Recap

- Many problems can be viewed as a search problem
- Needs means to explore the search space.
- Uninformed search approaches: dfs, bfs, iterative deepening
- Informed - use heuristics; A* algorithm
- Adversarial search: minimax, alphabeta pruning

Monte Carlo Tree Search

Monte Carlo Tree Search

- Minimax and alpha beta pruning are useful approaches
- Alpha beta pruning effectively results in computing the square root of the branching factor
- In many cases, game trees have too a high a blocking factor and evaluating the leaf nodes is not possible
- Instead, use a estimation function instead.

Monte Carlo Tree Search

- Continually estimate values of tree nodes
- Combines ideas from classical tree search and machine learning
- Balances exploration of unseen space with exploitation of best known move/action

Monte Carlo Tree Search

- Explores all potential options at that time
- Identify the best move and search for other options while validating how good the current action is
- Continue searching for other options at that time.
- Search for other options while validating how good the current options is
- Use *play out* - simulation considering random actions

Monte Carlo Tree Search

- Selection - pick suitable path
- Expansion - expand from that path
- Simulation - simulation (play outs)
- Back propagation - update states

Monte Carlo Tree Search

- Tradeoff between exploration and exploitation
- The balance is usually controlled by a number of parameters/factors
- Include: Number of wins, simulations, exploration cost
- Used in a large number of domains - AlphaGo - DeepMind, many video games

Local Search Algorithms

- In many domains, we are only interested in the goal state
- The path to finding the goal is irrelevant

- Main idea is to maintain the current state and to repeatedly try an improve upon the current state
- Little memory overhead
- Can find reasonable solutions even in large or infinite spaces

- Suitable for certain classes of problems
- Maximising (or minimising) certain criteria among a number of candidate solutions
- Local search algorithms are incomplete algorithms - they may not find the optimal solution.

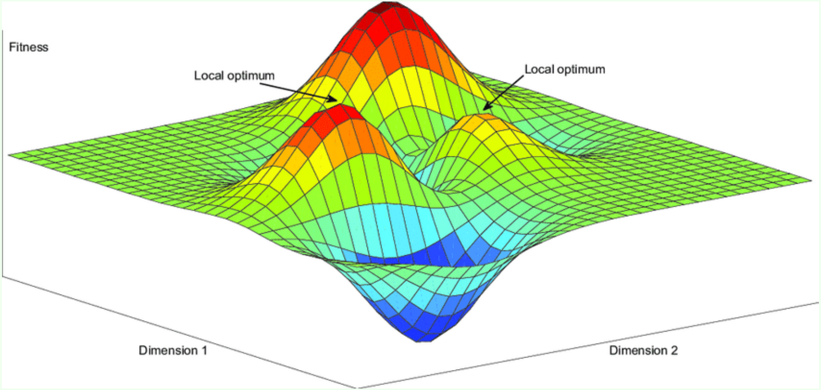

Figure: Fitness Landscape - T. Nygaard (c)

Hill Climbing

- Move in the direction of increasing value
- Stop when you reach a peak
- No look ahead involved
- Maintain only the current state and the objective function

- Problem: local maxima (or minima)
- The success of the algorithm depends on shape of the search space
- Approaches: random re-start

Hill climbing

```
currentnode = initialstate
do while ( stopping criteria )
    neighbour = fittest neighbour of current
    if value(neighbour) > value (current)
        current = neighbour
```

Sample problem

- 8 queens problems
- Goal: to place 8 queens on a chess board so that no two queens are attacking each other
- Formally, given an 8*8 grid, identify 8 squares so that no two squares are on the same row, column or diagonal.
- Note: can be generalised to be an N*N board
- Problem: generate an algorithm to find a solution
- Use hill climbing; note that algorithm may get caught in a local maximum

Simulated Annealing

- Tries to avoid being stuck in local maxima
- Randomly moves to another state
- Move is based on fitness of current state, new state and the 'temperature'; a parameter which decreases over time and reduces the probability of changing states

Well known optimisation problems

Knapsack Problem

- There are a set of items - each with a value and a weight
- Goal: select a subset of items such that the weight of these items is below a threshold and the sum of the values if optimised/maximised.
- Problem: how to select those items?

Knapsack Problem

- More formally, given two n-tuples of values $< v_0, v_1, \ldots v_n >$ and $< w_0, w_1, \ldots w_n >$, chose a set of items i from the n items such that $\sum_i v(i)$ is maximised and $\sum_i w(i) \leq T$
- Brute force approach: enumerate all subsets and keep the subset that gives greatest payoff. $O(2^n)$
- Greedy approach: more efficient but won't give best solution
- Other approaches?

Knapsack Problem

- Many variations. The version described above it the 0/1 Knapsack problem. There are scenarios where part of an item may be taken.
- Constraints over the items where the value of an item is dependent on another item being chosen or not chosen.
- Multiple knapsacks.

Travelling Salesperson Problem

- Given N cities, devise a tour that involves visiting every city once.
- The distance or cost between pairs of cities is known.
- Goal: minimise the cost of the tour
- Many applications in scheduling and optimisation.

Travelling Salesperson Problem

- Brute force - enumerate all tours and chose cheapest. Computationally intractable.
- Greedy approach?
- Other?

Summary/Exercise

- Consider the TSP and the Knapsack problem, can you identify a local search approach? What would consider a neighbouring state?