Pergamon

Computers chem. Engng Vol. 20, No. 9, pp. 1065–1080, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0098-1354(95)00221-9
0098-1354/96 $15.00 + 0.00

THE SIMPLEX–SIMULATED ANNEALING APPROACH TO CONTINUOUS NON-LINEAR OPTIMIZATION

MARGARIDA F. CARDOSO, R. L. SALCEDO† and S. FEYO DE AZEVEDO
Departamento de Engenharia Química, Faculdade de Engenharia da Universidade do Porto, Rua dos Bragas, 4099 Porto, Portugal

(Received 30 November 1994; final revision received 5 April 1995)

Abstract—An algorithm suitable for the global optimization of nonconvex continuous unconstrained and constrained functions is presented. The scheme is based on a proposal by Press and Teukolsky (*Comput. Phys.* **5**(4), 426, 1991) that combines the non-linear simplex and simulated annealing algorithms. A non-equilibrium variant will also be presented, whereby the cooling schedule is enforced as soon as an improved solution is obtained. The latter is shown to provide faster execution times without compromising the quality of the attained solutions.

Both the algorithm and its non-equilibrium variant were tested with several severe functions published in the literature. Results for nine of these functions are compared with those obtained employing a robust adaptive random search method and the Nelder and Mead simplex method (*Comput. J.* 7, 308, 1965). The proposed approach is shown to be more robust and more efficient in what concerns the overcoming of difficulties associated with local optima, the starting solution vector and the dependency upon the random number sequence. The results obtained reveal the adequacy of the algorithm for the global optimization of a broad range of problems encountered in chemical engineering practice. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

The optimization of non-linear constrained problems is relevant to chemical engineering practice. Non-linearities are introduced by process equipment design relations, by equilibrium relations and by combined heat and mass balances, and constraints are due to limits imposed on design variables and on inequality relations between dependent and independent variables. The design variables may be continuous [non-linear programing (NLP) problems] or some may be integer [mixed-integer non-linear programming (MINLP) problems]. For the case of NLP problems, the general statement is:

$$\text{minimize } F(\boldsymbol{\theta}) \tag{1}$$

subject to the inequality constraints:

$$g_j(\boldsymbol{\theta}) \geq 0 \quad j = 1, 2, \ldots, M \tag{2}$$

and to the bound constraints:

$$\alpha_i \leq \boldsymbol{\theta}_i \leq \beta_i \quad i = 1, 2, \ldots, N \tag{3}$$

where $\boldsymbol{\theta}$ represents the vector of optimization parameters and (α_i, β_i) are bounds on parameter θ_i.

The efficiency of optimization routines is linked to their ability in reaching the optimum with a desired accuracy within a finite number of iterations. With nonconvex functions and constraints, the presence of local optima is difficult to deal with, since most optimization algorithms are of the "local search" type and will eventually get trapped in a local optimum. This is particularly true for derivative algorithms, but still highly relevant for direct (derivative-free) methods. It should be noted that currently available algorithms for nonconvex NLP problems can lead to sub-optimal solutions as nonconvexities may cut off the global optimum from the current search regions (Wang and Luus, 1978; Kocis and Grossmann, 1988; Floudas *et al.*, 1989).

Adaptive random search methods are particularly attractive for the global optimization of nonconvex problems and are easily applicable to constrained functions. Some examples include the Luus and Jaakola algorithm (Luus and Jaakola, 1973) and improved variants (Gaines and Gaddy, 1976; Heuckroth *et al.*, 1976; Wang and Luus, 1977; Martin and Gaddy, 1982; Salcedo *et al.*, 1990), modified Matyas algorithms (Mihail and Maria, 1986; Maria, 1989) and search clustering algorithms (Price, 1978; Törn, 1978). Although beyond the scope of this paper, a further advantage of adaptive random search methods is in their ability to handle MINLP problems directly, at least of moderate dimension (Campbell and Gaddy, 1976; Salcedo, 1992).

† Author to whom all correspondence should be addressed.

The ability of random search methods to escape difficult local optima can be attributed to the random nature of the search space. Such desirable feature is significantly enhanced when the random method incorporates shifting strategies allowing for "wrong-way" moves to be produced. These strategies may be simple heuristics that penalize the current solution vector, guiding the algorithm towards diverse regions within the search space (Salcedo *et al.*, 1990) or more sophisticated penalizing functions.

Simulated annealing is a powerful technique for combinatorial optimization, i.e. for the optimization of functions that may assume several distinct discrete configurations (Metropolis *et al.*, 1953; Kirkpatrick *et al.*, 1983; Kirkpatrick, 1984). Algorithms based on simulated annealing employ a stochastic generation of solution vectors and employ similarities between the physical process of annealing (i.e. melting a solid by heating it, followed by slow cooling and crystallization into a minimum free energy state) and a minimization problem. During the cooling process, transitions are accepted to occur from a low to a high energy level through a Boltzmann probability distribution. For an optimization problem, this corresponds to "wrong-way" movements as referred to above.

Other direct methods, albeit deterministic, also possess some ability to escape local optima. One of the most robust with respect to this feature, for unconstrained problems, is the non-linear simplex method of Nelder and Mead (1965). This is a direct algorithm that does not require derivative (either first or second order) information. The simplex method has been exhaustively studied by many researchers, among them Olson and Nelson (1975) and Barabino *et al.* (1980). Basically (Nelder and Mead, 1965; Tao, 1988; Edgar and Himmelblau, 1988), the methods proceeds by generating a simplex (with N dimensions, a simplex is a hypergeometric figure generated by joining $N+1$ points in the N-dimensional space) which evolves at each iteration through reflexions, expansions and contractions in one direction or in all directions, so as mostly to move away from the worst point.

In this work, we present an algorithm based on the combination of the non-linear smplex and simulated annealing algorithms (the SIMPSA algorithm) which is shown to be adequate for the global optimization of an example set of unconstrained and constrained NLP functions. The algorithm relies on a scheme proposed by Press and Teukolsky (1991) incorporating important additional features, such as the ability to handle constraints and appropriate termination criteria. The adoption of a non-equilibrium simulated annealing scheme (Cardoso *et al.*, 1994) leads to a variant (the NE-SIMPSA algorithm) which decreases the computational burden without significantly compromising the quality of the attained solutions.

Comparison is made with a robust adaptive random search method available (Salcedo *et al.*, 1990; Salcedo, 1992) and with the Nelder and Mead (1965) simplex algorithm. The approach proposed in this paper shows a superior performance in overcoming local optima and accuracy. All nine test functions employed for the analysis of performance were proposed by independent authors and cover a broad range of difficult optimization problems. Thus, the SIMPSA and the NE-SIMPSA algorithms represent interesting alternatives for the global optimization of NLP problems typical of chemical engineering practice.

SIMULATED ANNEALNG FOR COMBINATORIAL MINIMIZATION

Annealing is the physical thermal process of melting a solid by heating it, followed by slow cooling and crystallization into a minimum free energy state. If the cooling rate is not carefully controlled or the initial temperature is not sufficiently high, the cooling "solid" does not attain thermal equilibrium at each temperature. In such circumstances, local optimal lattice structures may occur, which translate into lattice imperfections. Thermal equilibrium at a given temperature is characterized by a Boltzmann distribution function of the energy states. Under these conditions, even at a low temperature, albeit with a small probability, a transition may occur from a low to a high energy level. Such transitions are assumed to be responsible for the system reaching a minimum energy state.

The Metropolis algorithm (Metropolis *et al.*, 1953; Kirkpatrick *et al.*, 1983; Kirkpatrick, 1984) was the first proposed to simulate this process. Starting from a high energy state, corresponding to a particular system temperature T^j, a series of new energy states E^j are stochastically generated. Each new system configuration is accepted if $E^{j+1} \leqslant E^j$. Otherwise, and by analogy with the Boltzmann distribution for energy states at thermal equilibrium, E^j will be accepted as an improved state with a probability determined by $P(\Delta E) = \exp(-(E^{j+1}-E^j)/(K_B T^j))$, where T^j is the current system temperature and K_B is Boltzmann's constant. At high temperatures this probability is close to one, *viz.* most energy transitions are permissible. As the system temperature decreases the probability of accepting a higher energy state as being an improved

energy state approaches zero, and it is assumed that thermal equilibrium is reached at each temperature.

From an optimization point of view, simulated annealing explores the key feature of the physical annealing process of generating transitions to higher energy states, applying to the new states an acceptance/rejection probability criterion which should naturally become more and more stringent with the progress of the procedure. For solving a particular problem with a simulated annealing algorithm, the following steps are thus necessary:

(i) Definition of an objective function to be minimized.

(ii) Adoption of an annealing cooling schedule, whereby the initial temperature, the number of configurations generated at each temperature and a method to decrease it are specified.

(iii) At each temperature in the cooling schedule, stochastic generation of the alternative combinations, centered on the currently accepted state.

(iv) Adoption of criteria for the acceptance or rejection of the alternative combinations, against the currently accepted state at that temperature.

