A Direct Search Simulated Annealing Algorithm for Optimization Involving Continuous Variables

Montaz Ali

Turku Centre for Computer Science (TUCS), Turku, Finland

Aimo Törn

Department of Computer Science, Åbo Akademi University, Finland

Sami Viitanen

Department of Computer Science, Åbo Akademi University, Finland

Turku Centre for Computer Science
TUCS Technical Report No 97
March 1997

ISBN 951-650-970-3
ISSN 1239-1891

Abstract

A memory-based simulated annealing algorithm is proposed which fundamentally differs from the previously developed simulated annealing algorithms for continuous variables by the fact that a set of points rather than a single working point is used. The implementation of the new method does not need any properties of the function being optimized. The method is well tested on a range of problems classified as easy, moderately difficult and difficult. The new algorithm is compared with other simulated annealing methods on both test problems and practical problems. Results showing an improved performance especially for difficult problems are given.

Keywords: Global optimization, simulated annealing, stochastic, continuous variable, centroid

1 Introduction

A global optimization algorithm aims at finding a global minimizer, or its close approximation, of a function $f : S \subset R^n \to R$. A point x^* is said to be a global minimizer of f if $f^* = f(x^*) \leq f(x), \forall x \in S$. We assume that the function f is essentially unconstrained, i.e., all of its minimizers are in the interior of S. In practice, especially in applied sciences and in engineering, it is required that an approximation $\hat{x}^*$ of a global minimizer of f be found with $f(\hat{x}^*) - f^* < \varepsilon$. In recent years, a number of deterministic and stochastic algorithms have been proposed (Ratschek and Rokne [19]; Törn and Žilinskas [22]; Horst and Tuy [13] and Floudas and Pardalos [12]) for solving a global optimization problem. In this paper we restrict ourselves to a class of stochastic methods, namely the simulated annealing method.

Simulated annealing (SA) is a stochastic method for global optimization which has received much attention over the last few years. Recent interest began with the work of Kirkpatrick *et al* [14] and Cerny [8]. The physical annealing process is known in condensed matter physics as a thermal process for obtaining low energy states of a solid in a heat bath. This has been successfully modelled as a Monte Carlo simulation. In the fifties, Metropolis *et al* [15] introduced a simple method known as the Metropolis algorithm, to simulate a collection of atoms at a given temperature. Kirkpatrick *et al* showed how the model for simulating the annealing of solids proposed by Metropolis *et al* could be used for optimization problems, where the objective function to be minimized corresponds to the energy of the states of the solid. The name of the algorithm, therefore, derives from an analogy between solving an optimization problem and simulating the annealing of a solid. The Metropolis algorithm can be described briefly as follows.

Starting from an initial state, x, the system is perturbed at random to a new state, y, in the neighbourhood of x and the value of the objective function f_y is calculated. If the change, $\Delta f_{xy} = f_y - f_x$, represents a reduction in the value of the objective function, then the new state is accepted. If the change represents an increase in the objective function value, then the transformation is accepted with probability $\exp(-\Delta f_{xy}/T)$, where T is a control parameter corresponding to temperature in the analogy. By repeating this process for a large enough number of iterations Metropolis *et al* showed that the equilibrium distribution (Boltzmann distribution) for a system of particles is approached for a given temperature. Thus the Metropolis procedure from statistical mechanics provides a mechanism that enables a simulated annealing algorithm to avoid becoming trapped in a local minimum in its search for the global minimum.

Since the work of Kirkpatrick *et al* [14] much work has been done on simulated annealing for discrete variables and it has been used in a wide range of contexts. Brief descriptions of different modifications to the discrete simulated annealing algorithm can be found in a recent review by Eglese [11]. We will here discuss the simulated annealing algorithm for continuous variables.

