10. Simulated annealing

[Gould-Tobochnik p. 607-610]

10.1. Main idea

In traditional processing of metals, a standard method to improve the quality of the metal is to heat it up to high temperatures, then slowly cool it down. Sometimes this is done in repetitive cycles. This method is called **annealing** (in Finnish "hehkutus", "lämpökäsittely", "karkaisu" or "mellotus"). What physically roughly speaking happens when this is carried out is that the annealing removes defects from the crystal. Thus the average potential energy per atom is decreased during the annealing.

So the whole thing can be considered a macroscopic energy minimization scheme.

This has lead to the use of an analogous process in minimization, called **simulated annealing.** The idea here is to introduce a variable T (which may not have anything at all to do with any thermodynamic temperature). Then a generalized Metropolis scheme is used to simulate the system at the temperature T, while T is gradually cooled down from some high starting temperature.

Simulated annealing can be used in a wide variety of optimization problems, and is especially useful in high-dimensional cases which are complex to handle by other methods.

10.1.1. Generalized Metropolis algorithm

As hinted at in the previous full section, it is easy to generalize the Metropolis algorithm to a non-atomic system. The numbering is identical to the previous presentation; hence steps 3 and 10 are now missing.

We consider a system with a state described by an N-dimensional vector $\mathbf{x}$, for which the function to be minimized is $f(\mathbf{x})$. The function should return a scalar quantity.

The generalized temperature T is just a scalar quantity which has the same dimensions as f.

0 a° Select the initial configuration $\mathbf{x}$.

0 b° Set the number of Monte Carlo steps $n_{MCS} = 0$

1° Choose a transition $\Delta\mathbf{x}$ at random

2° Calculate the function value before the transition $f_b = f(\mathbf{x})$

4° Do the trial transtion as $\mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$

5° Calculate the energy of the system after the transition $f_a = f(\mathbf{x})$

6° Calculate $\Delta f = f_a - f_b$, then :

- **7°** If $\Delta f \leq 0$ accept the state
- **8°** If $\Delta f > 0$:
  - **8 a°** Generate a random number u between 0 and 1
  - **8 b°** Accept the state only if $u < e^{-\Delta f/T}$

9° If the state is rejected, return to the previous state: $\mathbf{x} = \mathbf{x} - \Delta\mathbf{x}$

11° Set $n_{\mathrm{MCS}} = n_{\mathrm{MCS}} + 1$

12° If $n_{\mathrm{MCS}} < n_{\max}$ return to step 1

To use this for minimization, we start at some high T, with a magnitude comparable to the maximum possible values of Δf in the beginning, then gradually lower it to 0.

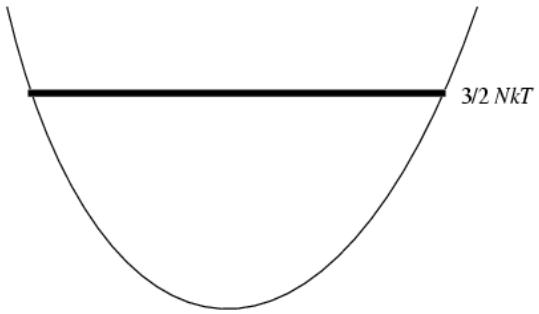
In doing this, it may well be a good idea to do make the transition size $\Delta \mathbf{x}$ depend on the "temperature", so that large changes are done at high temperatures, and small ones at lower T's.

10.1.2. How to use Metropolis for minimization

[Own reasoning and (limited) experience – *caveat emptor*]

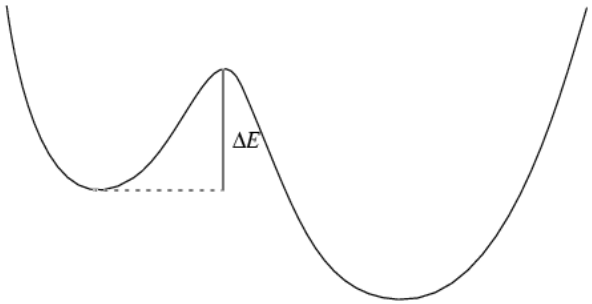
Despite the short explanation given above, it may not be immediately obvious why the Metropolis scheme, which is meant to describe particles in the NVT ensemble, actually can be used to minimize energy. Hence I now attempt to take a pedagogical approach to explaining qualitatively why and how this works.

