Simulated Annealing

As its name implies, Simulated Annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system.

The algorithm is based upon that of Metropolis *et al.* [1953], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by Pincus [1970], but it was Kirkpatrick *et al.* [1983] who proposed that it form the basis of an optimization technique for combinatorial (and other) problems.

SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm employs a random search which not only accepts changes that decrease objective function f, but also some changes that increase it. The latter are accepted with a probability

$$p = \exp\left(-\frac{\delta f}{T}\right), \tag{1}$$

where δf is the increase in f and T is a control parameter, which by analogy with the original application is known as the system 'temperature' irrespective of the objective function involved.

The implementation of the SA algorithm is remarkably easy. Figure 1 shows its basic structure. The following elements must be provided:

- a representation of possible solutions,
- a generator of random changes in solutions,
- a means of evaluating the problem functions, and
- an *annealing schedule* — an initial temperature and rules for lowering it as the search progresses.

Solution Representation & Generation

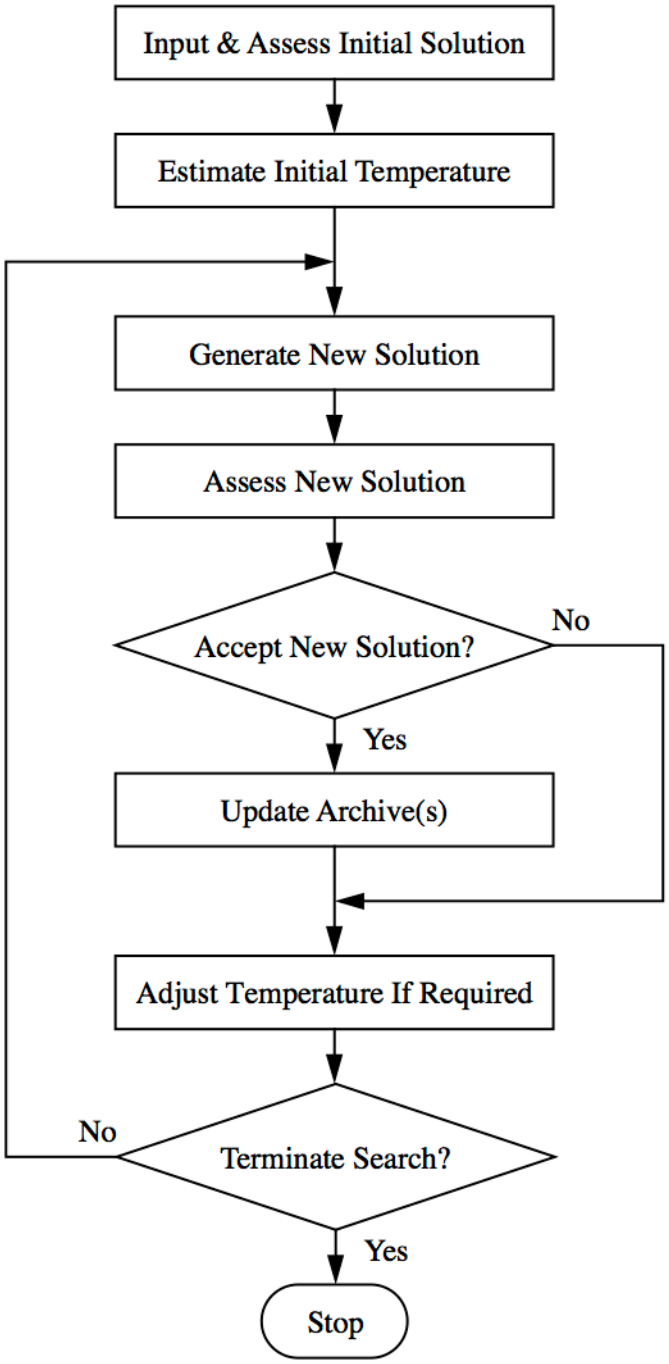
When attempting to solve an optimization problem using the SA algorithm, the most obvious representation of the control variables is usually appropriate. However, the way in which new solutions are generated may need some thought. The solution generator should

- introduce small random changes, and
- allow all possible solutions to be reached.

