CZ3005 Artificial Intelligence Assignment 1

Lab Group: TS4

Team Members and Respective Contributions:

Zhu Weiji (U1922876G) – All tasks

Muhammad Shafiq (U1922134K) – All tasks

1.1 Introduction

Graph traversal and pathfinding are well studied problems with a large body of research. Standard algorithms such as BFS, UCS and A* Search have good performance and are complete.

Given that the entire network has been parsed and will not change during our analysis, we can define the environment of this problem as having the following properties: *accessible* (fully observable), *deterministic*, *episodic* (non-sequential), *static* and *discrete*. Edges have two properties – distance and energy cost. The objective is to pathfind for the shortest overall distance, whilst checking to ensure that the energy cost constraint is not violated. This task is described in literature as a *resource constrained shortest path problem.*

1.2 Results

Task 1

```
1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->
1259->1249->1246->963->964->962->1002->952->1000->998->994->995->996->987->988->979->980->969->977->
989->990->991->2369->2366->2340->2338->2339->2333->2334->2329->2029->2027->2019->2022->2000->1996->
1997->1993->1992->1989->1984->2001->1900->1875->1874->1965->1963->1964->1923->1944->1945->1938->19
37->1939->1935->1931->1934->1673->1675->1674->1837->1671->1828->1825->1817->1815->1634->1814->1813-
>1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->54
18->5431->5425->5424->5422->5413->5412->5411->66->5392->5391->5388->5291->5278->5289->5290->5283->5
284->5280->50
```

Distance: 148648.63, Iterations: 32514, Time taken: 0.3364 seconds

Task 2

```
1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->
1259->1249->1246->963->964->962->1002->952->1000->998->994->995->996->987->988->979->980->969->977->
989->990->991->2465->2466->2384->2382->2385->2379->2380->2445->2444->2405->2406->2398->2395->2397->
2142->2141->2125->2126->2082->2080->2071->1979->1975->1967->1966->1974->1973->1971->1970->1948->19
37->1939->1935->1931->1934->1673->1675->1674->1837->1671->1828->1825->1817->1815->1634->1814->1813-
>1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->54
18->5431->5425->5424->5422->5413->5412->5411->66->5392->5391->5388->5291->5278->5289->5290->5283->5
284->5280->50
```

Distance: 150335.55, Energy Cost: 259087, Iterations: 16600000, Time taken: 119.09 seconds

Task 3

```
1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->
1259->1249->1246->963->964->962->1002->952->1000->998->994->995->996->987->986->979->980->969->977->
989->990->991->2369->2366->2340->2338->2339->2333->2334->2329->2029->2027->2019->2022->2000->1996->
1997->1993->1992->1989->1984->2001->1900->1875->1874->1965->1963->1964->1923->1944->1945->1938->19
37->1939->1935->1931->1934->1673->1675->1674->1837->1671->1828->1825->1817->1815->1634->1814->1813-
>1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->54
18->5431->5425->5429->5426->5428->5434->5435->5433->5436->5398->5404->5402->5396->5395->5292->5282-
>5283->5284->5280->50
```

Distance: 150784.61, Energy Cost: 287931, Iterations: 3527, Time taken: 0.0153 seconds

2.1 Uninformed Search

In tasks 1 and 2, uninformed search is used to determine the shortest path. For both tasks we can analyse the graph as a directed graph since moving in reverse along the same edge doubles both distance and energy cost with no benefit. The graph may contain cycles. We use UCS to explore the graph in task 1.

Task 1

Time Complexity Number of nodes which have a shortest distance path to the target node than the starting node	**Space Complexity** Same as time complexity
Completeness Yes	**Optimality** Yes

Empirical Analysis

Number of nodes explored: 32514 nodes, V+E = 65028, Depth of answer: 131 nodes deep

Avg. branching factor of graph: b, Assuming b = 2,

Complexity of BFS for same depth = $O(b^d)$ = 2.72×10^{39}

UCS performs ~1.02×10^{33} times better than BFS for this problem

Task 2

When optimising only for shortest distance, UCS performs well because the nodes at the frontier are guaranteed to have the shortest distance from the starting node. Thus, only one path needs to be cached for each node and nodes are explored at most once, minimising both time and space complexity. In the worst case, all nodes (vertexes) and edges are traversed once, for $O(V + E)$ complexity.

However, this task requires the consideration of two resources. In this case, the first exploration of a node might provide shortest overall distance but large energy cost. Therefore, nodes may have to be explored more than once, greatly increasing overall complexity.

Without consulting the literature, two possible solutions can be conceived heuristically. BFS to explore all possible paths to the target, or a UCS using an evaluation function $f(n) = \sum k_i \times c_i(n), i \in N$ (otherwise called best-first search). For this variation, instead of using solely distance to measure cost several cost functions can be combined to provide an overall cost, with weights assigned to each cost function (e.g., distance might be judged to be 2x as important as cost).

Best-First Search

By tuning the weights of the cost functions, we can optimise for either the best distance or best energy cost result.