(a) First of all let us consider an atomic system with a single potential well and thus a single minimum.



Recall that by the equipartition theorem, there will be equally much kinetic and potential energy in a monatomic system. Hence for a given T in an N-atom system, the potential energy will on average be $\frac{3}{2}NkT$ above the minimum (thick line in figure). Now from the figure above it is obvious that when we decrease T, we will approach the minimum and eventually reach it.

(b) Consider now only one step more in complexity, a system with two minima:



Now if we simulate at a temperature T we may be in either minimum. But we want of course to reach the lower minimum. If we then are in the higher minimum, we need to cross the energy barrier ΔE. This is an activated process, so the rate of crossing is

$$\nu = \nu_0 e^{-\Delta E/kT}$$

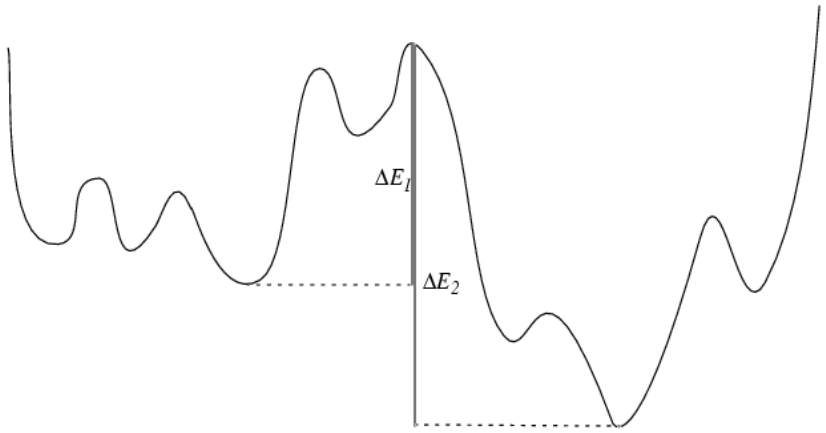
In Metropolis MC simulations we do not know the real time scale, and hence can not predict the rate. But in case we begin with a

$$kT > \Delta E$$

and simulate for many MC steps at this T, we are sure that we will have a large probability of crossing, since the exponential becomes > 1.

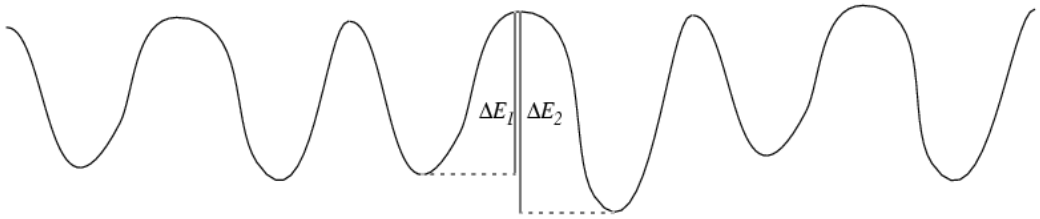
This illustrates that it is best to start at a quite high temperature in simulated annealing, one clearly higher than the barriers which may exist. If you do not know what barrier heights there may be, test your way forward.

(c) In reality, things are not likely to be anything close to this simple. In most systems of interest (which are complicated enough that it is worth using MC for them in the first place), the range of local minima and maxima is likely to form a complex landscape:



Whether we will ever be able to find the right minimum is not clear other than by extensive testing. But in a landscape such as the above, if we spend a long time in the temperature region $\Delta E_1 < kT < \Delta E_2$ there is good hope the system will find its way from the wrong local minimum to the global one, and not jump back.