Continuous Control Variables

For problems with continuous control variables, a simple (if rather crude) way of generating

Figure 1: The Structure of the Simulated Annealing Algorithm.



new trial solutions is to use:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \boldsymbol{C}\mathbf{u} \ , \tag{2}$$

where $\mathbf{u}$ is a vector of uniform random numbers in the range $(-1, 1)$ and $\boldsymbol{C}$ is a (constant) diagonal matrix which defines the maximum change allowed in each variable.

Vanderbilt and Louie [1984] recommend that new trial solutions be generated according to the formula:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \boldsymbol{Q}\mathbf{u} \,, \tag{3}$$

where $\mathbf{u}$ is a vector of random numbers in the range $\left(-\sqrt{3}, \sqrt{3}\right)$, so that each has zero mean and unit variance, and $\boldsymbol{Q}$ is a matrix that controls the step size distribution. In order to generate random steps with a covariance matrix $\boldsymbol{S}$, $\boldsymbol{Q}$ is found by solving

$$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{Q}^T \tag{4}$$

by Cholesky decomposition, for example. $\boldsymbol{S}$ should be updated as the search progresses to include information about the local topography:

$$\boldsymbol{S}_{j+1} = (1-\alpha)\,\boldsymbol{S}_j + \alpha\omega\boldsymbol{X} \,, \tag{5}$$

where matrix $\boldsymbol{X}$ measures the covariance of the path actually followed and the damping constant α controls the rate at which information from $\boldsymbol{X}$ is folded into $\boldsymbol{S}$ with weighting ω. One drawback of this scheme is that the solution of equation (4), which must be done every time $\boldsymbol{S}$ is updated, can represent a substantial computational overhead for problems with high dimensionality. In addition, because the probability of an objective function increase being accepted, given by equation (1), does not reflect the size of the step taken, $\boldsymbol{S}$ must be estimated afresh every time the system temperature is changed.

An alternative strategy suggested by Parks [1990] is to generate solutions according to:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \boldsymbol{D}\mathbf{u} \,, \tag{6}$$

where $\mathbf{u}$ is a vector of uniform random numbers in the range $(-1, 1)$ and $\boldsymbol{D}$ is a diagonal matrix which defines the maximum change allowed in each variable. After a successful trial, i.e. after an accepted change in the solution, $\boldsymbol{D}$ is updated:

$$\boldsymbol{D}_{j+1} = (1-\alpha)\,\boldsymbol{D}_j + \alpha\omega\boldsymbol{R} \,, \tag{7}$$

where $\boldsymbol{R}$ is a diagonal matrix the elements of which consist of the magnitudes of the successful changes made to each control variable, i.e.:

$$R_{kk} = \left| D_{kk}\, u_k \right| , \tag{8}$$

and the damping constant α controls the rate at which information from $\boldsymbol{R}$ is folded into $\boldsymbol{D}$ with weighting ω. This tunes the maximum step size associated with each control variable towards a value giving acceptable changes. Suitable values of α and ω are 0.1 and 2.1.

If this formulation is to be used, the control variables should be rescaled so that within the optimization routine they vary over the same range $(0, 1)$ or $(-1, 1)$ and therefore have similar step sizes. For some problems it can also prove helpful to set upper and lower limits on the values of the elements of $\boldsymbol{D}$ to prevent individual values becoming too large or too small.

When using this strategy it is recommended that the probability of accepting an increase in f be changed from that given by equation (1) to:

$$p = \exp\left(-\frac{\delta f}{T\bar{d}}\right), \tag{9}$$

where $\bar{d}$ is the actual step size, i.e.:

$$\bar{d}^2 = \sum_{k=1}^{N} R_{kk}^2 , \tag{10}$$

where N is the number of control variables, so that $\delta f/\bar{d}$ is a measure of the effectiveness of the change made. As the size of the step taken is considered in calculating p, $\boldsymbol{D}$ does not need to be adjusted when T is changed.

Integer/Combinatorial Control Variables

For problems with integer control variables, the simple strategy whereby new trial solutions are generated according to the formula:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{u} , \tag{11}$$

where $\mathbf{u}$ is a vector of random integers in the range $(-1, 1)$ often suffices.

