

Physics 170

The Metropolis Algorithm

Statistical Systems and Simulated Annealing

When studying systems with a great many particles, it becomes clear that the number of possible configurations becomes exceedingly (unimaginably?) large very quickly. Even for extremely simple binary models, in which each particle may exist in one of two possible states, the number of configurations of the system grows extremely rapidly with N , the number of particles. In the statistical mechanics course we began by dividing the room into two equal halves, and inquired into the number of ways of partitioning the N air molecules between the two halves. The same mathematics describes a simple Ising model of magnetic spins, in which each spin-1/2 particle may either be “spin up” or “spin down” with respect to a chosen axis. Since each of the N particles may exist in 2 possible states, provided that we can distinguish the particles the number of configurations is

$$\Omega = 2^N \tag{1}$$

Imagine a model of a square lattice of such spins with $32 \times 32 = 2^{10}$ spins. This is absolutely puny on the scale of a macroscopic sample of matter. Nonetheless, there are

$$\Omega = 2^{2^{10}} \approx 10^{308}$$

distinct configurations of this system. The basic postulate of statistical mechanics is that in equilibrium, an isolated system is equally likely to be found in any one of these configurations that is consistent with known macroscopic properties (such as volume, number of particles, total magnetic moment, etc.). To compute the time-averaged properties one measures experimentally, we instead compute an ensemble average over *all* the accessible configurations.

More typically, a system is not energetically isolated from its surroundings, but may exchange energy with them. This exchange is characterized by a temperature T , which quantifies how willingly the environment shares its energy. The greater T , the more willing the environment is to give energy to the system; the smaller the temperature, the more the environment puts a premium on having the system in a low-energy state. The Boltzmann factor, $e^{-E/T}$, is proportional to the probability that the system will be found in a particular configuration at energy E when the temperature of the environment is T (I am using energy units for temperature here; divide by the Boltzmann constant to convert to kelvins, if desired). That is,

$$P \propto e^{-E/T} \tag{2}$$

To study a model of a thermal system, it is hopeless to imagine investigating each configuration, or to average over all of them, unless we can manage the sums analytically. When the particles don't interact with one another, we can indeed manage the sums analytically; this describes spins in an external magnetic field, ideal gases, black body radiation, lattice vibrations, and a few other simple systems. When the particles do interact to an appreciable extent, we can almost never perform the sums analytically. We have no choice but to seek an approximation, either analytically or numerically.

A Numerical Approach

The Metropolis algorithm is based on the notion of detailed balance that describes equilibrium for systems whose configurations have probability proportional to the Boltzmann factor. We seek to sample the space of possible configurations in a thermal way; that is, in a way that agrees with Eq. (2). We accomplish this by exploring possible transitions between configurations.

Consider two configurations A and B , each of which occurs with probability proportional to the Boltzmann factor. Then

$$\frac{P(A)}{P(B)} = \frac{e^{-E_A/T}}{e^{-E_B/T}} = e^{-(E_A-E_B)/T} \quad (3)$$

The nice thing about forming the ratio is that it converts relative probabilities involving an unknown proportionality constant (called the inverse of the partition function), into a pure number. In a seminal paper of 1953,¹ Metropolis *et al.* noted that we can achieve the relative probability of Eq. (3) in a simulation by proceeding as follows:

1. Starting from a configuration A , with known energy E_A , make a change in the configuration to obtain a new (nearby) configuration B .
2. Compute E_B (typically as a small change from E_A).
3. If $E_B < E_A$, assume the new configuration, since it has lower energy (a desirable thing, according to the Boltzmann factor).
4. If $E_B > E_A$, accept the new (higher energy) configuration with probability $p = e^{-(E_B-E_A)/T}$. This means that when the temperature is high, we don't mind taking steps in the "wrong" direction, but as the temperature is lowered, we are forced to settle into the lowest configuration we can find in our neighborhood.

If we follow these rules, then we will sample points in the space of all possible configurations with probability proportional to the Boltzmann factor, consistent with the theory of equilibrium statistical mechanics. We can compute average properties by summing them along the path we follow through possible configurations.

