6-1992

Simulated Annealing

Per Brinch Hansen
Syracuse University, School of Computer and Information Science, pbh@top.cis.syr.edu

SU-CIS-92-12

Simulated Annealing

Per Brinch Hansen

June 1992

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, NY 13244-4100

Simulated Annealing[1]

PER BRINCH HANSEN

Syracuse University, Syracuse, New York 13244

June 1992

This tutorial describes simulated annealing, an optimization method based on the principles of statistical mechanics. Simulated annealing finds near-optimal solutions to optimization problems that cannot be solved exactly because they are NP-complete. The method is illustrated by a Pascal algorithm for the traveling salesperson problem. The performance of the algorithm was measured on a Computing Surface.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: Probabilistic algorithms (Monte Carlo)

General Terms: Algorithms

Additional Key Words and Phrases: Simulated annealing, traveling salesperson

CONTENTS

INTRODUCTION
1. NAIVETE
2. ANNEALING
3. CONFIGURATIONS
4. COOLING
5. SEARCHING
6. REARRANGEMENT
7. PARAMETERS
8. COMPLEXITY
9. EXPERIMENTS
10. SUMMARY
ACKNOWLEDGEMENTS
REFERENCES

INTRODUCTION

This tutorial describes *simulated annealing*, an optimization method based on the principles of statistical mechanics. The method imitates the process by which melted metal forms an atomic lattice with minimal energy when it is cooled slowly. Simulated annealing finds near-optimal solutions to optimization problems, which cannot be solved exactly in reasonable amounts of computing time.

Simulated annealing was introduced by Kirkpatrick et al. [1983]. The method has been applied to a large number of optimization problems in science and engineering [Aarts and Korst 1989].

The *traveling salesperson* problem is probably the most famous combinatorial optimization problem: A salesperson must visit each of n cities once and return to the initial city. The aim is to find the shortest possible tour [Lawler et al. 1985].

The traveling salesperson and many other optimization problems belong to the class of *NP-complete* problems for which no efficient algorithms are believed to exist [Garey and Johnson 1979]. For these intractable problems we must be satisfied with approximation algorithms that find near-optimal solutions.

We will explain simulated annealing by developing a Pascal algorithm for the traveling salesperson problem. In a few minutes this algorithm finds a near-optimal tour of 100 cities by sampling fewer than one million of the 5×10^{150} possible tours!

1. NAIVETE

For the traveling salesperson, the most obvious idea is to examine all the $(n-1)!/2$ possible tours. Suppose this computation takes $n!$ μs. In that case, we can find a minimum tour of 15 cities in about two weeks. However, a 24-city problem would require 20 billion years, which is about four times the age of Earth [Sagan 1980]. So *exhaustive search* is out of the question except for very small problems.

Since it is impractical to consider all possible tours of n cities, we will examine only a random sample of tours. The idea is to make random changes of an initial tour in the hope of finding shorter and shorter tours. This statistical approach is an example of the *Monte Carlo* method of computing.

Consider first a *greedy search*, which always makes the choice that looks best at the moment. The initial tour is a randomly chosen sequence of the n cities. We now randomly select two cities and exchange them in the tour. The new tour is accepted if it is shorter than the previous one. The random exchange of cities continues until the tour no longer decreases.

Unfortunately, there is no guarantee that this algorithm will even come close to finding an optimum solution. In most cases, it will be trapped in a *local minimum* in the huge solution space.

2. ANNEALING

Annealing is the process of heating a metal until it melts and then lowering the

temperature slowly to allow the atoms sufficient time to form a uniform lattice with *minimal energy.* If the metal is cooled too quickly, the atoms form a slightly irregular lattice with higher energy due to internal stress.

Annealing can be viewed as a stochastic process which finds an arrangement of atoms that minimizes their energy. At high temperatures the atoms move freely and will often move to positions that temporarily increase the total energy. As the temperature is lowered, the atoms gradually move toward a regular lattice and will only occasionally increase their energy.

The occasional increase of energy plays a crucial role in annealing. These *uphill changes* enable the atoms to escape from local minima by increasing their energy temporarily. At high temperatures such jumps occur with high probability. At low temperatures they seldom occur.

The temperature must be lowered slowly to maintain *thermal equilibrium.* When the atoms are in equilibrium at temperature T, the probability that their total energy is E is proportional to $e^{-E/kT}$, where k is Bolzmann's constant [Feynman et al. 1989]. Consequently, the probability that the energy is $E + dE$ can be expressed as follows:

$$\text{Prob}(E + dE) = \text{Prob}(E)\ e^{-dE/kT}$$

In other words, the probability that the energy changes from E to $E + dE$ is $e^{-dE/kT}$. As the temperature decreases, so does the probability of energy increases.