For combinatorial, or permutation, optimization problems, the solution representation and generation mechanism(s) will necessarily be problem-specific. For example, in the famous *traveling salesman problem* (TSP) of finding the shortest cyclical itinerary visiting N cities, the most obvious representation is a list of the cities in the order they are to be visited. The solution generation mechanism(s), or *move set*, must, obviously, be compatible with the chosen representation. For combinatorial problems, it is common for the move set to permute a small, randomly chosen, part of the solution. For example, the move set suggested by Lin [1965] for the TSP makes two types of change:

- a section of the route is removed and replaced by the same cities running in the opposite order, e.g. $87|164|253 \rightarrow 87|461|253$; or
- a section of the route is moved (cut and pasted) from one part of the tour to another, e.g. $87|164|253 \rightarrow 8725|164|3$.

Solution Evaluation

The SA algorithm does not require or deduce derivative information, it merely needs to be supplied with an objective function for each trial solution it generates. Thus, the evaluation of the problem functions is essentially a 'black box' operation as far as the optimization algorithm is concerned. Obviously, in the interests of overall computational efficiency, it is important that the problem function evaluations should be performed efficiently, especially as in many applications these function evaluations are overwhelming the most computationally intensive activity.

Some thought needs to be given to the handling of constraints when using the SA algorithm.

In many cases the routine can simply be programmed to reject any proposed changes which result in constraint violation, so that a search of feasible space only is executed. However, there are three important circumstances in which this approach cannot be followed:

- if there are any equality constraints defined on the system,
- if it is suspected that the problem is highly constrained, so that generating feasible solutions is not easy, or
- if the feasible space defined by the constraints is (suspected to be) disjoint, so that it is not possible to move between all feasible solutions without passing through infeasible space.

In any of these cases the problem should be transformed into an unconstrained one by constructing an augmented objective function incorporating any violated constraints as penalty functions:

$$f_A(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{T}\mathbf{w}^T\mathbf{c}_V(\mathbf{x}), \tag{12}$$

where $\mathbf{w}$ is a vector of nonnegative weighting coefficients and the vector $\mathbf{c}_V$ quantifies the magnitudes of any constraint violations. The inverse dependence of the penalty on temperature biases the search increasingly heavily towards feasible space as it progresses.

Annealing Schedule

Through equation (1), the annealing schedule determines the degree of uphill movement permitted during the search and is, thus, critical to the algorithm's performance. The principle underlying the choice of a suitable annealing schedule is easily stated — the initial temperature should be high enough to 'melt' the system completely and should be reduced towards its 'freezing point' as the search progresses — but "choosing an annealing schedule for practical purposes is still something of a black art" [Bounds, 1987].

The standard implementation of the SA algorithm is one in which Markov chains of finite length (i.e. random sequences of solutions) are generated at decreasing temperatures, so that a typical annealing schedule is of the form shown in Figure 2. The following parameters should therefore be specified:

- an initial temperature T_0;
- a final temperature T_f or a stopping criterion;
- a length for the Markov chains; and
- a rule for decrementing the temperature.

Initial Temperature

Kirkpatrick [1984] suggested that a suitable initial temperature T_0 is one that results in an average probability χ_0 of a solution that increases f being accepted of about 0.8. The value of

T_0 will clearly depend on the scaling of f and, hence, be problem-specific. It can be estimated by conducting an initial search in which all increases in f are accepted and calculating the average objective increase observed $\overline{\delta f}^+$. T_0 is then given by:

$$T_0 = -\frac{\overline{\delta f}^+}{\ln(\chi_0)} . \tag{13}$$

Alternatively, if σ_0 the standard deviation of the variation in the objective function observed during this initial search is calculated, then the formulation of White [1984], which has been found to be effective on many problems, can be used:

