Permutation0 Beta = 100	"	"	10	(–1, 1)	0
22	Powersum	Neumaier web site Singular minimum among very flat valleys	"	8	(0, 2)	0

No	Name	Source / Property	Reported by	No. of Vars.	Upper Bound, Lower Bound	Global Optimum
23	Trid	Neumaier web site Convex, quadratic, tridiagonal hessian, best solved by gradient	"	10	(–100, 100)	–210
24	Schwefel's Sine Root	web site MATLAB / TEST/Lazauskas second local minimum which is very far from global minimum traps algorithms	web site MATLAB TEST/Lazauskas	10	(–500, 500)	0
25	Rastrigin	web site MATLAB / TEST/Lazauskas Highly multimodal and difficult	"	10	(–5, 12, 5, 12)	0
26	Rosenbrock	Rosenbrock (1970) Long curved only slightly decreasing valley, unimodal.	"	4	(–2.048, 2.048)	0
27	Ackley	web site MATLAB / TEST/Lazauskas	"	10	(–30, 30)	0
28	Griewank	http://iridia.ulb.ac.be/langerman/ICEO.html 4 local minima	http://iridia.ulb.ac.be/langerman/ICEO.html	10	(−600, 600)	0
29	Three Hump Camel Back	Levy et al. (1981)	Jansson and Knüppel (1994)	2	(–2, 4)	0
30	Levy 1	Levy et al. (1981) 3 local minima	"	1	(–4, 4)	7
31	Levy 2	Levy et al. (1981) 19 local minima	"	1	(–10, 10)	–14.508
32	Levy 3	Levy et al. (1981) 760 local minima	"	2	(–10, 10)	–176.541
33	Levy 5	Levy et al. (1981) 760 local minima	"	2	(–10, 10)	–176.137
34	Levy 8	Levy et al. (1981) 125 local minima	"	3	(–10, 10)	0
35	Levy 9	Levy et al. (1981) 625 local minima	"	4	(–10, 10)	0
36	Levy 10	Levy et al. (1981) 10^5 local minima	"	5	(–10, 10)	0
37	Levy 11	Levy et al. (1981) 10^8 local minima	"	8	(–10, 10)	0
38	Levy 12	Levy et al. (1981) 10^{10} local minima	"	10	(–10, 10)	0
39	Levy 13	Levy et al. (1981) 900 local minima	"	2	(–10, 10)	0
40	Levy 14	Levy et al. (1981) 2700 local minima	"	3	(–10, 10)	0
41	Levy 15	Levy et al. (1981) 71000 local minima	"	4	(–10, 10)	0
42	Levy 16	Levy et al. (1981) 10^5 local minima	"	5	(–5, 5)	0
43	Levy 18	Levy et al. (1981) 10^8 local minima	"	7	(–5, 5)	0
44	Beale	Schwefel (1991)	"	2	(–4.5, 4.5)	0