The vast majority of applications of simulated annealing have been related to combinatorial minimization problems, such as the classical traveling salesman problem (Press *et al.*, 1986; Salcedo *et al.*, 1993; Cardoso *et al.*, 1994), heat-exchanger and pressure-relief header networks (Dolan *et al.*, 1989, 1990; Cardoso *et al.*, 1994), complex rectification column sequencing (Floquet *et al.*, 1994), graph partitioning (Johnson *et al.*, 1989), optimization of physical data tables (Corey and Young, 1989), batch process scheduling (Das *et al.*, 1990; Ku and Karimi, 1991; Patel *et al.*, 1991), graph colouring and number partitioning (Johnson *et al.*, 1991) and imaging applications (Silverman and Addler, 1992; Groisman and Parker, 1993). An overview of simulated annealing as applied to combinatorial minimization can be found in Aarst and Korst (1989) and in Ingber (1993).

SIMULATED ANNEALING FOR CONTINUOUS OPTIMIZATION

Simulated annealing has also been applied to the optimization of multimodal continuous spaces (Vanderbilt and Louie, 1984; Bohachevsky *et al.*, 1986; Corana *et al.*, 1987; Press and Teukolsky, 1991) and this is the subject of the present work. For such problems, the system state (or configuration) is simply the continuous vector $\boldsymbol{\theta}$ and one must provide some means of generating alternative configurations starting from the current one. This seems to be the major stumbling block for the effective application of simulated annealing to the optimization of continuous spaces (Vanderbilt and Louie, 1984; Press and Teukolsky, 1991).

Corana *et al.* (1987) derived a global continuous optimizer based on simulated annealing, by coupling the acceptance/rejection criteria of the Metropolis algorithm with a random search that progresses along each coordinate axis. The algorithm was tested against both the Nelder and Mead (1965) simplex method and an adaptive random search method (Masri *et al.*, 1980; Pronzato *et al.*, 1984); for difficult unconstrained functions, up to 10 dimensions, it was found to be more robust in overcoming local optima, albeit at a great expense in computational burden. This algorithm may, however, produce poor results for the optimization of cost functions having "valleys" not directed along the coordinate axis (Corana *et al.*, 1987). It is noteworthy to mention that, apart from the occurrence of a few local optima, the simplex method required much fewer function evaluations to locate the global optima than both the simulated annealing and the adaptive random search algorithms (by about a factor of 500–1000). Groisman and Parker (1993) have applied the Corana *et al.* (1987) algorithm for the solution of least-squares problems in applied photometry.

Bohachevsky *et al.* (1986) used a slightly different scheme, whereby the random directions of simulated annealing were computed from independent standard normal variates and the random steps were problem dependent. They have applied their algorithm to the solution of simple multimodal 2-D unconstrained functions and to a more complex constrained design problem in neuroscience involving 11 degrees of freedom.

Vanderbilt and Louie (1984) proposed a self-regulatory search mechanism for the continuous vector $\boldsymbol{\theta}$, which guarantees that the annealing proceeds in an efficient and anisotropic way, i.e. maintaining a biased random walk towards the global minimum. They have applied their algorithm to difficult unconstrained mathematical functions and found it to be competitive with adaptive random search and search cluster methods (Price, 1978; Törn, 1978). Vanderbilt and Louie (1984) further suggest that their approach could be integrated with search clustering or with the simplex method, to improve performance.

Press and Teukolsky (1991) have reviewed the basic approaches behind the application of simulated annealing to continuous optimization, and have proposed, in our opinion, a very interesting and potentially robust scheme — combining the

stochastic simulated annealing algorithm with the simplex method of Nelder and Mead (1965). However, these authors do not show any data to substantiate the performance of the proposed algorithm.

THE SIMPSA AND NE–SIMPSA ALGORITHMS

In this work, the original Metropolis algorithm and its non-equilibrium variant (NESA/Metropolis; Cardoso *et al.*, 1994) were combined with the simplex algorithm as proposed by Press and Teukolsky (1991), giving rise to the SIMPSA and NE–SIMPSA algorithms.

The role of simulated annealing in the overall approach is to allow for wrong-way movements, simultaneously providing (asymptotic) convergence to the global optimum. The main aspects are the acceptance/rejection criteria, the initial annealing temperature and the adoption of a suitable cooling schedule. The role of the non-linear simplex is to generate system configurations. The main differences between the proposed algorithms and that of Press and Teukolsky are:

- the ability to deal with constraints
- the adoption of a cooling schedule based on the global centroid
- the adoption of a contraction scheme for the parameter search space, commanded by the cooling schedule, whereby toggling is made between the global and the contracted search space
- the inclusion of an additional stopping criterium which produces a faster non-equilibrium variant of simulated annealing.

All of these aspects of the algorithms are described next.

The equilibrium and non-equilibrium Metropolis algorithms

With the Metropolis algorithm, the probability of acceptance of new configurations (solutions) is as follows:

$$\begin{aligned} P &= 1 && \Delta C < 0 \\ &= \exp(-\Delta C/K_B T) && \Delta C \geq 0 \end{aligned} \qquad (4)$$

where ΔC is the difference in cost between a newer configuration and the current solution, K_B is Boltzmann's constant and T is the annealing temperature. The Metropolis algorithm enforces the cooling schedule only after a large number of trials have been evaluated, in order to reach equilibrium at every temperature level. To reduce the computational burden associated with this scheme, the cooling schedule may be enforced as soon as an improved solution is obtained. This modification corresponds to a non-equilibrium situation and to a change in the termination criterion for the inner loop.

At the start of the optimization, the solution is most probably far from the optimum and CPU time is not wasted in the search for a near-equilibrium state. As the optimization proceeds, the "temperature" control parameter drops and solution acceptance is decided according to the Metropolis algorithm. It will eventually drop to a point where no more poorer acceptances are allowed, thus assuring convergence to a local optimum, as in the patterns observed for the original simulated annealing procedures. This non-equilibrium scheme has been tested with combinatorial minimization and shown to produce significantly faster convergence to the global optimum (Salcedo *et al.*, 1993; Cardoso *et al.*, 1994). In the present paper, we show that the non-equilibrium variants of simulated annealing also give good results for non-linear continuous optimization, in a fraction of the time needed by the equilibrium Metropolis algorithm.

Initial annealing temperature

The initial control temperature is an important parameter which can be obtained by specifying the fraction of generated solutions to be accepted initially (Aarst and Korst, 1989; Ku and Karimi, 1991; Patel *et al.*, 1991). It is important that the initial temperature control parameter be estimated such that the acceptance probabilities are close to one. For this purpose, in this work, the initial temperature T_i was estimated by:

$$X = \frac{m_1 + m_2 \exp\left(\dfrac{-\Delta f^+}{T_i}\right)}{m_1 + m_2} \qquad (5)$$

where the acceptance ratio X was set to 95% (Aarst and Korst, 1989). For a universe of m_0 combinations, and in a sequential analysis, m_1 represents the number of successful moves ($E^k < E^{k-1}$ for $k = 2$ to m_0), m_2 the number of unsuccessful moves ($E^k \geq E^{k-1}$) and Δf^+ the average increase in cost for the m_2 unsuccessful moves.

As with combinatorial optimization, m_0 was set to $100 \times N$, where N represents the number of dimensions (Cardoso *et al.*, 1994). To apply equation (5) to the continuous case, a preliminary high temperature was estimated by multiplying the absolute value of the objective function corresponding to the starting solution vector by a large positive value (e.g. 10^5). This high temperature allows all moves to be initially accepted, and after m_0 moves corresponding to a full Metropolis cycle are completed, the initial

temperature T_i in equation (5) is computed from the values of m_1, m_2 and Δf^+. The cooling schedule will then proceed with this temperature value.

Cooling schedule

The cooling schedule employed was the Aarst and van Laarhoven (1985) scheme:

$$T^{j+1} = \frac{T^j}{1 + \dfrac{T^j \cdot \ln(1+\delta)}{3\sigma}} \tag{6}$$

where δ and σ are trajectory parameters and j is the current iteration. The parameter δ controls the cooling rate and is a measure of the desired closeness to equilibrium (Das *et al.*, 1990; Patel *et al.*, 1991). Small values (<1) produce slow convergence and large values (>1) produce convergence to poor local optima. The parameter σ is the standard deviation of all cost configurations at the current temperature T^j.

Alternative cooling schedules are usually of the exponential type (Kirkpatrick *et al.*, 1983), where a constant multiplicative factor is employed to obtain the new temperature. Tests performed with combinatorial minimization (Aarst and Korst, 1989; Das *et al.*, 1990; Dolan *et al.*, 1990; Patel *et al.*, 1991; Cardoso *et al.*, 1994) lead to verify that simulated annealing performed better with the Aarst and van Laarhoven scheme [equation (6)], rather than with the exponential decrease in temperature.

At this point, it is important to distinguish non-equilibrium simulated annealing (NESA) from simulated quenching (Ingber, 1993), which results from the use of faster cooling schedules in order to reduce the computational burden. With non-equilibrium simulated annealing, slow cooling schedules should be employed to provide smooth temperature trajectories similar to those found in the original Metropolis scheme and minimize the chances of convergence to local optima. The cooling schedule and the acceptance criterion have a coupled effect on convergence. Consequently, the same value of δ will produce significantly different trajectories for the nonequilibrium and equilibrium variants. With combinatorial minimization, it was found (Cardoso *et al.*, 1994) that the δ value for the non-equilibrium algorithm had to be about three orders of magnitude smaller than that corresponding to the equilibrium algorithm, in order to produce similar smooth cooling schedules. The same ratio was thus employed in the present work.