$$T_0 = \sigma_0 . \tag{14}$$

Final Temperature

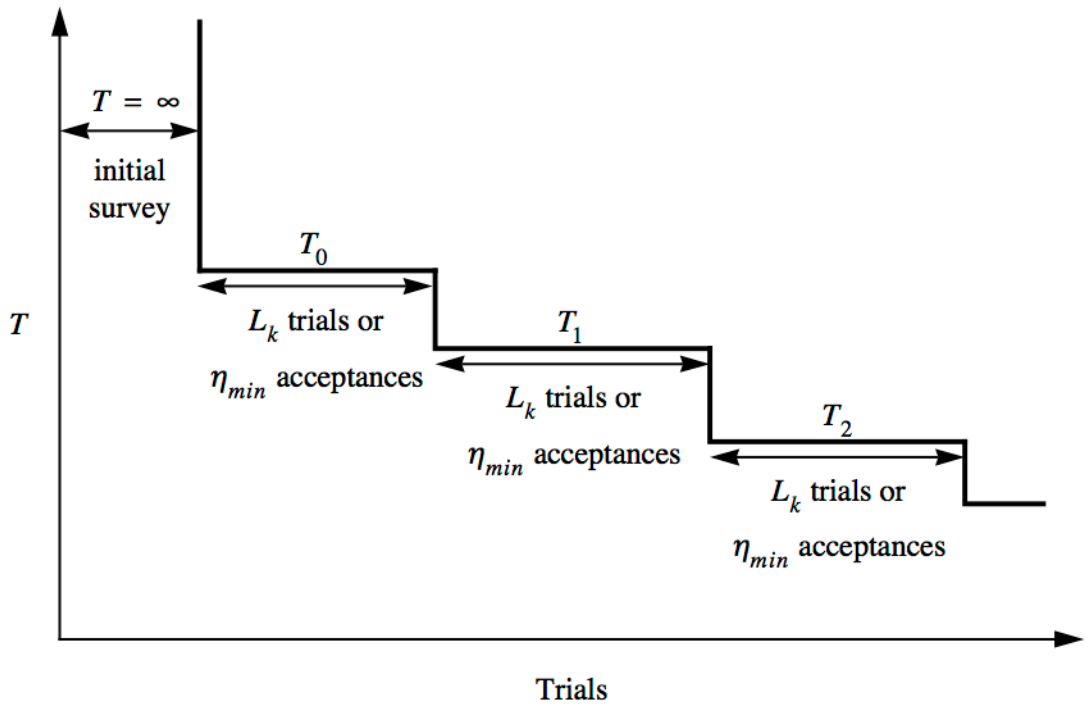
In some simple SA implementations the final temperature is determined by fixing

- the number of temperature values to be used, or
- the total number of solutions to be generated.

Alternatively, the search can be halted when it ceases to make progress. Lack of progress can be defined in a number of ways, but a useful basic definition is

- no improvement (i.e. no new best solution) being found in an entire Markov chain at one temperature, combined with
- the solution acceptance ratio (the fraction of solutions, both increasing and decreasing f, being accepted) falling below a given (small) value χ_f.

Figure 2: Typical Annealing Schedule.



Length of Markov Chains

An obvious choice for L_k, the length of the k-th Markov chain, is a value that depends on the size of the problem, so L_k is independent of k. Alternatively it can be argued that a minimum number of trials $\eta_{\min}$ should be *accepted* at each temperature. However, as T_k approaches 0, trial solutions are accepted with decreasing probability so the number of trials required to achieve $\eta_{\min}$ acceptances approaches ∞. Thus, in practice, an algorithm in which each Markov chain is terminated after

- L_k trials or
- η_{min} acceptances,

whichever comes first, is a suitable compromise. Typically $\eta_{min} \approx 0.6 L_k$.

Decrementing the Temperature

The simplest and most common temperature decrement rule is:

$$T_{k+1} = \alpha T_k \,, \tag{15}$$

where α is a constant close to, but smaller than, 1. This *exponential cooling scheme* (ECS) was first proposed Kirkpatrick *et al.* [1982] with $\alpha = 0.95$.

Many researchers have proposed more elaborate annealing schedules, most of which are in some respect adaptive, using statistical measures of the algorithm's current performance to modify its control parameters. These are well reviewed by van Laarhoven and Aarts [1987]. One relatively simple adaptive formulation, which has been found to be quite effective, is that due to Huang *et al.* [1986]. In this scheme the temperature is decremented as:

$$T_{k+1} = \alpha_k T_k \,, \tag{16}$$

with

$$\alpha_k = \max\left[0.5, \exp\left(-\frac{0.7 T_k}{\sigma_k}\right)\right] \,, \tag{17}$$

where σ_k is the standard deviation of the objective function values accepted at temperature T_k.