The hardest part about implementing the Metropolis algorithm is the first step: how to generate "useful" new configurations. How to do this depends on the problem. As an illustration, let's consider that classic of computer science, The Traveling Salesperson.

¹ N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, *J. Chem. Phys.* **21** (1953) 1087-1092.

The Traveling Salesperson

A traveling salesperson must visit N cities once and only once, returning to home base at the end of the journey. What is the optimal route she must take? “Optimal” depends on your priorities, but for simplicity I will take optimal to mean that the total distance traveled is minimized. More realistically, it might mean that some combination of flight time and cost is minimized.

In physical terms, we seek the ground state of a complicated system. That is, we would like to find the minimum energy configuration (or in this case, itinerary).

How many itineraries are there? For N cities, including the salesperson’s home base, there are $(N - 1)!$ possible paths that visit each city only once: $N - 1$ choices for the first city, $N - 2$ for the next, etc. For $N = 10$ cities, this is not too bad: only 362,880. It wouldn’t be too expensive to have the computer grind through each of them. However, the factorial functions grows *very* rapidly with its argument. Stirling’s approximation (in simple form),

$$n! \approx n^n e^{-n} \tag{4}$$

shows that the factorial increases faster than exponentially, and we can use it to show that for $n = 20$, $n! \approx 2 \times 10^{17}$. For a massively parallel system running 100 teraflops, this still takes hours. For $n = 40$, forget about it.

We will look for a solution to the traveling salesperson problem by analogy with the physical process of annealing.² Metals, semiconductors, and glasses are often filled with undesirable structural imperfections because of their thermal history or various mechanical process to which they have been subjected. Annealing them by raising their temperature significantly for a period of time, and gradually cooling provides a way to remove much of this damage. While hot, the atoms are more mobile and able to move out of high-energy metastable states to find lower-energy equilibrium states. As the temperature cools, they settle (ideally) into their lowest energy configuration, producing a perfect crystal.

In the salesperson problem, the path length L plays the role of energy, and the temperature parameter has the dimensions of a length. We will investigate its application by placing N dots at random within the unit square, as shown in Fig. 1, and seeking the path through all N cities that minimizes the total length L .

The results of a simulated annealing solution to the traveling salesperson problem with $N = 50$ is shown in Fig. 2. Each of the two “best” configurations was found after ap-

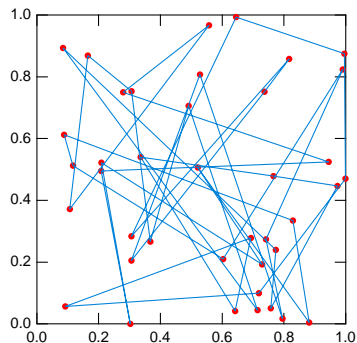


Figure 1: The traveling salesperson problem with $N = 40$ cities. The salesperson must visit each city once and only once, returning to her home town at the end. The indicated path reflects the order in which the cities were randomly placed.

²Kirkpatrick *et al.* were the first workers to apply the Metropolis algorithm to problems of function minimization. See, Kirkpatrick, S., Gelatt, C. D. Jr., and Vecchi, M.P., “Optimization by Simulated Annealing,” *Science* **220** (1983) 671-680.