Simulated annealing is a computational method that imitates nature's way of finding a system configuration with minimum energy. We will discuss this method in the context of the traveling salesperson problem. To emphasize the analogy between real and simulated annealing, we will use the terminology of statistical mechanics: Each tour is a possible *configuration* of the cities. The tour length represents the *energy* of the configuration. A variable T plays the role of *temperature.* Since T is a fictional entity, we replace Bolzmann's constant k by 1.

The aim is to lower the temperature slowly while changing the configuration until we reach near-minimal energy.

3. CONFIGURATIONS

In a plane a *city* is defined by two real coordinates:

```
type city =
  record x, y: real end
```

Algorithm 1 defines the Euclidean *distance* between two cities p and q.

```
function distance(p, q: city)
  : real;
var dx, dy: real;
begin
  dx := q.x − p.x;
  dy := q.y − p.y;
  distance :=
    sqrt(dx*dx + dy*dy)
end
```

Algorithm 1

A *tour* of n cities is defined by an array of cities:

```
type tour =
  array [1..n] of city
```

The salesperson visits the cities in numerical order $1, 2, \ldots, n$ before returning to city number 1.

The *length* of a tour is the sum of the distances between successive cities (Algorithm 2).

```
function length(var a: tour)
  : real;
var i: integer; sum: real;
begin
  sum := distance(a[n], a[1]);
  for i := 1 to n − 1 do
    sum := sum +
      distance(a[i], a[i + 1]);
  length := sum
end
```

Algorithm 2

4. COOLING

Simulated annealing begins at a high temperature $T_{\max}$, which is lowered in a fixed number of steps. At each step we keep the temperature T constant while searching for a shorter tour. The temperature is then reduced by a factor *alpha*, which is slightly less than 1 (Algorithm 3).

```
procedure anneal(var a: tour;
  Tmax, alpha: real; steps,
  attempts, changes: integer);
var T: real; k: integer;
begin
  T:= Tmax;
  for k := 1 to steps do
  begin
    search(a, T, attempts,
      changes);
    T := alpha*T
  end
end
```

Algorithm 3

The parameters of the search procedure will be explained below.

5. SEARCHING

In order to reach near-equilibrium at a given temperature T, we must examine many possible tours before lowering the temperature further. Algorithm 4 defines the Monte Carlo *search* for a shorter tour at temperature T.

The search procedure randomly selects two cities a_i and a_j and considers the possibility of exchanging them:

$$\text{select}(a,\ i,\ j,\ dE)$$

The select procedure also computes the resulting energy change dE.

The function value

$$\text{accept}(dE,\ T)$$

defines whether or not the energy change dE will be accepted at temperature T.

If the energy change is accepted, the tour is changed by exchanging cities a_i and a_j:

$$\text{change}(a,\ i,\ j)$$

The search continues until a fixed number of *changes* have been accepted. At high temperatures most changes are accepted, and the tour looks quite random. At low temperatures most random changes are likely to be rejected since they increase a tour that is already fairly short. To limit the search for shorter tours at low temperatures, the search algorithm also limits the total number of *attempts*.

```
procedure search(var a: tour;
  T: real; attempts, changes:
  integer);
var i, j, na, nc: integer;
  dE: real;
begin
  na := 0; nc := 0;
  while (na < attempts)
    and (nc < changes) do
  begin
    select(a, i, j, dE);
    if accept(dE, T) then
    begin
      change(a, i, j);
      nc := nc + 1
    end;
    na := na + 1
  end
end
```

Algorithm 4

6. REARRANGEMENT

Preliminary experiments showed that random *city exchanges* produced longer tours than the random *path reversals* suggested by Lin [1965]. Our final algorithm uses a variant of Lin's idea.

In a tour

$$a_1, a_2, \ldots, a_n$$

we randomly pick two cities a_i and a_j. The successors of these cities are denoted a_{si} and a_{sj}, respectively:

$$\ldots a_i, a_{si}, \ldots, a_j, a_{sj}, \ldots$$

A new tour is generated by reversing the order of the cities from a_{si} to a_j:

$$\ldots a_i, a_j, \ldots, a_{si}, a_{sj}, \ldots$$

The *select* procedure generates two random city indices i and j and computes the energy change dE that will be caused by reversing the path from city number si to city number j (Algorithm 5).

```
procedure select(var a: tour;
  var si, j: integer; var dE:
  real);
var i, sj: integer;
begin
  generate(i, j);
  si := i mod n + 1;
  sj := j mod n + 1;
  if i <> j then
    dE := distance(a[i], a[j])
      + distance(a[si], a[sj])
      - distance(a[i], a[si])
      - distance(a[j], a[sj])
  else dE := 0.0
end
```

Algorithm 5

The energy change is computed in constant time using the coordinates of the two cities and their successors. There is no need to compute the total length of any tour (except the final one).

