Experiences with Stochastic Algorithms for a class of Constrained Global Optimisation Problems

ARTICLE *in* RAIRO - OPERATIONS RESEARCH · MARCH 2000

Impact Factor: 0.41 · DOI: 10.1051/ro:2000110

4 AUTHORS, INCLUDING:

Abdel Salhi
University of Essex
73 PUBLICATIONS **252** CITATIONS
SEE PROFILE

L.G. Proll
University of Leeds
46 PUBLICATIONS **389** CITATIONS
SEE PROFILE

RAIRO Operations Research
RAIRO Oper. Res. **34** (2000) 183-197

EXPERIENCES WITH STOCHASTIC ALGORITHMS FOR A CLASS OF CONSTRAINED GLOBAL OPTIMISATION PROBLEMS (*)

by Abdellah SALHI ([1]), L.G. PROLL ([2]), D. RIOS INSUA ([3]) and J.I. MARTIN ([3])

Communicated by J.A. FERLAND

Abstract. – *The solution of a variety of classes of global optimisation problems is required in the implementation of a framework for sensitivity analysis in multicriteria decision analysis. These problems have linear constraints, some of which have a particular structure, and a variety of objective functions, which may be smooth or non-smooth. The context in which they arise implies a need for a single, robust solution method. The literature contains few experimental results relevant to such a need. We report on our experience with the implementation of three stochastic algorithms for global optimisation: the multi-level single linkage algorithm, the topographical algorithm and the simulated annealing algorithm. Issues relating to their implementation and use to solve practical problems are discussed. Computational results suggest that, for the class of problems considered, simulated annealing performs well.*

Keywords: Global optimisation, stochastic methods, constraints, multistart, simulated annealing.

1. INTRODUCTION

The *Global Optimisation (Minimisation) Problem* (GO) can be stated as: let f be a function from R^n to R and $A \subset R^n$, then find $x^* \in A$ such that $\forall x \in A$, $f(x^*) \leq f(x)$. The problem is known to be hard, both from a theoretical and a practical viewpoint (Murty and Kabadi 1987).

(*) Received September 1998.

([1]) Department of Mathematics, University of Essex,Wivenhoe Park, Colchester CO4 3SQ, U.K.
([2]) Scheduling and Constraint Management Group, School of Computer Studies, University of Leeds, Leeds LS2 9JT, U.K.
([3]) Dipartimento de Inteligencia Artificial, Universidad Politecnica de Madrid, 28660-Madrid, Spain.

Recherche opérationnelle/Operations Research, 0399-0559/00/02/$ 7.00

Our need to solve GO problems arises from attempts to implement a framework for discrete multi-criteria decision making which is heavily based on mathematical programming (Proll *et al.* 1993). The package is intended to be used as an aid to decision-makers' understanding of the implications of, and possible inconsistencies in, their judgements during such events as decision conferences (French 1992). This environment implies the need for quick and "automatic" solution of the mathematical programmes, probably on a PC. The sensitivity analysis algorithm filters the set of alternatives through four phases, *dominance*, *potential optimality*, *adjacent potential optimality* and *distance analysis*. Each phase leads to a mathematical programme of different structure, which also depends on the form of the evaluation function and on the distance metric used. Some classes of problem may be nonlinear and nonconvex, so we cannot rely on local optimisation as this may convey a false impression of insensitivity. Although some of the classes exhibit special characteristics, we attempt to handle them as general problems as it is impractical from a software development viewpoint to implement a different algorithm for each of the many classes of problem which arise. Hence we require a robust method. This is reinforced by the fact that, of necessity, we need to solve the problems generated by our framework "unseen" and in near real-time; there is no opportunity to "tune" the optimiser.

In the following we shall be concerned with representatives of two popular stochastic algorithms for global optimisation: *Multistart* and *Simulated annealing*. Two different variants of the multistart algorithm are considered: the Multi-level Single Linkage algorithm, (MLSL), (Rinnooy Kan and Timmer 1987b), and the Topographical method, Törn and Viitanen 1992a). Issues related to their implementation and use to solve practical problems arising in our sensitivity analysis context are discussed.

