CIS*3700

Assignment 2

Gary (Gurpreet) Dhillon

Student # 0802321

Introduction:

The following report outlines the different computational searches used to find the minimum step solution in solving a given iteration of the "Rush Hour Game" as outlined in the Assignment 2 description. The report will follow along with the order the problems are presented on the Assignment 2 outline.

Code Submission:

The files submitted were written in python and should all be runnable on a system with python 2. The files submitted are as follows:

SearchProblem.py (this file was re used from Assignment 1 to make use of the bfs algorithm I wrote earlier)

RushHour.py (contains all code relevant to generating the game board and possible moves of cars)

Testing Instructions:

The values for the game should be entered into the provided "test.txt" file which already contains a past iteration of the game I used while testing.

With the desired car positions entered into the "test.txt" next you need to choose which of the 3 searches you would like to conduct in order to find a minimum path solution. Open up the "RushHour.py" file and navigate to line 149. Simply change the function called at the end to the desired search:

.bfs() – default, begin a brute force search in a breadth first manner to find a minimum path solution.

.bfes(0) – begin a best first search (using heuristic 1 which will be explained later) to find a minimum path solution.

.bfes(1) – begin a best first search (using heuristic 2 which will also be explained later) to find a minimum path solution.

Finally, type "python RushHour.py" into your terminal while in the source directory to begin the search.

Part A.

1. The brute force algorithm employed in this program is the breadth first search (bfs) algorithm I developed for the previous assignment. The bfs algorithm is designed to search all nodes (or states) on a particular depth (in the case of the rush hour game the depth would be the number of moves required to get to the current state). Thus, the program will not be checking possible solutions of length 4 until it has checked all possible solutions of length 3 first. This ensures all possibilities are exhausted before moving on resulting in a brute force approach with no strategy.

2. The best first search algorithm I developed works similarly to the bfs algorithm except that instead of entering new unvisited states into a regular queue the best first search algorithm enters them with a priority alongside the state. The priority is calculated using one of two heuristics I implemented (which will be explained further along in the report) which then assigns lower numbers (or higher priorities) to states that seem to be heading towards the goal state. When items are popped off of the minheap (which acts as a priority queue) the states which are deemed closer to the finish point will have higher priorities and thus be closer to the top of the heap.

3, 4

Euclidean Distance

The first heuristic I implemented involves calculating the shortest straight-line distance from the red car to the finish point (the finish point in this case is when the red car is in position (4,2) as this would only require 1 more right move to be outside the board and out of "rush hour"). The shortest straight line distance of course would always be in the range of 1-4 as the car can only move horizontally and thus must always have a vertical difference of 0 for the game to be solvable.

Is this heuristic admissible?

For the search to arrive at a minimal step answer it needs to ensure that the priority numbers being assigned to states are never over estimations so that the lowest possible cost solution is found first. For each state the following must be calculated and stored as its priority:

F(n) = G(n) + H(n)

Where F is the total cost to move to the new state, G is the cost incurred so far to move to the current state (or depth) and H is the value assigned by the heuristic to measure how close this state is to the goal state.

By using the shortest straight-line distance to the goal state the Euclidean Distance Heuristic always provides the minimum number of moves the red car would need to make it to the goal state. This is akin to a "best case" scenario where the player is free to move the red car straight from its current position to the exit. Thus for all scenarios this heuristic will always provide a distance equal to or less than the actual number of moves and so total cost F will never be an overestimate.

Number of Vehicles between red car and goal

The second heuristic involves calculating the number of spaces in front of the red car that are currently occupied by vehicles. With this method we will also only ever see values in the range 1-4 as the red car can have at max 4 spaces between it and the goal state and thus a max of 4 occupied squares ahead of it. This also means that it can never overestimate the number of moves from the red car's current position to the goal state as in the event there are 4 occupied spaces ahead of the red car (the max) the heuristic will predict a minimum of 4 moves (evidently much less than the actual required amount). In the event there are no vehicles ahead of the red car (the min) the heuristic will predict 0 moves to the goal state which is not ideal however also does not make the heuristic inadmissible as value F in the equality:

F(n) = G(n) + H(n)

still holds true as not being an overestimate of the number of moves to the goal state.

