COMP 472

Artificial Intelligience:

Mini Project 2

Submitted to:

Leila Kossem

December 3rd, 2022

By
Hamzah Sheikh, Student 40103993
Muhammad Ferreira, Student 40113326
Hamza Shaikh, Student 40129291

Concordia University

Analysis :

**Excel sheet can be found in the Github repository **

1. The length of the solutions across algorithms and heuristics. When do you have the lowest-cost solution?

Heuristic	Count of Heuristic	Average of Length of the Solution
Heuristic 1	**100**	**14.34375**
A/A*	50	13.47916667
GBFS	50	15.20833333
Heuristic 2	**100**	**14.34375**
A/A*	50	13.47916667
GBFS	50	15.20833333
Heuristic 3	**100**	**15**
A/A*	50	14.79166667
GBFS	50	15.20833333
Heuristic 4	**100**	**14.44791667**
A/A*	50	13.58333333
GBFS	50	15.3125
N/A	**50**	**13.47916667**
UCS	50	13.47916667
Grand Total	**450**	**14.41666667**

Table 1: Average Length of the Solution for each Heuristic

Algorithm	Count of Algorithm	Average of Length of the Solution
A/A*	**200**	**13.83333333**
Heuristic 1	50	13.47916667
Heuristic 2	50	13.47916667
Heuristic 3	50	14.79166667
Heuristic 4	50	13.58333333
GBFS	**200**	**15.234375**
Heuristic 1	50	15.20833333
Heuristic 2	50	15.20833333
Heuristic 3	50	15.20833333
Heuristic 4	50	15.3125
UCS	**50**	**13.47916667**
N/A	50	13.47916667
Grand Total	**450**	**14.41666667**

Table 2: Average length of the solution for each algorithm

By calculating the average of length of solution across all heuristics and algorithms, we noticed that the heuristics with the lowest-cost solution are both heuristic 1 and heuristic 2 with an average length of 14.34375.

Heuristic 1 checks the number of blocking vehicles in a game whereas heuristic 2 determines the number of blocked positions. Heuristic 4 calculates the number of blocking vehicles in addition to the number of vehicles blocking those in the way of the ambulance and heuristic 3 is value of heuristic 1 multiplied by a value which we chose to be 5. These heuristics affect the length of the solution as it is used to determine how our priority queue will be sorted. These heuristics will be expanded in section 2

In terms of the algorithms, the lowest-cost solution is UCS followed by A/A* and GBFS which follows the theory learned in class. The reason UCS has the lowest-cost is UCS looks at all nodes at every depth until it reaches the a solution. This is done through the function f(n)=g(n) where g(n) is the path from initial node to node n. As a result, the solution will always be the lowest cost available, however this may result in longer execution times. For GBFS, the solution is found according to the heuristic provided through the function f(n) = h(n) where h(n) is the value of the heuristic from node n to the goal node. This will result in a faster execution time due to the fact this is an informed search, however this will not always result in the most optimal solution. For A/A* algorithm, it will choose the cheapest path according to the function f(n)=g(n)+h(n) where g(n) is the path from the initial node to a node n and h(n) is the value of the heuristic function at that given node. This is a mix between both GBFS and UCS, resulting in performance with results falling in between the aforementioned algorithms. Note that this algorithm is defined as A* algorithm for heuristics 1 and 2 as they guarantee the lowest cost solution whereas it is defined as an algorithm for heuristics 3 and 4 as they have higher average length of solution than UCS which guarantees the lowest cost solution. This impacts the admissibility of the algorithm which will be further elaborated in section 2.

2. The admissibility of each heuristic and its influence on the optimally of the solution.

For a heuristic to be admissible if it never overestimates the cost of reaching the goal:

$$h(n) \leq h*(n) \text{ for all } n$$

To validate if the heuristics were in fact admissible, we compared the average length of solution for the A algorithm for each heuristic to that of the average length found using the UCS algorithm. This is due to the fact that UCS always produces the smallest solution length.

Note that admissible heuristics may temporarily reach some non-goal states as it goes along a sub-optimal path. Also, for A* algorithms, if we have a node in n in a open or in closed queue. We may come across a node n again, but with a lower f(n) due to a lower g(n) and same h(n). As a result, we may need to update the info of a node n in open or put n back into open although it has already been visited which could be costly.