2. STOCHASTIC ALGORITHMS

2.1. Multistart

Multistart represents a broad class of algorithms designed primarily to improve upon *Pure Random Search* (Rinnooy Khan and Timmer 1987a). An issue which is central to the efficient implementation of such algorithms is that of sampling. Such methods characteristically need to generate a sample of points which in some sense "cover" the search space in order that there is some confidence that all of the best local minima have been detected. Unfortunately it is well known that sampling in regions defined by general constraints is difficult (Rubinstein 1982; Smith 1984) so that it is not

straightforward to adapt MLSL to deal with the constrained GO problem. Consequently, our approach is to sample from the enclosing hypercube defined by the bounds on the variables and use a local optimiser which accepts infeasible starting points.

2.2. Multi-level single linkage

A popular stochastic algorithm for GO is a version of multistart called the *Multi-level Single Linkage* algorithm due to Rinnooy Kan and Timmer (1986, 1987a, 1987b, 1989).

It can be described as follows:

At iteration k

1. Draw a uniform random sample of N points in A; select the γN points with lowest function value, where γ is the reduction parameter.
2. Compute the threshold distance

$$r_k = \pi^{-1/2}\left[\Gamma\left(1+\frac{n}{2}\right)v(A)\frac{\sigma\log(kN)}{kN}\right]^{1/n}$$

   where $v(A)$ is the volume of A, σ is a positive parameter.
3. Perform local optimisation from each selected point $\mathbf{x_i}$ unless there is a previously processed point $\mathbf{x_j}$ for which $f(\mathbf{x_j}) < f(\mathbf{x_i})$ and $d(\mathbf{x_i}, \mathbf{x_j}) < r_k$.
4. Stop if $\frac{(\gamma kN-1)w}{\gamma kN-w-2} \leq w + 0.5$ where w is the number of distinct local minima found so far.

The practical implementation of this algorithm depends on the choice of the parameters (N, γ, σ) which make up the threshold distance. This distance influences the amount of work required by the search and the quality of the search before the stopping rule is satisfied: too short and most points in the reduced sample will be starting points to local optimisation; too long and very few local optimisations will be required, thus increasing the risk of missing the global optimum. Yet it is not clear how these parameters can be chosen other than arbitrarily.

2.3. The topographical algorithm

The underlying strategy of the topographical algorithm (TOPO) of Törn and Viitanen (1994, 1996) is that a topograph may be constructed by evaluating the objective function at randomly sampled points in A. The topograph is a directed graph in which nodes represent sample points and

arcs, directed towards the node with larger function value, connect each node to its k nearest neighbours. The minimum points in the topograph are those with no incoming arcs and are good starting points for possible local optimisations. TOPO can be described as follows:

1. draw a random sample of N points such that, for each pair of points $\mathbf{x_i}, \mathbf{x_j}$, $d(\mathbf{x_i}, \mathbf{x_j}) > \delta$ where $d(.,.)$ is the Euclidean distance and δ a threshold distance.
2. Identify k nearest neighbours of each sample point.
3. Compute the function values at each sample point and identify the minima in the topograph, *i.e.* points for which all k nearest neighbours do not have a better function value.
4. Perform local minimisation from (some of) the minima in the topograph.

Some experimentation is necessary to decide on an appropriate value for δ and the (arbitrarily) chosen sample size N. N can more properly be regarded as a secondary sample size because points in the topograph are obtained by sequentially generating a much larger sample and rejecting points which are closer to others in the sample than the threshold distance. In our experience, the rejection rate is extremely high and, consequently, this process is expensive. A sample of suitable values for $n = 2, ..., 10$ and $N = 100, 200$ for the unit hypercube can be found in Törn and Viitanen (1994). It is also not clear what value to choose for k. Qualitatively, as k increases, fewer local minimisations will be performed. This increases the risk of missing the global minimum, so some compromise is necessary. Algorithm TOPO is essentially a direct method in which the stopping rule is implicit. It stops when all promising points are used to start local optimisations, leading to the choice of the candidate for global minimum.

We considered this method because it had potential advantage for our application, which involves many minimisations over the same search space. In such a case, the overhead of generating the sample can be shared.