According to Press and Teukolsky (1991), the temperature should be decreased after a fixed number of iterations within the simplex method. Here, this number was set at $100 \times N$, as stated above for the evaluation of the initial annealing temperature. With the non-equilibrium variants, the temperature also decreases as soon as an improved solution is obtained. Since a representative point on which to measure this improvement is needed, the global centroid (incorporating all current simplex vertices without considering constraints) was employed. The main idea is that the global centroid is the most appropriate indicator of the simplex movement. Tests performed showed that better results were obtained using the global centroid rather than using the centroid with the exclusion of the worst point, as is usually done in the simplex method. This global centroid was also used by Adelman and Stevens (1972) in their implementation of a constrained version of the simplex method, *viz.* a variant of the Complex method of Box (1965).

Generating the initial simplex

To solve an N-D problem, it is necessary to generate $N+1$ points that form the vertices of the N-D simplex.

For unconstrained problems, this was performed by the following rule:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_0 + (0.5\text{-rnd}) \times 2 \times \text{abs}(\boldsymbol{\theta}_0) \tag{7}$$

where $\boldsymbol{\theta}_0$ is any initial N-D point, rnd is a pseudo-random number between 0 and 1 and abs is the absolute value of the N components of the initial vector $\boldsymbol{\theta}_0$. However, other schemes are equally possible.

For constrained problems, the initial simplex was generated by:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_0 + (0.5\text{-rnd}) \times K^j \times (\beta_i - \alpha_i) \tag{8}$$

which is the continuous parameter generation scheme employed in the SGA/MSGA algorithms (Salcedo *et al.*, 1990; Salcedo, 1992). K^j is a variable factor, as shown below [equation (9)], and j is the current global iteration.

It should be stressed that equation (8) does not guarantee feasibility, with respect either to the inequality [equation (2)] or to the bound [equation (3)] constraints. It was found that the proposed algorithms sometimes collapsed on infeasible points if the initial solution vector $\boldsymbol{\theta}_0$ did not obey all bound constraints. In this case, equation (8) might not be able to generate simplex vertices within the bound constraints, evolving towards a region of total infeasibility from where the algorithms may not recover. Thus, in the SIMPSA and NE–SIMPSA algorithms, equation (8) is repeatedly employed until all vertices obey all bound constraints (but not necessarily the

inequality constraints), starting from either a feasible or an infeasible point. In this work, for comparison purposes, we have employed the starting points proposed by independent authors. In a general application, a simple choice for a starting solution vector which obeys all bound constraints is the middle of the search intervals. This has also been tested, having led to similar results.

Generation of system configurations

The generated configurations are options presented to the system. For combinatorial minimization, the generation of each system configuration on an N-D space, specified by the vector components θ_j ($j = 1, N$), from the previous solution θ_i ($i = 1, N$), may be problem dependent. It may be obtained by simple random changes (Dolan *et al.*, 1989) or may include some heuristic guidance (Press *et al.*, 1986).

For the case of continuous optimization, the generation of the system state is based on the simplex method of Nelder and Mead (1965), whereby the design vector θ is replaced by an N-D simplex. Press and Teukolsky (1991) add a positive logarithmic distributed variable, proportional to the control temperature T, to the function value associated with every vertex of the simplex. Likewise, they subtract a similar random variable from the function value at every new replacement point. These schemes are represented by the following relations:

$$(F_{\text{perturbed}})_k = F_k - T \times \ln(\text{rnd}); \quad k = 1, N+1 \tag{9}$$

$$(F_{\text{perturbed}})_{\text{new}} = F_{\text{new}} + T \times \ln(\text{rnd}) \tag{10}$$

where F_k is the function value for vertex k, F_{new} is the function value at the replacement point and $F_{\text{perturbed}}$ is the perturbed function value. For a minimization problem, the $N+1$ vertices are perturbed towards higher function values, according to equation (9), whereas the replacement point is perturbed towards a lower value, following equation (10). Thus, if the replacement point corresponds to a lower cost, this method always accepts a true downhill step. If, on the other hand, the replacement point corresponds to a higher cost, an uphill move may be accepted, depending on the relative costs of the perturbed values. This is the principle behind the Metropolis algorithm, and has the advantage of reducing to the simplex method as $T \to 0$. This approach has been retained in the present work.

Dealing with constraints

This is an important and difficult question that affects most optimization problems with interest to chemical engineering. However, this point is not addressed by Press and Teukolsky (1991). Box (1965) has developed an optimization algorithm applicable to constrained functions (the Complex method) based on the simplex method of Nelder and Mead. Thus, it seemed logical to replace, within the proposed scheme, the simplex method with the Complex method of Box (1965). Basically, Box proposed that if one bound constraint [equation (3)] is violated, the variable takes the value of the constraint, and if an inequality (or implicit) constraint is violated [equation (2)], the new point would move half-way from the distance that separates it from the centroid of the remaining points (Adelman and Stevens, 1972; Edgar and Himmelblau, 1988).

We tested several problems with this approach, and in most cases degenerate simplexes would occur collapsing on an infeasible centroid. This could in principle be solved by restarting periodically the algorithm or by using more complex vertex replacement schemes (Umeda and Ichikawa, 1971), but a more convenient and simple method was sought — this consists of substituting points produced by the simplex movement that do not obey either bound or implicit constraints by randomly generated points centered on the current best vertex, through the use of equation (8), where θ_0 is now the best vertex. Infeasible points may still be replaced by infeasible points, except that they are now centered on the best vertex and obey all bound constraints.

To ensure convergence of the evolving simplex, the parameter K^j in equation (8) was made variable with the global iteration j, following the Aarst and van Laarhoven (1985) scheme [equation (6)]. This shrinking effect on the size of the region centered on the current best vertex may, however, guide the algorithm too fast towards a local optimum from which it may not be able to escape. Thus, the compression factor K^j is only activated once every two global iterations, *viz.* the algorithm toggles between a search with the global intervals and a search with compressed intervals, *viz.*:

$$K^j = 1 \qquad j \text{ odd}$$

$$K^j = K^{j-2}\frac{T^j}{T^{j-1}} \qquad j \text{ even}. \tag{11}$$

All infeasible points are penalized by assuming a very large positive value for a minimization problem or a very large negative value otherwise. It may obviously occur that all points in the simplex end up penalized. However, quantitative comparison between these points, which is needed both for the simplex algorithm and for the Metropolis algorithm, is still possible, since a random perturbation proportional to the temperature control parameter is superimposed on the function values [equations (9)

and (10)]. The simplex will then proceed with reflections, expansions or contractions through the centroid, defined here as usually, *viz.* incorporating all current vertices with the exception of the worst point.

Thus, unlike direct random search methods such as the MSGA algorithm, which are feasible path methods, the proposed algorithms are infeasible path methods, since they can proceed through comparisons of infeasible vertices. For highly nonconvex problems, it may be very difficult to generate points on the surface described by inequality constraints. With degenerate simplexes, the periodic setting of $K^j = 1$ makes the overall search space available for the generation of new solutions. This avoids the need of periodically restarting the algorithm, since the probability of obtaining a feasible vertex increases. On the other hand, when some vertex is feasible, the adoption of a shrinking search space increases the probability of finding a nearby feasible point, thus guiding the algorithm towards feasibility.

As the objective functions of the proposed algorithms do not include any information about the extent of constraint violations, there is no guarantee that a feasible point will be found. However, the proposed algorithms were able to converge to feasible points that correspond to several active inequality constraints, which is the case for the global optima of all constrained problems tested in the present work.

Termination criteria

The proposed algorithm includes two convergence tests. One is inherent to the simplex method, as implemented by Press *et al.* (1986) and Press and Teukolsky (1991) and is a measure of collapse of the centroid. The second criterium was employed before for combinatorial minimization and is based on an averaged gradient of the objective function with respect to the number of function evaluations. This has been shown to be an efficient criterion for the non-equilibrium simulated annealing algorithms (Cardoso *et al.*, 1994). To implement such a criterion while smoothing out fluctuations in the objective function, groups of $5 \times N$ values were averaged. The normalized gradient was computed from these averages, as follows:

$$\frac{1}{C_i^*}\frac{\mathrm{d}C^*}{\mathrm{d}N} = \frac{C_i^* - C_{i-1}^*}{C_i^*(N_i - N_{i-1})} < \varepsilon \qquad (12)$$

where $0 < \varepsilon \ll 1$, N_i is the cumulative number of function evaluations at iteration i and C_i^* the respective averaged cost. Since the factor $N_i - N_{i-1}$ is constant (in our case equal to $5 \times N$), the gradient given by equation (12) is simply an expression for the relative error in the objective function, averaged over a fixed number of function evaluations. Here, just as for the cooling schedule, the global centroid is employed to compute the cost term needed for the successive comparisons.

NUMERICAL IMPLEMENTATION AND CASE-STUDIES

The SIMPSA and the NE–SIMPSA algorithms were written in FORTRAN 77 and all runs were performed with double precision on an HP730 workstation, running compiler optimized FORTRAN 77 code. To state the quality of the proposed algorithms, comparison with another global optimizer is needed. The MSGA adaptive random search algorithm (Salcedo *et al.*, 1990; Salcedo, 1992) was implemented in the same workstation and used for this purpose.

Simulated annealing and random search methods are stochastic, and as such can be sensitive to the sequence of pseudo-random numbers. In this work, a Lehmer linear congruential generator (Shedler, 1983) was used, which was tested for multidimensional uniformity and found to be satisfactory (Salcedo *et al.*, 1990).

