CS 180

Written Report

The comparison of the three heuristics: zero heuristic, block heuristic and the invented desire heuristic and their effectivity

Prepared by:

Gueco, Maria Rosario T. - 200664076

Yao, Faneallrich Li - 201455331

The Abstract

The Abstract of this paper is to show the author's attempts to solve the Rush Hour problem through the use of the three Heuristics for the famous A-star search (A* search) The first heuristic is the trivial zero heuristic or an A* without a heuristic function which will function like a normal BFS search. The second heuristic is the blocking heuristic wherein we will count the number of cars blocking the primary car from the goal state. Finally, the third heuristic is the advanced Heuristic which we named the desire-block heuristic wherein the opposing blocking can show what move they desire to show effective movement.

This paper will contain the description of the A* search with the three heuristics, the data of the program running these three heuristics and the thereotical in-depth analysis of the three heuristic with the data as a justification to these theoretical analyses.

Description of the A* Search:

The A* Search implementation uses two queues, one for explored nodes and another for unexplored nodes. At the start of the search, the initial node is found in the queue for the unexplored nodes and the queue for explored nodes is empty. A loop is used where in each iteration a node is taken from the unexplored queue and expanded then placed in the unexplored queue.

Each child of the expanded node is compared first to the nodes in the explored queue then the nodes in the unexplored queue to check for a similar state. If a node already exists that contains he same state, the possible cost returned by the heuristic function is then compared. If the new node has a lesser value, the node with the repeated state is removed from the queue it exists in and the new node is placed in the unexplored queue in order according to its possible cost value. This also applies to when the new node has a unique state. The new node is discarded if it has a state that already exists and has a lesser possible cost.

The search halts if it has found the goal node or the unexplored queue has been exhausted.

Description of the Desire Heuristic:

The Desire Heuristics returns a cost according to the number of cars blocking the main car, the max number out of the minimum number of cars that need to be removed for the blocking cars to exit and whether or not the main car will take one or two moves to reach the goal state had the board been clear.

First, each car is checked if it is blocking the main car. If it is blocking the main car, the space above and below is checked if there is enough clearance for the car to move and give way to the main car (ignoring any other car occupying the space). If there is enough space, the cars in the way of the blocking car's exit above or below are counted. Let's call this the exit value. If there is space above and below, we will take the minimum exit value out of the two. The exit value will then be compared to the other blocking car's exit value. Since all blocking cars must be removed before the main car can pass through, we will take the maximum exit value. This ensures that we can remove the blocking car with the most cars blocking its fastest exit.

The number of blocking cars is then added as well as the least number of moves (one or two) for the main car to reach the goal.

Data Processing:

in.txt

6

1 2 h 2

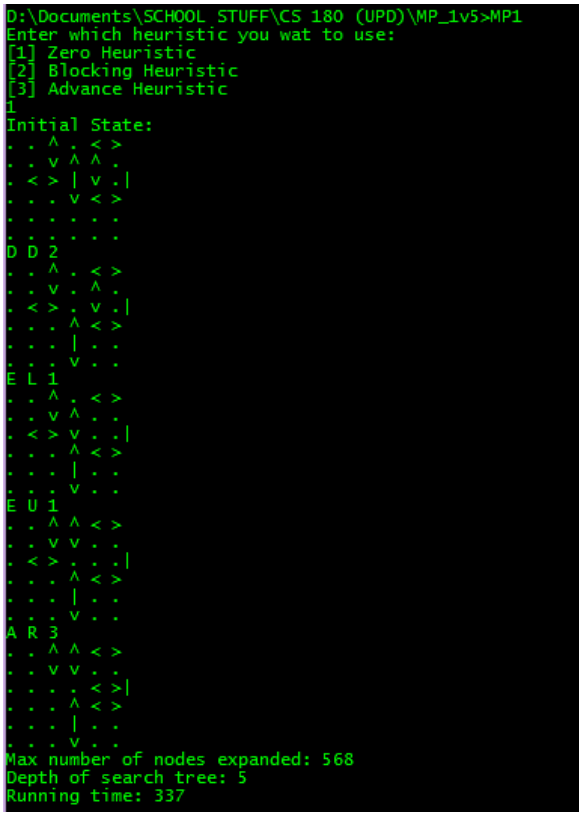
2 0 v 2

4 0 h 2

3 1 v 3

4 1 v 2

4 3 h 2



out.txt

ZERO H

Cost: 4

D D 2

E L 1

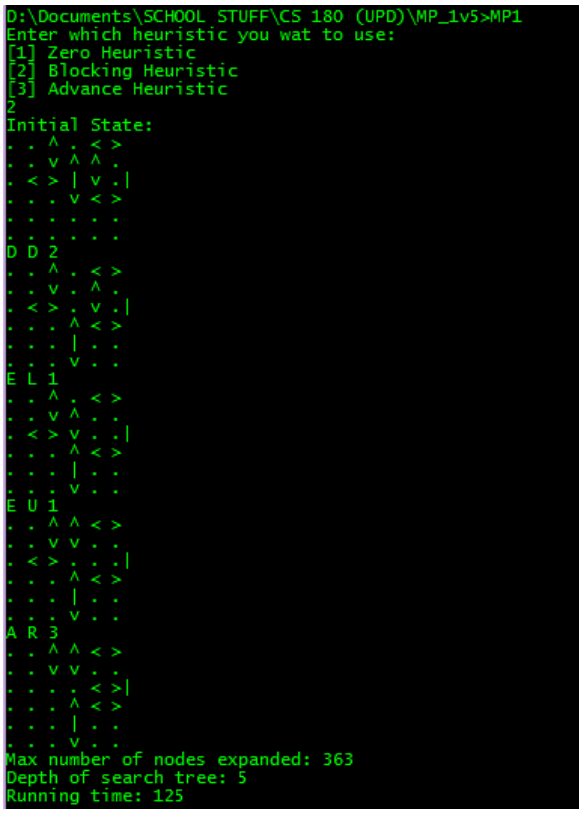
E U 1

A R 3

Max number of nodes expanded: 568

Depth of search tree: 5

Running time: 337



out.txt

BLOCKING H

Cost: 4

D D 2

E L 1

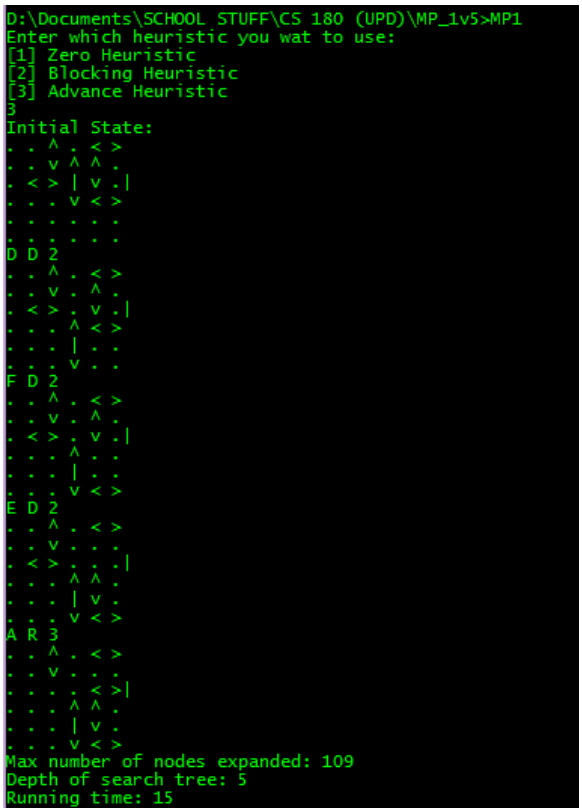
E U 1

A R 3

Max number of nodes expanded: 363

Depth of search tree: 5

Running time: 125



out.txt

ADVANCE H

Cost: 4

D D 2

F D 2

E D 2

A R 3

Max number of nodes expanded: 109

Depth of search tree: 5

Running time: 15

Analysis of the Zero Heuristic:

The Zero Heuristic or may be referred to as the BFS search will exhaust every possible search node until it reaches the goal state or the solution state. The BFS search has an exponential time complexity and space complexity and may take a while in exhausting states in extremely large inputs. Furthermore, the completeness of the BFS is evident as it will be able to search for every possible node in the given graph until it reaches the goal state. However, we can question the fact that if it is optimal the fact that it is possible in the worst case scenario that it will exhaust the whole graph and in large inputs that will take a generally large amount of time.

Proof: suppose we have a Graph G where G(V,E) where V is the number of Vertices and E is the number of edges. Doing a BFS search is possible that the with the start state Q and the goal state P wherein the traversal will visit every node adjacent to Q until the nth iteration. In this traversal once it reaches the goal state the algorithm will terminate. However, in the worst case is that every possible state will be exhausted in order to achieve the goal state hence showing how the BFS is complete but not optimal.