No	Name	Source / Property	Reported by	No. of Vars.	Upper Bound, Lower Bound	Global Optimum
45	Schwefel 3.1	Schwefel (1991)	"	3	(–10, 10)	0
46	Perturbed Schewel	Schwefel (1991)	"	3	$(-10^{-6}, 10^{-6})$	0
47	Booth	Schwefel (1991)	"	2	(–5, 5)	0
48	Box 3D	Schwefel (1991)	"	3	(–10, 30)	0
49	Kowalik	Schwefel (1991)	"	4	(0, 0.42)	0.000307
50	Powell	Singular, hessian at origin	"	4	(–2, 4)	0
51	Matyas		"	2	(–30, 30)	0
52	Schewel 3.7	Schwefel (1991) Singular hessian at $x^* = 0$	"	5	(–0.005, 0.89)	0
53	Schewel 3.2	Schwefel (1991) Singular hessian at $\mathbf{x}^* = 0$	"	2	(–10, 10)	0
54	Branin	Törn and Zilinskas (1989) 3 local minima, slow gradient	Törn and Zilinskas (1989)	2	X_0: (–5, 10) X_1: (0, 15)	0.3978
55	Goldstein-Price	4 local minima	"	2	$(-2, 2)$	3
56	Hartman	4 local minima	"	3	(0,1)	–3.862782
57	Six Hump Camel Back Valley	Törn and Zilinskas (1989)	"	2	(–5, 5)	–1.031628
58	Rastrigin		"	2	(–1, 1)	–2
59	Shekel 5	Törn and Zilinskas (1989) 5 local minima	"	4	(0, 10)	–10.153
60	Shekel 7	Törn and Zilinskas (1989) 7 local minima	"	4	(0, 10)	–10.402
61	Shekel 10	Törn and Zilinskas (1989) 10 local minima	"	4	(0, 10)	–10.536
62	Hartman	Törn and Zilinskas (1989) 4 local minima	"	6	(0, 1)	–3.32236
63	Ingber's f0	Corana et al. (1987) 10^{20} local minima, gradient search is useless	Ingber (1995)	4	$(-1000, 1000)$	0
64	p3 – (2D Schubert)	Alluffi, Pentini (1985) 760 local, 18 global minima	Dekker and Aarts	2	(–10, 10)	–186.7309
65	p8	Alluffi, Pentini (1985) 5^3 local minima	"	3	(–10, 10)	0
66	p16	Alluffi, Pentini (1985) 15^5 local minima	"	5	(–5, 5)	0
67	p22	Alluffi, Pentini (1985) Origin is a local minima	"	2	(–20, 20)	–24775
68	WingoA	Wingo (1983)	Breiman and Cutler (1993)	1	(3, 17)	–28.1617*
69	WingoB	"	"	1	(2, 26)	–73.452*
70	WingoC	"	"	1	(4.1, 2746)	–407.61*
71	Exp 2	Breiman, Cutler (1993)	"	2	(–1, 1)	0.3678*

No	Name	Source / Property	Reported by	No. of Vars.	Upper Bound, Lower Bound	Global Optimum
72	Exp 4	"	"	4	(–1, 1)	0.1353*
73	Cos 2	"	"	2	(–1, 1)	–2.2*
74	Cos 4	"	"	4	(–1, 1)	–4.4*
75	Easom	Easom (1990) A singleton maximum in a horizontal valley	Stuckman and Easom (1992)	2	(–100, 100)	–1
76	De Jong f2	De Jong (1975)	Wodrich and Bilchev (1997)	2	(–2.048, 2.048)	–3906.98
77	De Jong f3 – Step Function	"	http://iridia.ulb.ac.be/langerman/ICEO.html	5	(–5.12, 5.12)	–30

Note * : Solution may not be optimal.

8. Appendix (Second Set of Test Functions)

	Name	Source / Property	Reported by	No. of Vars.	Upper Bound. Lower Bound	Global Optimum
1	Baluja-1	Baluja (1994) High interdependence among variables	Wodrich and Bilchev (1997)	100	(–2.56, 2.56)	–100000
2	Baluja-2	Baluja (1994) High interdependence among variables	"	100	(–2.45, 2.56)	–100000
3	Baluja-3	Baluja (1994) No interdependence among variables	"	100	(–2.56, 2.56)	–100000
4	Sphere	http://iridia.ulb.ac.be/langerman/ICEO.html Unimodal (14th in the first set)	Bilchev and Parmee (1996)	30	(–5.12, 5.12)	0
5	Michalewicz	http://iridia.ulb.ac.be/langerman/ICEO.html (13th in the first set)	"	10	$(0, \pi)$	–4.687
6	Powersum	Neumaier web site Singular minimum among very flat valleys (22nd in the first set)	"	64	(0, 2)	0
7a	Schwefel's Sine Root	web site MATLAB / TEST/Lazauskas Second local minimum which is very far from global minimum traps algorithms (24th in the first set)	Web site MATLAB /TEST/Lazauskas	20	(–500, 500)	0
7b	Schwefel's Sine Root	Same as above	Same as above	50	(–500, 500)	0

