Back to heuristics for global optimisation

Simulated Annealing

Simulated Annealing is an heuristic method for finding ‘good’ solutions to difficult optimisation problems. The technique was first applied to combinatorial optimisation problems by Kirkpatrick *et al.* [2] and independently by Cerny [1]. They showed that ideas used in simulating the physical process of *annealing* as applied to solids, first proposed by Metropolis *et al.* [3], could be extended to solve optimisation problems in general, especially combinatorial optimisation problems.

Physical Analogy and the Metropolis Algorithm

Annealing is a thermal process used in *condensed matter physics* to obtain solids with low energy states. The process has two main steps, melting the solid and then **carefully** reducing the temperature so that the particles of the solid arrange themselves into the *ground state*. When the solid is melted to a liquid, the particles are arranged randomly, whereas in the ground state of the solid particles are arranged in an organised lattice that achieves a minimum energy configuration. If cooling is not done carefully, the ground state will not be reached, instead *quenching* occurs, whereby the solid takes on a *meta-stable state* with less than minimum energy. In either case we have reached a steady *frozen* state.

Metropolis *et al.* [3] modeled this physical process with the introduction of an algorithm based on *Monte Carlo* techniques. (Monte Carlo techniques involve the selective application of randomisation.) The algorithm generates a sequence of simulated states of the solid by successively perturbing states in the following manner:

- A particle of the present state is given a small displacement to produce a new state and the resulting change in the energy of the system, *delta*, is calculated.
- If *delta*<0 then the new state is accepted as the current state, otherwise the state is accepted according to the *Metropolis criterion (test).* The change is accepted with a certain probability given by:

  $$\exp\left(\frac{-\delta}{\kappa_B T}\right),$$

  where *T* is the *temperature* and *kappa* a physical constant known as the *Boltzmann constant*.
- If the temperature is slowly and carefully lowered (achieved in the algorithm by carrying out many iterations at each stage) then the solid will reach thermal equilibrium at each temperature. Thermal equilibrium is categorised by a Boltzmann probability distribution where the probability of being in state *i* with *E(i)*, the energy of state *i*, at temperature *T* is:

  $$\frac{1}{Z(T)}\exp\left(\frac{-E_i}{\kappa_B T}\right),$$

  and *Z(T)* is a normalising partition function defined as:

  $$Z(T) = \sum_i \exp\left(\frac{-E_i}{\kappa_B T}\right).$$

The Simulated Annealing Algorithm

The application of the Metropolis algorithm to optimisation problems, originally suggested by Kirkpatrick *et al.* [2] and Cerny [1], is possible after making an analogy between the problem and the Metropolis algorithm determined by the following equivalences.

- Solutions in the optimisation problem are equivalent to states of the physical system.
- The cost of a solution is equivalent to the energy of a state.
- A neighbouring solution is equivalent to a change of state.
- The *control parameter* is equivalent to the temperature.
- An heuristic solution to the problem is equivalent to a frozen state.

An optimisation problem can be solved using the Metropolis algorithm by repeatedly sampling neighbourhoods randomly, starting from an initial solution, and accepting as a new solution any configuration of superior quality and any inferior solution that passes the Metropolis test. The Metropolis test is dependent on the control parameter and the magnitude of the increase in such a way that small increases (assuming minimisation is the goal) are more likely to be acceptable than large increases. Also, when the temperature is high, the algorithm is likely to accept almost all configurations regardless of quality, but when the temperature is low, only solutions of a better quality are likely to pass the test. By accepting some inferior solutions the annealing algorithm seeks to avoid becoming trapped in a local optimum that is far from globally optimal (i.e. quenched). For a combinatorial optimisation problem, assuming minimisation of the objective function is the goal, a formulation is: minimise *f(x)* where *x* is a element of *S* and *S* is the solution space, *x* a solution, *f(x)* the objective function, and *N(x)* the neighbourhood structure. If the number of iterations spent at each temperature is identified as *L* and the temperature as *T* then a generalised annealing algorithm for such a problem is shown figure 2.6.

```
begin
   select an initial solution i ∈ S
   select an initial temperature T
   while stopping criterion not true
       count=0
       while count < L
           choose a neighbour j ∈ N(i)
           δ = f(j) − f(i)
           if δ < 0 then
             i := j
           elsif random[0, 1) < exp (−δ/T) then
             i := j
           end if
           count=count+1
       end while
       reduce temperature T
   end while
end
```

Figure 2.6. Pseudo-algorithm for the Simulated Annealing method. The algorithm repeatedly searches the neighbourhood of the current solution for a configuration to become the new current solution. Better quality configurations are accepted automatically, solutions of a lower quality are accepted probabilistically, dependent on T and the magnitude of the difference in solution quality.

Since *T* is now a control parameter, Boltzmann's constant has no analogy in the optimisation problem and *kappa* has been dropped from the Metropolis test. However it is still usual to refer to *T* as the temperature and the rate and manner by which *T* is reduced as the *cooling (or annealing) schedule*. The speed of convergence of the algorithm will depend largely on the choice of *L* and *T* for each iteration.

The Simulated Annealing algorithm in figure 2.6 is both simple and generally applicable since it is essentially a *local search* algorithm with the additional feature of the control parameter regulating some amount of uphill moves. The success of the algorithm has been accounted for by applying a Markov Chain model to the technique and demonstrating the asymptotic convergence properties of Simulated Annealing.(Any system that consists of random transitions from one state to another over an index (usually time) together with the Markov property is a *Markov Chain*. The Markov property essentially says that the choice of states available at the next stage of the system is unaffected by past history. Only the current state affects the selection of the next state.) In purely theoretical terms it has been shown that the algorithm converges to a set of globally optimal solutions as time tends to infinity. However when implementing an annealing algorithm practical considerations must be given to the annealing schedule to ensure convergence to a 'good' quality solution occurs in finite time.

