A2 Lab Report

Johnny Le
Trung Le
CS 4300 - 001
09/07/2016

1.Introduction - Trung Le

The goal of the assignment is to implement the A* search algorithm on the Wumpus world problem using the Manhattan distance as a heuristic. The A* search algorithm is categorized as a best-first approach meaning the next state explored is based on its' estimated cost. The experiment will implement the priority queue in two different ways. The first option is to insert a node of equal cost before a node that has greater or equal cost. The second option is to insert a node of equal cost after a node that has less than or equal cost. Using analysis, the data will be used to determine if option 1 is 10% better than option 2 at 95% confidence.

1. What are the odds of success for the A* algorithm?
2. What is the mean number of tree nodes generated by options one and two?
3. Is the solution provided by the A* algorithm the most efficient path to the goal?

2. Method - Johnny Le

Six functions were developed for the analysis. **CS4300_A2driver.m** is used to run the simulation which begins with CS4300_generate_board.m. This first function creates a board with a 20% chance of having pits as well as returns the board and the location of the gold.

Once the board is obtained, **it is first checked to see if it is solvable and that data is stored**. Then it is converted using CS4300_conversion.m to set the initial starting location to the bottom left corner. This modified board is then passed into the main bulk of our code which is in CS4300_Wumpus_A_star1. Within this function, the initial state, the goal state, the heuristic function (CS4300_A_star_Man.m), and the option is taken in.

Processing begins with the initial parameters being put in place. Five main matrices are used to simulate data structures. The first is a container map that connects certain integers to a state. Second is another container map that connects those same integers to its corresponding cost which is calculated using the heuristic function and the goal state. The Manhattan distance is calculated by finding the distance from the initial state to the goal state. Continuing the list of data structures, there are two single row matrices representing two arrays which is the frontier and the explored.

Exploration is completed by a series of checks. CS4300_explore.m cycles through all options and selects the available moves. Once done, it will return the three states that can be taken if all three steps (for each action) were available.

For both option one and two, the bulk is the same. It begins with removal of the initial state from the frontier and placing it into the explored once all of its children are found. Then the children are compared with the previous states to ensure no duplicates exist within the three found and this is sent into the frontier stage which differs between the two options.

In option 1, the frontier stage will take all candidate children and compare them to the first non-zero index stored in frontier. This comparison progresses down the matrix until the child to be inserted is arrives at an integer associated with cost greater than the cost associated with the child. The child is then inserted and the remainder within the list is shifted down one space in the matrix.

For option 2, the frontier stage will take all candidate children but will instead compare them to the last non-zero index stored in the frontier and approach the list from the opposite direction. By doing so, the same comparisons will be made ensuring the priority queue design is followed but the insertion will always occur to the right of the equal cost associated indices.

In both of the above methods, it will complete with **children added to the map of known states and have it entered into appropriate list once the processing it complete**. Continuing, this entire process will repeat with the first non-zero index from the frontier becoming the next integer associated state to be explored.

At the beginning of each removal from the frontier, checks are done to determine if a state has been explored already and if the agent is killed or the gold is found.

The solution is collected by taking the goal state and using the CS4300_Traceback.m function which simply follows the track from children to its parents and stores the solution into a matrix.

2000 trials with each using a newly randomized board is completed and the number of successes as well as the number of steps taken is recorded.

This data was plotted on charts to show the distribution of the steps and the rate of success.