The *change* procedure defines a path reversal of cities number i through j (Algorithm 6). The number of cities on the path is denoted n_{ij}. The path is reversed by swapping pairs of cities, starting at both ends of the path and working toward the middle. If n_{ij} is odd, the middle city is left alone.

```
procedure change(var a: tour;
  i, j: integer);
var k, nij: integer;
begin
  nij := (j - i + n) mod n + 1;
  for k := 1 to nij div 2 do
    swap(a[(i + k - 2) mod n + 1],
         a[(j - k + n) mod n + 1])
end
```

Algorithm 6

Algorithm 7.1 defines the criterion for *accepting* an energy change dE at temperature T. A tour of shorter (or unchanged) length is always accepted. A longer tour is accepted with probability $e^{-dE/T}$. The latter possibility is simulated by comparing the probability with a random number between 0 and 1.

```
function accept(dE, T: real)
  : Boolean;
begin
  if dE > 0.0 then
    accept := exp(−dE/T) > random
  else accept := true
end
```

Algorithm 7.1

Moscato and Fontanari [1989] found a simpler deterministic criterion that works just as well (Algorithm 7.2). This is the criterion used in our experiments.

```
function accept(dE, T: real)
  : Boolean;
begin accept := dE < T end
```

Algorithm 7.2

7. PARAMETERS

The choice of annealing parameters requires educated guessing and experimentation.

Consider a volume uniformly filled with atoms of the same type and imagine that the volume is divided into subvolumes. When the atoms form a lattice during annealing, the same stochastic process takes place in every subvolume. From a macroscopic point of view, it is as if every subvolume goes through the same average sequence of energy changes.

For simulated annealing, this intuitive argument suggests that the total number of attempted and accepted energy changes should be proportional to the number of cities, that is, $O(n)$.

The initial temperature $T_{\max}$ must be high enough to ensure that most energy changes are accepted. However, once the initial tour is random, it is a waste of computer time to attempt to make it "more random." So $T_{\max}$ should not be too high.

We will assume that the cities are placed on a square area. When you compare different tours, it is obviously the *relative* distances between cities that matter. We can therefore select a square of any dimension without changing the computation. We will use a square of n units. This choice makes the average density of cities independent of the problem size n.

On a square of area n, the distance between two successive cities cannot exceed the length of the diagonal, which is $O(\sqrt{n})$. A path reversal changes two distances in

the tour. Assuming that the initial tour is random, the average energy change caused by an initial path reversal is

$$dE = O\left(\sqrt{n}\right)$$

We will use an initial temperature of the same order of magnitude to make sure that most initial changes will be accepted:

$$T_{\max} = O\left(\sqrt{n}\right)$$

The final temperature $T_{\min}$ must be so low that most energy increases will be rejected when we have found a near-optimal tour. Beyond that point, nothing is gained by considering further changes. So $T_{\min}$ should not be too low.

If n cities are uniformly distributed on a square of area n, the average distances from each city to its nearest neighbors are $O(1)$. This is easy to see if you subdivide the square into n subsquares of area 1 and place a city on each subsquare.

When we are close to an optimal solution, the smallest possible energy increase is comparable to the distance between neighboring cities:

$$dE_{\min} = O(1)$$

Most energy increases will be rejected if the final temperature is of the same order of magnitude:

$$T_{\min} = O(1)$$

At this point the algorithm soon gets trapped in a local (near-optimal) minimum.

After the last search the final temperature is

$$T_{\min} = T_{\max}\ alpha^{\text{steps}-1}$$

The termination condition $T_{\min} = O(1)$ is satisfied when

$$(1/alpha)^{\text{steps}-1} = O(T_{\max})$$

By taking the logarithm on both sides we find

$$\begin{aligned} \text{steps} &= O(\log(T_{\max})) \\ &= O\left(\log\left(\sqrt{n}\right)\right) \\ &= O(\log n) \end{aligned}$$

These considerations and the folklore of simulated annealing led to the following *cooling schedule*:

Tmax	=	sqrt(n)
alpha	=	0.95
steps	=	20 ln(n)
attempts	=	100 n
changes	=	10 n

The procedure call

```
anneal(a, sqrt(n), 0.95,
  trunc(20*ln(n)), 100*n, 10*n)
```

replaces an initial tour a by a near-optimum tour (see Algorithm 3).