	Name	Source / Property	Reported by	No. of Vars.	Upper Bound. Lower Bound	Global Optimum
7c	Schwefel's Sine Root Sine Root	Same as above	Same as above	100	(–500, 500)	0
7d	Schwefel's Sine Root	Same as above	Same as above	150	(–500, 500)	0
7e	Schwefel's Sine Root	Same as above	Same as above	200	(–500, 500)	0
7f	Schwefel's Sine Root	Same as above	Same as above	400	(–500, 500)	0
8a	Rastrigin	web site MATLAB / TEST/Lazauskas Highly multimodal (25th in the first set)	Web site MATLAB /TEST/Lazauskas	20	(–5.12, 5.12)	0
8b	Rastrigin	Same as above	Same as above	50	(–5.12, 5.12)	0
8c	Rastrigin	Same as above	Same as above	100	(–5.12, 5.12)	0
8d	Rastrigin	Same as above	Same as above	200	(–5.12, 5.12)	0
8e	Rastrigin	Same as above	Same as above	400	(–5.12, 5.12)	0
9a	Griewank	http://iridia.ulb.ac.be/langerman/ ICEO.html (28th in the first set) 4 local minima	Http://iridia.ulb.ac be/langerman/ ICEO.html	10	(–600, 600)	0
9b	Griewank	Same as above	Same as above	20	(–600, 600)	0
9c	Griewank	Same as above	Same as above	100	(–600, 600)	0
9d	Griewank	Same as above	Same as above	200	(–600, 600)	0
9e	Griewank	Same as above	Same as above	400	(–600, 600)	0
9f	Griewank Original	Griewank (1981)	Törn and Zilinskas (1989)	50	(–100, 100)	0
10a	Schewel, 3.7	Schwefel (1991) Singular hessian at $\mathbf{x}^* = 0$ (52nd in the first set)	Jansson and Knüppel (1994)	10	(–0.002, 0.63)	0
10b	Schewel, 3.7	Same as above	Same as above	30	(–0.005, 0.36)	0
11a	Ackley's Fnc.	web site MATLAB / TEST/Lazauskas, easy	web sit MATLAB TEST/Lazauskas	30	(–30, 30)	0
11b	Ackley's Fnc.	Same as above	Same as above	100	(–30, 30)	0
11c	Ackley's Fnc.	Same as above	Same as above	200	(–30, 30)	0
11d	Ackley's Fnc.	Same as above	Same as above	400	(–30, 30)	0

References

Aarts, E.H. and Korst, J.H.M. (1991), Boltzmann machines as a model for parallel annealing, Algorithmica, 6: 437–465.

Aarts, E.H., de Bont, F., Haberts, E. and van Laarhoven, P. (1986), Parallel implementations of the statistical cooling algorithm, VLSI Journal, 4: 209–238.

Ali, M.M. and Storey, C. (1994), Topographical Multi Level Single Linkage. Journal of Global Optimization, 5: 349–358.

Aluffi-Pentini, F., Parisi, V. and Zirilli, F. (1985), Global optimization and stochastic differential equations. Journal of Optimization Theory and Applications, 47: 1–16.

Androulakis, G.S. and Vrahatis, M.N. (1996), OPTAC: a portable software packaage for analyzing and comparing optimization methods by visualization. Journal of Computational and Applied Mathematics 72: 41–62.

Balardi, F. and Orlando, S. (1989), Strategies for a massively parallel implementaion of simulated annealing, Parallel Architectures and Languages, PARLE '89: 273–287.

Baluja, S. (1994), Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Working Paper, CMU-CS-94-163, School of Computer Science, Carnegie Mellon University.

Bilchev, G. and Parmee, I. (1996),Inductive search. IEEE Trans. 832–836.

Botsaris, C.A. (1996), A curvilinear optimization method based upon iterative estimation of the eigensystem of the hessian matrix. J. of Mathematical Analysis and Applications, 63: 396–411.