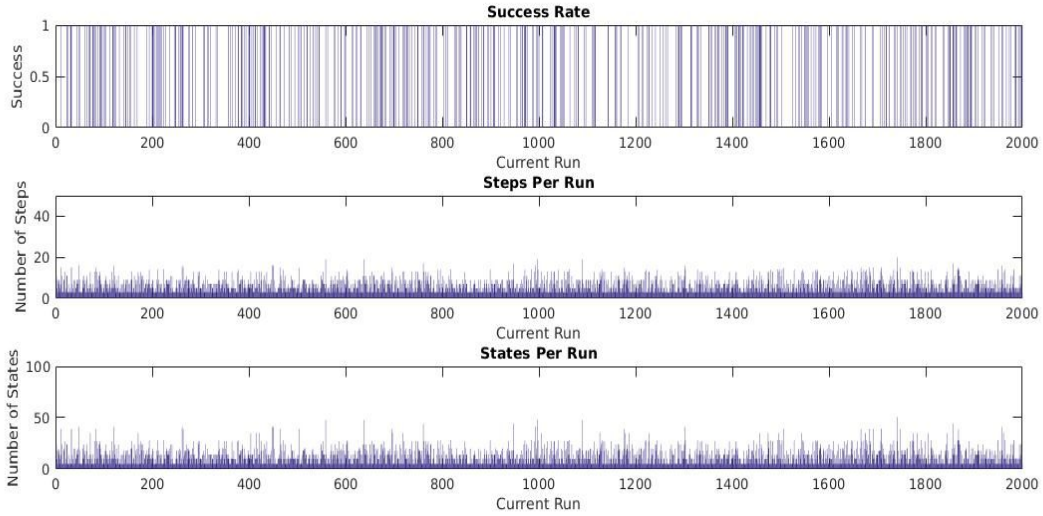
3. Verification of Program - Trung Le

In testing three boards (if this is the printed version, see attached for the comparison with the computed results in section 4 below), the written trials were mainly with the computed versions; however, trials done by hand had resulted in far more complications such as human error during the mapping. Only through closer examination, after observing the results of the computation, were some solutions found to be more efficient than ours.

On example one, both the hand written and the computed had failed to find the gold, but the computed one took more steps to reach death. Example two, the computed version found gold but had taken one additional step than what was hand written drawing it closer to optimal pacing. The best result, however, came in example three where the gold and pit were next to the initial state. Both the hand-written and the computed solutions resulted in six total states and 2 total steps to find the solution using the same path.

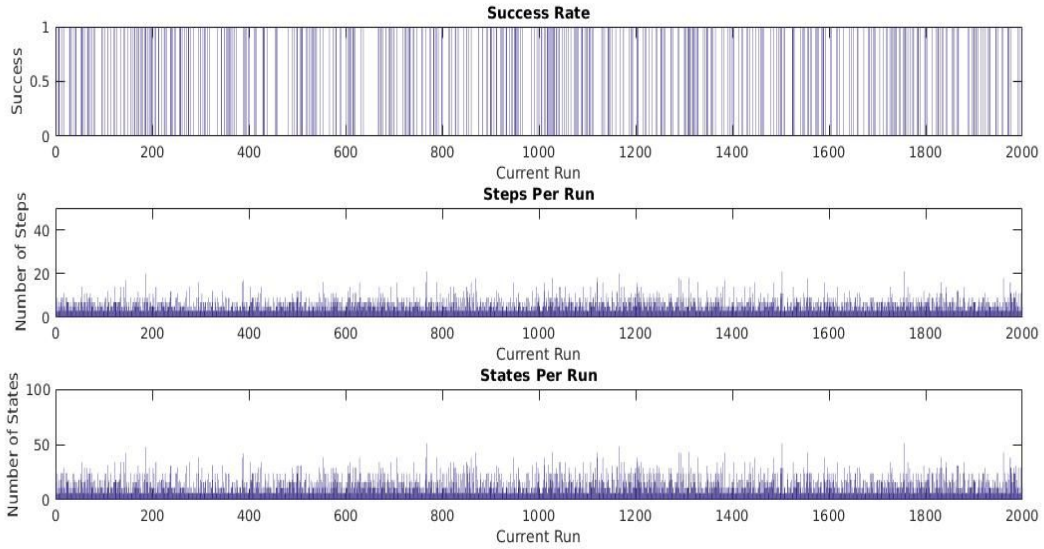
4. Data and Analysis - Johnny Le

Data for an instance of Option 1



Mean Result	0.2250
Mean Number of Steps	6.2555
Mean Number of States	12.6600
Variance of Results	0.1745
Variance of Number of Steps	11.1718
Variance of Number of States	64.6167
Result Confidence Interval	[0.2067, 0.2433]
Number of Steps Confidence Interval	[6.1090, 6.4020]
Number of States Confidence Interval	[12.3077, 13.0123]

Data for an instance of Option 2



Mean Result	0.2315
Mean Number of Steps	6.0490
Mean Number of States	14.3460
Variance of Results	0.1780
Variance of Number of Steps	9.3093
Variance of Number of States	70.2044
Result Confidence Interval	[0.2130, 0.2500]
Number of Steps Confidence Interval	[5.9153, 6.1827]
Number of States Confidence Interval	[13.9788, 14.7132]

For the given board

0	0	0	0
0	0	0	0
0	2	0	0
->	0	0	0

Using option 1:
The number of states generated is 10
The number of steps to get to the end state is 5
Using option 2:
The number of states generated is 11
The number of steps to get to the end state is 5

Example One
For the given board

0	3	0	2
1	0	0	0
0	0	1	1
->	1	0	0

The A* algorithm using option 1 resulted in the following solution path and node list

Solutions:

X	Y	Dir	Action	
1	1	0	0	
1	1	1	3	
1	2	1	1	
1	2	0	2	
2	2	0	1	
2	2	3	2	
2	1	3	1	It did not reach the gold and died at (2,1)