8. COMPLEXITY

Simulated annealing goes through $O(\log n)$ temperature steps. For each temperature the search examines $O(n)$ attempted and accepted changes. The computation rejects a change of the current tour in $O(1)$ time. If a change is accepted, the average path reversal involves $O(n)$ city exchanges. Consequently, the run time T_n of simulated annealing has the complexity

$$T_n = O((n^2 + n) \log n))$$

Since most steps take place at low temperatures, where most changes are rejected, the $O(n \log n)$ term is not negligible compared to the $O(n^2 \log n)$ term.

9. EXPERIMENTS

We reprogrammed simulated annealing in occam and tested it on a Computing Surface with T800 transputers using the random number generator of Park and Miller [1988].

The first test case was a *square grid* of n cities separated by horizontal and vertical distances of length 1 (Fig. 1).

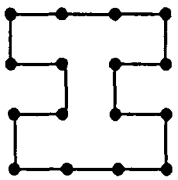

Fig. 1 A city grid

A tour of the cities consists of n distances, each of which is at least of length 1. So every tour is at least of length n. If n is the square of an even number, an optimal tour of length n exists (see Fig. 1). This test case, which has a known optimal solution, gives an idea of the accuracy of simulated annealing.

Since the algorithm is probabilistic in nature, we tried each experiment ten times with different initial values of the random number generator. The trials were performed in parallel on a Computing Surface with ten transputers [Brinch Hansen 1992].

Table I shows measured (and estimated) run times T_n (in minutes) for grids of 100 to 2500 cities. It also shows the shortest, average, and longest tours obtained from ten trials. The shortest tours are 0 to 4 percent longer than the optimal tours. The longest tours are 1 percent longer than the shortest ones.

Table I. City Grid

n	T_n m	E_{min}	E_{aver}	E_{max}
100	2 (2)	100	101	101
400	14 (14)	406	407	410
900	50 (48)	921	924	927
1600	130 (129)	1651	1657	1665
2500	280 (290)	2602	2611	2619

The second test case was a *random distribution* of n cities on a square area of n units.

Table II. Random Cities

n	T_n m	E_{min}	E_{aver}	E_{max}
100	2 (2)	76	78	80
400	14 (14)	307	310	314
900	50 (48)	717	723	728
1600	129 (129)	1276	1288	1297
2500	280 (290)	2009	2022	2037

The empirical formula

$$T_n = (0.26n + 240)n \ln(n)\ ms$$

defines the estimated run times shown in parentheses in Tables I and II.

10. SUMMARY

Simulated annealing is an effective method for finding near-optimal solutions to optimization problems that cannot be solved exactly. When the method is applied to the NP-complete problem of the traveling salesperson, it finds short tours of hundreds of cities in 2–50 minutes.

Simulated annealing exploits an interesting analogy between combinatorial optimization and the statistical behavior of a physical system that slowly moves toward a state of minimal energy. It is yet another example of a fundamental computation with a subtle theory and a simple algorithm.

ACKNOWLEDGEMENTS

I thank Jonathan Greenfield for helpful comments.

REFERENCES

AARTS, E., and KORST, J. 1989. *Simulated Annealing and Boltzmann Machines.* John Wiley, NY.

BRINCH HANSEN, P. 1992. Parallel Monte Carlo trials. School of Computer and Information Science, Syracuse University, Syracuse, NY.

FEYNMAN, R., LEIGHTON, R. B., and SANDS, M. L. 1989. *The Feynman Lectures on Physics.* Vol. I. Addison-Wesley, Redwood City, CA.

GAREY, M. R., and JOHNSON, D. S. 1979. *Computers and Intractability.* A guide to the Theory of NP-Completeness. W. H. Freeman, NY.

KIRKPATRICK, S., GELATT, C. D., and VECCHI, M. P. 1983. Optimization by simulated annealing. *Science 220*, pp. 671–680.

LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., and SHMOYS, D. B. (eds.) 1985. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley, Chichester, England.

LIN, S. 1965. Computer solutions of the traveling salesman problem. *Bell System Tech. J. 44*, pp. 2245–2269.

MOSCATO, P., and FONTANARI, J. F., 1989. Stochastic vs. deterministic update in simulated annealing. California Institute of Technology, Pasadena, CA.

PARK, S. K., and MILLER, K. W. 1988. Random number generators: good ones are hard to find. *Comm. ACM 31*, pp. 1192–1201.

SAGAN, C. 1980. *Cosmos.* Random House, NY.