Simulated annealing type algorithms for continuous variables have been proposed by fewer authors than for the discrete variable case. For example, Vanderbilt and Louie [23] introduced the idea of a covariance matrix for controlling the transition probability and the generation of random points to try to take into account the local structure of the objective function. Aluffi-Pentini *et al* [5] proposed a method to compute global minimizer(s) by following the paths of a system of stochastic differential equations. Bohachevsky *et al* [7] presented a generalised simulated annealing algorithm in which the approach followed is basically that of a random direction method, in which, at each step, a random point is generated on the surface of a sphere with a pre-fixed radius, centered on the current point. However, firstly these methods are somewhat different from the original simulated annealing approach to discrete optimization and secondly enough numerical evidences justifying their reliabilities are missing. Dekkers and Aarts [9] gave a simulated annealing algorithm for the continuous problem which is a direct transformation from the original discrete case to the continuous case. The convergence proof of their method is based on the equilibrium distribution of Markov chains (a Markov Chain in the simulated annealing algorithm is a sequence of trials). This method claims to be a reliable simulated annealing method for continuous optimization. The convergence of the simulated annealing algorithm in the continuous case is considered by Bélisle [6].

The principle complication introduced in going from the discrete to the continuous application of SA lies in the point generation mechanism, g_{xy}, for generating a point y from a point x. Dekkers and Aarts proposed the following generation mechanism (GM):

$$g_{xy} = \begin{cases} LS(x), & \text{if } \omega \geq q, \\ \frac{1}{m(S)}, & \text{if } \omega < q, \end{cases} \tag{1}$$

where q is a fixed number in $(0, 1)$, ω a uniform random number in $(0, 1)$ and $m(S)$ is the Lebesgue measure of the set S. Generating y from x by a uniform distribution on S is given by $g_{xy} = 1/m(S)$. $LS(x)$ denotes a local search that generates a point y in a descent direction from x. The local search from x is not a complete local search but only a few steps of some appropriate descent search. Thus $f_y < f_x$ but y is not necessarily a local minimum. With this GM, although $g_{xy} \neq g_{yx}$, it can still be shown that the method converges to a neighbourhood of the optimal solution. They have also chosen the acceptance probability, $A_{xy}(T)$, (i.e., the probability of accepting a point y if x is the current point and y is generated as a possible new point) to be the same as in the Metropolis algorithm, i.e.,

$$A_{xy}(T) = \min\{1, \exp(-(f_y - f_x)/T)\} \ . \tag{2}$$

Dekkers and Aarts' implementation (LSA) is, no doubt, a step forward but the drawback remains. The fact is, the LSA algorithm is still a randomization device, which by means of an acceptance/rejection criterion allows some ascent steps during the optimization process. It is therefore quite possible that at some fixed temperature level the procedure will visit the optimal solution but due to the acceptance/rejection mechanism it will leave the best solution and arrive at a worse solution. The LSA procedure, therefore, is memoryless in the sense that new solutions are accepted disregarding previously obtained intermediate results. Ali and Storey [3] have recently developed a new aspiration based simulated annealing algorithm with memory (ASA) where this issue has been addressed. A modification to the cooling schedule and to the acceptance criterion in [9] are made in order to make ASA memory based. An extensive numerical investigation of some global optimization algorithms including LSA and ASA has been carried out using a number of practical problems in [1] with the conclusion that ASA is much better than LSA both in terms of number of function evaluations (FE) and reliability in finding the global minimizer.

However, when some methods were directly compared on the practical problems requiring that the same estimate of the global minimum be achieved ([1]), it was found that some direct search methods needed much less FE than ASA. This is due to the fact that both LSA and ASA uses numerical differentiation in GM. On one hand numerical differentiation needs extra function evaluations giving rise to a higher total number of function evaluations, and on the other hand in many cases instability may occur. Therefore a memory based simulated annealing algorithm which does not use gradient based local search is essential. In this paper such a simulated annealing algorithm is proposed. In the next section we describe our new algorithm. In section 3 numerical results are compared both for test problems and for practical problems and in section 4 conclusions are made based on the numerical comparisons.

