Alternative Search Methods

CSCU9A3

Re-cap

- Problem solving can be viewed as trying lots of different combinations of actions until one that solves the problem is found
  - This is done by searching
  - Tree searches are structured ways of covering an entire search space, but are uninformed

Informed Search

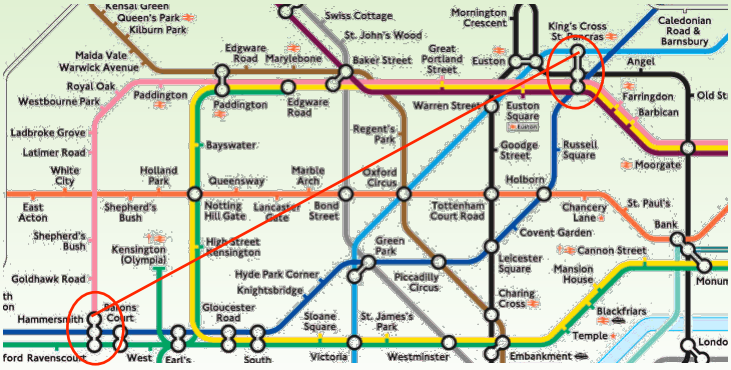
- Often, when searching, you get feedback from the environment on how well you are doing
- Often you know what the goal looks like, but not where it is, so you can compare your current state with the goal state to see how far away you are, and move accordingly
- This comparison is often called a distance metric, or fitness score, and the function that measures it is the evaluation function

Back to the Underground

- In the last lecture, we tried to get from Hammersmith to King's Cross on the London Underground with an uninformed tree search
- It was uninformed because we didn't know where King's Cross was
- Now we try again, with the help of a GPS that tells us the straight line distance from us to King's Cross
- Our model knows the coordinates of each station, so it can simulate the GPS

Hammersmith to King's Cross

- When we start, the distance is (say) 10 miles

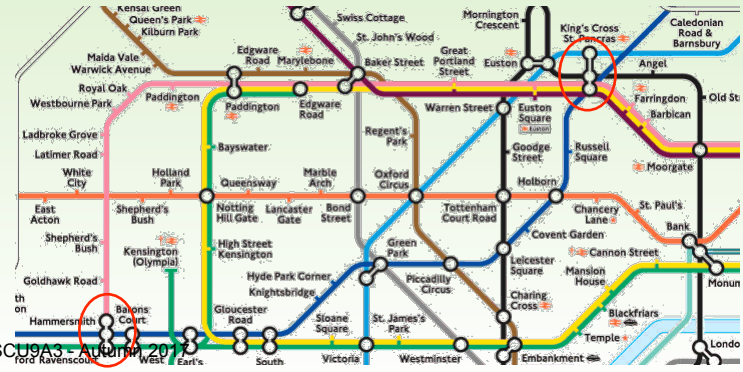


Greedy Search

- Greedy search chooses the step that takes us closest to the goal at each branch of the decision tree
- It then repeats the process until it finds either
  - The goal
  - The end of a path in the tree
  - A pre-defined number of steps

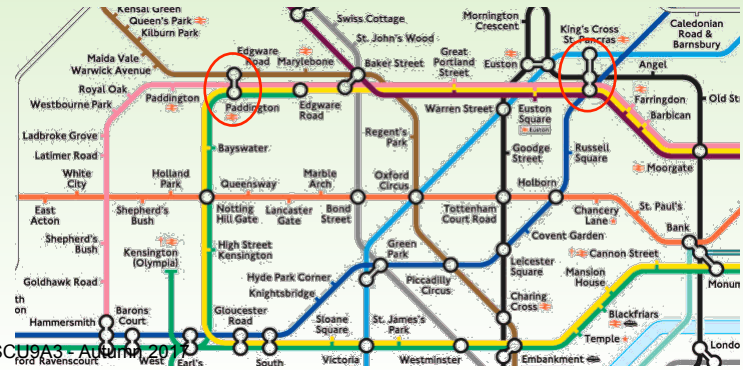
Greedy Search

- From Hammersmith, calculate the distance for Paddington, Barons Court, and stations west. Choose Paddington



Greedy Search

- From Paddington calculate the distances from Edgeware road and Baker Street. Choose Baker Street



And So On

- Usually speeds up a search considerably by ignoring many options at each step
- Sometimes you have to go backwards to go forwards, which it is not able to cope with
- Search is still systematic, with previous states being stored for recursive search as dead ends are found

Local Search

- Imagine you have climbed a hill, but it has got dark before you are back down. Your goal is to get to the bottom of the hill. Your problem is finding it. You have no GPS, so greedy search is not an option.
- Oh, and you fell and hit your head, so you have amnesia – you can't remember where you have just been
- Your options are...