Heuristic	Count of Heuristic	Average of Length of the Search Path
Heuristic 1	100	1083.12
A/A*	50	1479.36
GBFS	50	686.88
Heuristic 2	100	1083.12
A/A*	50	1479.36
GBFS	50	686.88
Heuristic 3	100	779.92
A/A*	50	872.96
GBFS	50	686.88
Heuristic 4	100	769.43
A/A*	50	996.9
GBFS	50	541.96
N/A	50	1874.34
UCS	50	1874.34
Grand Total	**450**	**1033.946667**

Table 3: Average Length of Search Path

Heuristic 1:

Heuristic 1 takes into consideration the number of blocking vehicles in front of the ambulance. Although in normal game rule sets, this would be not much different than Heuristic 2, in the analyzed version of the game, a valet system can remove horizontally placed cars in front of the ambulance. As such, this is a valid way to determine progress towards the goal state.

When analyzing if this is an admissible heuristic, we can see that the average length of solution is of around 13.48 for the A/A* algorithm. When comparing this to the UCS, which also produces an average solution length of 13.48, we can conclude that H1 is admissible.

<u>Heuristic 2:</u>

Heuristic 2 is calculated based on the number of blocked spaces in front of the ambulance. Unlike H1, a horizontally placed car in front of the ambulance will be counted for all the positions it occupies. This is a valid way to analyze progress towards the goal state in more traditional puzzle configurations of the game, where cars cannot be placed horizontally in front of the ambulance. In cases where a car is placed horizontally in front of an ambulance, this one lacks in performance, as it will indicate that the progress is much further than it actually is.

When analyzing if this is an admissible heuristic, we can see that the average length of solution is of around 13.48 for the A/A* algorithm. When comparing this to the UCS, which also produces an average solution length of 13.48, we can conclude that H2 is admissible.

<u>Heuristic 3:</u>

Heuristic 3 is the value of the of Heuristic 1 multiplied by a constant value of 5. As such, a difference in performance for this heuristic can only be viewed in the A/A* algorithm. As A/A* algorithm uses both the node depth and heuristic value, this heuristic value is given more weight as opposed to the depth of the node. As such, this is a valid heuristic for the A/A* algorithm, but redundant to H1 in the GBFS algorithm.

When analyzing if this is an admissible heuristic, we can see that the average length of solution is of around 14.79 for the A/A* algorithm. When comparing this to the UCS, which produces an average solution length of 14.48, we can conclude that H3 is not admissible as it produces a longer average solution length.

<u>Heuristic 4:</u>

This heuristic is the total number of cars blocking the ambulance, in addition to the number of cars blocking the movement of these cars themselves. This gives us a better insight on the board positioning of not just all cars in front of the ambulance, but also if these cars themselves can be moved. As such, it is an adequate way to analyze the progress towards the goal state.

When analyzing if this is an admissible heuristic, we can see that the average length of solution is of around 14.79 for the A/A* algorithm. When comparing this to the UCS, which produces an average solution length of 13.58, we can conclude that H4 is not admissible as it produces a longer average solution length.

<u>Benefits of H3 and H4</u>

Although H3 and H4 are not admissible, it can be concluded that they result in a shorter search path length on average as compared to H1 and H2. These will result in shorter execution times, with the trade-off of potentially finding a non-optimal solution.

3. The execution time across algorithms and heuristics. Is an informed search always faster?

Heuristic	Count of Heuristic	Average of Execution Time (in seconds)
Heuristic 1	**100**	**17.2574**
A/A*	50	23.9388
GBFS	50	10.576
Heuristic 2	**100**	**17.375**
A/A*	50	24.0788
GBFS	50	10.6712
Heuristic 3	**100**	**11.7325**
A/A*	50	12.8044
GBFS	50	10.6606
Heuristic 4	**100**	**11.5571**
A/A*	50	15.0896
GBFS	50	8.0246
N/A	**50**	**29.8596**
Grand Total	**450**	**16.18928889**

Table 4: Average Execution Time for Each Heuristic

Algorithm	**Count of Algorithm**	**Average of Execution Time (in seconds)**
A/A*	**200**	**18.9779**
Heuristic 1	50	23.9388
Heuristic 2	50	24.0788
Heuristic 3	50	12.8044
Heuristic 4	50	15.0896
GBFS	**200**	**9.9831**
Heuristic 1	50	10.576
Heuristic 2	50	10.6712
Heuristic 3	50	10.6606
Heuristic 4	50	8.0246
UCS	**50**	**29.8596**
N/A	50	29.8596
Grand Total	**450**	**16.18928889**

Table 5: Average Execution Time for each algorithm

Given the following data, it can be inferred that, generally speaking, the Fastest Search is the Greedy Best First Search, while the slowest algorithm is the Uniform Cost Search. This falls in line with the theory of each algorithm, as the Uniform Cost Search will explore the entire tree until the shortest path reveals itself, while the Greedy Best First Search searches the tree based on the lowest Heuristic, meaning it will find a solution if one exists, but that solution will not always be the most efficient/best solution. The trade off for the UCS, is while it will always provide the shortest path/best solution, it will explore more of tree to do so, meaning the search will be costly, and in turn, take up more time, as is reflected in the above results.