2.4. Simulated Annealing

Simulated annealing (SA) is a well-established technique for combinatorial optimisation problems and has been reported to perform well on such problems in high dimensions with a large number of local minima (Eglese 1990). Based on this success, variants of SA for continuous global optimisation have been developed (Corana *et al.* 1987; Dekkers and Aarts 1991). The SA algorithm for continuous optimisation considered here is due

to Dekkers and Aarts (1991). It is modelled as a Markov chain with the transition probability

$$P(B \mid x;T) = \begin{cases} \int_{y\in B} p_{xy}(T)dy & \text{if } x \notin B, \\ \int_{y\in B} p_{xy}(T)dy + \left(1 - \int_{y\in B} p_{xy}(T)dy\right) & \text{if } x \in B, \end{cases}$$

where $p_{xy} = \beta(T)g_{xy}(T)$, g_{xy} being the probability distribution function for generating a point y from a point x at a fixed value of the controle parameter $T \in R^+$, $\beta(T)$ the acceptance criterion given by $\beta(T) = \min(1, e^{-\frac{f(y)-f(x)}{T}})$ and $B \subset A$.

According to Dekkers and Aarts (1991), a procedure based on a Markov chain with the above transition probability will converge asymptotically to a local minimum x of f in $B \subset A$, starting from any point x_0. Formally,

$$\forall \epsilon > 0 : \lim_{T\downarrow 0}\lim_{k\to\infty} Prob\{x_k \in B_f(\epsilon) \mid T\} \geq 1 - \epsilon, \ \forall x_0.$$

Here, $B_f(\epsilon > 0)$ is the set of points in A with value close to that of the minimal point.

The conditions of convergence of such a procedure to the set of minimal points of f are as follows:

1. A is a bounded subset of R^n;
2. f is a real-valued function defined over A;
3. the number of minima of f over A is finite and they are interior to A;
4. the acceptance criterion is $\beta(T)$ defined above;
5. the neighbourhood of a point $x_0 \in A$ is a subset of $A - x_0$ over which the generation probability distribution function $g_{xy}(T)$ is defined by:
   - $\forall x_0 \in A$, $\forall B \subset A$: $m(B) > 0 \Rightarrow \int_{y\in A} g_{x_0y}(T) > 0$, where $m(B)$ is the Lebesgue measure of the set B;
   - $g_{xy}(T) = g_{yx}(T)$;
   - $g_{xy}(T)$ is independant of T.

These conditions, however, are sufficient, but not necessary.

Note that such a procedure can be perceived as an infinite number of homogeneous Markov chains of infinite length, which makes it impracticable to implement. A practicable version, however, can be described as the following SA algorithm.

1. Set $\mathbf{x}$ to $\mathbf{x_0} \in A$, f^* to $f(\mathbf{x})$, T_k to T_0, L_k to L_0 and k to 0;
2. if stopping rule satisfied then Stop;

3. for $l = 1$ to L_k do
   generate $\mathbf{y}$ as a random neighbour of $\mathbf{x}$;
   if $(f(\mathbf{y}) \leq f(\mathbf{x}))$ then
   $\mathbf{x} = \mathbf{y}$;
   elseif $\left(e^{\frac{f(\mathbf{x})-f(\mathbf{y})}{T_k}} > random[0,1)\right)$ then
   $\mathbf{x} = \mathbf{y}$;
   endif
   endfor;
4. $k = k + 1$;
5. find L_k and T_k;
6. go to 2.

In this algorithm parameter T_k is commonly referred as the temperature. It slows down the algorithm if it is too high and it removes the global aspect of the algorithm, *i.e.* uphill moves, if it is too small (Schoen 1991; Eglese 1990). For T_k, we require an initial value, a decrement function for decreasing it and a final value to use in the stopping condition. We also need to set the length L_k of each Markov chain corresponding to each T_k. This set of parameters is usually referred to as the *cooling schedule*. An important difference between SA and multi-start methods is that SA follows a path in the search space rather than attempts to "cover" it. As we discuss later, this may allow general constraints to be handled more effectively.

The difference between the cooling schedule we implented and that of Dekkers and Aarts (1991) is in the way a point in the neighbourhood of the current one is generated. Also, while they use a local search procedure, we adapt the *coordinate directions* method described in Berbee *et al.* (1987) for detecting non-redundant constraints. This allows us to exploit the structure of our constraints which comprise linear equations which are "non-overlapping", *i.e.* variables with non-zero coefficients are present in one equation at most, together with general linear inequalities. Neighbours of the current point are found, as described below, by generating a random direction and a random step length under conditions which allow us to keep feasibility.

Let A be defined by a system of linear equations and inequalities $\mathbf{a_i} * \mathbf{x}(\leq, =)b_i$. Without loss of generality, we assume that in each equation at least two variables have non-zero coefficients.