5. To determine which of the two heuristics will give higher admissible estimates it is necessary to explore how the estimation equation ties in to the moves/rules of the Rush Hour Game.

The Euclidean Distance heuristic will estimate lower priority (higher estimate) to any edge which does not involve the red car moving closer to the goal state. This means that for any given state, any edge which does not move the red car closer to the goal state will receive the same estimation value as the other same-depth edges resulting in behavior very similar to a breadth-first search of all edges share the same priority value. This will amount to this heuristic giving larger admissible estimates due to only one edge having a chance at a lower priority.

Looking at the “Number of Vehicles between the red car and goal” heuristic, any edge which frees up a space ahead of the red car will receive higher priority (lower estimate). Any edge which does not free up a space ahead of the red car will receive a priority estimate equal to the number of occupied spaces between the red car and goal state (at max can only be equal to the current Euclidean distance and in many other cases will be less).

Therefore we can conclude that the Euclidean Distance heuristic will provide higher admissible estimates than the Number of Vehicles between the red car and goal heuristic.

6.

Compare the search space explored by each approach

What follows is the test data across 15 iterations of the Rush Hour game, the iterations were generated by the provided Rush Hour website in the assignment 2 outline. Included is the contents of the test.txt file followed by the solution, depth and states visited.

```
X02H

B11H

C14H

D34H

E03V

F22V

G30V

O33H

P50V

Q00H
```

Euclidean:

```
Solution:  BL FU OL PD GD QR QR QR FU XR ED OL OL DR GD GD XR XR FD QL GD PU OR OR
OR EU EU CL FD FD BR FD OL OL EU OL GU XL XL GU GU QL QL GU XR XR DL PD PD PD
XR

('Depth: ', 51)

('states visited: ', 2663)
```

VehiclesInPath:

```
Solution:  BL ED FU OL OL OL GD GD DR GD PD QR QR QR FU XR XR XR GD FD OR QL EU PU
EU OR OR CL FD FD FD OL OL BR EU XL OL GU XL GU GU QL QL GU PD DL PD PD XR XR
XR

('Depth: ', 51)
```

```
('states visited: ', 2759)


BFS:

Solution:  BL DR ED FU GD OL OL OL GD GD GD OR EU OR PD QR QR QR FU XR XR XR EU
CL FD QL PU OR FD BR EU FD XL XL FD OL OL OL GU GU DL GU PD PD PD QL QL GU XR
XR XR

('Depth: ', 51)

('states visited: ', 2862)


X02H

A03H

B04V

C14V

D21V

E23V

F43V

O32V

P53V

Q25H


Euclidean:

Solution:  DU XR PU OU QR OU ED AR AR FU FU AR EU QL CU BU PU QL PU QL AR OD OD
OD XR CU CU CU XL OU QR QR QR OU OU BU ED AL AL BU AL AL EU QL QL OD OD QL OD
XR FU XR PD PD PD XR

('Depth: ', 56)

('states visited: ', 6484)

VehiclesInPath:

Solution:  DU PU QR ED AR OU OU AR PU PU FU FU BU AR AR EU QL CU OD QL QL OD OD
XR XR CU CU CU XL OU QR OU QR QR BU OU AL ED AL BU AL AL EU PD PD QL PD FU OD
QL QL OD OD XR XR XR

('Depth: ', 56)

('states visited: ', 6489)


BFS:
```

Solution: DU XR FU FU FU OU OU PU PU PU QR ED AR AR AR AR BU BU BU CU EU OD OD QL QL QL OD XR CU CU CU XL OU OU OU AL PD PD QR QR QR ED AL AL AL EU OD OD QL PD QL QL OD XR XR XR

('Depth: ', 56)

('states visited: ', 6694)

X32H

A33V

B44H

O20V

P03H

Q51V

Euclidean:

Solution: QU AD PR PR PR OD OD OD XL XL XL OU OU OU PL PL PL AU AU AU PR PR PR OD OD OD XR AU XR XR OU OU OU PL BL QD QD QD XR

('Depth: ', 39)

('states visited: ', 275)

VehiclesInPath:

Solution: QU AD PR PR PR OD OD OD XL XL XL OU OU OU PL PL PL AU AU AU PR PR PR OD OD OD AU XR XR XR OU OU OU PL BL QD QD QD XR

('Depth: ', 39)

('states visited: ', 271)

BFS:

Solution: AD PR PR QU PR OD OD OD XL XL XL OU OU OU PL PL PL AU AU AU AU BL PR PR PR OD OD OD XR XR XR OU OU OU PL QD QD QD XR

('Depth: ', 39)

('states visited: ', 280)

X02H

A10V

B31V

C24V

D54V

E35H

O30H

P13H

Q03V

R42V

Euclidean:

Solution: XR QU QU QU PL BD BD XR DU ER OL BD PR AD OL RU RU PR AD AD AD PL PL BU XL BU BU PR QD OL BU XR EL DD PR PR CU EL AU EL EL CD AU AU PL PL RD RD RD XR XR

('Depth: ', 51)

('states visited: ', 3974)

VehiclesInPath:

Solution: XR QU QU QU PL BD BD DU ER RU BD XR OL AD OL PR RU PR AD AD AD PL PL XL BU BU BU PR QD OL BU EL PR AU DD PR XR AU AU CU EL EL EL CD PL PL RD RD RD XR XR

('Depth: ', 51)

('states visited: ', 3974)

BFS:

Solution: XR DU ER OL QU QU QU PL BD BD XR AD BD OL PR RU RU PR AD AD XL AD PL PL BU BU BU EL DD PR PR AU PR CU EL EL EL CD PL QD OL BU XR AU AU PL RD RD RD XR XR

('Depth: ', 51)

('states visited: ', 4131)

X12H

A41V

B14V

C24V

```
D34H

O30V

P52V

Q35H

Euclidean:

Solution:  CU XL CU CU CU XR PU QL BU OD DR QL QL OD OD XR BU BU BU XL OU OU DL
DL DL OD OD XR AU XR PD PD XR

('Depth: ', 33)

('states visited: ', 5528)


VehiclesInPath:

Solution:  AU BU OD PU DR OD CU QL QL QL OD XR BU BU BU XL OU OU DL XL CU DL DL
OD OD PD PD DL CD XR XR XR XR

('Depth: ', 33)

('states visited: ', 5533)


BFS:

Solution:  XL AU BU CU CU CU CU XR OD PU DR OD QL QL QL OD XR BU BU BU XL OU OU
DL DL DL OD OD XR XR PD PD XR

('Depth: ', 33)

('states visited: ', 6493)


X02H

A03H

B14V

C25H

O32V

P53V


Euclidean:

Solution:  XR AR OU OU AR PU PU AR PU AR OD OD BU CL OD XR BU BU BU XL OU OU OU
AL AL AL OD OD OD XR XR PD PD PD XR

('Depth: ', 35)
```

('states visited: ', 704)

VehiclesInPath:

Solution: AR OU OU AR BU PU PU PU AR AR OD CL OD OD XR XR BU BU BU XL OU OU OU AL AL AL OD OD OD PD PD PD XR XR XR

('Depth: ', 35)

('states visited: ', 772)

BFS:

Solution: XR AR OU OU AR AR BU CL PU PU PU AR OD OD OD XR BU BU BU XL OU OU OU AL AL AL OD OD OD XR XR PD PD PD XR

('Depth: ', 35)

('states visited: ', 806)

X22H

A10V

B43H

C45H

O33V

P50V

Euclidean:

Solution: XL OU OU OU BL BL BL OD OD CL CL CL OD XR XR PD PD PD XR

('Depth: ', 19)

('states visited: ', 666)

VehiclesInPath:

Solution: XL OU OU OU BL PD PD CL PD CL BL BL OD OD CL OD XR XR XR

('Depth: ', 19)

('states visited: ', 741)

BFS:

```
Solution:  XL OU CL CL CL OU OU BL BL BL OD OD OD XR XR PD PD PD XR

('Depth: ', 19)

('states visited: ', 789)


X32H

A40H

B01H

C33H

D44H

E02V

F22V

G34V

O10H

P21H

Q05H

R51V
```

Euclidean:

```
Solution:  OL AL RU CR GU FD XL XL GU ED DL XL FU DL DL GD GD CL RD RD PR FU CL CL
GU QR ED CL GU DR DR FD FD XR AR OR QR QR FD CR EU EU BR EU EU CL FU XL FU DL
DL DL FD XR GD XR XR QL RD XR

('Depth: ', 60)

('states visited: ', 1204)
```

VehiclesInPath:

```
Solution:  OL AL RU CR FD XL XL GU ED XL GU DL FU DL DL GD GD CL RD RD PR FU CL CL
GU QR ED GU DR CL DR FD FD XR QR QR FD CR EU BR EU EU CL FU QL RD AR OR EU XL
FU DL DL DL FD GD XR XR XR XR

('Depth: ', 60)

('states visited: ', 1240)
```

BFS:

Solution: ED FD XL XL XL FU OL AL RU CR GU GU DL DL DL GD GD CL RD AR OR RD PR FU CL CL GU GU DR DR QR ED CL FD BR FD XR QR QR FD CR EU EU EU CL EU XL FU FU DL DL DL FD XR GD XR XR QL RD XR

('Depth: ', 60)

('states visited: ', 1276)

X12H

A14V

B24H

C43V

O30V

P53V

Euclidean:

Solution: CU BR PU PU BR OD OD OD XR AU AU AU AU XL OU OU BL BL BL OD OD XR CD XR PD PD XR

('Depth: ', 27)

('states visited: ', 846)

VehiclesInPath:

Solution: CU BR PU PU BR OD OD OD XR AU AU AU AU XL OU OU BL BL BL OD OD CD PD PD XR XR XR

('Depth: ', 27)

('states visited: ', 811)

BFS:

Solution: AU CU BR OD PU PU BR OD OD XR AU AU AU XL OU OU BL BL BL CD OD OD XR XR PD PD XR

('Depth: ', 27)

('states visited: ', 990)

X02H

A35V

B24H

C14V

O30V

P52V

Euclidean:

Solution: XR BR PU BR OD OD CU AL AL OD XR CU CU CU XL OU OU BL BL BL OD OD XR XR PD PD XR

('Depth: ', 27)

('states visited: ', 824)

VehiclesInPath:

Solution: AL CU PU AL BR BR OD OD OD XR XR CU CU CU XL OU OU BL BL BL OD OD PD PD XR XR XR

('Depth: ', 27)

('states visited: ', 879)

BFS:

Solution: XR AL BR CU AL OD PU BR OD OD XR CU CU CU XL OU OU BL BL BL OD OD XR XR PD PD XR

('Depth: ', 27)

('states visited: ', 979)

X22H

A20V

B02V

C12V

D41V

E44V

F54V

G40H

H43H

O04H

Euclidean:

Solution: HL HL OR CU BD HL CU XL XL AD BD HL GL GL DU EU EU OR FU FU OR AD AD XR XR AD HR HR CD CD CD CD HL HL AU XL XL AU AU GL GL AU XR XR HR HR CU OL CU OL ED XR FD XR

('Depth: ', 54)

('states visited: ', 5194)

VehiclesInPath:

Solution: CU CU XL OR BD XL GL AD GL DU BD HL HL HL EU FU HL AD EU OR FU OR AD XR XR CD AD HR HR CD CD CD HL XL HL XL GL AU GL AU AU AU OL FD HR HR CU XR XR CU OL ED XR XR

('Depth: ', 54)

('states visited: ', 5157)

BFS:

Solution: CU CU XL GL HL FU FU HL EU HL OR BD XL AD BD GL DU EU HL AD OR OR AD XR XR AD CD GL GL HR HR CD CD XL XL CD HL HL AU AU AU AU XR XR HR HR CU CU OL FD OL ED XR XR

('Depth: ', 54)

('states visited: ', 5274)

X02H

A51V

B53V

C14V

D24H

E25H

O30V

P42V

Euclidean:

Solution: XR AU BU PU DR DR OD OD CU EL OD XR CU CU CU XL OU OU DL DL DL OD OD XR PD PD XR BD XR

('Depth: ', 29)

('states visited: ', 2184)

VehiclesInPath:

Solution: AU OD PU CU DR BU DR EL OD OD XR XR CU CU CU XL OU OU DL DL DL OD OD PD PD BD XR XR XR

('Depth: ', 29)

('states visited: ', 1919)

BFS:

Solution: XR AU BU CU EL OD PU DR DR OD OD XR CU CU CU XL OU OU DL BD DL DL OD OD XR PD PD XR XR

('Depth: ', 29)

('states visited: ', 3109)

X02H

B23V

C35H

O30H

P42V

Euclidean:

Solution: XR XR CL PD XR XR

('Depth: ', 6)

('states visited: ', 58)

VehiclesInPath:

Solution: CL PD XR XR XR XR

('Depth: ', 6)

('states visited: ', 72)

BFS:

Solution: XR XR CL PD XR XR

('Depth: ', 6)

('states visited: ', 156)

X32H

B03H

C14V

D44H

O51V

P33V

Euclidean:

Solution: XL XL PU BR PU PU BR BR OU BR PD PD PD XR CU CU CU CU XL PU PU DL PU DL BL BL BL DL PD PD PD XR XR OD OD OD XR

('Depth: ', 37)

('states visited: ', 577)

VehiclesInPath:

Solution: XL XL PU PU BR PU BR OU BR BR PD PD PD XR CU CU CU CU XL PU PU PU BL BL BL PD DL DL DL PD PD OD OD OD XR XR XR

('Depth: ', 37)

('states visited: ', 587)

BFS:

Solution: XL XL BR OU PU PU PU BR BR BR CU PD PD PD XR CU CU CU XL PU PU DL DL DL PU BL BL BL OD OD OD PD PD PD XR XR XR

('Depth: ', 37)

('states visited: ', 627)