Breiman, L. and Cutler, A. (1993), A deterministic algorithm for global optimization. Mathematical Programming, 58: 179–199.

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

Chen, H., Flann, N.S. and Watson, D.W. (1998), Parallel genetic simulated annealing: A massively parallel SIMD algorithm, IEEE Trans. on Parallel and Distributed Systems, 9: 126–136.

Corana, A., Marchesi, M., Marini, C. and Ridella, S. (1987), Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, ACM Trans. of Mathl. Software 13: 262–279.

De Jong, K. (1975), An analysis of the behaviour of a class of genetic adaptive systems. Ph.D. Thesis, University of Michigan, Ann Arbor.

Dekkers, A. and Aarts, E. (1991), Global optimization and simualted annealing. Mathematical Programming, 50: 367–393.

Demirhan, M., Özdamar, L., Helvacioğlu, L. and Birbil, Ş.I. (1999), FRACTOP: A Geometric partitioning metaheuristic for global optimization, Journal of Global Optimization, 14: 415–435.

Diekmann, R., Lüling, R. and Simon, J. (91193), Problem independent distributed simulated annealing and its applications. Working Paper, University of Paderborn, Dept. of Mathematics and Computer Science, Germany.

Easom, E. (1990), A survey of global optimization techniques. M. Eng. Thesis, University of Louisville, Louisville, KY.

Frost, R. (1993), Ensemble based simulated annealing, ftp.sdsc.edu/pub/sdsc/math/Ebsa, La Jolla, CA, University of California, San Diego.

Greening, D.R. (1990), Parallel simulated annealing techniques, Physica, D42: 293–306.

Jansson, C. and Knüppel, O. (1994), Numerical Results for a self-validating global optimization method. Working Paper, No. 94.1, Technical University of Hamburg, Harburg.

Horst, R. and Tuy, H. (1996), Global Optimization – Deterministic Approaches, 3rd ed. Springer Verlag.

Ingber, L. (1994), Simulated annealing: Practice versus theory. J. Mathl. Comput. Modelling, 18(11): 29–57.

Ingber, L. (1995), Adaptive simulated annealing (ASA): Lessons learned. To appear in Control and Cybernetics, (Polish Journal).

Rinnooy Kan, A.H.G. and Timmer, G.T. (1994), Stochastic methods for global optimization. American J. of Mathematical Management Science 4: 7–40.

Kearfott, B. (1979), An efficient degree-computation method for a generalized method of bisection. Numerical Mathematics 32: 109–127.

Kirkpatrick, A., Gelatt, Jr., C.D. and Vechi, M.P. (1983), Optimization by simulated annealing, Science 220: 671–680.

Kliewer, G. and Tschöke, S. (1998), A general parallel simulated annealing library and its applications in industry. Working Paper, University of Paderborn, Dept. of Mathematics and Computer Science, Germany.

Kravitz, S.A. and Rutenbar, R.A. (1987), Placement by simulated annealing on a multi-processor, IEEE Trans. on Computer Aided Design 6: 534–549.

Kurbel, K., Schneider, B. and Singh, K. (1998), Solving optimization problems by parallel recombinative simulated annealing on a parallel computer – An application to standard cell placement in VSLI design, IEEE Trans. on Systems, Man and Cybernetics-Part B-Cybernetics 28: 454–460.

Lee, S-Y. and Lee, K.G. (1996), Synchronous and asynchronous parallel simulated annealing with multiple Markov Chains. IIE Transactions on Parallel and Distributed Systems 7.

Levy, A.V., Montalvo, A., Gomez and S., Calderon, A. (1981), Topics in Global Optimization: Lecture Notes in Mathematics, No. 909, Springer-Verlag, Berlin.

Lin, F.T., Kao, C.Y. and Hsu, C.C. (1991), Incorporating genetic algorithms into simulated annealing, F.J Cantu-Ortiz et al. (eds), Proceedings of the Int. Symposium on Artificial Intelligence: pp. 290–297.

