Lecture 12 (Local Search)

1 Exploitiong Problem Structure

1. Problem specific optimisations to speed up CSP solving
2. Disconnected components can be independently solved
3. Tree-structured CSPs can be easily solved in $O(nd^2)$ time
4. Instantiate some variables and solve on the pruned graph
   - Find a subset of variables S, such that the remaining constraint graph becomes a tree after the removal of S (S is a cycle cut set)

2 Iterative Approaches to Solving CSPs - Local Search

1. Take an assignment with unsatisfied constraints
2. Reassign variable values
3. Repeat: till CSP does not have a solution
   i. Variable selection - randomly select any conflicted variable
   ii. Choose a new value which has the least number of conflicts - heuristic functions

Above idea is called "hill climbing" with $h(x)$. Generic idea is:

1. Start at a state
2. Repeat: move to best neighbouring neighbour
3. If no neighbour better than current, return

3 Optimisation Problems - Generic Setup

- Local search is an example of optimisation problem
- We attempt to minimise the cost function
- Compared to search algorithms, notion of "path" from initial state to goal state isn't important here

4 Preventing Stagnating at Local Maxima - Simulated Annealing

1. Allow some bad moves to escape local maxima
2. Repeat:
   i. Let X_i be a random neighbour of X
   ii. If $E_i > E$, $X \leftarrow X_i$ and $E \leftarrow E_i$
   iii. Else, with some probability p, $X \leftarrow X_i$ and $E \leftarrow E_i$
3. This algorithm is a form of Monte-Carlo Search
4. p is higher when $|E_i - E|$ is low and vice-versa
5. Exact formulation of $p = \exp -\frac{E - E_i}{T}$, T is the dynamic variable that slowly reduces to 0 over time

5 Local Beam Search

1. Track k states
2. Begin with k randomly sampled states
3. Loop:
   i. Generate successors of each of the k states
   ii. If any of them has the goal, algorithm halts
   iii. Select only the k best successors from the list and repeat
4. States become concentrated in a small region of space