COMP3702 - Assignment 1

Roy Portas - 43560846

August 26, 2016

1 Formal Definition of a Navigation Agent

The navigation agent in this assignment is designed to output the shorted route between two points.
The navigation agent achieves this by gathering information about the environment, which in this case is the adjacency matrix))
The navigation agents uses this data of the environment to find the shortest route

The navigation agent can be divided into three sections:

1. Action space
   The set of all actions the agent can do, which in this cause is all possible paths between every point
2. Percept space
   The set of all things the agent can perceive in the world, which in this case is all the points and the costs between every point
3. State space
   The internal state of the world and the agent

The navigation agent will use the above the select an action it believes will minimize the cost between two points.

2 Characteristics of Navigation Agent

The navigation agent will be discrete, because the points and paths between them are finite.

The agent is deterministic, because the agent knows exactly what state it will be in after moving along a path

The agent is fully observable, because the agent knows the entire state of the world at any time. Additionally there are no external factors that could change the state

The agent is static, because the world does not change while the agent is determining the shortest path.

3 A* Search Heuristic

Choose the heuristic of the node to be the minimum cost of the distances to it's children.

If this heuristic is applied to every node, it should allow the algorithm to choose the next node in such a way that it's children are easy to reach. Since we want the heuristic to be admissible, it must never overestimate. If we choose the minimum cost of the children, the heuristic will always underestimate the cost of the children nodes.

With the heuristic in place, the algorithm then can choose the next node based on the cost to the node and the minimum cost of the children of the child nodes. By choosing the minimum distance of the children nodes, it allows the algorithm to reduce the cost in the long run. Since when it moves to the next node, it will have the shortest distance to it's children compared to the other nodes.

4 Comparision of Uniform Cost and A*

Uniform cost search involves a few steps:

- Dequeue the maximum priority element from the queue
- If the path is ending in the goal state, write the path to the file and exit
- Expand the nodes and insert the children into the priority queue, each child's cost is the cost up to the child plus the child cost itself

The uniform cost search does not use any heuristics. Additionally, UC is complete as long as the branching factor is finite and all edges have a valid cost. Furthermore, it will generate an optimal solution if all edges have a positive cost. However since UC is a blind search algorithm, it does not estimate the cost from the current node to the goal.

The Uniform Cost search algorithm has a space and time complexity of

$$O(branch\ factor^{1+floor(\frac{cost\ of\ optimal\ solution}{min\ step\ cost}})$$

However the A* search, with heuristic, will be much slower. This is because the heuristic visits every node, and all children to each node. The worse case scenario would be when all nodes are connected to each other, resulting in a heuristic time complexity of $O(n^2)$. However it is worth noting that for a given environment, the heuristic only needs to be calculated one. Meaning after the first distance calculation the algorithm will have a space and time complexity equal to the Uniform Cost search.

5 Will a larger graph cause the A* algorithm to take longer?

True, however this depends on how sparse the graph is. It is possible to have a graph with more vertices yet is sparse, meaning it would not take longer to do.

Assuming that two graphs are approximately the same sparseness, the larger graph will take longer to run. This is because at A* is fundermentally a smarter BFS search. Thus the more vertices the graph has the longer it will take to compute.