Michalewicz, Z. (1994), Genetic Algorithms+Data Structures=Evolution Programs, Springer Verlag, Berlin.

Moore, R.E. (1979), Methods and applications of interval analysis. SIAM, Philadelphia.

More, B. J., Garbow, B.S. and Hillstrom, K.E. (1981), Testing unconstrained optimization. ACM Trans. Math. Software 7: 17–41.

Moore, R.E. and Ratschek, H. (1988), Inclusion functions and global optimization II. Mathematical Programming 41: 341–356.

Özdamar, L. and Birbil, I.Ş. (1998), A hybrid genetic algorithm for the capacitated lot sizing and loading problem, European Journal of Operational Research 110: 525–547.

Özdamar, L. and Demirhan, M. (2000a), Experiments with new stochastic global optimization search techniques, Computers and OR 27: 841–865.

Özdamar, L and Demirhan, M. (2000b), Comparison of partition evaluation measures in an adaptive partitioning algorithm for global optimization, to appear in Fuzzy Sets and Systems.

Özdamar, L. (1999), A genetic algorithm approach for the multi-mode resource-constrained project scheduling problem under general resource categories. IEEE Transactions on Systems, Man and Cybernetics 29: 44–69.

Pinter, J. (1992), Convergence qualification of adaptive partitioning algorithms in global optimization, Mathematical Programming 56: 343–360.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992), Numerical Recipes, The Art of Scientific Computing,. Cambridge University Press, New York.

Price, W.L. (1998), A controlled random search procedure for global optimization, in L.C.W. Dixon and G.P. Szegö (eds), Towards Global Optimization 2, North-Holland, Amsterdam.

Rosenbrock, H.H. (1970), State-Space and Multivariable Theory, Nelson, London.

Schaffer, J.D. (1989), A study of control parameters affecting online performance of genetic algorithms for function optimization. Proc. of the Third Int. Con. on Genetic Algorithms: 51–60.

Schwefel, H. (1991), Numerical Optimization of Computer Models, Wiley, New York. Srinivas, M. and Patnaik, L.M. (1994), Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 24: 656–667.

Stenger, F. (1975), Computing the topological degree of a mapping in $\aleph^n$, Numerical Mathematics 25: 23–38.

Stuckman, B.E. and Easom, E.E. (1992), A comparison of Bayesian/Sampling global optimization techniques, IEEE Transactions on Systems, Man and Cybernetics 22: 1024–1032.

Törn, A. (1978), A search clustering approach to global optimization, in L.C.W. Dixon and G.P. Szegö (eds), Towards Global Optimization 2, North-Holland, Amsterdam.

Törn, A. and Zilinskas, A. (1989), Global Optimization. Lecture Notes in Computer Science, Springer-Verlag, Berlin.

Törn, A. and Viitanen, S. (1994), Topographical global optimization using pre-sampled points. Journal of Global Optimization 5: 267–276.

Törn, A., Ali, M.M. and Viitanen, S. (1999), Stochastic global optimization: Problem classes and solution techniques, Journal of Global Optimization 14: 437–447.

Wendt, O. and König, W. (1998), Cooperative simulated annealing: How much cooperation is enough?, Working Paper, Frankfurt University.

Wingo, D.R. (1983), Estimating the location of the Cauchy distribution by numerical global optimization. Communications in Statistics Part B. Simulation and Computation 12: 201–212.

Wodrich, M. and Bilchev, G. (1997), Cooperative distributed search: The ant's way. Working Paper, University of Cape Town, South Africa.

Zacharias, C.R., Lemes, M.R. and Pino, A.D. (1998), Combining genetic algorithms and simulated annealing: a molecular geometry optimization study. THEOCHEM – Journal of Molecular Structure 430: 29–39.

Zilinskas, A. (1981), Two algorithms for one-dimensional multimodal minimization, Optimization 12, 53–63.