COMP30030: Introduction to Artificial Intelligence

Neil Hurley

School of Computer Science
University College Dublin
neil.hurley@ucd.ie

October 16, 2018

Simulated Annealing

Analogy with a thermal process for obtaining low energy states of a solid in a heat bath. The process consists of the following two steps;

1. Increase the temperature of the heat bath to a maximum value at which the solid melts
2. Decrease **carefully** the temperature of the heat bath until the particles arrange themselves in the ground state of the solid.

Simulated Annealing I

- In the ground state, the molecules are arranged in a highly structured lattice and the energy of the system is minimal.
- The Metropolis algorithm simulates the evolution of a solid in a heat bath to thermal equilibrium. The algorithm generates a sequence of states of the solid in the following way:
    1. Given a current state i of the solid with energy E_i.
    2. Apply a perturbation which transforms the current state into a new state j by a small distortion.
    3. If the energy difference $E_j - E_i$ is ≤ 0 accept j as the current state. If the energy difference is > 0, the state j is accepted with a certain probability which is given by

       $$\exp\left(-\frac{E_j - E_i}{k_B T}\right)$$

       where T denotes the temperature and k_B is the Boltzmann constant.

Simulated Annealing II

- If the lowering of the temperature is done sufficiently slowly, the solid can reach thermal equilibrium at each temperature value.

Analogy with Optimisation Problems

<u>feasible solutions to opt problem</u> = <u>states of the physical system</u>
<u>cost function</u> = <u>energy of a state</u>

- Let (S, f) be a combinatorial optimisation problem and i, j be two solutions with cost $f(i)$ and $f(j)$, then the acceptance criterion determines whether j is accepted from i by applying the following <u>acceptance probability</u>

$$\begin{aligned}\text{Prob(accept } j) &= 1 \text{ when } f(j) \leq f(i) \\ &= \exp\left(-\frac{f(j) - f(i)}{T}\right) \\ &\quad\ \text{when } f(j) > f(i)\end{aligned}$$

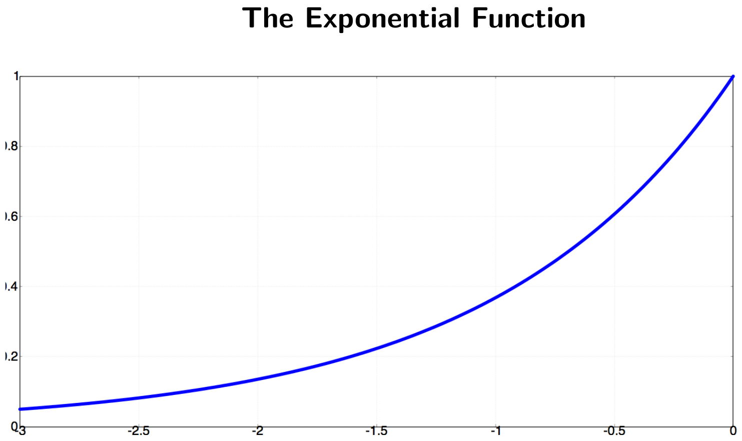
The Effect of T I

- Large values of T mean that almost all transitions will be accepted. As $T \to 0$ no deteriorations will be accepted at all.

The Effect of T II

$$\exp\left(-\frac{f(j)-f(i)}{T}\right)$$

The Effect of T III



The Effect of T IV

The exponential function

- Always returns a number between 0 (0% probability) and 1 (100% probability)
- If T is large, the argument of exp() is very small, so the probability is close to $\mathbf{1}$. (Since $\exp(0) = 1$)
- If T is close to zero, then the argument of exp() is close to $-\infty$, so the probability is close to $\mathbf{0}$ (Since $\exp(-\infty) = 0$)
- In general, T is contolling the chance that a bad neighbouring state will be accepted.
- T will start at a high value, meaning the chance of accepting bad states is high.
- T will be gradually reduced, so that gradually it becomes less and less likely that bad states are accepted.

Simulated Annealing Algorithm

```
Simulated_Annealing(){
  INITIALISE($i_{start}$, $T_0$, $L_0$);
  k=0;
  i = $i_{start}$;
  while (!stopCriterion()) {
      for len = 1 to $L_k$ {
        GENERATE(j from $S_i$);
        if(f(j) < f(i) )
          i = j;
        else
        if (exp$\left(-\frac{f(j)-f(i))}{T_k}\right)$ > random[0,1])
          i = j;
      } % end for
      k := k + 1;
      CALCULATE_LENGTH($L_k$);
      CALCULATE_TEMP($T_k$);
  } % end while
} % end simulated annealing
```

Simulated Annealing

- T_k is the temperature at the k-th iteration of the Metropolis algorithm. L_k is the number of transitions generated at the k-th iteration of the Metropolis algorithm.
- Note that for a particular value of T, the probability of a transition from a state i to a state j is determined completely by i. This sort of random transition is called a Markov chain
- Simulated Annealing is a simple and generally applicable algorithm that can escape local minima.
- Convergence is determined by the choice of L_k, T_k.