2 The Direct Search Simulated Annealing Algorithm (DSA)

Although the development of discrete simulated annealing and its direct transformation to simulated annealing for continuous variables by Dekkers and Aarts was motivated by the behavior of a physical system, we should make the scope of our investigation very clear. Any practical implementation of SA should permit us to always keep track of the best solution discovered during the search procedure so far. However, except ASA, little is presently known about how to implement a simulated annealing algorithm so that the best solution is retained. Even so, despite its success ASA implements the gradient based generation mechanism GM. This need for gradient calculations has been removed from DSA. In our new algorithm a feature is also included which allows us to retain the best solution during a run while preserving the cooling schedule proposed in [9]. Unlike other simulated annealing algorithms, including LSA and ASA, where only one point in the Markov chain moves in the search space, DSA associates a set of points with each Markov chain. The benefit of using this set of points is two-fold: firstly it helps keep track of the best solution and secondly some trial points are generated from the configuration of existing points in the set. How this is done will be explained below.

2.1 Generation Mechanism and Restoration of the Best Solution

The generation mechanism GM implemented for both LSA and ASA generates 25% of the trial points in each Markov chain by local descent requiring gradient calculation. In effect these points are accepted as they provide better function values. DSA, however, avoids this gradient calculation by using a subset of N ($N > n$) points stored in an array A like in the Controlled Random Search algorithms [16, 17, 18]. In CRS trial points are generated using such a set of current points as a base, with better points replacing the old ones in the set, and the algorithm terminates when the function values of all points in the set are within some preset limit of each other. Before describing in detail how this is done in DSA we first define the generation rule (GR) which replaces the local search in GM. The new generation mechanism for DSA is therefore given by

$$g_{xy} = \begin{cases} GR, & \text{if } \omega \geq q, \\ \frac{1}{m(S)}, & \text{if } \omega < q \end{cases} \quad . \tag{3}$$

The generation rule, GR, is calculated from $n+1$ points in A. A random selection of n points, $p_2, p_3 \ldots, p_{n+1}$, is first made from A and then the centriod G is calculated from $p_1, p_2 \ldots, p_n$; p_1 being the best point in A. A trial point x_p calculated in GR is then given by

$$x_p = 2G - p_{n+1} \quad . \tag{4}$$

We will now explain the methodology of our approach and justify the introduction of GR which replaces LS in GM.

In a simulated annealing algorithm a complete Markov chain is executed at each temperature. In any regular SA algorithm it is not unlikely that the function value of the final point in the Markov chain is worse than or is on the same level as its initial point. Given this, one finds it difficult to justify a gradient based local search which certainly spends many function evaluations in each Markov chain – given that 25% of the trial points in each Markov chain be generated by local descent. In the vicinity of the global minimizer such local search would produce a better approximation but this could also easily be achieved by refining the final solution. However, as shown in posterior, our proposed methodology could also produce good approximate solutions. The algorithm starts with N points and their function values stored in A and a single starting point of the initial Markov chain; both the N points and the starting point are selected at random from S. We call f_h the highest and f_l the lowest function values of points in A. If a point is accepted during the course of a Markov chain then this new point replaces the worst point in the array A and the new f_h and f_l are found in A before the process continues again. (We do not, however, order the whole array but keep the worst and the best points at the top and bottom of A respectively.) This process continues until a Markov chain ends. So, the current point in the Markov chain is always in A. We replace the Metropolis acceptance criterion by

$$A_{xy}(T) = \min\{1, \exp(-(f_y - f_h)/T)\} , \tag{5}$$

where the trial function value, f_y, is only compared with f_h, the worst point in A. If f_y is accepted then y and f_y replace the worst point and its corresponding function value f_h in A respectively. In this way, unlike the worst point, the current best point remains in A even if it is replaced by a new best. A random replacement could also be introduced in A excluding f_l, its best point. In this case rather than replacing only the worst point by the new accepted point, a random selection could be made within $N-1$ points of A and the selected point would then be replaced by the new accepted point. Our experiment showed that utilizing random replacement governed by a small probability worked well.