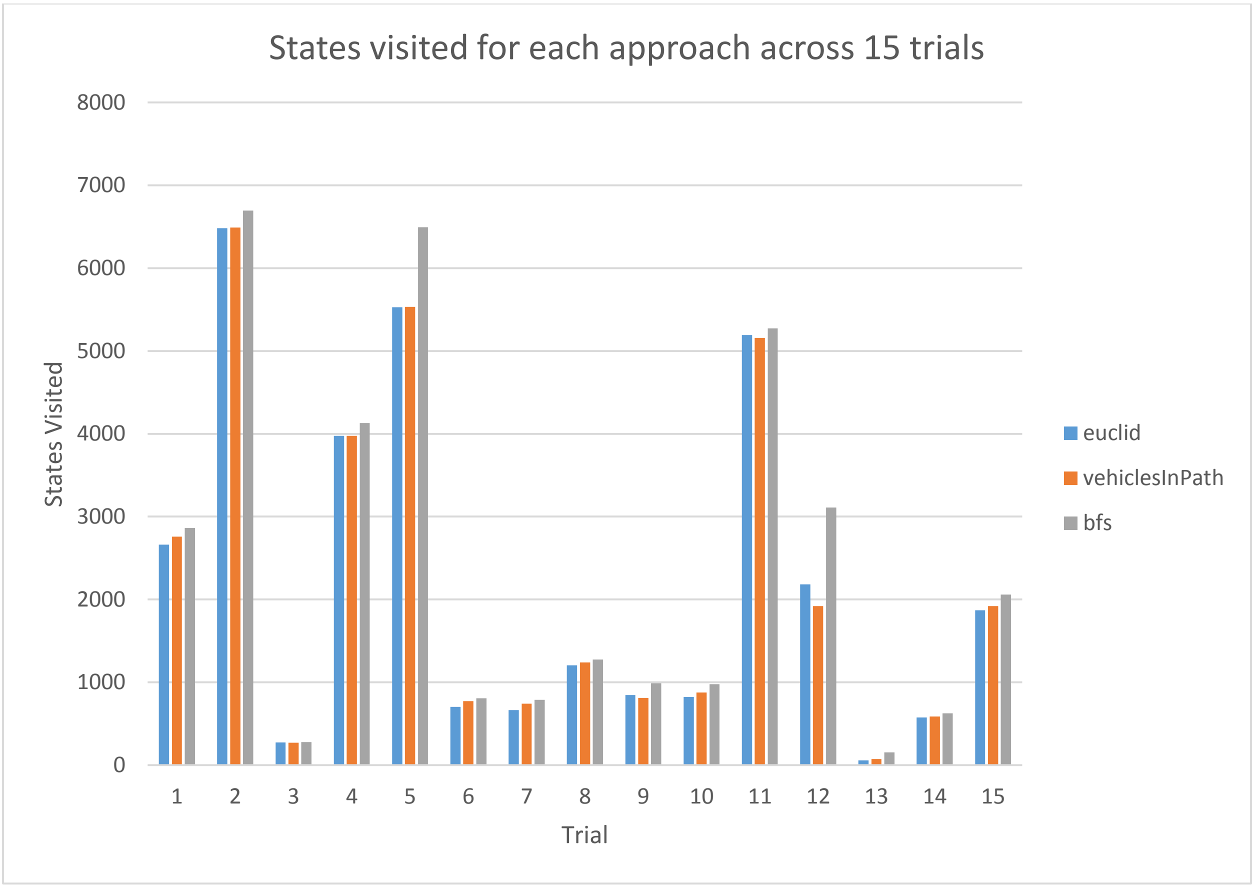
X02H

A20V

B22V

```
C14V

D34H

E25H

O31V

P53V


Euclidean:

Solution:  BD XR PU PU DR OD CU EL OD XR CU CU CU XL XL BU OU OU DL DL DL DL BD
XR OD OD XR XR PD PD XR

('Depth: ', 31)

('states visited: ', 1869)


VehiclesInPath:

Solution:  CU PU PU DR OD EL OD BD XR XR CU CU CU XL OU OU XL DL BU DL DL OD OD
PD PD DL BD XR XR XR XR

('Depth: ', 31)

('states visited: ', 1922)


BFS:

Solution:  BD XR CU EL PU PU DR OD OD XR CU CU CU XL XL BU OU OU DL DL DL DL BD
XR OD OD XR XR PD PD XR

('Depth: ', 31)

('states visited: ', 2058)
```