Implementing the Algorithm

To implement the algorithm for a particular problem it is necessary to first address two issues, *generic* and *problem specific* considerations. Problem specific considerations include the choice of a solution space, the form of the cost function, and an appropriate neighbourhood scheme. Generic considerations involve the choice of the initial temperature, the manner in which temperature is reduced, and the stopping condition. Both factors can greatly affect the speed of convergence.

- **Problem specific choices**: The solution space for any given problem must be clearly defined. In the case of the TSP the solution space is all solutions resulting from different orderings of a sequence of cities in a tour. Once the solution space is determined unambiguously, a neighbourhood structure and a method of evaluating solution quality must be imposed. The correct choice of neighbourhood structure is important if the annealing process is to work effectively. The neighbourhood should not be too large or too small. A neighbourhood that is too small will potentially give rise to slow or no convergence and a neighbourhood that is too large could lead to premature convergence. Ideally the neighbourhood should also be smooth in the sense that the quality of configurations in the neighbourhood of a solution should not be vastly different from either the quality of the solution itself or from the qualities of other solutions in the neighbourhood.

  The form of the evaluation function is quite clear for many problems, taking the value of the function that is being optimised. In the TSP case this is simply the sum of the intercity distances in a (cyclic) tour. With many combinatorial optimisation problems however, the evaluation function is not as straightforward, especially when searching for *any* feasible solution to some problem. In these cases, a penalty function is generally applied, which can distinguish between two infeasible solutions to determine which one is closer to feasibility. Thus the solution space for the annealing algorithm does not necessarily consist entirely of feasible solutions, although the algorithm will tend to discourage the selection of infeasible solutions as the temperature decreases.

  With a solution space , a neighbourhood scheme, and an evaluation function defined, the algorithm can now determine which other solutions can be reached in a single move, from a particular solution, and whether they are better or worse than the current position. Also, to begin the algorithm, a method of constructing an initial solution is required. This is usually achieved in either a randomised or a systematic manner.

- **Generic choices**: The set of implementation decisions that remain unchanged for all problem types is collectively known as the cooling or annealing schedule. The annealing schedule consists of:

- An initial temperature.
- The number of iterations spent at each temperature, L.
- A function to reduce the temperature.
- A decision rule to halt the search.

Finding an initial temperature is analogous to heating a solid in the true annealing process to a temperature when all particles are free to move about in the solid, meaning the system is equally likely to be in any state. This can be achieved in the algorithm by selecting some arbitrary initial temperature and performing a preliminary 'heatup' process. This typically involves repeatedly increasing the temperature until a certain high percentage of moves are being accepted.

Once the initial temperature has been determined and the algorithm proper has begun, the time spent at each value of the temperature needs to be decided upon. Returning to the true annealing process, notice that temperature reduction must be performed *carefully* in order to reestablish the thermal equilibrium of the system, categorised by the Boltzmann distribution. Every time the temperature is reduced in the algorithm, L should be the number of iterations needed before the distribution of states returns to equilibrium. This is quite a theoretical consideration and L is often determined arbitrarily, e.g. fixed at some constant value. Alternatively some degree of feedback can be used in the algorithm to determine when equilibrium has (approximately) been reached.

As noted above, temperature reduction needs to be performed carefully. If the temperature is reduced too quickly then the system can quickly become trapped in a local optimum of poor quality, analogous to quenching in the physical system. If the temperature is reduced too slowly then time until the algorithm converges to a local optimum can become prohibitively long. There is a trade off between the size of the incremental changes in the temperature and the number of iterations required to reestablish thermal equilibrium. The larger the change, the longer it will take to reestablish equilibrium at each temperature. If small changes are used then equilibrium is found relatively quickly, larger changes require more time to regain the Boltzmann distribution. Again the exact manner in which temperature is reduced can be determined arbitrarily, such as a geometric decrease $T=aT$ where $a<1$ (although usually quite close to 1), or through a feedback mechanism. In general it is advisable not to spend too long at high temperatures, since then the algorithm is effectively performing a random search. Too much time should also not be spent at low temperatures, since the algorithm is then simply performing a local search.

The algorithm should terminate when there is little chance of escaping from the current solution to a better quality solution, and if temperature reduction has been done carefully, the quality of this local optimum will be close to that of the global optimum. Many methods of determining when to stop can be used. For example, if the current solution has not changed for a certain number of temperature reductions then the algorithm terminates, an arbitrary method. Alternatively, information from the algorithm can be used to measure the rate of change in solution quality (as a function of temperature) at successive temperatures, and when this rate has fallen below a certain threshold then the algorithm terminates.

References

1. V. Cerny. *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm.* Journal of Optimization Theory and Applications, 45(1):41--51, 1985.
2. S. Kirkpatrick, Jr. C.D. Gelatt, and M.P. Vecchi. *Optimization by Simulated Annealing.* Science, 220(4598):671--679, 1983.
3. Nicholas Metropolis, Arianna W. RosenBluth, Marshall N. Rosenbluth, Augusta H.Teller, and Edward Teller. *Equation of State Calculations by Fast Computing Machines.* The Journal of Chemical Physics, 21(6):1087--1092, 1955.

This page is under development. *Last updated 17 January 1995*

Grant Telfar
grant@isor.vuw.ac.nz