Under these circumstances, in our new methodology the search will be guided by the control parameter T as well as by the configuration of N points in A. Generating trial points by GR will be an exploration if the points are scattered and it is unlikely that the N points will form a dense cluster when the temperature is high. On the other hand when T is small, it is more likely that points will be accepted with function values less than f_h. In this case the N points in A would shrink to form a dense cluster. We now describe the cooling schedule used for the implementation of both DSA and LSA.

2.2 The Cooling Schedule

In any implementation of simulated annealing, a cooling schedule must be specified. The temperature parameter, T, is set to an initial value T_o; this is generally relatively high, so that most trials are accepted and there is little chance of the algorithm zooming in on a local minimum in the early stages. A scheme is then required for reducing T and for deciding how many trials are to be attempted at each value of T (the length of the Markov chain). Finally a stopping criterion is required to terminate the algorithm. The choice of a cooling schedule clearly has an important bearing on the performance of a SA algorithm. For the benefit of the reader we now summarize the cooling schedule as introduced by Dekkers and Aarts.

2.2.1 Initial value of the temperature

The basic assumption underlying the calculation of the initial value of the control parameter T is that it should be sufficiently large so that approximately all transitions are accepted at this value. This can be achieved by generating a number of trials, say m_o, and requiring that the initial acceptance ratio $\chi_0 = \chi(T_o)$ is close to one, where $\chi(T)$ is defined as the ratio between the number of accepted transitions and the number of proposed transitions. This initial value of T is then obtained from the following expression

$$T_o = \overline{\Delta f^+} \left(\ln \frac{m_2}{m_2 \chi_o + (1 - \chi_o) m_1} \right)^{-1} \tag{6}$$

where m_1 and m_2 denote the number of trials ($m_1 + m_2 = m_o$) with $\Delta f_{xy} \leq 0$ and $\Delta f_{xy} > 0$, respectively, and $\overline{\Delta f^+}$ the average value of those Δf_{xy}-values for which $\Delta f_{xy} > 0$.

2.2.2 Length of the Markov chains

No generally acceptable solution has been presented for this value, which decides how many trials has to be generated at each temperature. The optimal value of this parameter, which has to depend on the problem size, can not be determined in a rigorous way. Dekkers and Aarts choose the value to be

$$L_t = L_o \times n \ , \tag{7}$$

where n is the problem dimension. Note that this choice leads to a chain length which is constant for a given problem instance n.

2.2.3 Decrement of the control parameter

The new value of T_t (t is the temperature counter), T_{t+1}, is calculated from the following expression

$$T_{t+1} = T_t \left(1 + \frac{T_t \ln(1+\delta)}{3\sigma(T_t)}\right)^{-1} \tag{8}$$

where $\sigma(T_t)$ denotes the standard deviation of the values of the cost function at the points in the Markov chain at T_t. The constant δ is known as the distance parameter and determines the rate of decrease of the control parameter.

2.2.4 Stopping Criterion

The last problem to be addressed is that of finding a criterion to terminate the annealing. The stopping criterion is based on the idea that the average function value $\bar{f}(T_t)$ over a Markov chain decrease with T_t, so that $\bar{f}(T_t)$ converges to the optimal solution as $T_t \to 0$. If no changes have occured in $\bar{f}(T_t)$ in two consecutive Markov chains the procedure will stop. Therefore the simulated annealing algorithm is terminated if

$$\mid \frac{d\bar{f}_s(T_t)}{dT_t} \frac{T_t}{\bar{f}(T_o)} \mid < \varepsilon_s \ . \tag{9}$$

Here $\bar{f}(T_o)$ is the mean value of f at the points in the initial Markov chain, $\bar{f}_s(T_t)$ is the smoothed value of $\bar{f}$ over a number of chains in order to reduce the fluctuations of $\bar{f}(T_t)$, and ε_s is a small positive number, called the stop parameter.

