**AC21007 – Algorithms and Artificial Intelligence**

**Team members:**Ahmid Omarzada – 2527809  
Ahmed Youssef – 2507690  
Anas Saad – 2510059  
Huzaifa Ijaz – 2541544  
Ramin Hashemi – 2508573

**Word count: <1880 Words – 1500 +/-10%>**

**AI PROJECT REPORT:   
Finding the shortest path to a petrol station of the lowest price, within a limited travel distance considering fuel consumption.**

**INTRODUCTION:**

The project is about locating the most cost-effective path to a petrol station of the lowest price, or closest to a user specified price, within a certain travel distance. Each location is either a petrol station, or a non petrol station, and has latitudinal and longitudinal coordinates, by which the distance and fuel consumption between each is calculated.

**PROBLEM DEFINITION:**

**State:** a state is represented by the location’s ID, the type of the location (whether it’s a petrol station or not), latitude, longitude, and the fuel price at that specific state if applicable.

**Initial State:** the initial state will be the location with ID 13 which is not a petrol station and carries the coordinates (53.796851 -3.017311) of latitude and longitude respectively.

**Actions:**

* **Travel to different location:** the agent moves to a different location within a limited travel distance of 75 km.

**Transition Model:** the state is changed by moving from the current location to the next location and updating the current coordinates to the new location’s coordinates. The decision of where to move is based upon the search strategy used in that context.

**Goal Test:** the goal is met when the agent reaches a location that is a petrol station, and is offering fuel of the lowest price. The search can also aim for the closest price to a user specified price. In the default example’s case, the price being checked for it will be 137.1 pence per litres.

**Path Cost:** Each movement between one location and the other will cost whatever the distance was between both location, that will be used along with a constant fuel consumption rate of 0.5 per km travelled, which will then be multiplied by the price of the petrol station reached to finally get the amount of money paid for the whole trip.

**WHY IT MATTERS:**

It is important to find a solution to the petrol price based navigation system optimisation challenge for a number of reasons. Finding the most cost effective path to a station with the lowest price may save a lot of money for both people and businesses. Shorter travel path/distances on efficient paths result in reduced fuel usage. This immediately contributes to lower carbon emissions, which is very beneficial to the environment. Ultimately, this would support the more general objective of maximising benefits while utilising the least amount of fuel.

**THE DATA:**

The dataset for this project is a mixture of multiple larger datasets of petrol station locations and their prices reduced down to fit the scope of the project. The modified dataset contains 165 locations.

The dataset was then organised like the following:

* ID
* Latitude
* Longitude
* Type of location
* Fuel price

The data was then remodelled into a list of nodes along with an adjacency matrix to represent the connections between a location and the other within the limited travel distance.

**RESULTS:**

We decided to use these 2 uninformed search strategies and 1 informed search strategy with one extra, which are:

* Bidirectional search (uninformed)
* Depth limited search (uninformed)
* A\* search (informed)
* Greedy best first search (extra informed search)

We also agreed before applying the search algorithms that the A\* search should theoretically be the best search algorithm amongst them as it uses path cost along with heuristic values to reach the goal. (keeping in mind the limited travel distance is 75 km meaning that each node expands based on this distance)

We also decided to measure and compare runtimes in microseconds instead of milliseconds or seconds as the algorithms were running too fast.

**Bidirectional search**

For the bidirectional search, the start node was the location/node of ID 13. The goal node has to be known beforehand (the closest node with the price 137.1 pence per litre) which was the node with the ID 134. After the applying the search, we retrieved the following information:

**Intersection node’s ID:** 2  
**Path:** 13, 2, 7, 134   
**Distance travelled:** 177 KM  
**Fuel consumed:** 88L  
**Path cost:** £1.37 \* 88L = £120.60  
**Average runtime:** 617.7 microseconds

Bidirectional search is complete but not optimal in our case since there are cases where the goal node is directly reachable from the initial node, but the bidirectional search tries to find and stops at an intersection node, thus leading to a path that is at least 3 nodes in length, which is not an optimal path.

In terms of complexity, the time complexity for bidirectional search is O(bd/2) where “b” is the branching factor and “d” is a constant depth from the start node to the end node. This is because it explores nodes from both the start and the end, reducing the depth of the search to d/2. Space complexity will be the same as the time complexity because it needs to store the explored nodes in the forward and backward directions until intersection.

**Depth limited search**

For the depth limited search, we chose to set the depth to 7 levels. The following results were outputted:

**Goal node’s ID:** 134  
**Path:** 13, 163, 161,159, 150, 148, 157, 134   
**Distance:** 288 KM  
**Fuel consumed:** 144L  
**Path cost:** £1.37 \* 144L = £197.30  
**Average runtime:** 120.4 microseconds

With this search being not complete, the depth limited search found a solution which was the same location/node with the lowest price as the bidirectional search with a much lower runtime of **120.4 microseconds**, but a much higher path cost of **£197.30** compared to **£120.60**. This solution is not optimal as there are goal nodes much closer to the initial node, and different paths that lead to less fuel consumed.