(d) The worst case may look something like the following:



I.e. we have many minima with similar widths which are almost as deep. In this case, it is quite unlikely we will end up in the right minimum - when $kT \gtrsim \Delta E_2$ it may jump into any minimum with almost equal probability, and also out of the correct minimum. With extremely slow annealing in the range $\Delta E_1 < kT < \Delta E_2$ we should in principle find the right minimum, but because of the stochastic nature of the process, there are no guarantees.

In which of these cases is simulated annealing (MC) the most appropriate to use? Case **(a)** is so simple any minimization routine should find the right answer, and something like conjugate gradients [see Numerical Recipes] would probably be much more efficient at it. In case **(b)** MC almost certainly would work, but so might something simpler. In **(c)** MC may actually be the best tool to use. Case **(d)** is clearly a challenge for any minimization method, but at least in some

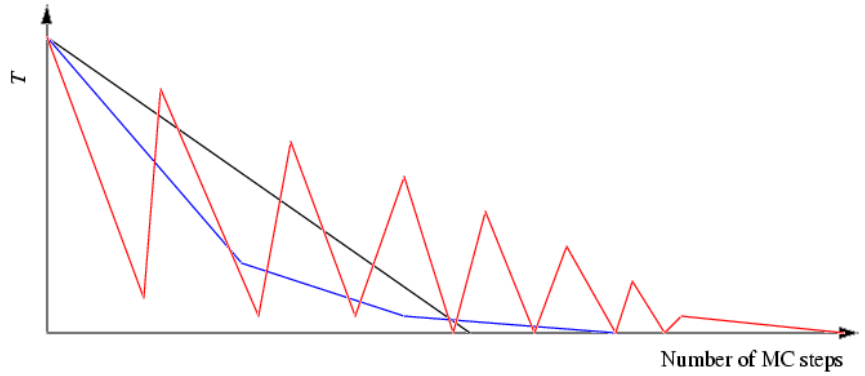
contexts genetic algorithms has been found to work well in this kind of a landscape of local and global minima.

But in many applications it may actually not be necessary to find the global minimum. Instead, a good local minimum may be adequate. In this case, the MC method will almost certainly work for any of the functional landscapes presented above.

It may also be very useful to combine different schemes. For instance one can use MC to get oneself into the correct minimum basin, then conjugate gradients to get accurately into the bottom of the minimum. Sometimes MD ran at high temperatures may be useful to locate reasonable candidates for minima, and then MC swapping between different minima candidates can be good for locating the global minimum. [Laura Nurminen, PhD thesis].

10.1.3. Different kinds of heat treatment schemes

[Own reasoning and (limited) experience – *caveat emptor*]



The easiest heat treatment scheme is of course just starting at a very high temperature and linearly coming down to 0. This may sometimes work quite well (black curve in figure).

If we want to have the final state with good accuracy, a linear approach starting from a very high temperature may come down to 0 too quickly for an accurate finding of the minimum. An easy way to solve this is to reduce the cooling rate at low temperatures (blue curve).

Finally, much more complicated schemes may sometimes work better, for instance a sawtooth-like decreasing pattern (red curve). The idea here is that each "tooth" in the pattern will remove some of unfavourable configurations, which are not likely to return on additional heating to slightly lower

temperatures than before. This may make removing other unfavourable configurations easier during the next stage.

10.2. Example: crystal structure of Cu

[Own development]

As a first application, we revisit the Morse Cu system discussed in the previous full section. There we did Metropolis MC simulation of 256 Cu atoms in a periodic system. When we started from a random atom configuration, we found that we were **not** able to reach the correct ground state configuration, the close-packed FCC lattice. But there the temperature was always constant. The results obtained were:

T	djump	Nsteps	fraction of failures	Eave
2000	0.2	1e6	0.412087	-3.061507
1000	0.2	1e6	0.268201	-3.210508
1000	0.4	1e6	6.099085E-02	-3.202355
600	0.2	1e6	0.172835	-3.269553
600	0.2	3e6	0.171711	-3.284650
300	0.2	3e6	7.716486E-02	-3.304991
300	0.1	3e6	0.311042	-3.316886
100	0.1	3e6	0.110436	-3.334477
100	0.03	10e6	0.577190	-3.347841

Let us now see if we can achieve a better result using simulated annealing. This is actually a very

natural place to use simulated annealing: after all, the whole method gets its name and idea from the thermal processing of metals!