2.3 The DSA Algorithm

The cooling schedule described above is a finite-time realisation as it implements the Markov chain of finite length at a finite sequence of (decreasing) values of the control parameter T. This implementation leads to the following DSA algorithm.

begin

initialize (T_o, x);

initialize an array A of N points with function values;

stopcriterion:=false;

```
while not stopcriterion do
begin
    for i:=1 to L_t do
    begin
        generate y from x using GR
        if f(y) - f_h <= 0
        then accept
        else if exp(-(f(y) - f_h)/T_t) > random[0,1)
        then accept;
        if accept then x := y;
        replace worst point in A, find new worst, best points;
    end
    lower T_t
end

end.
```

3 Numerical Results

The performance of DSA has been compared with LSA and ASA using classes of easy and hard problems ([21]) and it is measured by the criteria based on the number of function evaluations (FE), the running time (cpu) and the quality of the final result. We have coded DSA and run it on a SUN-Sparc ELC workstation. Random numbers were always generated using the well tested procedure given in [20]. The data for both LSA and ASA were taken from [1, 3, 4].

3.1 Choice of Parameters

In our numerical work we have used the same cooling schedule as was used by Dekkers and Aarts. To make the comparison more fair we have also used the parameter values as suggested by Dekkers and Aarts. We took $\chi_o = 0.9$, $\delta = 0.1$, $\varepsilon_s = 10^{-4}$ and $L_o = 10$. The generation mechanism (1) of LSA and ASA uses $q = 0.75$. We took $q = 0.50$ in (3), i.e., half way between the intensification and diversification of search.

3.2 Numerical Comparison

In this section the computational result of the DSA method is summarized. To begin with, we study some easy problems (according to the study done by Törn *et al* [21]). When the algorithm stops we consider the best point in A as the final solution.

We also compared DSA with LSA and ASA on these same test problems. Because LSA and ASA were implemented on different machines the standard unit time was introduced. One unit of time is the running time needed for 1000 function evaluations of the problem function Shekel 5 (S5, [10]). All results are summarized in Table 1. Here FE denotes the average number of function evaluations, cpu the running time in standard time, f_w^* the function value of the worst of three runs and ℓ refers to a local minimum. DSA was run with $N = 6(n+1)$. The data for both LSA and ASA were taken from [1, 3]. Note that the data for LSA and ASA are from a single run.

Table 1: Comparison of DSA with LSA and ASA for easy problems

f	DSA			LSA			ASA		
	FE	cpu	f_w^*	FE	cpu	f^*	FE	cpu	f^*
GP	663	1.20	3.01	1102	2.00	30.00^ℓ	734	2.66	3.00
S5	1218	2.67	-10.15	1120	2.22	-10.15	524	1.33	-10.15
S7	1591	2.94	-10.40	1122	2.66	-10.40	524	2.00	-10.40
S10	1244	3.44	-10.54	1179	2.67	-10.54	524	1.77	-10.54
H3	1066	2.67	-3.86	1252	4.00	-3.86	451	1.33	-3.86
H6	3549	30.80	-3.32	1817	7.33	-3.32	558	3.11	-3.32
P16	1920	5.22	0.00	4027	8.22	0.00	2123	6.21	0.00
Total	11251	48.94		11719	29.10		5438	18.41	

From the comparison between DSA and LSA over the easy problems of Table 1 it is clear that although the later uses gradient based local search the former performs very close to it in terms of function evaluations. In terms of cpu both LSA and ASA have outperformed DSA, this is mostly because of the overhead in using the array A. An overall comparison shows that ASA performs best of the three on these problems, with DSA and LSA closely matched.