Restarts

If the SA search is no longer making progress, i.e. many iterations have passed since a new best solution was last found, a strategy by which the search is restarted from the best solution found thus far (while keeping the temperature unchanged) can prove effective. Such a 'return to base' strategy must be used with caution, as, if the conditions under which a restart is made are met too easily, only a limited part of the search space (possibly only a local minimum) will be explored.

Example: Travelling Salesman Problem

Figure 3 shows solutions of the EUR100 Travelling Salesman Problem (Figure 4) at four very different stages of the SA algorithm. The initial tour (a) (T = 17.85) looks very chaotic; as the value of the annealing temperature decreases the tours become less chaotic, (b) T = 4.46 and (c) T = 1.28, respectively; the final tour (d) (T = 0.06) is highly regular, which is reflected by a small cost — its value is very close to the known global optimum.

Figure 3: SA Solutions of the EUR100 Travelling Salesman Problem.

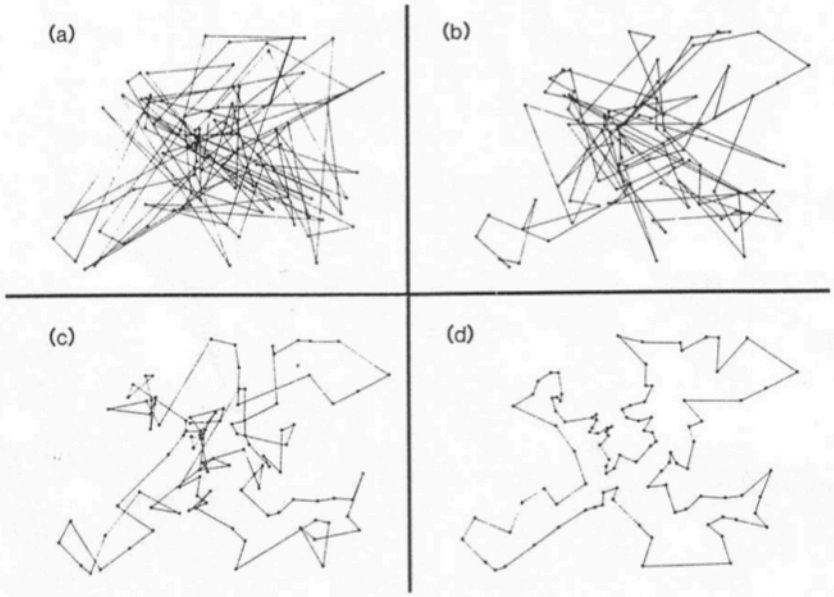
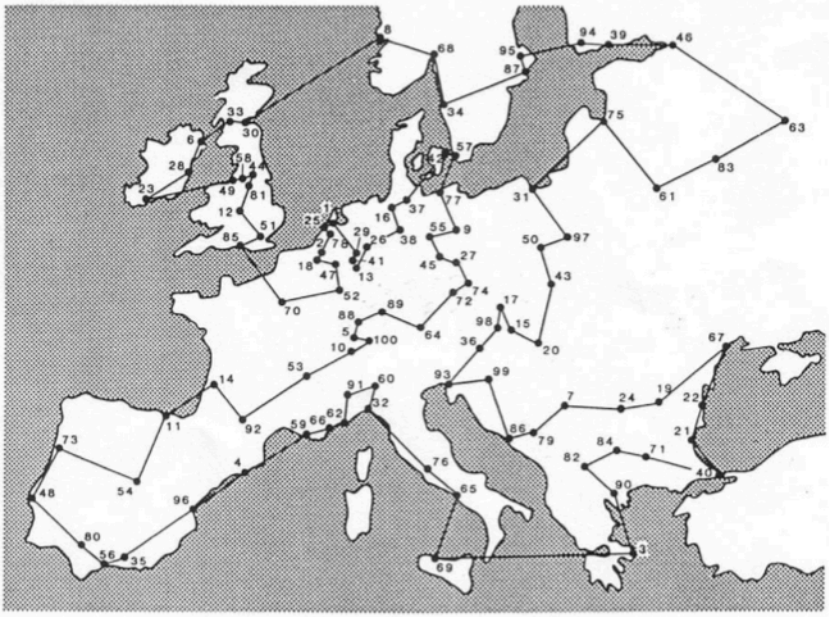


Figure 4: The Optimal Solution of the EUR100 Travelling Salesman Problem.