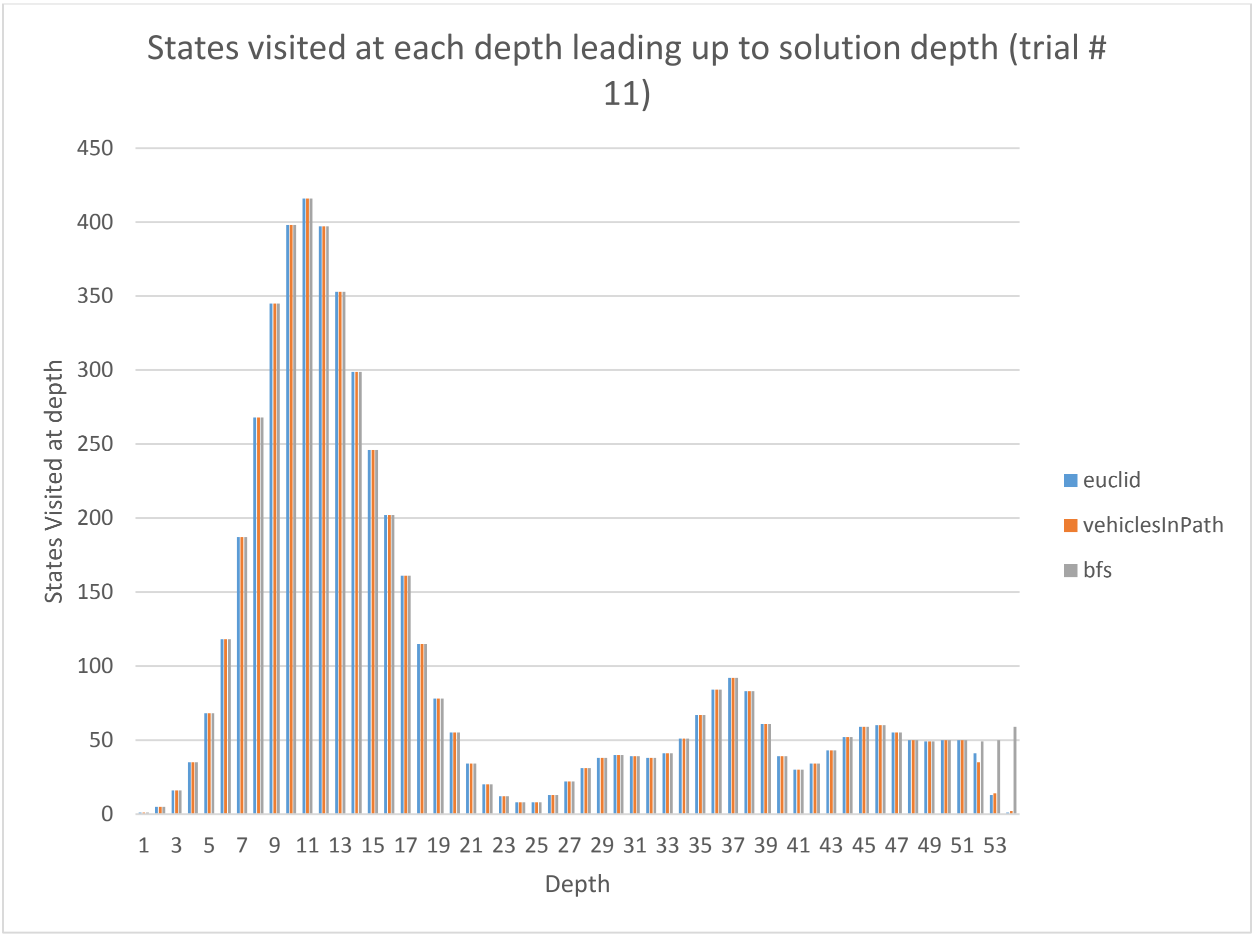
Search space explored across 15 trials



Based on the graph above one can see that while the Euclid and vehiclesInPath heuristics tend to be close, with one having more/less states visited on a trial by trial basis, the bfs search is consistently higher in its number of states visited, sometimes by a large margin such as with trials 5 and 12.

This would make sense seeing as how our two heuristics try to find a solution while using a specific strategy (although it is still a very shallow and limited strategy). Due to its limited nature it also makes sense that we see some trials (generally the ones with small solutions available) where bfs visits only slightly more states than the two heuristic approaches.

One such trial where the total visited states by bfs is only slightly higher is trial # 11, on the next page we can see a breakdown of the number of states visited at each depth leading up to the solution depth of 54.



Looking at this breakdown we can see that for the most part both heuristic searches visit the same amount of states at each level of depth as bfs. However, we can see that as the solution state draws closer the heuristic searches are able to begin using their strategy to cut down on the number of states visited starting at depth 52. This is due to the fact that new states at a depth level of 52 are showing more promise than the other states due to the heuristic rule in place. What we can draw from this is that our simplistic heuristic cannot take effect until much of the work of a regular bfs has been done already, resulting in only a marginal decrease in states visited overall.