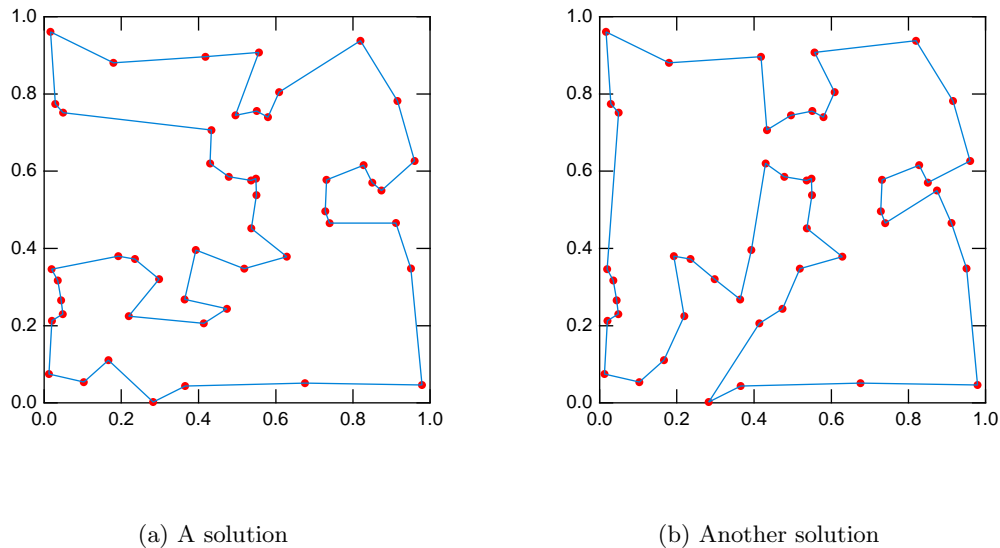


Figure 2: (a) A (near) optimal path for the salesperson to take in traveling to each of 50 cities on a closed circuit. The cities are randomly positioned in a square of size 1. The optimal path was sought by the method of simulated annealing. This path has a length of 6.09, and was the best path found after sampling 16,960 paths. (b) A somewhat longer path, found for the same set of 50 cities as in (a) merely by cooling down again from a slightly lower temperature, but starting at the configuration of (a). This one has length 6.21.

proximately 17,000 configurations were tested. This represents less than $3 \times 10^{-57}\%$ of all possible configurations, and yet the two paths look fairly convincingly close to optimal. In fact, configuration (b) is slightly longer than (a), and I cannot guarantee that (a) is indeed the shortest possible. However, the simulated annealing approach finds a local minimum that is typically quite close to the optimal configuration.

Implementation Details

There are two primary challenges to coding a solution to the Traveling Salesperson:

- Developing a method for producing a new test configuration B , given the current configuration A .
- Implementing a cooling schedule to reduce the temperature gradually as the simulation runs, thereby getting the system to settle into the lowest energy states.

Generating New Configurations

I have implemented two strategies for generating new configurations. Each step consists in trying one of the two strategies, which are chosen at random with equal probability. The first strategy picks two indices a and b in the range $0 \dots (N - 1)$ and reverses the path

between a and b . The values a and b are ordered so $a < b$; if $a = b$, then b is incremented by 1. In the case where $a = 0$ and $b = N - 1$, b is decremented by 1. The second method takes the cities in the range from a to b and inserts this segment into the remaining list of cities at a random position p . Care must be taken in this case not to try to insert the segment into itself.

For computational efficiency, in evaluating a new configuration we compute only the change in length associated with the proposed changes. For large N , the savings are significant.

Cooling Schedule

The second challenge is to determine a cooling schedule. Starting from a high temperature allows the system to explore all sorts of configurations. What is high? Well, $T = L_0$ is a very high temperature. A new configuration having double the original length L_0 would have only a $1 - e^{-1} = 63\%$ chance of being rejected. Virtually all configurations are allowed. The system will evolve far from its initial configuration in short order.

The downside to such a high temperature is that even if the system finds a very low-energy configuration, the odds that it will stick near it are negligible. So, after a few iterations at high temperature, it is appropriate to begin cooling. The lower the temperature gets, the more iterations the system will typically need to explore possibilities. Exactly what should be considered a low temperature depends a bit on the number of cities N .

Press *et al.* recommend that one generate several rearrangements to get a sense for the scale of the energy variations. Then set the initial temperature to a value several times larger than the largest ΔE typically encountered. Run at this temperature for some number of iterations (maybe $100N$) or until $10N$ successes occur. Then decrease the temperature by 10% and run some more. Stop when you get discouraged!

Applications

The Metropolis algorithm is a mainstay of Monte Carlo methods and statistical simulations. It is how properties of spins systems in 3 or more dimensions are studied, and it has been used to optimize the layout of hundreds of thousands of circuit elements on a silicon chip to minimize interference.³ See Chapter 17 of Gould and Tobochnik for lots more information and plenty of interesting projects to try.

³See section 10.9 of *Numerical Recipes*. I suspect the number should be millions by now!