1. Find a feasible point $\mathbf{x}$.
2. Generate a direction vector $\mathbf{v}$ with equal probability from one of the n coordinate vectors, *i.e.* generate a random index k in 1,...,n. Set $v_k = 1$, $v_j = 0$ for $j \neq k$.

3. If x_k has non-zero coefficient in equality constraint i, generate a random index l, $(l \neq k)$ in 1,...,n such that x_l has non-zero coefficient in constraint i.
   Set $v_l = -a_{ik}/a_{il}$.
4. For each inequality j, compute $\lambda_j = (b_j - \mathbf{a_j} * \mathbf{x})/(\mathbf{a_j} * \mathbf{v})$.
5. If $\exists j$ such that $\lambda_j = 0$ and $\mathbf{a_j} * \mathbf{v} > 0$, set $\lambda^+ = 0$ else $\lambda^+ = \min\{\lambda_j : \lambda_j > 0\}$.
   If $\exists j$ such that $\lambda_j = 0$ and $\mathbf{a_j} * \mathbf{v} < 0$, set $\lambda^- = 0$ else $\lambda^- = \max\{\lambda_j : \lambda_j < 0\}$.
6. Generate u from a uniform distribution on (0,1) and set $\mathbf{y} = \mathbf{x} + (\lambda^- + u(\lambda^+ - \lambda^-))\mathbf{v}$.

The scheme above guarantees that $\mathbf{y} \in A$. Although, in our application a user-supplied feasible point is always available (Proll *et al.* 1993a), such a point can also be found through

(i) linear programming, or

(ii) local optimisation from an arbitrary starting point.

The SA process performs a local optimisation if a point $\mathbf{x}$ is accepted for which $f^* - f(\mathbf{x}) > \theta \mid f^* \mid$ where f^* is the current best value of the objective function and θ is a small positive value. It will stop when no change of more than $\alpha\%$ is observed in f after p successive decreases of temperature.

3. COMPUTATIONAL EXPERIMENTS

The algorithms described above were coded in Fortran77 and initially compared on a set of eleven problems arising in the distance analysis phase of the sensitivity analysis algorithm. These problems have many local minima, are subject to nontrivial constraints and include problems of higher dimension than is often encountered in the literature. The tests were performed on a 33 MHz 486 running in 386 mode under MS DOS Version 5.00 and Salford FTN77 Version 2.67. The results of these tests suggested that SA was the most promising algorithm for our purposes. To confirm this a second set of tests was undertaken comparing SA and MLSL on other problems arising in the sensitivity analysis algorithm. These tests were performed on a 66 MHz 486 running under MS DOS Version 6.00 and Salford FTN77 Version 2.67.

All test problems have a nonlinear objective function, which may be nonsmooth, and linear constraints comprising both inequalities and "non-overlapping" equalities.

3.1. Test problems 1

Problem statistics are listed in Table 1. Problems of type L1 have objective function max $\Sigma_{j=1}^{n} \mid x_j - w_j \mid$. Those of type L2 have objective function max $\Sigma_{j=1}^{n} (x_j - w_j)^2$, where w_j is a known constant and n, the number of variables. For L1 problems, the global maximum is known since it can be computed by integer linear programming (Proll 1997). For L2 problems, upper bounds can be computed manually. Global maxima for these, of course, can be computed by one of a number of algorithms (Pardalos and Rosen 1987), but codes were not available to us.

TABLE 1
Test problem statistics 1.

Problem No.	No. of variable	No. of constatins	Objective type
1	6	1	L1
2	6	3	L1
3	10	4	L1
4	11	6	L1
5	13	5	L1
6	28	7	L1
7	10	7	L2
8	11	7	L2
9	12	11	L2
10	20	12	L2
11	28	7	L2

3.2. Results and Discussion

In Tables 2-7 below, the column headings refer to the following:

Cycles: In MLSL: number of times a sample of N points is drawn;
In SA : number of times factor T is decreased;

Eval: Number of function evaluations;

NLO: Number of local optimisations performed;

DLM: Number of distinct local maximum values discovered;

Time: CPU time in seconds;

Value: Best objective value returned;

Ratio: Ratio of the best objective value returned to the value of the global optimum, if known, or to an upper bound on the global optimum.