Using the data to further justify, the BFS visits the highest amount of nodes which shows the evidence of the proof. Despite the BFS generates the same output, as the two heuristic, the high number of nodes expanded shows the zero heuristic as less optimal compared to the other two heuristics Henceforth, this will proves that the trivial search despite producing a solution is not the most effective way to solve the rush hour problem.

Analysis of the Block Heuristic:

We can see that the Blocking Heuristic will be able to decrease the amount of possible nodes due to no longer visiting repetitive states as these are no longer part of the solution. With sharing the same worst case scenario of the BFS, the complexity of the Block Heuristic is also exponential in terms of space and time. However in most cases we must take into consideration that the Blocking Heuristic will no longer be considering some of the states in the graph due to the irrelevance of the state for the heuristic hence we can see that it is complete and is more optimal compared to the BFS as it will be able

to arrive at the goal state and in the worst case scenario, it will be similar to the zero heuristic. Hence we can say that there will be a decrease of n number of states where n is greater than or equal to 0. Although in large input, same as the BFS it will take a huge amount of time. However, due to the blocking heuristic the amount of time will decrease due to the n number of states that will be removed.

Proof: Suppose we have a Graph G where G(V,E) where V is the number of Vertices and E is the number of edges. In the extreme worst case, doing an A* with the Blocking Heuristic search is possible that with the start state Q and the goal state P wherein the traversal will visit every node adjacent to Q until the nth iteration. In this traversal once it reaches the goal state the algorithm will terminate. However, in the worst case is that every possible state will be exhausted in order to achieve the goal state hence showing how the Blocking Heuristic is complete and has the same optimization of the BFS in terms of the worst case input. However, in most cases the n number of nodes where n is greater than 0 will be taken out by the blocking Heuristic as it will no longer traverse there. In Comparison, having to visit V number of states it will be visiting V – n hence showing that V – n will be lesser than V due to the fact that n is greater than 0. Hence it shows in most inputs, the blocking Heuristic is more optimal compared to the zero Heuristic.

By observing the data, we can see a huge difference between the Zero Heuristic and blocking Heuristic in terms of both the number of nodes expanded and the time. However, if the input is in the extreme worst case, they will have the same yield as the BFS search.

In Conclusion the A star search with the Blocking Heuristic is at least as effective as the zero-heuristic but in general is more effective compared to the zero- heuristic. As shown in the proof and the data provided.

Analysis of the Desire Heuristic:

It is an application of the Blocking Heuristic with the addition of a voting system or we call desire system. In terms of complexity it is also exponential. However, in its average case it is more optimal compared to both Blocking and Zero Heuristic. As we can

see the Zero Heuristic will exhaust everything while the Blocking Heuristic will consider the number of cars blocking the main car hence will no longer need to explore all the states. In this Heuristic, the blocked cars have a desire system so we will know where they should be moved thus further narrowing down the search and be able to remove the need to traverse some states say we have k number of states that will no longer be searched. Comparing the number of states in their average cases BFS will need to search V number of states while the Blocking Heuristics will need to search V – n states where n is greater than 0. The Advanced Heuristic however will only need to search V-n-k states where k and n is greater or equal to 0.

Proof: As referenced by the proof above the extreme worst case will be similar to the BFS. However, Suppose we have a graph G(V,E) where V and E correspond to the number of vertices and edges. At the average case of input P to goal state Q the algorithm will be imitating the blocking heuristic in removing number of edges. However, instead of V-n where n is greater than or equal to zero it will remove k(where k is greater than or equal to 0) more states from being searched thus having to search only V-n-k edges by examining the two proofs above, V-n-k < V – n < V – n – k. Thus showing that the solution is more optimal compared to the previous searches.

In the data provided it is clear that the proof is justified as it has the least number of nodes expanded and has the fastest running time compared to the other two searches. Hence this justifies why our advanced Heuristic is more effective compared to the other two searches in both space and time.

Conclusion:

Overall, the Desire Heuristic shows the consistency and effectiveness of the A-star search. With this heuristic, it drastically cuts down the needed space by the Blocking Heuristic by being able to search through lesser number of states and a faster running time. Despite the fact that space can be easily foregone in the field of Computer Science, the fact that we have a heuristic that will be able to reduce the cost in terms of time is already crucial in our study. However, the Rush Hour Problem is still an interesting study in the field of approximation algorithms and further studies can show a better Heuristic

that will be able to reduce the complexity costs. In the program provided by the two students, it is clear that the solutions of the A-star search will be unique and similar to other solutions yet will add their own twists to contribute to the decrease of complexity costs of the A* search which is the purpose of Computer Science in finding more efficient and consistent solutions.