The GBFS will always provide the fastest solution, as it searches based on lowest heuristic value rather than considering the shortest solution path length. This means it leaves much of the tree unexplored, meaning the shortest path to the goal may be missed/left unaccounted for. This does however mean the search will be much fast, and not at all costly, leading to a very short search time, as shown in the above results.

Looking finally at our A/A* search, which uses a mix of heuristics as well as considering the shortest path length, we see that it as well has a lower search time than the UCS. This value lies almost exactly in the middle of both the UCS and the GBFS, as it is a compromise of the two algorithms, that being that UCS is only concerned with shortest path length, and therefore searches the entire tree to find the lowest cost/depth, while GBFS is only concerned with heuristic value, and follows only the path that presents the lowest face value heuristic cost until the goal is achieved, and iterates to the next if no goal is found. The A/A* algorithm works similarly to the GBFS, in that it follows the path that presents the lowest A* value. Where it differs from GBFS, is that the A/A* value is a mix of the Heuristic as well as the shortest path length of the tree, meaning depending on what the heuristic value is, we can make this algorithm favor either the face value depth of the face value heuristic. This causes the algorithm to find a better/more optimal solution path than the GBFS, but likely a worse one than the UCS. This means it takes up more resources than the GBFS, but less than the UCS, and is therefore in the middle of both in terms of solution speed.

From this we can conclude that an informed search is always faster than a non-informed search, as a non-informed search searches the entire tree, while informed searches use rules to filter out parts of the tree, thereby avoiding searching the entire tree, saving resources and improving execution times. We can further conclude, looking at the heuristics for each algorithm, that on average Heuristic 4 shows the greatest results in terms of execution time, however for each we see for each informed search (A/A* and GBFS) has their own most efficient heuristic, that being Heuristic 4 for GBFS (8.0246 Seconds) and Heuristic 3 for A (12.8044 Seconds). We can also observe that the slowest execution time for heuristics on average can be observed as the Heuristic 2, which also holds true when looking at the informed searches individually, that being 24.0788 Seconds for A* and 10.6712 Seconds for GBFS. It must be noted, the Heuristic 1 and Heuristic 3 are essentially the same for the GBFS, and the discrepancy between their execution time could be due to the difference in the difficulty in calculating each heuristic, while there should be a large difference between Heuristic 1 and 3 for the A/A* algorithm.

4. Other interesting facts that you deem worthy of describing.

An interesting fact that came about while trying to optimize our search algorithms execution time was the influence of other softwares on our program. We were able to notice a great increase in performance as we closed auxiliary programs that were running in the background (google chrome, discord, etc…etc). On some games, our solution time nearly halved (going from 88 seconds to 39 seconds for Solution 2, Heuristic 1, GBFS [Later 1.39 Seconds after further optimizing our algorithms]).

Another interesting fact/experiment we did, was we ran our algorithms against the hardest games of Rush Hour we could find on the internet. The game we tested has a 51 move solution game we found on Michael Fogleman's[1] blog, where he does extensive research into algorithms used to solve Rush Hour Games. Our solutions gave us the following results:

[1] https://www.michaelfogleman.com/rush/

Algorithm	Moves To Complete	Execution Time (in seconds)
A/A*		
Heuristic 1	51	38.76
Heuristic 2	51	40.04
Heuristic 3	53	26.61
Heuristic 4	51	33.42
GBFS		
Heuristic 1	54	17.68
Heuristic 2	54	17.72
Heuristic 3	54	17.92
Heuristic 4	57	15.0
UCS		
N/A	51	37.03

Table 6: Execution Time and Moves for "Hardest Rush Hour Game"

As you can see above, we were actually surprised to see that the A/A* algorithm was essentially as efficient as the UCS algorithm, with the exception of Heuristic 3, which saw a rise in moves from 51, to 53. This goes to show that the informed searches, while they don't necessarily guarantee the absolute most efficient result, could often achieve it, so long as the parameter they use to measure is properly adjusted for the use case/data set. Furthermore, we can observe that the trends we noticed were still intact, those being GBFS was the algorithm that found the most inefficient route, but took the least amount of time to execute, the A/A* algorithm provided the second most efficient answer set, and was somewhere between the UCS and the GBFS (although as we previously mentioned, the results were more than accepted and exceeded expectation), and finally the UCS was the most costly in terms of resources and and took the most time, but consistently produced the most efficient goal path.

In this game case, we concluded that the A/A* algorithm, paired with heuristic 4 would be best in this use case. This is due to its balance of resources taken, and as well as the seemingly guaranteed best path result as opposed to the long execution time of UCS and the inefficiency of the paths produced from GBFS.