We count as distinct those local maxima whose values differ by more than 1%. Tables 2 and 3 record results for MLSL with parameters $N = 100$, $\gamma = 0.2$, $\sigma = 4.0$. Tables 4 and 5 record results for TOPO with parameters $N = 100, k = 10$. Tables 6 and 7 record results for SA with parameters $L_0 = 10$, $\chi_0 = 0.9$, $m_0 = 5n$, $\delta = 0.1$, $\theta = 0.01$, $p = 5$, $\alpha = 1$. These values are largely those suggested by Dekkers and Aarts. In all cases, table entries represent average performance over 4 independent runs.

The MLSL code was allowed to run for a maximum of 10 cycles, corresponding to a sample size of 1000 points. The notation $p(q)$ under Cycles implies that q of the 4 runs were halted before the termination condition was reached. Table 2 shows the disappointing performance of MLSL in that, firstly, poor estimates of the global maximum were obtained for problems 1, 2 and 5 and secondly run times were long. This led us to consider using a composite objective which incorporates a measure of the infeasibility of the sample point with respect to the linear constraints. Rinnooy Kan and Timmer (1986) use a similar but more formal approach based on double exact penalty functions. Results using this objective are given in Table 3 and show some improvement in the robustness of the algorithm at the expense of run time.

TABLE 2
MLSL: normal objective.

Problem	Cycles	Eval	NLO	DLM	Time	Value	Ratio
1	1(0)	1094	13	1	0.80	4.87	0.696
2	4(0)	3717	37	2	2.76	8.88	0.740
3	10(4)	16971	142	14	18.46	44.00	1.000
4	10(4)	21881	145	26	31.61	73.92	1.000
5	10(4)	28271	171	25	45.72	54.34	0.938
6	10(4)	73398	199	23	342.10	106.94	0.948
7	10(4)	11722	121	28	15.92	0.12	1.000
8	10(4)	15238	145	28	22.50	0.13	1.000
9	6(0)	18545	87	6	36.97	225.18	1.000
10	10(4)	91665	195	18	411.36	461.20	1.000
11	10(4)	146636	197	13	761.09	691.10	1.000

For the topographical method, we chose values for the sample size and number of nearest neighbours commensurate with those used by Törn and Viitanen (1994). This allowed their values for threshold distance to be used as a basis for ours. This was necessary since there is no formula for obtaining the threshold distance and, in our application, it would not be feasible to experiment in order to find a "good" threshold distance. It should be noted that the times reported in Tables 4 and 5 do not include those required for sample generation. Generation times are substantial and far outweigh solution

TABLE 3
MLSL: composite objective.

Problem	Cycles	Eval	NLO	DLM	Time	Value	Ratio
1	8(1)	9012	111	7	6.57	6.50	0.929
2	8(1)	8107	89	7	5.97	10.47	0.873
3	10(4)	15654	143	14	17.80	44.00	1.000
4	10(4)	21374	142	27	30.75	73.92	1.000
5	10(4)	28148	177	22	45.33	58.01	0.998
6	10(4)	75812	200	27	344.19	106.39	0.943
7	10(4)	11941	126	35	15.39	0.12	1.000
8	10(4)	10924	115	35	13.31	0.13	1.000
9	6(0)	21653	102	6	42.44	225.18	1.000
10	10(4)	90974	193	18	405.48	461.17	1.000
11	10(4)	147281	197	13	758.11	691.16	1.000

TABLE 4
TOPO: normal objective.

Problem	Eval	NLO	DLM	Time	Value	Ratio
1	496	4	1	0.79	4.86	0.694
2	816	6	1	1.05	8.86	0.738
3	1317	8	5	1.93	40.00	0.909
4	2215	10	5	3.69	73.92	1.000
5	1722	8	6	3.06	49.22	0.847
6	2844	7	5	16.68	101.37	0.899
7	1093	7	5	1.99	0.12	1.000
8	630	4	3	1.47	0.12	0.953
9	1370	5	4	3.77	225.05	1.000
10	1824	4	3	8.46	461.19	1.000
11	6272	8	5	40.15	673.18	0.974

times, *e.g.* generating a sample of 100 points with a threshold distance of 1.750 for problem 10 required generating over 600 000 points and took over 300 secs. Given this load, we followed Törn and Viitanen's suggestion to sample from a unit hypercube, mapping resulting points onto the search space. This meant that we could use the same sample, for example, for problems 3 and 7 despite the fact that they reference different hypercubes. Comparison of Tables 4 and 5 shows that use of the composite objective did not bring the same benefits as for MLSL.