Performance evaluation was carried out by statistical evaluation of nine severe unconstrained and constrained functions taken from the literature. The constrained functions are all nonconvex (multimodal), used by other authors to test robustness in surpassing local optima. The unconstrained functions include difficult ill-conditioned least-squares examples.

The error criteria used in the present work were set at 10^{-8}, both for the convergence of the simplex algorithm as given by Press and Teukolsky (1991) and for the smoothed function values as given by equation (12). For the unconstrained functions, the cooling schedule δ values were set at 10 and 10^{-2}, respectively for the equilibrium (SIMPSA) and non-equilibrium (NE–SIMPSA) algorithms. The Rosenbrock function in four dimensions was tested with another value for δ to provide some indication of the influence of this parameter. For the constrained functions, which are more difficult, lower values for δ were employed, respectively $10^{-1}/10^{-4}$. Whenever different values for δ were used, in order to improve results, this is explicitly referred to in the discussion.

All problems were run with 100 different seeds. For the SIMPSA, NE–SIMPSA and unconstrained and constrained simplex, this corresponds to the generation of 100 different initial simplexes. It also corresponds to stochastic movements, whenever bound constraints are violated, through the use of

Table 1. Test functions for optimization procedures

Case number	Authors	Type of problem	Type of optimization	Number of parameters	General comments
1	Rosenbrock (1960)	MF	UMIN	2/4	Steep-sided parabolic valley
2	Colville (1968)	MF	UMIN	4	Difficult saddle point
3	Dixon (1973)	MF	UMIN	10	Difficult local optimum
4	Meyer and Roth (1972)	MF	ULS	3	Extremely ill-conditioned
5	Nash and Walker-Smith (1987)	MF	ULS	6	Triple exponential; difficult to fit
6	Luus (1974)	MF	CMAX	3	Four local maxima
7	Luus and Jaakola (1973)	CEP	CMIN	3	Difficult local minima
8	Luus (1975)	CEP	CMAX	5	Difficult local maxima
9	Grossmann and Sargent (1979)	CEP	CMIN	7	Optimization of multicrude pipeline

Type of Problem: MF = mathematical function; CEP = chemical engineering problem.
Type of Optimization: CMAX = constrained maximization; CMIN = constrained minimization; UMIN = unconstrained minimization; ULS = unconstrained least-squares.

equation (8). Further, for the SIMPSA and NE–SIMPSA algorithms, the different seeds also provide different optimization paths since a random perturbation is superimposed on the simplex vertices, as discussed before. For the MSGA algorithm, the different seeds simply provide different optimization paths as they directly affect the stochastic generation of solution vectors.

Some problems were also run with different starting points to provide a measure of the influence of the initial conditions. The initial search regions, for constrained optimization, were determined from the bound constraints. Where appropriate, results are also compared on the basis of accuracy in reaching a value of the objective function within a percentage of the global optimum. The average number of function evaluations as well as the average CPU times are also given. With the MSGA algorithm, the number of function evaluations is approximately constant at around 35000, irrespective of the problem dimension or difficulty (Salcedo *et al.*, 1990; Salcedo, 1992).

RESULTS AND DISCUSSION

Unconstrained minimization

The problems represented by functions 1–5 of Table 1 were chosen to illustrate the behavior of the SIMPSA algorithms for unconstrained minimization. The corresponding functional expressions and data are given in Table 2. Here, sets of 100 different seeds were employed with each algorithm, for every starting point.

The results are summarized in Table 3. For function 1, 2-D, all algorithms performed well, reaching the global optimum of $(1, 1)^T$. The average execution times were 0.35 s, 0.41 s, 0.67 s and 0.81 s,

Table 2. Unconstrained function

Function	Test function	Initial values	Global optimum (author)
1a Rosenbrock (1960)	$y = 100(\theta_1^2 - \theta_2)^2 + (1 - \theta_1)^2$	$\boldsymbol{\theta} = [-1.2, 1]^T$ $F = 24.2$	$\boldsymbol{\theta} = [1, 1]^T$ $F = 0$
1b Rosenbrock (1960)	$y = \sum_{i=1}^{3} \{100(\theta_i^2 - \theta_{i+1})^2 + (1 - \theta_i)^2\}$	$\boldsymbol{\theta} = [-1.2, 1, -1.2, 1]^T$ $F = 532.4$	$\boldsymbol{\theta} = [1, 1, 1, 1]^T$ $F = 0$
2 Colville (1968)	$y = 100(\theta_1^2 - \theta_2)^2 + (\theta_1 - 1)^2 + (\theta_3 - 1)^2 + 90(\theta_3^2 - \theta_4)^2 + 10.1((\theta_2 - 1)^2 + (\theta_4 - 1)^2) + 19.8(\theta_2 - 1)*(\theta_4 - 1)$	$\boldsymbol{\theta} = [-3, -1, -3, -1]^T$ $F = 19192$	$\boldsymbol{\theta} = [1, 1, 1, 1]^T$ $F = 0$
3 Dixon (1973)	$y = (1 - \theta_1)^2 + (1 - \theta_{10})^2 + \sum_{i=1}^{9} (\theta_i^2 - \theta_{i+1})^2$	$\boldsymbol{\theta} = [-2, \ldots, -2]^T$ $F = 342$	$\boldsymbol{\theta} = [1, \ldots, 1]^T$ $F = 0$
4 Meyer and Roth (1972)	$y = a^* \exp[b/(c + x)]$	$\boldsymbol{\theta} = [0.02, 4000, 250]^T$ $F = 1.7 \times 10^9$	$\boldsymbol{\theta} = [0.0056, 6181.4, 345.2]^T$ $F = 88$
5 Nash and Walker-Smith (1987)	$y = \sum_{i=1}^{3} [\theta_{2i-1} \exp - (\theta_{2i} x)]$	$\boldsymbol{\theta} = [1, 1, 1, 2, 1, 3]^T$ $F = 12.1$	$\boldsymbol{\theta} = [0.0951, 1, 0.8607, 3, 1.5567, 5]^T$ $F = 0$

With the following values applicable:

Function 4:	x	50	55	60	65	70	75	80	85	90	95	100	105	110	115	120	125
	y	34780	28610	23650	19630	16370	13720	11540	9744	8261	7030	6005	5147	4427	3820	3307	2872

Function 5: $y = 0.951 \exp(-x_i) + 0.8607 \exp(-3x_i) + 1.5567 \exp(-5x_i)$; $x_i = 0.05(i - 1)$, $i = 1$–24.

Table 3. Results for the unconstrained functions 1–5 ($\delta = 10$ for SIMPSA and $\delta = 10^{-2}$ for NE–SIMPSA)

Function	Starting point	Search interval	MSGA N_{Succ}*	Simplex N_{Succ}	Simplex N_{fobj}†	SIMPSA N_{Succ}	SIMPSA N_{fobj}	NE–SIMPSA N_{Succ}	NE–SIMPSA N_{fobj}
1a	$(-1.2, 1)^T$	$[-1.2, 1.2]$	100	100	323	100	10780	100	4508
		$[-10, 10]$	100	100	352	100	12500	100	5007
		$[-100, 100]$	98	100	394	99	13446	100	5353
			Average: 99	100		100		100	
1b	$(-1.2, 1, -1.2, 1)^T$	$[-1.2, 1.2]$	100	96	938	99	21177	94	3053
						100	43573‡	99	17886‡
		$[-10, 10]$	100	78	959	96	23616	82	3191
						97	48282‡	89	18338‡
		$[-100, 100]$	70	73	1025	82	25046	74	3421
						93	52861‡	84	20060‡
			Average: 90	82		95		87	
2	$(-3, -1, -3, -1)^T$	2*ABS(**θ**)	81	100	1102	100	27483	100	3591
		$[-3, 3]$	100	99	1136	100	22615	100	3591
	local optimum§	2*ABS(**θ**)	99	100	930	100	24675	100	3493
	local optimum‖	2*ABS(**θ**)	100	100	948	100	24830	100	3443
	$(-9, -3, -9, -3)^T$	2*ABS(**θ**)	78	100	1162	100	28201	100	3769
			Average: 92	100		100		100	
3	$(-2, \ldots, -2)^T$	2*ABS(**θ***)	27	92	9509	97	73115	91	9792
	local optimum¶	$[-0.5, 3.5]$	46	96	9249	93	56513	94	8613
		$[-0.3, 3.7]$	55	84	8435	93	54860	92	8099
	$(2, \ldots, 2)^T$	$[-0.5, 3.5]$	44	79	6144	92	55089	77	6428
		$[0, 4]$	98	65	5565	93	52556	65	5882
			Average: 54	83		94		84	
4	$(0.02, 4000, 250)^T$	2*ABS(**θ***)	4($< F = 176$)	98	2569	92	19162	99	4426
		ABS(**θ***)	5($< F = 176$)	100	2586	99	18570	99	4424
5	$(1, 1, 1, 2, 1, 3)^T$	2*ABS(**θ***)	49($< F = 10^{-5}$)	100	3094	1	12913	99	6124

* Percent number of successes.
† Average number of function evaluations.
‡ $\delta = 1$ for SIMPSA and $\delta = 10^{-3}$ for NE–SIMPSA
§ Local optimum = $(-0.79633, 0.645242, 1.13839, 1.29675)^T$; $F = 3.33443$.
‖ Local optimum = $(1, 1, -0.94032, 0.89566)^T$; $F = 3.886615$.
¶ Local optimum = $(0.94177, 0.88658, 0.77885, 0.59291, 0.35278, 0.12024, 0.031941, 0.0014266, -0.016515, 0.50500)^T$; $F = 0.504$.

