

*Introduction to  
Simulated Annealing*  
22c:145

---

---

---

---

---

---

---

---

**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)

---

---

---

---

---

---

---

---

**Differences to Hill Climbing**

- The main difference is that SA allows downwards steps
- In SA, a move is selected at random and then decides whether to accept it
- In SA better moves are always accepted. Worse moves are not always accepted.

---

---

---

---

---

---

---

---

### The Problem with Hill Climbing

- Gets stuck at local minima
- Possible solutions
  - Try several runs, starting at different positions
  - Increase the size of the neighbourhood (e.g. in TSP try 3-opt rather than 2-opt)

---

---

---

---

---

---

---

---

### To accept or not to accept - SA?

$$P = e^{-c/t} > r$$

- Where
  - $c$  is change in the evaluation function
  - $t$  is the current temperature
  - $r$  is a random number between 0 and 1

---

---

---

---

---

---

---

---

### To accept or not to accept - SA?

Change	Temp	exp(-C/T)	Change	Temp	exp(-C/T)
0.2	0.95	0.810157735	0.2	0.1	0.135335283
0.4	0.95	0.656355555	0.4	0.1	0.018315639
0.6	0.95	0.53175153	0.6	0.1	0.002478752
0.8	0.95	0.430802615	0.8	0.1	0.000335463

---

---

---

---

---

---

---

---

### To accept or not to accept - SA?

- 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 Algorithm

- The most common way of implementing an SA algorithm is to implement hill climbing with an accept function and modify it for SA by introducing a *cooling schedule*.

---

---

---

---

---

---

---

---

### 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
- *Current* = MAKE-NODE(INITIAL-STATE[*Problem*])

---

---

---

---

---

---

---

---

### SA Algorithm (for maximum-value solutions)

**For**  $t = 1$  **to**  $\infty$  **do**

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

**if**  $T = 0$  **then return**  $\text{Current}$

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

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

**if**  $\Delta E > 0$  **then**  $\text{Current} = \text{Next}$

**else**  $\text{Current} = \text{Next}$  only with probability  $\exp(-\Delta E/T)$

---

---

---

---

---

---

---

---

### SA Algorithm - Observations

- 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**

---

---

---

---

---

---

---

---

### 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
- Trial-and-error to find a suitable starting temperature

---

---

---

---

---

---

---

---

### 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

---

---

---

---

---

---

---

---

### Temperature Decrement

- Linear
  - $temp = temp - x$
- Geometric
  - $temp = temp * x$
  - Experience has shown that  $x$  should be between 0.8 and 0.99, with better results being found in the higher end of the range. Of course, the higher the value of  $x$ , the longer it will take to decrement the temperature to the stopping criterion

---

---

---

---

---

---

---

---

### Iterations at each temperature

- A constant number of iterations at each temperature
- to only do one iteration at each temperature, but to decrease the temperature *very* slowly.
- E.g.,
  - $t = t/(1 + \beta t)$where  $\beta$  is a suitably small value.

---

---

---

---

---

---

---

---

### Iterations at each temperature

- An alternative is to dynamically change the number of iterations as the algorithm progresses

At lower temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored

At higher temperatures, the number of iterations can be less

---

---

---

---

---

---

---

---

### Performance Issues

- Quality of the solution returned
- Time taken by the algorithm
  
- We already have the problem of finding suitable SA parameters (cooling schedule).

---

---

---

---

---

---

---

---

## Acceptance Probability

- Why cannot we use a different acceptance criteria?
  - The exponential calculation is computationally expensive.
  - (Johnson, 1991) found that the acceptance calculation took about one third of the computation time.
  - Johnson experimented with
$$P(\delta) = 1 - \delta/t$$
  - This approximates the exponential

---

---

---

---

---

---

---

---