Tables 6 and 7 show that there is no strong effect on the SA process in starting from a point determined by a local optimisation, either in terms of value achieved or run time.

The results reported in Tables 2 through 7 are summarised in Figures 1 and 2. They show that SA, generally, provides a better estimate of the global optimum than both MLSL and TOPO. It also runs much faster than MLSL.

TABLE 5
TOPO: composite objective.

Problem	Eval	NLO	DLM	Time	Value	Ratio
1	1059	10	5	1.30	6.50	0.929
2	705	6	4	0.99	12.00	1.000
3	917	6	4	1.61	44.00	1.000
4	1051	6	5	2.22	72.82	0.985
5	1663	8	8	3.13	55.62	0.957
6	3059	9	8	15.16	95.31	0.845
7	577	4	4	1.25	0.11	0.875
8	842	6	5	1.94	0.12	0.953
9	1088	4	3	13.03	225.05	1.000
10	2244	5	4	10.20	461.19	1.000
11	7055	9	5	45.30	673.18	0.974

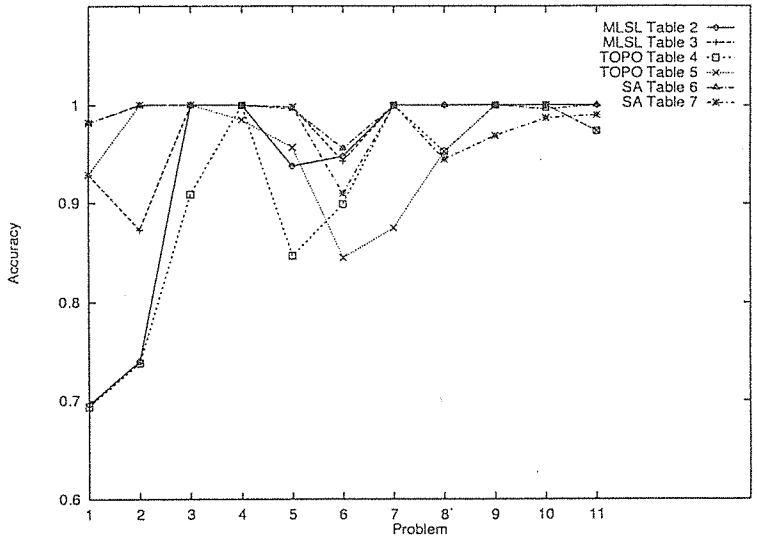

Figure 1. – Robustness comparison.

This is reinforced by the fact that, in most cases, the MLSL code was halted with the termination condition far from satisfied. The values achieved by SA suggest that there is little potential for the additional computational cost in allowing MLSL to run to termination to be offset against better solutions. Unsurprisingly, TOPO is the fastest of the three methods since the work required is simply to compare function values at a small number of points and perform a limited number of local optimisations. This speed is achieved at the expense of robustness. Its robustness could, in principle, be improved