Node List

Index	Parent	State	Action	Cost	Children
0	0	[1,1,0]	0	0	[2,3,4]
1	1	[2,1,0]	1	6	[0,0,0]
1	1	[1,1,3]	2	7	[5,0,0]
1	1	[1,1,1]	3	7	[6,0,0]
3	2	[1,1,2]	2	8	[0,0,0]
4	2	[1,2,1]	1	7	[7,8,9]
6	3	[1,3,1]	1	7	[0,0,0]
6	3	[1,2,0]	2	8	[10,11,0]
6	3	[1,2,2]	3	8	[0,0,0]
8	4	[2,2,0]	1	8	[12,13,14]
8	4	[1,2,3]	2	9	[0,0,0]
10	5	[3,2,0]	1	8	[0,0,0]
10	5	[2,2,3]	2	9	[15,16,0]
10	5	[2,2,1]	3	9	[0,0,0]
13	6	[2,1,3]	1	11	[0,0,0]
13	6	[2,2,2]	2	10	[0,0,0]

Example Two

For the given board

3	0	0	0
0	0	0	1
0	2	0	0
->	0	0	1

Solutions:

X	Y	Dir	Action
1	1	0	0
1	1	1	3
1	2	1	1
1	2	0	2
2	2	0	1

Node List

Index	Parent	State	Action	Cost Children
[2,3,4]	0	[1,1,0]	0	0
[0,0,0]	2	[2,1,0]	1	1
[5,0,0]	3	[1,1,3]	1	1
[6,0,0]	3	[1,1,1]	1	1
[0,0,0]	4	[1,1,2]	2	3
[7,8,9]	3	[1,2,1]	2	4
[0,0,0]	3	[1,3,1]	3	6
[10,11,0]	4	[1,2,0]	3	6
[0,0,0]	4	[1,2,2]	3	6
[0,0,0]	4	[2,2,0]	4	8
[0,0,0]	5	[1,2,3]	4	8

Example Three

For the given board

0	0	0	0
0	0	0	0
2	0	0	0
->	1	0	0

Solutions:

X	Y	Dir	Action
1	1	0	0
1	1	1	3
1	2	1	1

Node List

Index	Parent	State	Action	Cost	Children
0	0	[1,1,0]	0	0	[2,3,4]
1	1	[2,1,0]	1	1	[0,0,0]
1	1	[1,1,3]	2	2	[5,0,0]
1	1	[1,1,1]	3	2	[6,0,0]
3	2	[1,1,2]	2	3	[0,0,0]
4	2	[1,2,1]	1	2	[0,0,0]

5. Interpretation - Trung Le

A majority of the results were as expected with option 1 which was the method in which insertion was done before greater than or equal values. This was seen in the number of states as the number of overall states was lower in option suggesting a smaller space complexity. There was seen to be an average of 13 states between the two options. Some odd discrepancies in the data was seen in the average number of steps for each option as option 1 had a higher average than option 2 which is the opposite of our expectation. This may be due to the greater chance of falling into a pit or a wumpus since it is more optimized.

The odds for success in this algorithm were about 23% (option 1: 22.5%, option 2: 23.15%) which is surprisingly low. Although the algorithm solved the board quickly and often, a majority of the time it was still inefficient. Further optimization would require reacting to the environment such as with smelling and using other senses.

Despite its inefficiency though, when the A* algorithm found a solution, it was always the shortest path. Dropping time complexity may be another improvement for the experiment.

Overall, we could not find enough conclusive data to determine at a 95% confidence that it was 10% better to do option 1 although there were slight improvements.

6. Critique - Johnny Le

Through the implementation of our priority queue and exploration function from scratch, it allowed us to be aware of a number of issues throughout experimentation. The biggest issues we observed were the inefficiency of our algorithms and the potential of little variation in the randomized boards. Many boards that had been produced resulted in an almost immediate death and it took numerous trials to overcome this. Creating a matlab breadth first search was extremely difficult with a number of nuances without the provided data structures available in object oriented programming.

Our decision to utilize breadth first search benefited performance very well because it allowed us to find an end to the search much quicker. This was known due to the core difference between breadth and depth since breadth will determine the shortest path as it goes across each level.

Improvements that could be made is to develop the program in another programming language and doing the analysis in MatLab or doing a more thorough test of boards by setting certain probabilities instead of a single one.

7. Log - Trung Le

Deciphering the Assignment: Trung Le 3 hours
Coding the Assignment: Trung Le 12 hours, Johnny Le 14 hours
Analysis: Trung Le 4 hours, Johnny Le 4 hours