

Simulated annealing is a method that simulates a thermodynamic process in which a metal is heated to its melting temperature and then is allowed to cool slowly so that its structure is frozen at the crystal configuration of lowest energy. In this process, the atoms go through continuous rearrangements, moving toward a lower energy level as they gradually lose mobility due to the cooling. The rearrangements do not result in a monotonic decrease in energy, however. If the energy function has local minima, going uphill occasionally is desirable.

Metropolis et al. (1953) developed a stochastic relaxation technique that simulates the behavior of a system of particles approaching thermal equilibrium. (This is the same paper that described the Metropolis sampling algorithm.) The energy associated with a given configuration of particles is compared to the energy of a different configuration. In simulated annealing, the system moves through a sequence of states $s^{(1)}, s^{(2)}, \dots$, and the energy at each state, $f(s^{(1)}), f(s^{(2)}), \dots$, generally decreases. The algorithm proceeds from state $s^{(j)}$ by choosing a trial state r and comparing the energy at r , $f(r)$, with the energy at the current state, $f(s^{(j)})$. If the energy of the new configuration is lower than that of the previous one (that is, if $f(r) < f(s^{(j)})$), the new configuration is immediately accepted. If the new configuration has a larger energy, the new state is accepted with some nonzero probability that depends on the amount of the increase in energy, $f(r) - f(s^{(j)})$. In simulated annealing, the probability of moving to a state with larger energy also depends on a “temperature” parameter T . When the temperature is high, the probability of acceptance of any given point is high. When the temperature is low, however, the probability of accepting any given point is low.

Although the technique is heuristically related to the cooling of a metal, as in the original application, it can be used in other kinds of optimization problems. The objective function of the general optimization problem is used in place of the energy function of the original application. Simulated annealing is particularly useful in optimization problems that involve configurations of a discrete set, such as a set of particles whose configuration can continuously change, or a set of cities in which the interest is an ordering for shortest distance of traversal. Kirkpatrick, Gelatt, and Vecchi (1983) discussed various applications, and the method has become widely used since the publication of that article.

To use simulated annealing, a “cooling schedule” must be chosen; that is, we must decide the initial temperature and how to decrease the temperature as a function of time. If $T(0)$ is the initial temperature, a simple schedule is $T(k+1) = qT(k)$, for some fixed q , such that $0 < q < 1$. A good cooling schedule depends on the problem being solved, and often a few passes through the algorithm are used to choose a schedule. Another choice that must be made is how to update the state of the system; that is, at any point, how to choose the next trial point.

The probability of accepting a higher energy state must also be chosen. Although this probability could be chosen in various ways, it is usually taken

as

$$e^{-(f(r)-f(s^{(j)}))/T},$$

which is proportional to the probability of an energy change of $f(r) - f(s^{(j)})$ at temperature T and comes from the original application of Metropolis et al. (1953). In addition to the cooling schedule, the method of generating trial states, and the probability of accepting higher energy states, there are a number of tuning parameters to choose in order to apply the simulated annealing algorithm. These include the number of trial states to consider before adjusting the temperature. It is also good to adjust the temperature if a certain number of state changes have been made at the given temperature even if the limit on the number of trial states to consider has not been reached. Finally, there must be some kind of overall stopping criterion.

The simulated annealing method consists of the following steps:

1. Set $k = 1$ and initialize state s .
2. Compute $T(k)$ based on a cooling schedule.
3. Set $i = 0$ and $j = 0$.
4. Generate state r , and compute $\delta f = f(r) - f(s)$.
5. Based on δf , decide whether to move from state s to state r .
If $\delta f \leq 0$,
 accept;
otherwise,
 accept with a probability $P(\delta f, T(k))$.
If state r is accepted, set $i = i + 1$.
6. If i is equal to the limit for the number of successes at a given temperature, i_{\max} , go to step 1.
7. Set $j = j + 1$. If j is less than the limit for the number of iterations at the current temperature, j_{\max} , go to step 3.
8. If $i = 0$,
 deliver s as the optimum; otherwise,
 if $k < k_{\max}$,
 set $k = k + 1$ and go to step 1;
 otherwise,
 issue message that
 “algorithm did not converge in k_{\max} iterations”. ■

The traveling salesperson problem can serve as a prototype of the problems in which the simulated annealing method has had good success. In this problem, a state is an ordered list of points (“cities”), and the objective function is the total distance between all points in the order given plus the return distance

from the last point to the first point. One simple rearrangement of the list is the reversal of a sublist; that is, for example,

$$(1, \underline{2, 3, 4, 5, 6}, 7, 8, 9) \rightarrow (1, \underline{6, 5, 4, 3, 2}, 7, 8, 9).$$

Another simple rearrangement is the movement of a sublist to some other chosen point in the list; for example,

$$(1, \underline{2, 3, 4, 5, 6}, 7, 8, \uparrow 9) \rightarrow (1, 7, 8, \underline{2, 3, 4, 5, 6}, 9).$$

Both of these rearrangements are called “2-changes” because in the graph defining the salesperson’s circuit, exactly two edges are replaced by two others.

1. Write a program to use simulated annealing to solve the traveling salesperson problem for d cities. Your program should take as input a $d \times d$ matrix of distances between the cities. Use the simple cooling schedule, $T(k+1) = qT(k)$, and use random 2-changes as described above. The program should accept the beginning temperature $T(0)$ and the temperature reduction factor q . Finally, the program should also accept the control parameters i_{\max} , j_{\max} , and k_{\max} from the algorithm description above.

Although the program might seem rather complicated, it is not too difficult if it is built in separate modules. One special task, for example, is the implementation of the 2-change rule. You should write a module to perform these random changes and thoroughly test it before incorporating it in the full program.

2. Now, use your program to determine the optimal order in which to visit the cities in the mileage chart below. Assume that you return to the starting city (it does not matter which one it is).

Alexandria		↓																
Blacksburg	263		↓															
Charlottesville	117	151		↓														
Culpeper	70	193	47		↓													
Fairfax	15	249	102	55		↓												
Front Royal	71	203	124	44	57		↓											
Lynchburg	178	94	66	108	163	157		↓										
Manassas	28	238	91	44	23	45	157		↓									
Richmond	104	220	71	89	106	133	110	96		↓								
Roanoke	233	41	120	164	218	174	52	207	189		↓							
Williamsburg	148	257	120	133	150	177	165	140	51	215								

Distances Between Cities in Virginia

Reprinted from Gentle, James E. (2003), Random Number Generation and Monte Carlo Methods, second edition, Springer, New York.