For this system, I devised the following simulated annealing schemes:

Scheme 1:

0° Start at some high temperature T_0. Select $N_{\rm simann}$

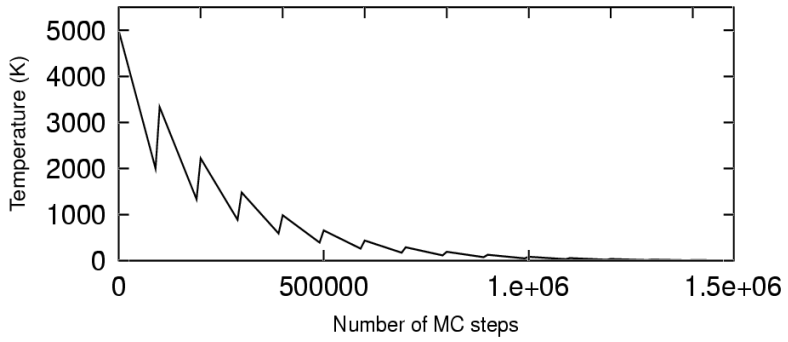
1° Over the next $N_{\rm simann}$ MC steps, lower the T linearly from T_0 to $1/3T_0$

2° Raise T to $2/3T_0$. Set $T_0 = T$

3° Set $N_{\rm anneal} = N_{\rm anneal} + 1$.

4° If $N_{\rm anneal} < 30$ return to step 1

So the temperature profile here looks e.g. as follows (example for $T_0 = 5000$ K and $N_{\rm simann} = 100000$):



This worked otherwise fine, but since the jump range $d_{\rm jump}$ was the same all the time, at high temperatures many jumps got accepted, but as the temperature lowered only a small fraction of them got. This lead to the natural extension to allow $d_{\rm jump}$ depend on T. This lead to modifying step 1° in scheme 1:

Scheme 2:

1 a$^\circ$ Over the next $N_{\rm simann}$ MC steps, lower the T linearly from T_0 to $1/3T_0$

1 b$^\circ$ Set $d_{\rm jump} = T/d_N$, with $d_N = 3000$

Otherwise as in scheme 1

This worked somewhat better, but did not lead to success either. This lead me to devise a couple of minor modifications of scheme 2, which helped a bit, but did not lead to full success either.

Scheme 3:

As scheme 2, but when $T < 800$ K reduce step down to $1/2$

Scheme 4:

As scheme 3, but use $d_N = 6000$

Here is a list of some of the results:

Torig	djump	scheme	simann_N	Efinal
1000	0.3	1	1000000	-3.36
3000	0.3	1	10000	-3.249
3000	0.3	1	100000	-3.324
4000	0.3	1	1000000	-3.43
5000	0.3	1	100000	-3.35
2000	-	2	1000000	-3.4412
2000	-	2	1000000	-3.37

3000	-	2	10000000	-3.4279
10000	-	2	100000	-3.348803
10000	-	2	1000000	-3.4413
20000	-	2	1000000	-3.36467
2000	-	3	1000000	-3.36
4000	-	3	1000000	-3.365
4000	-	3	1000000	-3.41
4000	-	3	10000000	-3.431
10000	-	3	100000	-3.35
10000	-	3	2000000	-3.364
4000	-	4	1000000	-3.40
20000	-	4	2000000	-3.39
50000	-	4	2000000	-3.444
50000	-	4	2000000	-3.436
50000	-	4	10000000	-3.43
100000	-	4	2000000	-3.38

Although we did not get to the right answer, a few things can be deduced from this. One is that raising the number of steps from 1 to 10 million per sequence does not seem to help. The other is that the best result is obtained when starting from very high temperatures, $\sim$ 50000 K. Although this is ridiculously high in experimental terms, it seems like this allows the system to jump over barriers which may be between the original, completely random state, and the correct fcc ground state.