Lost on a Hill

- Options
  - Circumnavigate the hill at the current altitude to see if the goal is at your current height, then take one step down and repeat until you are at the bottom
    - This exhaustive breadth first search is guaranteed to find the bottom, if you don't die first.
  - You could walk a straight line in any direction until you hit either the bottom or the top. This depth first search could have you going up and down for ages

Hill Descending

- Generally, you would make sure that every step you made took you down hill and, ignoring cliffs, as steeply as possible
- That is the fastest way to the bottom, unless:
  - You are at the bottom of a dip, and every step you might make takes you up hill
  - You are on a plateau and every step makes no difference

Back to Local Search

- Local search is for when you are lost on a hill in the dark with amnesia
- The hill is the evaluation function – the height tells you the current distance from your goal
- There is no memory, so no tree traversal, as you forget instantly which nodes have been traversed
- Actually, local search is highly effective if done well

Steepest Descent Search

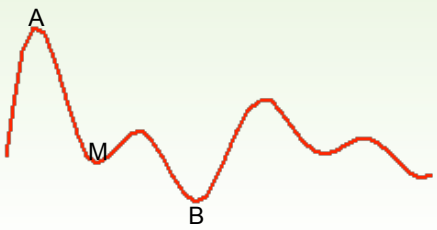
- Steepest descent searches are designed to take (you guessed it) the steepest path down the evaluation function
- The evaluation function equals zero at the goal and is positive elsewhere
- You actually don't need a goal, you could just be trying to minimise the cost of something
- Or reverse it all and try to maximise something

Real Examples

- Maximising the profit on an investment portfolio
- Cutting irregular shapes from square cloth with the least wastage
- Time tabling a whole university
- Producing e-fits of criminals
- Solving logical inference problems

Toy Example

- Here is a simple evaluation function
- Imagine we are at point A, and would like to be at point B, the goal
- Everything goes fine until we get to m, a local minimum. Then we are stuck.



Local Minima

- Local minima are the big problem with local search, especially as the agent has no memory
- A simple rule of: 'If down is not possible, pick a random direction and take a step up' will have you stepping up and down forever
- How do you get out?

Momentum

- As you head downhill, you pick up speed. The steeper and longer the hill, the more speed
- This gives you momentum (a bit of short term memory)
- You were probably going the right way before, so keep going that way even though it means going up hill for a while
- Momentum must run out, or you could go back up to the top of the hill by mistake

Simulated Annealing

- Or, shaking
  - Simulated annealing is like running around at random while you are near the top of the hill, but heading downhill with more purpose near the bottom
  - You are jumped out of local minima, but still tend to follow a downhill path
  - Actually based on picking at random and accepting up hill steps with a decreasing probability over time

Genetic Algorithms

- Many parallel searches are run (a population) of searches
- The evaluation function is known as the fitness function, but measures the same thing
- Searches with a good fitness score are bred, their offspring mutated, and a new generation is born!

Breeding and Mutation

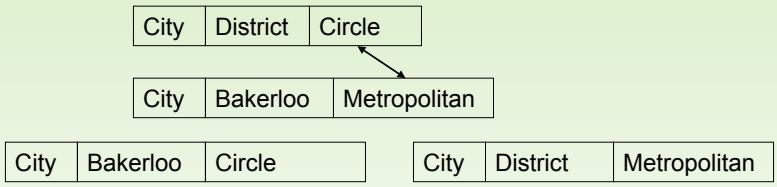
- Breeding is called crossover in the GA world
- The challenge in any specific problem is to find a representation where the result of mixing two quite good solutions is another quite good solution
- Also, need to consider whether sequences of actions can be joined

Tube Trains Again

- How do we represent a local search solution in our London underground problem?
- Current station is not a solution, entire route is, so a solution is a variable length list of lines taken:
  - City, Circle, Circle, Circle, for example would get you there

Tube Trains Again

- Can we breed routes?



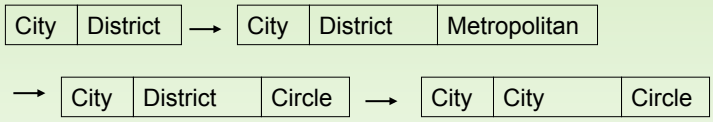
- Many will not be valid, but some will
- Invalid ones will have a very low fitness function and will not breed

Mutation

- New offspring solutions are changed slightly to increase the range of the search
- Only a small part of a solution's full description is changed at any one time
- Mutation is usually random, as trying every possible mutation to find a good one starts to look like an exhaustive search
- Sometimes, you know something about what is valid as a search pattern, and can avoid generating horrible mutants

Mutants on the Tube

- Mutating lists of tube stations might go:



- When we mutate, we can pick any random line, or look up which lines are valid from each station to cut down on wasted mutants.
- Mutations would need to grow and shrink the list as well as changing stations in it

Conclusion

- Walking downhill is not as easy as you would think
- Informed search algorithms try to move quickly towards a goal based on the distance metric from their current point
- Greedy search algorithms only follow paths in search space that bring them closest to the goal
- Local search algorithms have no memory to store tree structures, but work by intelligently covering selected parts of search space
- Population based search algorithms perform parallel searches and share information to produce new solutions