respectively, for the simplex, NE–SIMPSA, SIMPSA and MSGA algorithms. With four dimensions and four large search intervals, all algorithms decreased in robustness in arriving at the global optimum of $(1, 1, 1, 1)^T$. The SIMPSA algorithms produced better results with smaller values of the cooling parameter δ, but at a greater expense in computational burden. The average execution times were 0.46 s, 0.53 s, 1.46 s and 1.70 s, respectively, for the simplex, NE–SIMPSA, MSGA and SIMPSA algorithms, and increased to 1.52 s and 3.00 s with the slower cooling schedule, respectively, for the NE-SIMPSA and SIMPSA algorithms. The non-equilibrium variant (NE-SIMPSA) requires much fewer function evaluations than the original Metropolis version, with only a small decrease in robustness. For this problem, it can be stated that the simplex method of Nelder and Mead (1965) is almost as robust as the other three algorithms and requires much fewer function evaluations. Corana *et al.* (1987) also observed the efficiency of the simplex method compared to simulated annealing for the minimization of the Rosenbrock functions. However, their implementation of simulated annealing required about 5×10^5 function evaluations in two dimensions and 1.2×10^6 in four dimensions, *viz.* much larger values than those reported in the present work. The adaptive random search method tested by these authors (Masri *et al.*, 1980; Pronzato *et al.*, 1984) also required much more function evaluations with four dimensions than the MSGA algorithm, albeit starting from different solution vectors than those employed here.

For function 2, all algorithms performed well, but especially again the simplex and the NE–SIMPSA algorithms, since they required much fewer function evaluations than the MSGA or SIMPSA algorithms, and always arrived at the global optimum. The average execution times were 0.35 s, 0.42 s, 1.28 s and 1.41 s, respectively, for the simplex, NE–SIMPSA, SIMPSA and MSGA algorithms.

For function 3, the worst algorithm is the adaptive random search method, which, despite its general robustness, is here sensitive to the starting solution vector and search interval. It should be noted that this problem has 10 degrees of freedom and a difficult local optimum [local optimum (6) in Table 3]. Again, both the simplex and NE–SIMPSA algorithms require much fewer function evaluations than the SIMPSA algorithm, with only a small decrease in robustness. The average execution times were 1.07 s, 1.54 s, 3.15 s and 6.50 s, respectively, for the simplex, NE–SIMPSA, MSGA and SIMPSA algorithms.

Table 4. Constrained functions

Function	Test function and constraints		Global optimum (author)
6 Luus (1974)	$y=\theta_1^2+\theta_2^2+\theta_3^2$		$\theta_1=0.988$
	$4(\theta_1-0.5)^2+2(\theta_2-0.2)^2+\theta_3^2+0.1\theta_1\theta_2+0.2\theta_2\theta_3\leq 16$		$\theta_2=2.674$
	$2\theta_1^2+\theta_2^2-2\theta_3^2\geq 2$		$\theta_3=-1.884$
	$-2.3\leq\theta_i\leq 2.7 \quad i=1,2$		$F=11.67664$
7 Luus and Jaakola (1973)	Fuel allocation to power plants		
	$y=f_1\theta_3+g_1\theta_4$		$\theta_1=30$
	$f_1=1.4609+0.15186\theta_1+0.00145\theta_1^2$	$\theta_2=50-\theta_1$	$\theta_2=20$
	$g_1=0.8008+0.2031\theta_2+0.000916\theta_2^2$	$18\leq\theta_1\leq 30$	$\theta_3=0$
	$f_2=1.5742+0.1631\theta_1+0.001358\theta_1^2$	$14\leq\theta_2\leq 25$	$\theta_4=0.58366$
	$g_2=0.7266+0.2256\theta_2+0.000778\theta_2^2$	$0\leq\theta_3\leq 1$	$F=3.0521$
		$0\leq\theta_4\leq 1$	
	$(1-\theta_3)f_2+(1-\theta_4)g_2\leq 10$		
8 Luus (1975)	Cross-extraction problem*		
9 Grossman and Sargent (1979)	Optimization of multicrude pipeline†		

* A complete description is availabe in Luus (1975) or Salcedo *et al.* (1990).
† A complete description is available in Grossmann and Sargent (1979).

Functions 4 and 5 represent least-squares structures. It is well known that for this class of problems, it is strongly recommended to employ Gauss–Newton-type methods. These functions are included in the present set of tests since they represent very ill-conditioned problems.

Function 4 is extremely ill conditioned, since variations of only 0.01% in each parameter, centered around the global optimum, produce variations of several orders of magnitude in the objective function. Actually, the optimum parameter values reported for this function, as truncated and listed in Table 2, produce a value of 2017 for the objective function, instead of the reported value of 88. Thus, results within 100% of the global optimum, *viz.* $F<176$ are indeed acceptable. It can be seen that the simplex and NE–SIMPSA algorithms are very efficient in solving this least-squares problem. The average execution times were 0.44s, 0.71s, 1.81s and 3.26 s, respectively, for the simplex, NE–SIMPSA, SIMPSA and MSGA algorithms.

For function 5, a triple exponential, even more difficult to fit, the worst results occur with the SIMPSA algorithm, which only produced one objective function below 10^{-5}, whereas the NE–SIMPSA and the simplex algorithms produced much more accurate results. The forced decrease in temperature in the NE–SIMPSA version approximates the algorithm to the original simplex method, whereas the equilibrium version wanders in the search domain witout being able to significantly decrease the objective function. The behavior of the MSGA algorithm is somewhere between these two extremes. As seen in Table 3, all algorithms performed better with a smaller search region. The average execution times were 2.17 s, 5.17 s, 8.62 s and 21.9 s, respectively, for the simplex, NE–SIMPSA, SIMPSA and MSGA algorithms.

From the above discussion, we conclude that the simplex and NE–SIMPSA algorithms give more consistent results in arriving at the global optimum. Also, as expected, the simplex required fewer function evaluations. Comparing the equilibrium and non-equilibrium versions of the simulated annealing algorithms, the latter is much faster and generally as robust as the equilibrium version.

Constrained optimization

Constrained optimization problems are more interesting and probably more important than the unconstrained ones, at least from the point of view of process engineering. Table 4 lists for four cases, problems 6–9, selected for the present analysis. It further includes the functional expressions for problems 6 and 7. Problems 8 and 9 are described in more detail later in the text. The simplex method, which was originally developed for unconstrained problems, was modified according to the method given above (under the section entitled *Dealing with constraints*) and as such can be directly compared with the other algorithms. The only algorithmic difference between the constrained simplex algorithm and the SIMPSA and NE–SIMPSA algorithms is that, in the former, the temperature is always equal to zero, which deactivates the simulated annealing scheme. Obviously, the shrinking effect on the size of the region centered on the current best vertex following the cooling schedule, given by equation (11), is not applicable here, since the temperature remains always at zero.

Table 5 shows the results obtained by all algorithms in solving function 6. Here, the constrained simplex algorithm is the worst, and the SIMPSA variants the best, the non-equilibrium version requiring about half the number of function evaluations for only a small decrease in robustness. The

MSGA algorithm requires more function evaluations than either the SIMPSA or NE–SIMPSA implementations, and produces somewhat poorer results. The average execution times were 0.28 s, 1.00 s, 1.20 s and 1.70 s, respectively, for the simplex, NE–SIMPSA, MSGA and SIMPSA algorithms.

Table 6 shows the average number of function evaluations required by all algorithms to solve functions 7–9. Figures 1a and 1b show the behavior of all algorithms for the optimization of the fuel allocation problem (function 7), respectively for the two different starting points given in Table 6. These figures show the number of trials (out of 100 runs) which reached a value of the objective function within a percentage of the global optimum. As this function exhibits local minima which are about 0.5–1% of the global optimum, the results indicate that the SIMPSA, NE–SIMPSA and MSGA algorithms were all robust in solving this problem, the non-equilibrium algorithm requiring about one half the number of function evaluations as compared to the equilibrium version. Also, the simplex algorithm is very sensitive to the starting point, in opposition to what occurs with the other three algorithms. The average execution times were 0.28 s, 1.34 s, 1.49 s and 2.59 s, respectively, for the constrained simplex, MSGA, NE–SIMPSA and SIMPSA algorithms.