Cooling Schedule

- A <u>cooling schedule</u> specifies
    - a finite sequence of values of the control parameter, T i.e.
        - an <u>initial value</u> for T
        - a <u>decrement function</u> for decreasing the value of T
        - a <u>final value</u> of T specified by a <u>stop criterion</u>
    - a finite number of transitions at each value of T i.e.
        - a finite <u>length</u> of each Markov chain

Cooling Schedule

- For each value of the control parameter T_k, we would like to reach quasi equilibrium i.e. we would like to get sufficiently close to the stationary distribution at T_k after L_k trials
  - This means that after a while, the probability of any given state being visited settles down to a fixed value (that depends on T_k).
  - Unfortunately, it is difficult to know when the system has settled, as we cannot directly examine the probability of states being visited – generally it's a large state space, with too many states to examine.

Cooling Schedule

- In practise, to avoid an exponential running time for the SA algorithm, it is necessary to use some relaxed quantification of quasi equilibrium.
- By choosing T_0 sufficiently large i.e. accepting virtually <u>all</u> proposed transitions, quasi-equilibrium is immediately attained at T_0.

Cooling Schedule

- The length of the Markov chain and the decrement function must be chosen such that quasi equilibrium is restored at the end of each individual chain.
- Intuitively, large decrements of T_k will require longer chains in order to restore quasi equilibrium. Hence there is a trade-off between large decrements and small Markov chain lengths.

Cooling Schedule of Kirkpatrick et al I

1. **Initial Value of T** Require that the acceptance ratio is close to 1. In practice, this is achieved by starting with a small positive value of T_0 and multiplying it by a constant factor larger than 1, until the acceptance ratio, calculated from generated transitions is close to 1.
2. **Decrement of the Control Parameter**

   $$T_{k+1} = \alpha T_k$$

   for some α smaller than, but close to 1 e.g. $\alpha \in [0.8, 0.99]$.
3. **Final Value of T** Execution of the algorithm is terminated if the value of the cost function of the solution obtained in the last trial of a markov chain remains unchanged for a number of consecutive chains.

Cooling Schedule of Kirkpatrick et al II

4. **Length of the Markov Chain** The length of the Markov chain is based on the requirement that quasi equilibrium be restored. The intuitive argument is that it will be restored after acceptance of at least some fixed number of iterations. To avoid infinitely long chains, the length L_k is bounded by some constant.

Simulated Annealing is MCMC I

- Markov Chain Monte Carlo (MCMC) techniques provide a way of sampling from a distribution by using a "Markov chain" whose stationary distribution is the <u>target distribution</u> of interest.
- MCMC are nowadays used widely in <u>data analytics</u>.
- In general, the Metropolois algorithm provides a way to modify a Markov chain, so that its stationary distribution is a target distribution of interest and this has wide applications, wherever data modelling is required.
- The temperature parameter controls <u>exploitation vs exploration</u>. When the temperature is high, nearly all new states are explored, while when it is low, only better states are explored.

Graph Partitioning I

Definition

Let $G = (V, E)$ be a graph. A partition of the vertex set is a set of vertex sub-sets $\{P_1, \ldots, P_k\}$ such that $P_i \cap P_j = \emptyset$ and $P_1 \cup \cdots \cup P_k = V$. Given k, the balanced graph partitioning problem is to find a vertex partition such that $|P_1| = \cdots = |P_k|$ and the edge-cut is minimised. The edge cut is defined as

$$|\{(v, w) \in E | v \in P_i, w \in P_j \text{ s.t. } i \neq j\}|$$

- Lots of applications of this problem
  - Partitioning a computation across a parallel machine.
  - Module placement in VLSI design
  - FInding communities in social network
  - etc.

Representation of Graph Partitioning Problem I

- Consider the graph bi-partitioning problem (i.e. k=2). We can represent a solution to this problem (i.e. an example partition) by an indicator vector **x** of length $n =$ number of vertices, such that

$$\begin{aligned} x_i &= -1 \quad v_i \in P_0 \\ x_i &= 1 \quad v_i \in P_1 \end{aligned} \tag{1}$$

- The edge-cut objective can be expressed using the Laplacian matrix of the graph.
- The adjacency matrix A of a graph G is defined as

$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- Let D be a diagonal matrix, such that the value on the diagonal is the degree of vertex v_i.

Representation of Graph Partitioning Problem II

- The Laplacian matrix is defined as

$$\mathrm{L} = \mathrm{D} - \mathrm{A}$$

- Now consider the quadratic form

$$\begin{aligned}
\mathbf{x}^T \mathrm{L} \mathbf{x} &= \sum_{i=1}^{n} \sum_{j=1}^{n} l_{ij} x_i x_j \qquad (2) \\
&= \sum_i d_i x_i^2 - \sum_i \sum_j a_{ij} x_i x_j \\
&= \sum d_i - \sum_{x_i = x_j} a_{ij} + \sum_{x_i \neq x_j} a_{ij} \\
&= 2m - 2I + 2E \\
&\qquad (m = {}_{\text{no. edges}}\ I = {}_{\text{no. int edges}}\ E = {}_{\text{no. ext edges}}) \\
&= 2m - 2(m - E) + 2E \\
&= 4E
\end{aligned}$$

Representation of Graph Partitioning Problem III

- So we can write graph partitioning as the problem to find **x**

$$\mathbf{x} = \arg\min \mathbf{x}^T \mathrm{L}\mathbf{x} \qquad \text{s.t.} \sum_i x_i = 0 \quad x_i \in \{-1, 1\}$$

This is an integer (specifically, binary) **quadratic programming** problem with <u>linear</u> constraints.