In addition to the above easy test functions we also considered some highly complex problems from practical fields of application. They are concerned with the many-body potentials (MP) for silicon ($S_i(B)$ and $S_i(C)$) [2]. For these problems, a 3-particle problem has 3 variables, and the number of variables increases with 3 for each added particle. Some of these problems, especially the MP for five (P5) and six (P6) particles, are categorized as hard problems in [21]. Table 2 compares all three algorithms on MP for up to six particles. In the comparison we have only taken the number of function evaluations, the qualities of minima obtained and initial temperatures used. Numerical differentiation of MP has been used for both LSA and ASA. The data for both LSA and ASA were taken from [1, 4] and these were from a single run. The results for DSA were obtained for $N = 6(n+1)$ and the data presented for FE and T_o are averages of three runs. For DSA the worst function value (f_w^*) from three runs is given. The initial temperatures for LSA and ASA are the same as they use the same local search to calculate T_o from (6) by GM with $q = 0.75$. The T_o for DSA has also been found from (6) but simply by performing random moves, i.e., GM was implemented with $q = 1$.

Table 2: Comparison of DSA with LSA and ASA for hard problems

f	DSA			LSA			ASA	
$S_i(B)$	T_o	FE	f_w^*	T_o	FE	f^*	FE	f^*
P3	7615	782	-7.87	5556	13978	-7.87	13992	-7.87
P4	9159	3336	-15.70	7227	46458	-15.70	27988	-15.70
P5	9122	4581	-19.39	7052	112951	-20.31	36417	-20.31
P6	12170	18226	-24.35	10399	206166	-23.12	151109	-25.98
Total		26925			379553		229506	
$S_i(C)$	T_o	FE	f_w^*	T_o	FE	f^*	FE	f^*
P3	4060	736	-5.33	2761	12768	-4.95	6073	-5.33
P4	4922	3103	-7.99	3584	56365	-7.91	53485	-7.99
P5	4922	6331	-10.57	5258	114438	-11.30	113077	-10.14
P6	6156	18238	-12.61	5121	272927	-10.57	66274	-13.03
Total		28408			456498		238909	

Table 2 shows clearly that the initial temperature for LSA and ASA does not differ significantly from that of DSA. For MP the global minima of P3 and P4 are known for both $S_i(B)$ and $S_i(C)$ ([4]) and are achieved by all three, except LSA which failed to converge in P3 and P4 of $S_i(C)$. If the performance is measured solely by the number of function evaluations needed to achieve the same estimate of the global minimum then the new algorithm is significantly faster than the others. Given the same level of starting temperature, such a reduction in the number of function evaluations is a clear indication that a large number of function evaluations are needed in finding the descent direction numerically. The results certainly confirm the superiority of DSA over the other simulated annealing algorithms. However, when this criterion cannot be applied one could still make a comparison from the given data. In fact, the best values found by DSA during the three above runs were always better than the results obtained using LSA and ASA (with the exception of $S_i(C)$, P5 where LSA produced a slightly better result than DSA). The real difference, however, is seen from the total number of function evaluations performed, where the reduction is significant.

4 Discussion and Conclusion

The optimization of non-differentiable (including noisy or not exactly known functions) is a common problem occuring in various applications. Although numerical differentiations are used in many cases it is quite clear that this leads to much more function evaluations for finding descent directions as was shown for the many-body problems. Therefore DSA is a viable alternative to use when optimizing functions expensive to evaluate.

Although the DSA algorithm has a feature of Controlled Random Search ([16]) in having a relatively small array A, DSA differs from Controlled Random Search in two ways. First it can avoid the tedious stopping condition of agreement of N points of the array to certain decimal points, which is certainly not a sufficient condition for obtaining an accurate solution as shown in [1]. Second the algorithm is guided by the configuration of N points as well as by the temperature parameter which allows occasional ascents within the points of the array. It stops only when the temperature is small and the points in the array are relatively close.