The classical cross-extraction problem

This problem is given by Luus (1975) and was extensively tested by Salcedo *et al.* (1990). It is a very good example of a simple chemical engineering process and a challenge to an optimization algorithm due to the presence of difficult local optima. Figure 2 is a schematic diagram of the process, where the solvents Q and W are totally immiscible and f_i represents the fraction recycled from stage i to stage $i+1$. The xs and ys represent mass fractions of solute, respectively, on solvent phases Q and W. The equilibrium curves for the solute on both phases are given by Luus (1975) as two cubics:

$$y = 2.50x + 3.70x^2 - 113.0x^3 \quad x \leq 0.1 \tag{13}$$

$$y = 3.94x - 29.6x^2 + 74.0x^3 \quad x > 0.1. \tag{14}$$

Assuming a known feed rate Q and the maximum allowable extraction, the process has five degrees of freedom. The objective function is:

$$\max F = \{Q(x_f - x_1) - \lambda(w_1 + w_2 + w_3) - C[f_1 w_1 + f_2(w_1 f_1 + w_2)]\} \tag{15}$$

where λ is the relative cost of the extraction solvent W (i.e. cost of unit of solvent W per unit of value of extracted product) and C is the relative cost of recycling (i.e. cost of unit recycled per unit of value of extracted product). The five decision variables are $\{f_1, f_2, x_1, w_1, w_2\}$, subject to:

$$0 \leq f_i \leq 1; \quad i = 1, 2 \tag{16}$$

$$0 \leq x_1 \leq (x_f)_{\max} \tag{17}$$

$$0 \leq \sum_{i=1}^{3} w_i \leq \text{CAP} \tag{18}$$

$$0 \leq w_i; \quad i = 1\text{–}3 \tag{(19)}$$

where the value of $(x_f)_{\max}$ and CAP depend on the particular data. The pertinent data used are shown in Table 7. Also shown are the global optima for both cases, determined with the MSGA algorithm through optimized tunning of the search grids (Salcedo *et al.*, 1990). The objective function for case 1 exhibits one local optimum at $\{f_1, f_2, x_1, w_1, w_2; F\} = \{1, 0, 0.0257795, 0.4945684, 0; 0.1111452\}$, which is only about 0.25% lower than the global optimum. On the other hand, the objective function for case 2 is rather insensitive to the optimization parameters, with widely different values of the independent parameters producing values within 1% of the global optimum. This combination of multimodality and rippled search surface makes the problem challenging and the global optimum difficult to arrive at.

Table 5. Results for the constrained function 6 ($\delta = 10^{-1}$ for SIMPSA and $\delta = 10^{-4}$ for NE–SIMPSA)

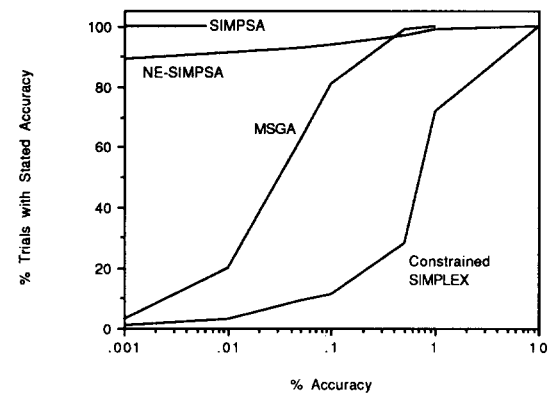
Starting point	MSGA	Simplex		SIMPSA		NE–SIMPSA	
	N_{Succ}*	N_{Succ}	$N_{f_{\text{obj}}}$†	N_{Succ}	$N_{f_{\text{obj}}}$	N_{Succ}	$N_{f_{\text{obj}}}$
$(1.04400, 2.48909, 1.78541)^T$	87	0	511	100	31950	95	16702
$(2.22288, -0.17259, 1.98899)^T$	71	0	624	100	31500	96	15742
$(1.57411, -1.79381, -1.75688)^T$	70	0	531	99	31588	87	15352
$(2.04195, -0.95967, -1.90526)^T$	78	4	577	99	31626	94	15575
$(-1.41862, -0.16050, 1.01260)^T$	76	32	943	98	29915	96	15714
$(1.40498, -0.24513, -1.00200)^T$	76	18	942	100	31195	92	15779
$(-1.3, -0.3, 0.63)^T$	75	29	958	96	30516	90	14610
Average	76	12		99		93	

* Percent number of successes.
† Average number of function evaluations.

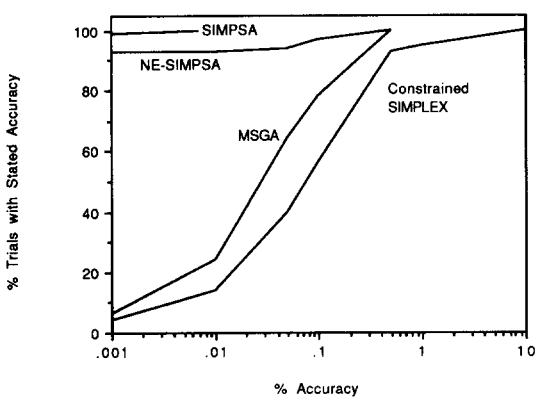
Table 6. Average number of objective function evaluation (100 runs) for the constrained functions 7–9 ($\delta = 10^{-1}$ for SIMPSA and $\delta = 10^{-4}$ for NE–SIMPSA)

Function	Starting point	Simplex	SIMPSA	NE–SIMPSA
7	$(25, 1, 1)^T$	845	44413	23839
	$(30, 0, 0.6)^T$	631	44622	24651
8	$(1, 1, 1, 0.3, 0)^T$	954	59842	23970
	$(0, 0, 0.3, 1.2, 0)^T$	1546	46734	23214
9	$(28, 14, 14, 110, 110, 110, 110)^T$	2040	52402	13780
			298228*	60037*

* $\delta = 10^{-2}$ for SIMPSA and $\delta = 10^{-5}$ for NE–SIMPSA.



(a)



(b)

Fig. 1. (a) Results for function No. 7 with starting point $(25, 1, 1)^T$. (b) Results for function No. 7 with starting point $(30, 0, 0.6)^T$.

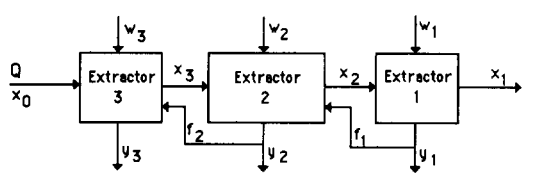


Fig. 2. Schematic diagram of cross-current extractors with wash recycle.

Figures 3a and 3b show the number of trials which reached a value of the objective function within a percentage of the global optimum, for both cases of Table 7. The following conclusions can be drawn:

(i) For case 1, the constrained simplex algorithm is the worst performer, since it is the least accurate and robust, as it is caught 66 times in the local optimum. For the other three algorithms, all results are within 1% of the global optimum. The MSGA algorithm, although not the most accurate, is capable of escaping most times (92 out of 100) the local optimum, whereas the SIMPSA is caught 14 times and the NE–SIMPSA 35 times. Of these, the NE–SIMPSA algorithm requires the least number of function evaluations.

(ii) For case 2, a similar ranking between algorithms occurs, with the constrained simplex the worst performer and the other three giving all results within 1% of the global optimum. Again, the NE–SIMPSA requires the least number of function evaluations of the three robust methods.

The average execution times were 0.42 s, 2.08 s, 2.29 s and 4.20 s, respectively, for the constrained simplex, MSGA, NE–SIMPSA and SIMPSA algorithms.

The results reported so far suggest that one should not expect to arrive at the global optimum with a single computer run, irrespective of the expected algorithm robustness.

Optimization of a crude pipeline distribution

The last problem studied was taken from Grossmann and Sargent (1979) and refers to the optimization of a multicrude pipeline. The schematic diagram can be seen in Fig. 4, and the purpose is to transport a number of fluids (in this example $n = 4$) with different mass flow rates (Q_i) and different physical properties, with constraints on minimum outlet pressures ($P^i_{m^2}$, $P^i_{m^3}$) and allocated flow periods per year. The pipe segments have known lengths (L_i) and variable constrained diameters (D_i).

Table 7. Input data and global optima for function 8

Case	Q	x_f	CAP	C	λ		f_1	f_2	x_1	W_1	W_2	F
1	1.0	0.2	1.2	0.105	0.05	Initial solution	1	1	0.1	0.3	0	0.0624
						Global optimum	1	1	0.0197961	0.318717	0.442742	0.1114302
2	2.6	0.6	2.4	0.03846	0.17308	Initial solution	0	0	0.3	1.2	0	0.5668
						Global optimum	0.886009	0.492992	0.0973779	2.31640	0	0.7880472

The objective is to obtain the optimum values for the three pipe diameters and the required pressures at the pump exit for all fluids in order to minimize a cost function. For a complete description of this problem and associated data, the reader is referred to Grossmann and Sargent (1979).

Starting from the point $\{D_1, D_2, D_3, P_1, P_2, P_3, P_4; F\} = \{28, 14, 14, 110, 110, 110, 110; 149833\}$, Grossman and Sargent (1979), using a variable metric method after linearizing all constraints, have obtained the lowest cost at {24.6, 15.6, 16, 105.6, 102.2, 105.6, 101.9; 130555}.

Figure 5a shows the results obtained with the four algorithms on a first approach to this problem, without any linearization, where a somewhat lower

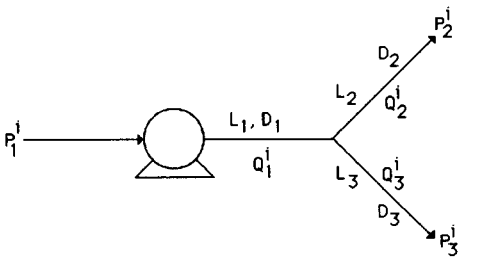
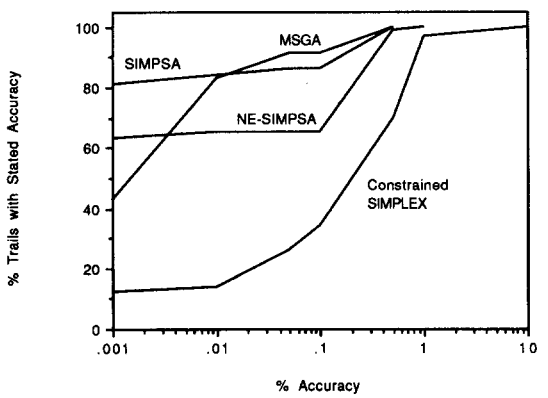


Fig. 4. Schematic diagram of multicrude pipeline distribution.



