CS4725/CS6705

Chapter 4: Beyond Classical Search

Local search

- Unfortunately, we will not spend very much time on this topic, but we will introduce it briefly.
- In many problems, the **path** to the solution is irrelevant. We only care about what the solution is. [Example: n queens]
- Local search: from the current state, move to a neighbour, searching for a goal

Hill-climbing search

- Moves to the neighbour with the best score (according to an evaluation function)
- Does not keep track of previously searched nodes, does not look ahead beyond immediate neighbours
- "...trying to find the top of Mount Everest in a thick fog while suffering from amnesia" (Russell & Norvig)
- "greedy local search"
- 8-queens example

Hill-climbing (cont'd)

- Often makes very quick progress toward a solution, but can get stuck (86% of the time with random 8-queens problems):
  - Local maxima/minima
  - Ridges
  - Plateaux

Variations on hill-climbing

- Stochastic hill climbing:
  - Choose at random from *all* neighbours that are better than the current state; the probability of choosing a neighbour depends on *how much* better it is.
- First-choice hill climbing:
  - Generate successors randomly. Choose the first one that is better than the current state.
- Random-restart hill climbing:
  - Perform hill climbing, starting with a random initial state. If you get stuck, then restart with a new random initial state.

Simulated annealing

- Picks a random move
  - If it is an improvement, accepts it
  - If it is not an improvement, accepts it with some probability, dependent on how bad the move is and how high the "temperature" is (starts high, decreases over time)
- If temperature is lowered slowly enough, probability of finding an optimal solution approaches 1

Local beam search

- Starts with k random states
- Generates successors for all of them
- If no goal is found, chooses the best k successors from the entire list and repeats
- Risk of all states being concentrated in one region of the state space
  - Stochastic beam search: chooses k successors at random, with probability dependent on a node's value

Genetic algorithms

- Similar to stochastic beam search
- Start with k randomly generated states, represented as strings
- States are rated by a **fitness function** (evaluation function)
- Random pairs are selected, with probability dependent on fitness, and then **mated**
- Offspring are generated by crossing over the parent strings
- Also: random **mutation**