In terms of complexity, the time complexity for this search is O(bl) where “l” is the depth limit chosen for the search. The calculated time complexity would be O(28^7). This time, the space complexity is different from the time complexity and is O(b\*l), which would be O(29\*7) = O(203).

**Heuristics**

For the heuristic function that will be used in the following informed searches, we needed to know the goal node because it is needed in guiding the algorithms towards it. We have a function that extracts the nearest goal node (nearest petrol station location with the lowest price) into a goal\_node object which is then passed into the heuristic function.

After testing 3 distance formulae to build our heuristic function upon, we discovered that Euclidean distance would be the best option to proceed with as it was the most accurate at measuring the distance in our case.

Our heuristic function uses Euclidean distance between the current node and the goal node to place a heuristic value on the current node, which will then be chosen from the list of neighbouring nodes if it possesses the lowest heuristic value, leading the algorithm to choose to travel to it.

**A\* search**

After applying the A\* search, we obtained the following results:

**Goal node’s ID:** 134  
**Path:** 13, 116, 109, 134  
**Distance:** 158 KM  
**Fuel consumed:** 79L  
**Path cost:** $1.37 \* 79L = $108.23  
**Average runtime:** 292.7 microseconds

The A\* search is complete as our dataset is finite, and is also optimal since our heuristic function is both admissible and consistent(\*). The goal node that the A\* search found was the same node with ID 134 as the other two searches. Although the runtime was slightly higher at **292.7 microseconds** than depth limited search at **120.4 microseconds**, the path cost came in at **$108.23** compared to both, almost halving the path cost of the depth limited search and saving **£89.07**, using significantly less fuel.

In an ideal situation where heuristic function is perfect and the problem is very well defined, A\* has a time complexity of O(bd), but this depends on the quality of the heuristic function implemented. In our case, the Euclidean distance is calculated in O(n), but A\* still has exponential time complexity. The space complexity for the A\* search could be potentially high as it has to store an open and closed set to keep track of explored and unexplored nodes, and it could quickly run out of space before time.

**(\*) Admissibility and consistency of heuristic function:** our heuristic function is admissible as it is a straight line distance directly from the current node to the goal. It is consistent because: consider the A\* path [13, 116, 109, 134]

h(13): 157.8 km  
cost(13, 116): 69.5 km  
h(116): 88.7 km

h(13) <= h(116) + cost(13, 116)  
157.8 <= 88.7 + 69.5  
157.8 <= 158.2

Which is true for all nodes in the path.

**Greedy best first search (extra search)**

After running the algorithm, we received the following results:

**Goal node’s ID:** 134  
**Path:** 13, 116, 109, 134   
**Distance:** 158 KM  
**Fuel consumed:** 79L  
**Path Cost:** $1.37 \* 79L = $108.23  
**Average runtime:** 342.9 microseconds

Greedy best first search is neither optimal nor complete, it can get stuck in loops or fail to find any solution, especially if the heuristic function is poorly designed of if the search space is too complex. In our case, it found a solution that is identical to the A\* search in goal node/location, path cost, and path, except for the runtime which was slightly higher by around **50 microseconds**. The solution it found was the optimal solution as there are no shorter paths to a goal node.

In terms of complexity, the time complexity for the greedy best first search is O(bm) where “m” is the maximum depth of the search space, which would be O(29m). The space complexity for this algorithm is the same as the time complexity.

**EXTRAS:**

We have included an interactive GUI that the user can use to enter and change each of the following:

* **The initial node**
* **Maximum travel distance**
* **Desired price (default is lowest)**
* **Search algorithm**
* **Depth limit (only for depth limited search)**

The search button is pressed to apply the algorithm to the entered details.

**FUTURE ENHANCEMENTS:**

A possible enhancement may include optimising the heuristic function that we implemented into the informed search algorithms to maybe consider both distance and fuel consumption as a combined heuristic with different weights on each variable. While the Euclidean distance heuristic was used in our A\* and Greedy Best-First Search algorithms, there is always room for improvement by investigating more complex heuristics that are customised to the unique features of the issue domain. Furthermore, a larger dataset could be useful in thing like adding real time data updates on price pricing and road conditions might improve our solution's relevancy and accuracy by directing users to the most cost effective petrol stations that are open when they need them.

**CONCLUSION:**

The A\* search has the best results from the perspective of prioritising fuel consumption and money paid for fuel. The depth first search was the worse in terms of the path cost as it was the highest, so it wouldn’t be considered a good option from that perspective.

In terms of runtime, the depth first search had the lowest runtime of all, followed by both A\* and greedy best first search, and then the worst of all was bidirectional search almost doubling or tripling them.

As a conclusion, A\* was the most optimal and best option to consider to solve this problem, because this solution has no other algorithm that can expand less nodes in a similar efficient manner.