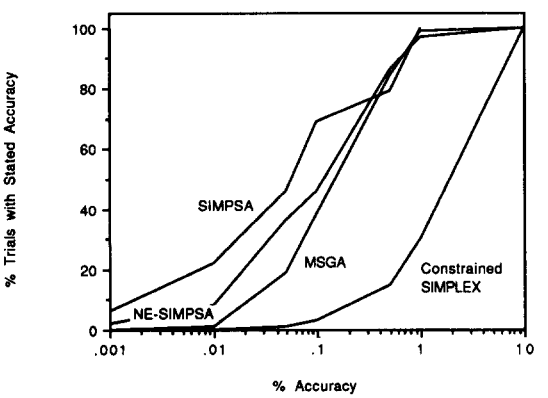
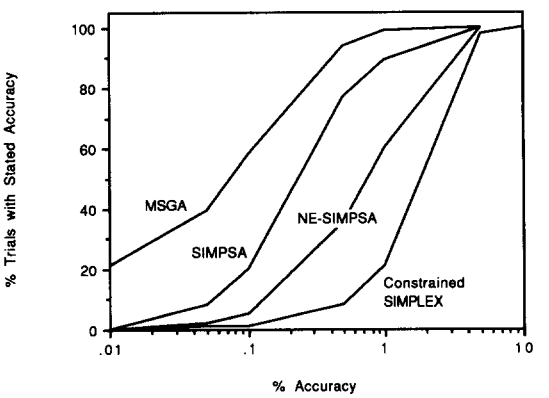


Fig. 3. (a) Results for case 1 of function No. 8 with starting point $(1, 1, 1, 0.3, 0)^T$. (b) Results for case 2 of function No. 8 with starting pont $(0, 0, 0.3, 1.2, 0)^T$.



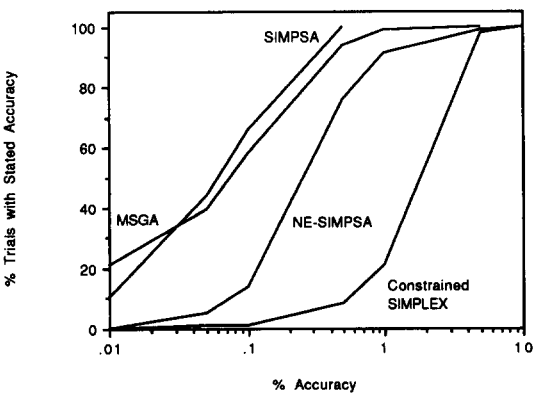


Fig. 5. (a) Results for function No. 9 with parameter $\delta = 10^{-1}$ for SIMPSA and $\delta = 10^{-4}$ for NE–SIMPSA. (b) Results for function No. 9 with parameter $\delta = 10^{-2}$ for SIMPSA and $\delta = 10^{-5}$ for NE–SIMPSA.

cost (about 0.6%) was obtained, corresponding to the point $\{D_1, D_2, D_3, P_1, P_2, P_3. P_4; F\} = \{26.908, 15.221, 16.000, 105.58, 102.17, 105.59, 101.91; 129842.8\}$. Again, the constrained simplex is the worst performer. Of the other three algorithms, the MSGA was the best, finding almost all results better than those reported by Grossman and Sargent, and the NE–SIMPSA required the least function evaluations. Figure 5b shows a second approach to this problem, where the SIMPSA and NE–SIMPSA algorithms were run with a lower value for the cooling parameter δ. The improvement is significant, since now all values obtained by the SIMPSA algorithm and about 80% of those obtained by the NE–SIMPSA algorithm are better than those reported. However, the number of function evaluations has increased significantly. The average execution times were 0.67 s, 3.41 s, 3.94 s and 11.3 s, respectively, for the constrained simplex, NE–SIMPSA, MSGA and SIMPSA algorithms. With the slower cooling schedule, these times increased to 11.4 s and 58.4 s, respectively, for the NE–SIMPSA and SIMPSA algorithms.

We conclude that the proposed constrained implementation of the simplex method is much less efficient than the SIMPSA, NE–SIMPSA or MSGA algorithms for the optimization of the difficult constrained functions employed in the present work. The SIMPSA algorithm seems to be the most robust of the tested algorithms, but usually at a greater expense in terms of the number of function evaluations. The NE–SIMPSA is also very robust, and requires less function evaluations (between 1/5 and 1/2) than the equilibrium version. For constrained optimization, the MSGA algorithm seems to be competititve with the SIMPSA and NE–SIMPSA algorithms, both in the number of function evaluations and robustness.

CONCLUSIONS

An optimization algorithm is presented, for continuous non-linear optimization (the SIMPSA algorithm), based on a scheme proposed by Press and Teukolsky (1991) that combines the simplex method of Nelder and Mead (1965) with the Metropolis algorithm (Metropolis *et al.*, 1953; Kirkpatrick *et al.*, 1983; Kirkpatrick, 1984). It has the ability to deal with arbitrary constraints through substition of infeasible points by randomly generated points centered on the best point of the current simplex. To ensure final convergence of the evolving simplex and at the same time avoiding too-fast convergence towards a local optimum, from which it may not recover, the algorithm toggles between a search with the global intervals and a search with compressed intervals. The compression of the search intervals follows the current decrease in the temperature control parameter, along the Aarst and van Larhoven (1985) cooling schedule.

A faster version of the algorithm, the NE–SIMPSA algorithm, which enforces a non-equilibrium simulated annealing scheme, is also proposed as a variant of the SIMPSA algorithm. Both variants include an appropriate termination criterium based on a smoothed gradient of the objective function with respect to the number of function evaluations.

The proposed algorithms were compared with the simplex method of Nelder and Mead, with a constrained variant of the simplex and with the MSGA adaptive random search method, for a variety of difficult unconstrained and constrained optimization functions, taken from the literature.

For unconstrained functions, the behavior of the algorithms showed some problem dependence, and no real discrimination becomes evident in what concerns the algorithms' robustness, although the simplex and NE–SIMPSA are more consistent. The unconstrained simplex is the most efficient, since it requires fewer function evaluations.

For the constrained functions examined in this work, the SIMPSA, NE–SIMPSA and MSGA algorithms are much more robust than the proposed constrained simplex. They are competitive with one another, but the NE–SIMPSA algorithm always requires less function evaluations than the SIMPSA algorithm, and usually less than the MSGA algorithm, thus, it is more efficient overall.

On a global basis, we conclude that the proposed SIMPSA and NE–SIMPSA algorithms are more robust, with the non-equilibrium version more efficient. Thus, they represent interesting alternatives for the global optimizatin of problems with interest to chemical engineering practice.

Since algorithms based on simulated annealing are usually applied to combinatorial minimization, a natural extension of the present work will be the development of an efficient SIMPSA-type algorithm for the optimization of the difficult mixed-integer non-linear programming (MINLP) problems.

Acknowledgement—This work was partially supported by JNICT (Portuguese Junta Nacional de Investigação Científica e Tecnológica), under contract No. BD/1457/91-RM, employing the computational facilities of Instituto de Sistemas e Robótica (ISR)—Porto.

NOMENCLATURE

C = Relative cost of recycling

C^* = Cost average over a fixed number of function evaluations
ΔC = Difference in cost between new and current configurations
CAP = Restriction on allowable solvent rates (Kg/h)
D_i = Pipe diameter (m)
E^j = Energy state of system configuration j
ΔE = Difference in energy between new and current configurations
F = Objective function
f_i = Recycled fraction from extractor i to $i+1$, $i = 1$–2
Δf^+ = Average increase in cost for the m_2 unsuccessful moves
g_j = Inequality constraint, $j = 1$–m
K_B = Boltzmann's constant
K^j = Variable compression factor at iteration j
L_i = Pipe length (m)
m = Number of inequality constraints
m_1 = Number of successful moves
m_2 = Number of unsuccessful moves
N = Number of independent parameters
N_i = Cumulative number of function evaluations at iteration i
P = Probability of acceptance/rejection of current solution
P_i = Outlet pressure of pipe i (Pa)
P^i_{mj} = Minimum outlet pressures of pipe j with fluid i (Pa)
Q = Mass feed rate to extraction system (kg/h)
Q_i = Mass flow rate of fluid i (kg/h)
rnd = Pseudo-random number between 0 and 1
T^j = System (annealing) temperature of iteration j
w_i = Mass flow rate of wash solvent (Kg/h) to extractor i, i = 1–3
X = Acceptance ratio
x_f = Mass fraction of solute in feed
x_i = Mass fraction of solute in outlet from extractor i, $i = 1$–3
y_i = Mass fraction of solute in wash current (extractor i), $i = 1$–3

Greek symbols

α_i = Lower bound on parameter i, $i = 1$–n
β_i = Upper bound on parameter i, $i = 1$–n
δ = Cooling rate control parameter
ε = Error criterion (relative)
λ = Relative cost of extraction solvent
σ = Standard deviation of all cost functions at current temperature
$\boldsymbol{\theta}$ = Vector of independent parameters
$\boldsymbol{\theta}^*$ = Initial solution vector

Abbreviations

CEP = Chemical engineering problems
CMAX = Constrained maximization problems
CMIN = Constrained minimization problems
SGA = Salcedo–Gonçalves–Azevedo algorithm
MSGA = Minlp Salcedo–Gonçalves–Azevedo algorithm
MINLP = Mixed integer non-linear programming
NLP = Non-linear programming
ULS = Unconstrained least-squares problems
UMIN = Unconstrained minimization problems

REFERENCES

Aarst E. and J. Korst, *Similated Annealing and Boltzmann Machines—a Stochastic Approach to Combinatorial Optimization and Neural Computers*. Wiley, New York (1989).