References

[1] M. M. Ali, Some Modified Stochastic Global Optimization Algorith-

ms with Applications, Loughborough University of Technology, *Ph.D Thesis*, 1994.

[2] M. Ali and R. Smith, The Structures of small Clusters Ejected by Ion Bombardment of Solids, *Vacuum*, Vol.44, No 3/4, 377-379, 1993.

[3] M. Ali and C. Storey, Aspiration based Simulated Annealing Algorithms, to appear *Journal of Global Optimization*, 1996.

[4] M.M. Ali, C. Storey and A. Törn, Application of some stochastic global optimization algorithms to practical problems, submitted to the *Journal of Optimization Theory and Applications*, 1995.

[5] F. Aluffi-Pentini, V. Parisi and F. Zirilli, Global Optimization and Stochastic Differential equations, *Journal of Optimization Theory and Applications*, Vol.47, pp.1-16, 1985.

[6] C. J. P. Bélisle, Convergence Theorems for a class of Simulated Annealing Algorithms on R^n, *Journal of Applied Probability*, Vol.29, pp.885-895, 1992.

[7] M. E. Bohachevsky, M. E. Johnson and M. L. Stein, Generalized Simulated Annealing for Function Optimization, *Technometrics*, Vol.28, pp.209-217, 1986.

[8] V. Cerny, Thermodynamical approach to the travelling salesman problem, An efficient simulation algorithm, *Journal of Optimization Theory and Applications*, Vol.45, pp.41-51, 1985.

[9] A. Dekkers and E. Aarts, Global Optimization and Simulated Annealing, *Mathematical Programming*, 50, pp.367-393, 1991.

[10] L.C.W. Dixon and G.P. Szegö, (Eds.), *Towards Global Optimization 2*, North-Holland, Amsterdam, Holland, 1978.

[11] R.W. Eglese, Simulated Annealing, A tool for Operational Research, *European Journal of Operational Research*, Vol.46, pp.271-281, 1990.

[12] A. Floudas and M. Pardalos (eds.), *Recent Advances in Global Optimization*; Princeton University Press, USA, 1992.

[13] R. Horst and H. Tuy, *Global Optimization (Deterministic Approaches)*, Springer-Verlag, Berlin, 1990.

[14] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, Vol.220, No. 4598, pp.671-680, 1983.

[15] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, Equation of State Calculation by Fast Computing Machines, *Journal of Chemical Physics*, Vol.2, No. 6, pp.1087-1091, 1953.

[16] W. L. Price, A Controlled Random Search Procedure for Global Optimization, *Towards Global Optimization 2*, Edited by L.C.W. Dixon and G.P. Szegö, North-Holland, Amsterdam, Holland, pp.71-84, 1978.

[17] W. L. Price, Global Optimization by Controlled Random Search, *Journal of Optimization Theory and Applications*, 40, pp.333-348, 1983.

[18] W. L. Price, Global Optimization Algorithms for a CAD Workstation, *Journal of Optimization Theory and Applications*, 55, pp.133-146, 1987.

[19] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester, 1988.

[20] S. Tezuka and P. L'Ecuyer, Efficient and Portable Combined Tausworthe Random Number Generators, *ACM Transactions on Modelling and Computer Simulation*, 1, pp.99-112, 1991.

[21] A. Törn, M. Ali and S. Viitanen, Stochastic Global Optimization : Problem Classes and Solution Techniques, TUCS Technical Report No 37, Turku Centre for Computer Science, submitted to *Journal of Global Optimization*, 1996.

[22] A. Törn and A. Žilinskas, *Global Optimization*, Springer-Verlag, Berlin, 1989.

[23] D. Vanderbilt and S. G. Louie, A Monte Carlo Simulated Annealing approach to Optimization over Continuous Variables, *Journal of Computational Physics*, Vol.56, pp.259-271, 1984.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**