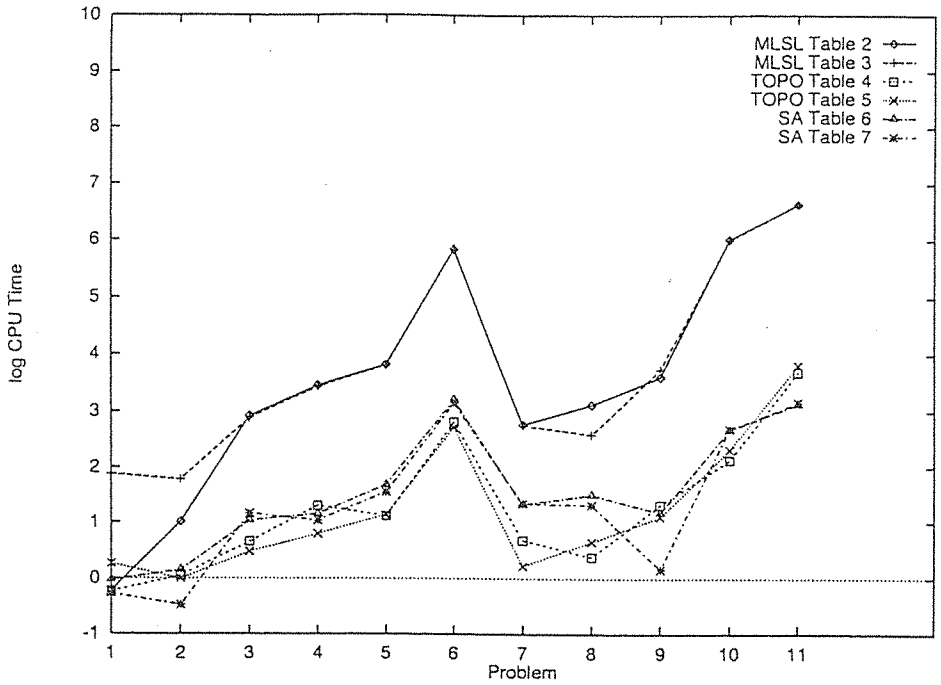

Figure 2. – Time comparison.

by choosing a smaller value for k. However experiments suggest that run time may rapidly increase. Clearly the lack of robustness of TOPO could be due to using the mapped sample. However, given the very substantial overhead in generating a sample directly, this is the only manner in which TOPO could be used in our application.

3.3. The problems 2

Problem statistics are listed in Table 8. Problems of type D have objective function min $(\psi_j(x) - \psi_i(x))$ where $\psi_j(x)$ is a bilinear evaluation function (Proll *et al.* 1993). They arise in checking whether alternative i dominates alternative j. Problems of type P have objective function min $\max\{\psi_j(x) - \psi_*(x) : j \neq *\}$ and arise in checking whether alternative j is potentially optimal. Problems of type C have objective function min $d(x) - p * \min\{0, \psi_j(x) - \psi_*(x)\}$. These problems arise in finding the nearest competitor of the currently optimal alternative, $*$.

Table 9 confirms the superiority of SA over MLSL in terms of time. The estimates of the global minimum obtained by these two methods do

TABLE 6
Initial point from local optimiser.

Problem	Cycles	Eval	NLO	Time	Value	Ratio
1	46	1653	3	0.97	6.88	0.982
2	52	1767	2	1.16	12.00	1.000
3	56	3010	1	2.81	44.00	1.000
4	47	2960	2	3.17	73.92	1.000
5	66	4950	4	5.42	57.93	0.996
6	56	9015	2	24.76	107.83	0.956
7	95	5007	2	3.80	0.12	1.000
8	98	5707	3	4.53	0.12	1.000
9	45	2932	1	3.26	225.05	1.000
10	91	9514	1	14.58	459.31	0.996
11	74	11108	1	22.95	691.16	1.000

TABLE 7
SA: initial point supplied.

Problem	Cycles	Eval	NLO	Time	Value	Ratio
1	42	1375	2	0.76	6.88	0.982
2	26	947	3	0.62	12.00	1.000
3	54	3295	7	3.20	44.00	1.000
4	30	2381	5	2.82	73.92	1.000
5	40	4045	4	4.73	58.06	0.998
6	20	6202	11	23.29	102.42	0.910
7	88	4443	4	3.84	0.12	1.000
8	74	4486	4	3.75	0.12	0.945
9	7	852	2	1.18	28.3	0.969
10	68	9140	31	14.72	454.78	0.987
11	23	7025	13	23.66	683.99	0.990

TABLE 8
Test problem statistics 2.

Problem No.	No. of variables	No. of constraints	Objective type
12	14	7	D
13	14	7	D
14	20	11	D
15	12	11	P
16	14	7	P
17	12	11	C
18	14	7	C
19	14	7	C
20	20	11	C

not differ by more than 1% except in one of the four trials for problem 19 for which MLSL achieved a value of 1.51 against a value of 0.92 for SA.

TABLE 9
SA vs. MLSL.

Problem	SA time	MLSL time
12	1.08	6.96
13	1.60	9.01
14	8.43	24.13
15	10.53	26.13
16	6.32	48.18
17	9.93	32.38
18	0.79	23.53
19	5.79	82.69
20	14.29	99.06

4. CONCLUSION

Our experience suggests that SA is robust and fast enough to be an appropriate tool for global optimisation in our application. This is likely to be due to the sequential sampling nature of SA which allows us, via an adaption of the coordinate directions method, to ensure that all sampled points are feasible. It has now been incorporated in our sensitivity analysis package where it has proved reliable. Algorithms such as MLSL and TOPO, which rely on uniform coverage of the search space, do not yet have a satisfactory mechanism for handling general constraints. Viitanen and Törn (1994) have suggested a mechanism for doing so in the topographical method but this is as yet unsupported by computational evidence and carries a much larger sampling overhead than the already substantial overhead incurred by TOPO. Thus it may well be worthwhile to explore whether our experience holds in more general contexts.