Aarst E. H. L. and P. J. M. van Laarhoven, Statistical cooling: a general approach to combinatorial optimization problems. *Philips J. Res.* **40**, 193 (1985).

Adelman A. and W. F. Stevens, Process optimization by the Complex method. *A. I. Ch. E. J.* **18**(1), 20 (1972).

Barabino G. P., G. S. Barabino, B. Bianco and M. Marchesi, A study on the performance of simplex methods for function minimization. *Proc. IEEE Int. Conf. Circuits Comput. ICCC 80*, 150–1153. IEEE, New York (1980).

Bohachevsky I. O., M. E. Johnson and M. L. Stein, Generalized simulated annealing for function optimization. *Technometrics* **28**(3), 209 (1986).

Box M. J., A new method of constrained optimization and a comparison with other methods. *Computer J.* **8**, 42 (1965).

Campbell J. R. and J. L. Gaddy, Methodology for simultaneous optimization with reliability: nuclear PWR example. *A. I. Ch. E. J.* **22**(6), 1050 (1976).

Cardoso M. F., R. L. Salcedo and S. F. de Azevedo, Non-equilibrium simulated annealing: a faster approach to combinatorial minimization. *Ind. Engng Chem. Res.* **33**(8), 1908 (1994).

Colville A. R., A comparative study of non-linear programming codes. IBM N.Y. Scientific Center, T.R. 320–2925 (1968).

Corana A., M. Marchesi, C. Martini and S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Software* **13**(3), 263 (1987).

Corey E. M. and D. A. Young, Optimization of physical data tables by simulated annealing. *Comput. Phys.* **3**(3), 33 (1989).

Das H., P. T. Cummings and M. D. Levan, Scheduling of serial mutliproduct batch processes via simulated annealing. *Computers chem. Engng* **14**(12), 1351 (1990).

Dolan W. B., P. T. Cummings and M. D. Levan, Process optimization via simulated annealing: application to network design. *A. I. Ch. E. J.* **35**(5), 725 (1989).

Dolan W. B., P. T. Cummings and M. D. Levan, Algorithmic efficiency of simulated annealing for heat exchanger network design. *Computers chem. Engng* **14**(10), 1039 (1990).

Dixon L. C. W., Conjugate directions without linear searches. *J. Ins. Math. Appl.* **11**, 317 (1973).

Edgar T. F. and D. M. Himmelblau, *Optimization of Chemical Processes*. McGraw-Hill, New York (1988).

Floquet P., L. Pibouleau and S. Domenech, Separation sequence synthesis: how to use simulated annealing procedure. *Computers chem. Engng* **18**(11/12), 1141 (1994).

Floudas C. A., A. Aggarwal and A. R. Ciric, Global optimum search for nonconvex NLP and MINLP problems. *Computers chem. Engng* **13**(10), 1117 (1989).

Gaines L. D. and J. L. Gaddy, Process optimization by flow sheet optimization. *Ind. Eng. Chem. Process Des. Dev.* **15**, 206 (1976).

Groisman G. and J. R. Parker, Computer-assisted photometry using simualted annealing. *Comput. Phys.* **7**(1), 87 (1993).

Grossmann I. E. and R. W. H. Sargent, Optimal design of multipurpose chemical plants. *Ind. Engng Chem. Process Des. Dev.* **18**, 343 (1979).

Heuckroth M. W., J. L. Gaddy and L. D. Gaines, An examination of the adaptive random search technique. *A. I. Ch. E. J.* **22**, 744 (1976).

Ingber A. L., Simulated annealing: practice versus theory. *J. Math. Comput. Modelling* **18**(11), 29 (1993).

Johnson D. S., C. R. Aragon, L. A. McGeoch and C. Schevon, Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Op. Res.* **37**(6), 865 (1989).

Johnson D. S., C. R. Aragon, L. A. McGeoch and C. Schevon, Optimization by simulated annealing: an

experimetnal evaluation; part II, graph coloring and number partitioning. *Op. Res.* **39**(3), 378 (1991).
Kirkpatrick S., Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.* **34**(5/6), 975 (1984).
Kirkpatrick S., C. D. Gelatt and M. P. Vechi, Optimization by simulated annealing. *Science* **220**, 671 (1983).
Kocis G. R. and I. E. Grossmann, Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Ind. Engng Chem. Res.*, **27**, 1407 (1988).
Ku H. and I. Karimi, An evaluation of simulated annealing for batch process scheduling. *Ind. Engng Chem. Res.* **30**(1), 163 (1991).
Luus R. R., Optimal control by direct search on feed back gain matrix. *Chem. Engng Sci.* **29**, 1013 (1974).
Luus R. R., Optimization of multistage recycle systems by direct-search. *Can. J. Chem. Engng* **53**, 217 (1975).
Luus R. and T. H. I. Jaakola, Optimization by direct search and systematic reduction of the size of search region. *A. I. Ch. E. J.* **19**, 760 (1973).
Maria G., An adaptive strategy for solving kinetic model concomitant estimation-reduction problems. *Can. J. Chem. Engng* **67**, 825 (1989).
Martin D. L. and J. L. Gaddy, Process optimization with the adaptive randomly directed search. *A. I. Ch. E. Symp. Ser.* **78**(214), 99 (1982).
Masri S. F., G. A. Bekey and F. B. Safford, A global optimization algorithm using adaptive random search. *Appl. Math. Comput.* **7**, 353 (1980).
Metropolis N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087 (1953).
Meyer R. R. and P. M. Roth, Modified damped least squares—an algorithm for non-linear estimation. *J. Instrum. Math. App.* **9**, 218 (1972).
Mihail R. and G. Maria, A modified Matyas algorithm (MMA) for random process optimization. *Computers chem. Engng* **10**, 539 (1986).
Nash J. C. and M. Walker-Smith, *Nonlinear Parameter Estimation*, Ch. 14. Marcel Dekker, New York (1987).
Nelder J. A. and R. Mead, A simplex method for function minimization. *Comput. J.* **7**, 308 (1965).
Olson D. M. and L. S. Nelson, The Nelder–Mead simplex procedure for function minimization. *Technometrics* **17**, 45 (1975).
Patel A. N., R. S. H. Mah and I. A. Karimi, Preliminary design of multiproduct noncontinuous plants using simulated annealing. *Computers chem. Engng* **15**(7), 451 (1991).
Press W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes: the Art of Scientific Computing*. Cambridge University Press, UK (1986).
Press W. H. and S. A. Teukolsky, Simulated annealing optimization over continuous spaces. *Comput. Phys.* **5**(4), 426 (1991).
Price W. L., A controlled random search procedure for global optimization. In L. C. W. Dixon and G. P. Szego, eds, *Towards Global Optimization 2*, p. 71. North-Holland, Amsterdam (1978).
Pronzato L., E. Walter, A. Venot and J. F. Lebruche, A general purpose global optimizer: implementations and applications. *Math. Comput. Sim.* **25**, 412 (1984).
Rosenbrock H. H., An automatic method for finding the greatest or least value of a function. *Comput. J.* **3**, 175 (1960).
Salcedo R. L., Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search. *Ind. Engng Chem. Res.* **31**(1), 262 (1992).
Salcedo R. L., M. F. Cardoso and S. Feyo de Azevedo, A fast simulated annealing algorithm for combinatorial minimization. *Proc. Escape 3*, Graz (Austria), Suppl. Vol., F. Moser, H. Schnitzer and H. J. Bart, eds, pp. 12–17 (1993).
Salcedo R., M. J. Goncalves and S. Feyo de Azevedo, An improved random-search algorithm for nonlinear optimization. *Computers chem. Engng* **14**(10), 1111–1126 (1990).
Shedler G. S., Generation methods for discrete event simulation. In *Computer Performance Modeling Hndbook*, pp. 227–251. Academic Press, New York (1983).
Silverman A. and J. Adler, Animated simulated annealing. *Comput. Phys.* **6**(3), 277–281 (1992).
Tao B. Y., Optimization via the simplex method. *Chem. Engng* **??**, 85–89 (1988).
Törn A. A., A search-clustering approach to global optimization. In L. C. W. Dixon and G. P. Szego eds, *Towards Global Optimization 2*, pp. 49–62. North-Holland, Amsterdam (1978).
Umeda T. and A. Ichikawa, A modified complex method for optimization. *Ind. Engng Chem. Process Des. Dev.* **10**(2), 229 (1971).
Vanderbilt D. and S. G. Louie, A Monte Carlo simulated annealing approach to optimization over continuous variables. *J. Comput. Phys.* **56**, 259 (1984).
Wang B. C. and R. Luus, Optimization of nonunimodal systems. *Int. J. Num. Meth. Engng* **11**, 1235 (1977).
Wang B. C. and R. Luus, Reliability of optimization procedures for obtaining global optimum. *A. I. Ch. E. J.* **24**, 619 (1978).