Since some of the states are quite close to the right answer, I thought maybe one should try raising the temperature of these states pretty much to try to jump over the final barriers. This corresponds to more drastic temperature steps than in the original schemes. This lead to scheme 5, which is the same as scheme 2 but with different parameter values:

Scheme 5:

1 a$^\circ$ Over the next $N_{\rm simann}$ MC steps, lower the T linearly from T_0 to $1/10T_0$

1 b$^\circ$ Set $d_{\rm jump} = T/d_N$, with $d_N = 6000$

2° Raise T to $8/10T_0$. Set $T_0 = T$

Otherwise as in scheme 2

This actually lead to success in the first try. Starting from 50000 K and doing 2 million steps in every heating interval lead to a potential energy of -4.499, exactly the right minimum.

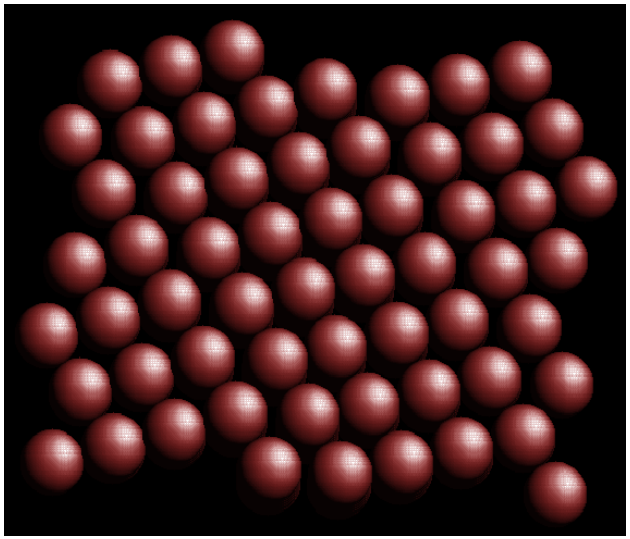
Here is a summary of the scheme 5 results:

```
Torig   djump   scheme  simann_N        Efinal
-----   -----   ------  --------        ------
```

```
10000 -5 100000 -3.357
10000 -5 300000 -3.367
50000 -5 300000 -3.4687
50000 -5 300000 -3.370
50000 -5 300000 -3.360
50000 -5 1000000 -3.440
50000 -5 2000000 -3.4989   SUCCESS!
50000 -5 2000000 -3.436
```

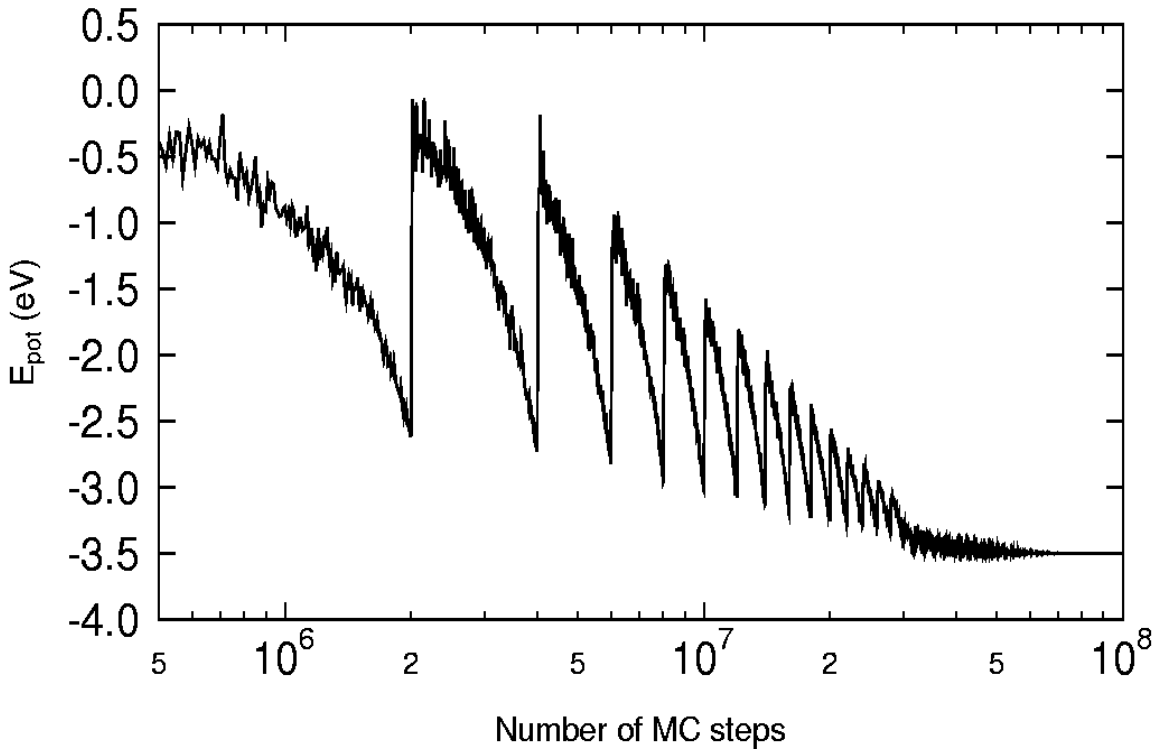
So even this formula does not always succeed, and needs plenty of steps to do so. But it is clearly better than the previous, in that states with a potential energy < -3.45 eV are achieved several times.

