Artificial Intelligence Methods (G52AIM)

Dr Rong Qu
rxq@cs.nott.ac.uk

Simulated Annealing
Hill Climbing – local optima
Hill Climbing – local optima

- Problem: Gets stuck at local minima!
  - Moves are always to better states

- Possible solutions
  - Try several runs, starting at different initial positions
  - Increase the size of the neighborhood (e.g. in TSP try 3-opt rather than 2-opt)
Simulated Annealing vs. HC

- Hill-climbing
  - moves are always to better states

- Simulated annealing
  - To escape a local optimum we must allow worsening moves
  - In a controlled way, allow downwards ("wrong-way", worsening) steps
Simulated Annealing vs. HC

- **Hill-climbing**
  - might consider many possible moves
  - evaluate many solutions, can be too expensive

- **Simulated annealing**
  - randomly select one state in the neighbourhood
  - decide whether to accept it or not
  - **better moves are always accepted**
  - **worsening moves are sometimes selected**
Simulated Annealing

- Motivated by the physical annealing process
- Material is heated and slowly cooled into a uniform structure
- Simulated annealing mimics this process
- The first SA algorithm was developed in 1953 (Metropolis)
Simulated Annealing

- Kirkpatrick (1982) applied SA to optimisation problems
Simulated Annealing - acceptance

The law of thermodynamics states that at temperature, $t$, the probability of an increase in energy of magnitude, $\delta E$, is given by

$$P(\delta E) = \exp(-\delta E / kt)$$

Where $k$ is a constant known as Boltzmann’s constant.
Simulated Annealing - acceptance

\[ P = \exp(-c/t) > r \]

- where
  - \( c \) is change in the evaluation function
  - \( t \) is the current temperature
  - \( r \) is a random number between 0 and 1

- Example
To accept or not to accept - SA?

<table>
<thead>
<tr>
<th>Change</th>
<th>Temp</th>
<th>exp(-C/T)</th>
<th>Change</th>
<th>Temp</th>
<th>exp(-C/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.95</td>
<td></td>
<td>0.2</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.95</td>
<td></td>
<td>0.4</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.95</td>
<td></td>
<td>0.6</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.95</td>
<td></td>
<td>0.8</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>
To accept or not to accept - SA?

<table>
<thead>
<tr>
<th>Change</th>
<th>Temp</th>
<th>exp(-C/T)</th>
<th>Change</th>
<th>Temp</th>
<th>exp(-C/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.95</td>
<td>0.810</td>
<td>0.2</td>
<td>0.1</td>
<td>0.13583</td>
</tr>
<tr>
<td>0.4</td>
<td>0.95</td>
<td>0.656</td>
<td>0.4</td>
<td>0.1</td>
<td>0.018339</td>
</tr>
<tr>
<td>0.6</td>
<td>0.95</td>
<td>0.532</td>
<td>0.6</td>
<td>0.1</td>
<td>0.0024852</td>
</tr>
<tr>
<td>0.8</td>
<td>0.95</td>
<td>0.431</td>
<td>0.8</td>
<td>0.1</td>
<td>0.000335</td>
</tr>
</tbody>
</table>

- Need to use a scientific calculator to calculate exp()
- Note that the temperatures here are just examples and not recommendations for values to use in real implementations.
Simulated Annealing - acceptance

- The probability of accepting a worse state is a function of both the temperature of the system and the change in the cost function.

- As the temperature decreases, the probability of accepting worse moves decreases.

- If $t = 0$, no worse moves are accepted (i.e. hill climbing).
SA – implementation

- The most common way of implementing an SA algorithm is to implement hill climbing with an accept function and modify it for SA.

- The example shown here is taken from Russell/Norvig (Artificial Intelligence: A Modern Approach).


SA – algorithm

**Function** `SIMULATED-ANNEALING(Problem, Schedule)`
returns a solution state

**Inputs**
- *Problem*: a problem
- *Schedule*: a mapping from time to temperature

**Local Variables**
- *Current*: a node
- *Next*: a node
- *T*: a “temperature” controlling the probability of downward steps
SA – algorithm

\[ Current = \text{MAKE-NODE}(\text{INITIAL-STATE}[\text{Problem}]) \]

\textbf{For } t = 1 \textbf{ to } \infty \textbf{ do}

\[ T = \text{Schedule}[t] \]

\textbf{If } T = 0 \textbf{ then return } Current

\[ Next = \text{a randomly selected successor of } Current \]

\[ \Lambda E = \text{VALUE}[Next] - \text{VALUE}[Current] \]

\textbf{if } \Lambda E > 0 \textbf{ then } Current = Next

\textbf{else } Current = Next \text{ only with probability } \exp(-\Lambda E/T)
SA – algorithm

- The cooling schedule is *hidden* in this algorithm - but it is important (more later)

- The algorithm assumes that annealing will continue until temperature is zero - this is not necessarily the case
SA – cooling schedule

- Starting Temperature
- Final Temperature
- Temperature Decrement
- Iterations at each temperature
**SA – cooling schedule**

- **Starting Temperature**
  - Must be *hot* enough to allow moves to *almost* neighbourhood state (else we are in danger of implementing hill climbing)
  - Must *not* be so hot that we conduct a random search for a period of time
  - Problem is finding a suitable starting temperature
SA – cooling schedule

- **Starting Temperature**
  - If we know the maximum change in the cost function we can use this to estimate
  - Start high, reduce quickly until about 60% of worse moves are accepted. Use this as the starting temperature
  - Heat rapidly until a certain percentage are accepted the start cooling
SA – cooling schedule

- **Final Temperature**
  - It is usual to let the temperature decrease until it reaches zero. However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used.
  - In practise, it is not necessary to let the temperature reach zero because the chances of accepting a worse move are almost the same as the temperature being equal to zero.
SA – cooling schedule

- **Final Temperature**
  - Therefore, the stopping criteria can either be a suitably low temperature or when the system is "frozen" at the current temperature (i.e. no better or worse moves are being accepted)

- Example: online demo
SA – cooling schedule

- **Temperature Decrement**
  - Theory states that we should allow enough iterations at each temperature so that the system stabilises at that temperature.
  - Unfortunately, theory also states that the number of iterations at each temperature to achieve this might be exponential to the problem size.
SA – cooling schedule

- **Temperature Decrement**
  - We need to compromise
  - We can either do this by doing a large number of iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two
SA – cooling schedule

- Temperature Decrement
  - Linear: $temp = temp - x$
  - Geometric: $temp = temp \times a$
    - Experience has shown that $a$ should be between 0.8 and 0.99
    - Of course, the higher the value of $a$, the longer it will take to decrement the temperature to the stopping criterion
SA – cooling schedule

- **Iterations at each temperature**
  - A constant number of iterations at each temperature
  - Another method, first suggested by (Lundy, 1986) is to only do one iteration at each temperature, but to decrease the temperature very slowly. The formula used is

\[ t = \frac{t}{1 + \beta t} \]

where \( \beta \) is a suitably small value
Learning Objectives

- SA basics
  - Cooling schedule (4 elements)
  - Acceptance criteria

- HC basics
  - Problem of local optima

- Be able to implement HC and SA in your coursework