Part B:

My program outputs the difficulty of a puzzle after running any of the 3 searches alongside the solution itself. The criteria for selecting whether or not a puzzle is ‘difficult’ is explained below:

Firstly, when considering difficulty, as with computational complexity, it would make sense to base it off of the minimum number of required steps in order to solve the problem. This would be the best measure of difficulty as the number of moves made to solve a given iteration of Rush Hour would be the only quantitative measure one could take from a player and compare it to the known minimum step solution.

With that in mind, one should then find out the absolute maximum difficulty (or minimum number of moves) that can occur in the Rush Hour game. According to this paper:

http://www.ulb.ac.be/di/algo/secollet/papers/crs06.pdf

A proof is provided that demonstrates the hardest known Rush Hour configuration would require a minimum of 92 steps to complete.

With this in mind I divided solutions in the 0-25, 25-50 and 50+ ranges in terms of difficulty. If it takes 25 or fewer moves to solve a puzzle it receives a difficulty of “easy”, 25-50 is “medium” and 50+ is “hard”. These thresholds were chosen with the absolute difficulty of 92 in mind.

Example of an easy puzzle:

X02H

B23V

C35H

O30H

P42V

Solution: XR XR CL PD XR XR

This puzzle was evaluated as easy by my program due to the fact that the solution is achievable in less than 25 moves. A human player would also likely find this puzzle very easy as there are only 2 other vehicles that need to be moved besides the red car.

Example of a medium puzzle:

X02H

A35V

B24H

C14V

O30V

P52V

Solution: XR BR PU BR OD OD CU AL AL OD XR CU CU CU XL OU OU BL BL BL OD OD XR XR PD PD XR

This puzzle was evaluated as having medium difficulty as the number of solutions is between 25 and 50. A human player might initially assume this puzzle is easy due to the fact that it only contains 6 total vehicles. However, it does require a minimum of 27 moves in order to solve it meaning that a human player would likely have some trouble with it.

Example of a hard puzzle:

X32H

A40H

B01H

C33H

D44H

E02V

F22V

G34V

O10H

P21H

Q05H

R51V

Solution: OL AL RU CR GU FD XL XL GU ED DL XL FU DL DL GD GD CL RD RD PR FU CL CL GU QR ED CL GU DR DR FD FD XR AR OR QR QR FD CR EU EU BR EU EU CL FU XL FU DL DL DL FD XR GD XR XR QL RD XR

This puzzle was evaluated as hard as it requires 60 moves minimum in order to solve. A human player would likely have a great deal of difficulty with a puzzle like this due to the sheer amount of moves required even for a minimum path solution.

The difficulty of a puzzle and the steps required to reach a solution:

There are other ways to gauge the difficulty of a puzzle besides the minimum number of steps required to reach it. However, many of these ways are not as easily quantifiable as the minimum step solution and thus are difficult to gauge a puzzle’s difficulty with. For example, in the start state of a game, if there is simply one obstructing vehicle between the red car and the finish state it may be very easy for a human player to shift around only vehicles they know are stopping that one vehicle from moving. Through the use of visual cues they could likely find the moves that will release that one obstructing car while visiting less total board states than a bfs. If there is a strategy apparent to the player based on the unique setup of that board’s start state they can exploit it to render a solution seemingly easier in comparison to other game states they can see no pattern or focal point in.

A puzzle can also be much more difficult than the minimum solution path implies. In the event that there is a puzzle that utilizes a large number of vehicles but has a short number of steps to a solution a human player may get confused and distracted moving vehicles that may not even be a part of the solution steps.

Conclusions:

What conclusions can be drawn from this experiment?

Firstly, with intelligent heuristics it is possible to find minimum path solutions with less search space visited than with a brute force approach. Although, if a heuristic is very simplistic this improvement could be very minor unless the minimum solution is a very long one.

Moreover, when judging the difficulty of a Rush Hour puzzle (or any puzzle) it is important to make note of the number of moves to a minimum path solution as it is a general guideline for difficulty however due to the way the human brain recognizes patterns and strategies in the layout of vehicles it may not always be a good indicator of what is difficult for a human player.