The coordinates looked as follows:



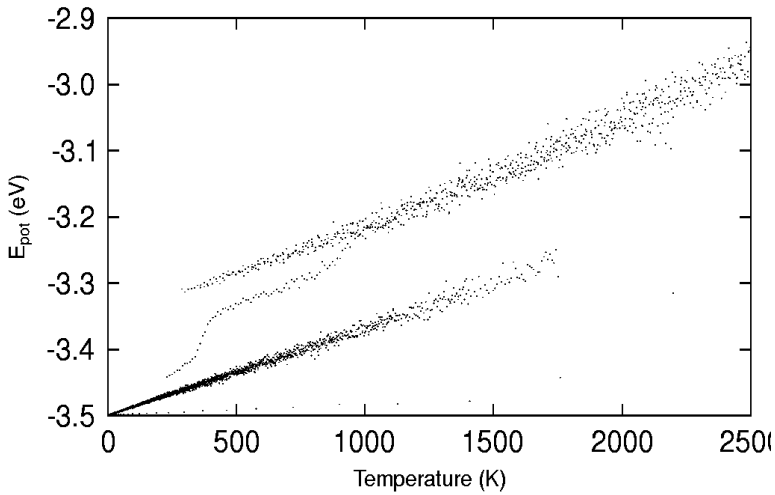
This is clearly a close-packed atom structure. The weird shape of the boundary is due to the fact that the structure need of course not be perfectly aligned with the periodic boundary, and the rotation which shows the atomic structure can lead to funny-looking effects. But this is of course not a real problem. If you look very closely, you will note that there are several stacking faults here. The energy difference between a close-packed structure, and one with stacking faults, is so small that there is virtually no hope of avoiding stacking faults, other than by pure good luck.

Let us still look at the development of the potential energy as a function of the number of MC steps for the successful run:



Otherwise the behaviour is pretty much as expected, but note that just before 3×10^7 MC steps, there is a sudden extra drop. Apparently at this stage, the system jumped over the crucial barrier between a false and the true minimum.

Another way to look at this is to plot the energy vs. the temperature. Since this of course follows a non-monotonous dependence, it is easiest to plot it as just a scatter plot (points):



The wide bands are regions where the structure stays essentially the same, and the energy is just a simple function of the temperature. The thin band of points in between must be the crucial final phase transformation.

10.2.1. Comments on result

So we saw that by using some 100 million MC steps, we can obtain a defect-free close-packed crystal structure starting from random atom coordinates with simulated annealing. Undoubtedly some more optimization of the method could have lowered this somewhat, maybe to 10 million steps.