Time Complexity Number of nodes with cost <= starting node	**Space Complexity** Same as time complexity
Completeness No.	**Optimality**

For some graph with only one path $n_1 \rightarrow \cdots n_n \rightarrow \cdots \rightarrow n$, where n_i are vertexes in the graph, if there are two paths that include n_n – the correct path with lower energy cost and a path that will exceed the overall energy cost (wrong path), given a cost function that penalizes distance, the wrong path will be chosen. Since all nodes are explored only once, the only valid path under the constraints will be discarded.	No. Optimal algorithms exist as a subset of complete algorithms. Since the algorithm is not complete, it is not optimal.

Number of nodes explored: 12616/26084 nodes, depending on parameter

Breadth-First Search

As analysed in Task 1, the complexity of BFS for this graph would make a complete traversal of the graph infeasible. By providing some conditions for the traversal, we can limit the number of nodes explored.

Number of nodes explored: 16600000 nodes

2.2 Informed Search

In task 3, A* Search is used to determine the shortest path. A* is guaranteed to return the least-cost path from the start to the goal if the heuristic used is admissible, whereby the heuristic will not overestimate the true cost (energy cost or distance) to the target.

The coordinates of each vertex is given, as well as the distance of an edge. Therefore, we can create a heuristic to calculate the estimated distance cost of path to the target. However, as no other energy information is given, we cannot estimate the energy cost.

We calculate the estimated distance to the target vertex with the equation

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

To derive a heuristic function, we can take the straight-line distance between two points. For some vertices $p_n = (x_2, y_2), p_g = (x_3, y_3)$, where p_n is the next vertex and p_g is the goal,

$$h(n) = sqrt((x_3 - x_2)^2 + (y_3 - y_2)^2)$$

The actual distance $d(p_n \rightarrow p_g)$ is equal to $\sum_{i=0}^{k-1} d(p_{n+i} \rightarrow p_{n+i+1})$, where $k =$ number of nodes on the shortest path to the goal and $d(p_i \rightarrow p_{i+1})$ is their true distance as given by the data.

For A* to return the least-cost path, the heuristic function must never overestimate the actual distance (must be admissible). Given that the heuristic calculates straight-line distance (the shortest path between two points) while the actual path must follow along the edges and therefore covers a circuitous route, the actual distance will always be larger than the estimated distance.

Furthermore, when comparing the calculated straight-line distance between two vertices against the actual distances given for each edge, the given distance is always equal to or larger than the calculated straight-line distance. (*Mean difference: 92.31, Std Dev: 100.80, n=40.5%*. [2] This discrepancy could be due to winding/curved roads, great-circle distance or other factors). However, the given distance is never smaller than the calculated distance. This is significant because if the calculated distance was larger than the actual distance, then $h(n)$ could be larger than the actual straight-line distance for some routes.

Therefore, we can conclude that $h(n)$ is admissible, and A* will return the shortest distance path.

Initially, we implemented A* keeping track of the set of nodes explored. Running the A* algorithm, we can determine the shortest distance path. In the event that the shortest distance path exceeds the energy cost, we must perform backtracking search. For least distance path $p_1 \rightarrow p_2 \rightarrow \cdots p_{t-2} \rightarrow p_{t-1} \rightarrow p_t$, we begin uninformed search from a previous node to completely search the space for a path minimises distance and satisfies the energy cost constraint.

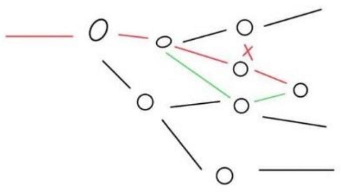

Fig 1. Backtracking Search
The red path is the original least distance path, and the green path is the constraint satisfactory path

By beginning uninformed search from p_{t-2} (the second last node), and propagating backwards if there is no solution, (start from the p_{t-3}, p_{t-4} etc.), we can iteratively propagate backwards until a valid solution is found.

Since this solution retraces the same paths many times, theoretically, memoising traversed paths will reduce the complexity of the overall solution.

The uninformed search used could be any algorithm from those previously mentioned, including Best-First search for optimal results. However, we implemented UCS using energy as the cost in order to really minimise the runtime of the overall algorithm.

However, we can simplify the approach by keeping track of the edges explored instead of the nodes explored. In this case, we can guarantee a complete traversal of the graph, while maintaining that paths that have an energy cost greater than the maximum will not be explored.~~s, such that any path~~

Time Complexity Generally, the complexity of A* is exponential wrt depth of target node d The complexity of the backtracking search from branch-off point of A* path n_b is $(depth(n_b \rightarrow \cdots \rightarrow n_t) \times b)!$, where b is the average branching factor of the graph	**Space Complexity** Same as time complexity
Completeness Yes, since UCS with energy cost is used as the backtracking search algorithm	**Optimality** No. A* requires the heuristic to be both admissible and monotonic for an optimal solution.

Number of nodes explored: 35277~~374~~

[1] – Average number of node references, given in verify.py

[2] – Comparison between calculated distance and given distance is done in investigate_distance.py

GitHub repository for this problem: https://github.com/zhuweiji/jubilant-tribble