Coursework Part 2 Report

Samuel Cattanach
40276600@live.napier.ac.uk
Edinburgh Napier University - Artificial Intelligence (SET09122)

1 Introduction

2 Dijkstra's Pathfinding Algorithm

First published in 1959 by Edsger Dijkstra, Dijkstra's shortest path first algorithm is the basis of many pathfinding AI's and has been expanded upon by many algrithms such as A*. One of the main applications of Dijkstra's algorithm is finding the quickest routes between locations using paths such as roads. The algorithm is implemented to be able to find the optimal route, if a route exists, in any scenario. the information required to find a path is a specific starting node, an end node, and either the coordinates of all nodes or the distances between each node. Starting at the specified node, Dijkstra's algorithm will find all traverseable nodes from that point, calculating the distance of each, and selecting the one which has the shortest distance regardless of whether it is closer to the specified end node. Two lists are maintained throughout the calulation of the shortest path. The first list is a priority queue consisting of nodes which have not been fully investigated yet which is sorted by the shortest distance first. The second list contains all nodes wich have been fully investigated and will not be further evaluated. Firstly, the travereable neighbouring nodes from the start point will be added to the priority queue and will have their travelled distances and parent nodes recorded. The starting node, once all travereable neighbouring nodes are added to the priority queue, will be added to the second list of investigated nodes. The algorithm then continues through the nodes, calculating the shortest routes first, until the end node is reached. Once the goal has been reached, the final path traversed is established by following the parent of each node. Through proper implementation of Dijkstra's algorithm the optimal path from start to finish will always be found.

In regards to the cave problem described previously, Dijkstra's algorithm can be implemented in order to find the optimal route of any series of caves, or identify if no route is available. As the euclidean distance can be calculated using each points coordinates and a connectivity matrix is supplied, Dijkstra's algorithm will be able to start at cave one and work through every travereable neighbouring node until the last cave is reached.

Dijkstra's algorithm was first implemented on the ARMAC computer, which had only 15 kilobytes of memory, because of this small amount of memory, there were limitations in how advanced Dijkstra's shortest path first algorithm could be. The most major limitation in comparison to some of its successors was that it did not take into account any heuristic value for the nodes. Since only the distance travelled so far is taken into consideration and not the distance from the end node, the algorithm may first calculate nodes leading away from the end node before going back towards it.

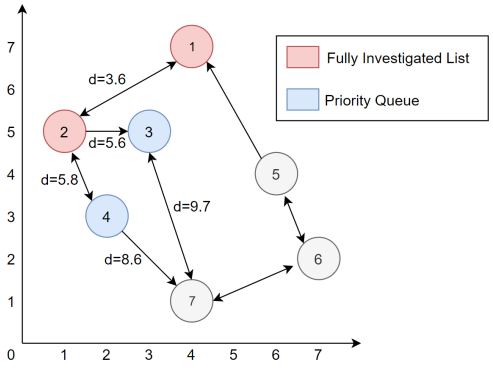

Figure 1: Dijkstra's algorithm

```
f(n) = g(n) + h(n)
```

Figure 2: A* shortest-first algorithm

3 The A Star Pathfinding Algorithm

Given the limitations of Dijkstra's algorithm, one of its successors, the A* search algorithm allows for an optimal route to be discovered much faster. A* would be a more suitable choice of algorithm for the given problem of navigating a system of connected caves because it takes into consideration not only the distance travelled but a heuristic value as well. This heuristic value may be the euclidean distance or the Manhattan distance for example. By calculating the euclidean distance of each node (h(n)) and the distance travelled so far (g(n)), using the below formula (Fig 2) to determine which node to explore first provides an optimal solution at a much faster rate.

In a situation where instead of using the euclidean distance as the distance travlled, each tunnel in a particular direction had a specified cost, the A* algorithm would not be the most suitable choice in finding an optimal path. In this scenario, the only given information to use when deciding which node to explore first is g(n), no heuristic is available and therefor the formula shown in figure 1 is

not usable. Instead of using A* in this situation, Dijkstra's algorithm may be the more sensible implementation as it only requires a g(n) value.

4 Conclusion

Appendices

Appendix A