ACKNOWLEDGEMENTS

The work reported in this paper was supported by UK Science and Engineering Research Council grant GR/F45141, by NATO Collaborative Research Grant CRG910546 and by a MEC-British Council grant.

REFERENCES

1. H.C.P. BERBERE, C.G.E. BOENDER, A.H.G. RINNOOY KAN, C.L. SCHEFFER, R.L. SMITH and J. TELGEN, Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Math. Programming* **37** (1987) 184–207.
2. A. CORANA, M. MARCHESI, C. MARTINI and S. RIDELLA, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Software* **13** (1987) 262–280.

3. A. DEKKERS and E. AARTS, Global optimization and simulated annealing. *Math. Programming* **50** (1991) 367–393.
4. R.W. EGLESE, Simulated annealing: A tool for operational research. *European J. Oper. Res.* **46** (1990) 271–281.
5. S. FRENCH, Strategic decision analysis and group decision support, edited by P. DeWilde and J. Vandewalle, *Computer Systems and Software Engineering*. Kluwer Academic Publishers, Dordrecht (1992) 217-230.
6. K.G. MURTY and S.N. KABADI, Some NP-complete problems in quadratic and nonlinear programming. *Math. Programming* **39** (1987) 117–130.
7. P.M. PARDALOS and J.B. ROSEN, Constrained Global Optimization: Algorithms and Applications. Springer Verlag, Berlin (1987).
8. L.G. PROLL Stronger formulations of mixed integer programs: An example. *Internat. J. Math. Ed. Sci. Tech.* **28** (1997) 707-712.
9. L.G. PROLL, D. RIOS INSUA and A. SALHI, Mathematical programming and the sensitivity of multi-criteria decisions. *Ann. Oper. Res.* **43** (1993) 109–122.
10. L.G. PROLL, A. SALHI and D. RIOS INSUA, A parallel implementation of a mathematical programming framework for sensitivity analysis in MCDM. Presented at *APMOD93 - Applied Mathematical Programming and Modelling*. Budapest (1993).
11. A.H.G. RINNOOY KAN and G.T. TIMMER, The multi-level single linkage method for unconstrained and constrained global optimization, edited by D.F. Griffiths and G.A. Watson, Numerical Analysis Longman Scientific & Technical, Harlow (1986) 173-186.
12. A.H.G. RINNOOY KAN and G.T. TIMMER, Stochastic global optimization methods. Part I: Clustering methods. *Math. Programming* **39** (1987) 27–56.
13. A.H.G. RINNOOY KAN and G.T. TIMMER, Stochastic global optimization methods. Part II: Multi-level methods. *Math. Programming* **39** (1987) 57–78.
14. A.H.G. RINNOOY KAN and G.T. TIMMER, Global optimization, edited by G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, *Optimization*, Chapter IX. North Holland, Amsterdam (1989) 631–662.
15. R.Y. RUBINSTEIN, Generating random vectors uniformly distributed inside and on the surface of different regions. *European J. Oper. Res.* **10** (1982) 205–209.
16. F. SCHOEN, Stochastic techniques for global optimization: A survey of recent advances. *J. Global Optim.* **1** (1991) 207–228.
17. R.L. SMITH, Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Oper. Res.* **32** (1984) 1296–1308.
18. A. TÖRN and S. VIITANEN, Topographical global optimization, edited by C.A. Floudas and P.M. Pardalos, *Recent Advances in Global Optimization*. Princeton University Press (1992) 384–398.
19. A. TÖRN and S. VIITANEN, Topographical global optimization using pre-sampled points. *Global Optim.* **5** (1994) 267–276.
20. A. TÖRN and S. VIITANEN, Iterative topographical global optimization, edited by C.A. Floudas and P.M. Pardalos, *State of the Art in Global Optimization*. Princeton University Press (1996) 353–363.
21. A. TÖRN and A. ZILINSKAS, *Global Optimization*. Springer-Verlag, Berlin (1989).