Example: Rosenbrock's Function $f(\mathbf{x}) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$

Figure 5 shows the progress of an SA search on Rosenbrock's function. Although one would not ordinarily choose to use SA on a problem which is amenable to solution by more efficient methods, it is interesting to do so for purposes of comparison. Each of the solutions accepted in a 1000 trial search is shown (marked by ○ symbols). The algorithm employed the adaptive step size selection scheme of equations (6) and (7). It is apparent that the search is wide-ranging but ultimately concentrates in the neighbourhood of the optimum.

Figure 6 shows the progress in reducing the objective function for the same search. Initially, when the annealing temperature is high, some large increases in f are accepted and some areas far from the optimum are explored. As execution continues and T falls, fewer uphill excursions are tolerated (and those that are are of smaller magnitude). The last 40% of the run is spent searching around the optimum. This performance is typical of the SA algorithm.

Figure 5: Minimization of Rosenbrock's Function by SA — Search Pattern.

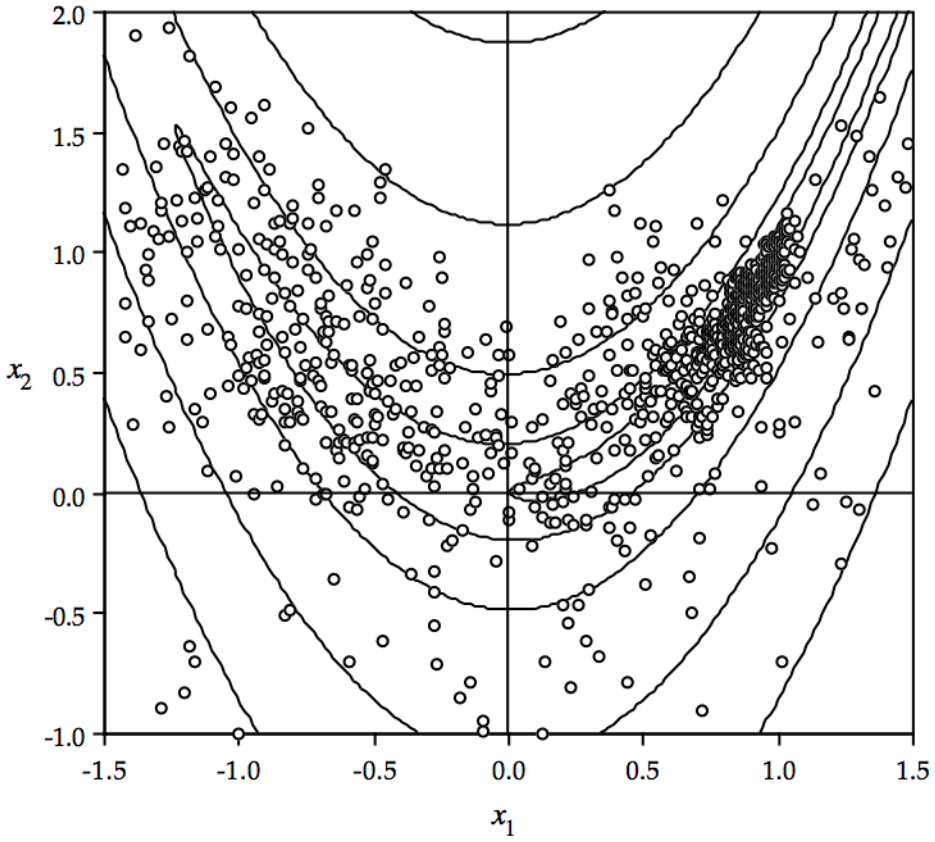
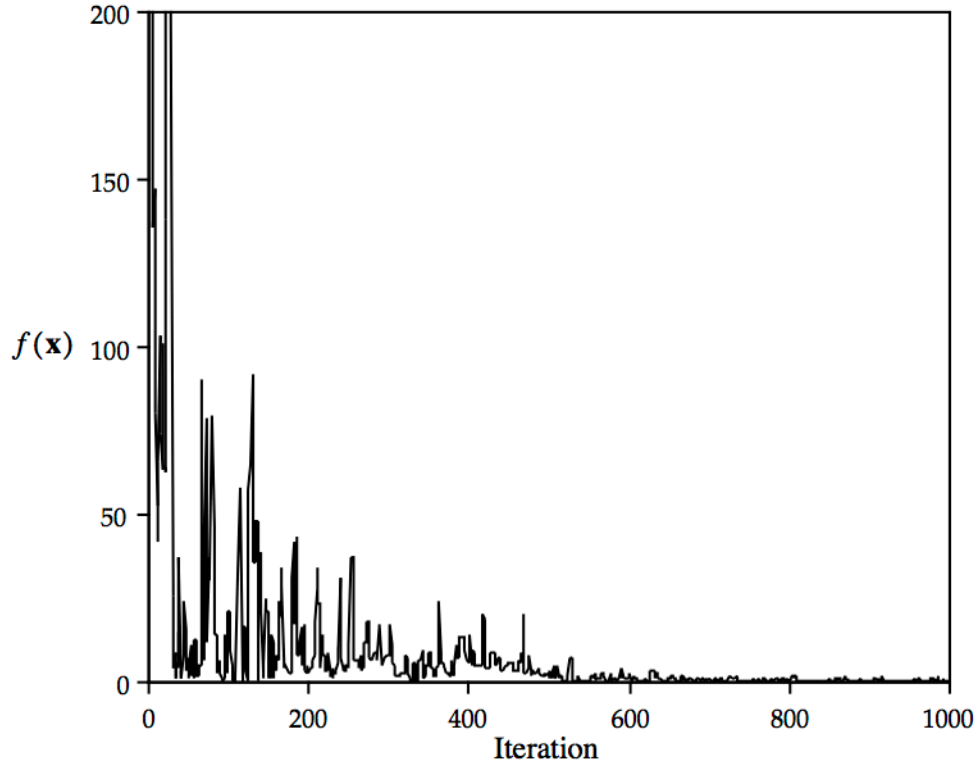