This is still pretty d–ned slow even for this simplest possible case. It means that in any real case, where the right answer is not known and the interactions and crystal structure more complex, this approach is probably too slow to ever obtain a ground state with certainty.

There are two other optimization schemes which might be better. One is genetic algorithms, which has been shown to obtain the ground state structure of several fullerenes correctly. The other is molecular dynamics - one could do similar heating-cooling stages with this. Since atom motion is correlated in MD, defective regions in a crystal can "see" each other via a strain field interaction, which makes it easier for them to recombine with each other.

The downside with MD is that it is not clear how one should extend that to other than particle systems. As we shall see in the next example, it is by contrast very easy to use Metropolis MC for systems which have nothing to do with atoms.

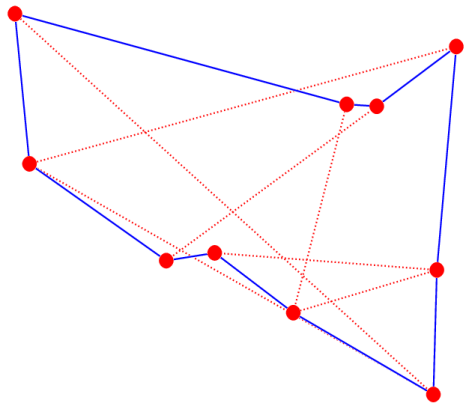
10.3. The traveling salesman problem

[Gould-Tobochnik p.?607-610]

A completely different field of science where simulated annealing can be highly useful is scheduling and design. The general idea here is to optimize some parameter, for instance the amount of material needed to construct something.

The classical example of a problem in this field is the *traveling salesman problem*. The problem can be stated as follows. A salesman (or -woman) is to travel through N cities, so that every city is visited exactly once, and the trip ends where it began. The question is, what is the optimal route between the cities.

This can be illustrated as follows:



The large red dots illustrate the positions of 10 "cities" chosen randomly in 2D. The red dotted lines show the initial, random route between the cities, and the blue solid lines the route optimized by simulated annealing. It is immediately obvious the simulated annealing route is much shorter.

This problem is actually a classical problem in graph theory and computer science. It is called an NP-complete problem. The NP refers to non-polynomial, which essentially means that there is currently no known way in which this class of problems could be solved in polynomial time (although the formal definition is much more complex). This again means that the time required for the solution of the problem with N objects can not be

$$O(N^a)$$

where a is some finite positive real number. Any known way for exact solution of the traveling salesman problem requires time of the order of

$$e^N$$

Hence it is easy to see that even for quite small N the time required for an exact solution becomes preposterous – e.g. for $N = 20$ we get an estimate of the order of 500 million.

One efficient way to solve the traveling salesman problem is by simulated annealing, utilizing the generalized Metropolis algorithm given above. The vector $\mathbf{x}$ is just the route between the cities, and the function $f(\mathbf{x})$ returns the length of the route, in whatever length unit used (the program does

not actually need to know the units). The "temperature" T will have the same units as f, and should start from some value comparable to the highest possible change in f, or maybe even higher to allow for efficient jumping over barriers in the beginning.

The trial can simply be changing two pairs of along the path. I.e. if we for instance have as the current path

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow \cdots$$

we could change it to

$$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow \cdots$$

by exchanging the positions of the particles 2 and 4.

10.3.1. Hints for practical implementation

Since the path always just goes from one city to another, the whole walk can very simply be expressed as an array of N integers given the order of moving through the cities.

The only thing to keep in mind in the implementation and printing of the results is to always count and print out also the last step, which returns the salesperson to the starting point.

When doing the trial by exchanging two cities a and b, it is not necessary to recalculate the length of the whole path. Only calculating the change by looking at the old and new distances to and from both a and b is enough.

Finally, if we are limited to a fairly small number of cities N, it is probably computationally most efficient to make in the beginning of the simulation an $N \times N$ matrix which tabulates the distance from every city to every other city. Then in the actual simulation any required distance can be directly obtained by reading in the value from the matrix.

To give you a feeling of all this, and the Metropolis method, you get to code this yourself in the exercises.