Figure 6: Minimization of Rosenbrock's Function by SA — Objective Reduction.



References

Arts, E.H.L., and J.H.M. Korst (1989) *Simulated Annealing and Boltzmann Machines*, Wiley (Interscience), New York.

Bounds, D.G., (1987) "New Optimization Methods from Physics and Biology", *Nature* **329**, 215-218.

Huang, M.D., F. Romeo and A. Sangiovanni-Vincentelli (1986) "An Efficient General Cooling Schedule for Simulated Annealing", *Proc. IEEE Int. Conf. Computer Aided Design*, 381-384.

Kirkpatrick, S., C.D. Gerlatt, Jr., and M.P. Vecchi (1982) "Optimization by Simulated Annealing", *IBM Research Report* **RC 9355**.

Kirkpatrick, S., C.D. Gerlatt, Jr., and M.P. Vecchi (1983) "Optimization by Simulated Annealing", *Science* **220**, 671-680.

Kirkpatrick, S., (1984) "Optimization by Simulated Annealing — Quantitative Studies", *J. Stat. Phys.* **34**, 975-986.

Laarhoven, P.J.M. van, and E.H.L. Aarts (1987) *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht, Holland.

Lin, S., (1965) "Computer Solutions of the Traveling Salesman Problem", *Bell Syst. Tech. J.* **44**, 2245-2269.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller (1953) "Equations of State Calculations by Fast Computing Machines", *J. Chem. Phys.* **21**, 1087-1092.

Parks, G.T., (1990) "An Intelligent Stochastic Optimization Routine for Nuclear Fuel Cycle Design", *Nucl. Technol.* **89**, 233-246.

Pincus, M., (1970) "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems", *Oper. Res.* **18**, 1225-1228.

Vanderbilt, D., and S.G. Louie (1984) "A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables", *J. Comput. Phys.* **56**, 259-271.

White, S.R., (1984) "Concepts of Scale in Simulated Annealing", *Proc. IEEE Int. Conf. Computer Design*, 646-651.