Rajul Ramchandani & Conan Zhang

Professor Thomas Henderson

CS 4300 - 001

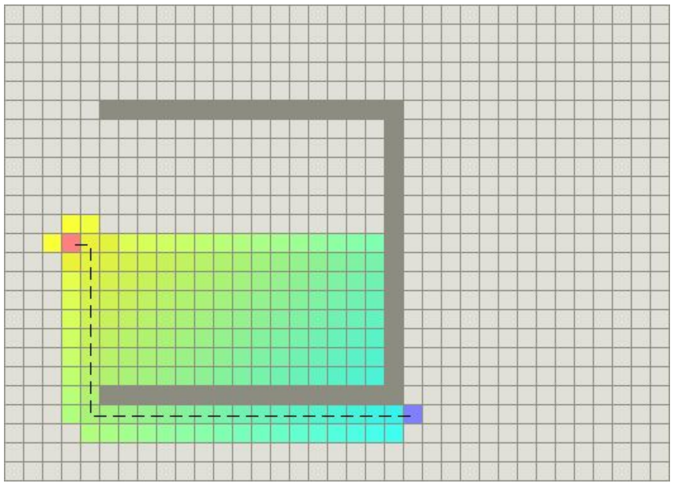
8 September 2016

Assignment A2: Problem Solving: Search

1. INTRODUCTION: RAJUL

This experiment, studies statistics of using variations of A* search on the Wumpus World problem provided in paper "Assignment A1". Current;y, the measure of complexity is the total number of nodes generated during a search.

An A* search consists of two main parts for this experiment: A frontier and Explored. Each of these signify whether or not a node (which is a combination of coordinates, direction and action) is currently being explored. The frontier is the edge of the exploring area(seen above) while the explored list



is the list of nodes already visited. The A* also uses a hueristic as an estimate of the distance to the goal. For this experiment, the hueristic was chosen to be the Manhattan distance to the goal which is good estimate. This is garunteed to be less than or equal to the actual distance to be travelled. This provides interesting problems and questions to be answered: What is the mean number of search tree nodes produced by A* options 1 and 2 on these problems). Test whether the Hypothesis that option 1 A* search is 10% better than option 2 at the 95 % confidence level. This will be run for 2000 trials. Are the (new) children nodes added to the tree immediately when a node is expanded from the frontier? When is a node checked to see if it is a solution? What are the number of nodes in the tree for every possible position of the gold on the board with no pits and wumpus? Is the experiment data similar to hand solved, real life data?

2. METHOD: CONAN

The algorithm we used for this experiment is A*, a best-first search that uses a heuristic and priority queue. The heuristic we use is the Manhattan Distance, the exact horizontal and vertical distance two nodes are from each other on a graph. This distance is a value that we can use to estimate the cost of transitioning to a cell on the board. We also use these costs as a way to prioritize nodes in a priority queue. That's what allows A* to be classified as a best-first search.

The priority queue was developed as a MATLAB class that can insert, pop, and peek. It also has two options of inserting for costs of equal value, before the same value or after. The priority queue acts as the frontier of our experiment, with children being added immediately to the node tree after a node is expanded. Nodes are checked if they are the solution every time we visit the node with top priority in the priority queue. Boards with no solution return as an empty array, with a node tree consisting of 64 nodes.

To be able to answer our question, we take the complexity of our A* algorithm to mean the number of nodes generated in the search. We will gather this data by generating 2000 random boards with a Wumpus and a 20% chance of a pit. We can then compare the complexity of A* with two different types of insertion for equal values, option 1 is before and option 2 after. Our hypothesis is that inserting before a cost of equal value is 10% better than inserting after with a 95% confidence interval.

3. VERIFICATION OF PROGRAM: RAJUL

The table provided below provide information for how many steps the agent would take. The coordinates of each cell below, starts from (1,1) from the bottom left and where each of the coordinates signifies the number of step needed to reach the gold in that cell. This used option 1 for inserting in priority queue.

Number of Nodes for Gold Positions			
4	6	7	8
3	5	6	7
2	4	5	6
1	2	3	4

Gold Board Position Solutions

For verification, let's consider the known solution for cell (2,2). For this, the agent must either (forward, turn left, forward) or (left, forward, right, forward). The first option being the shorter one, it is chosen. This means, the partial tree path would look like

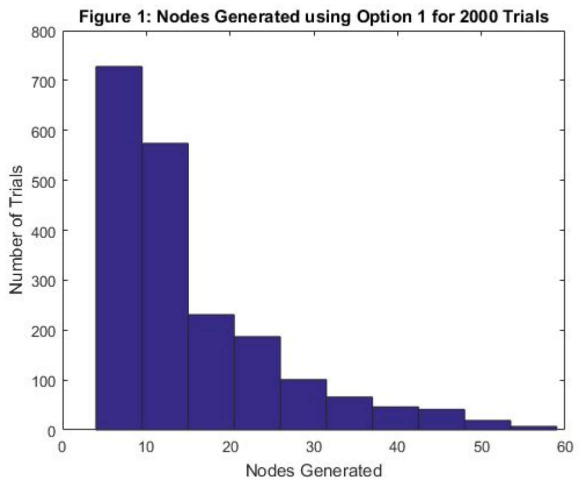
```
[1, 1,0]
        [1, 2 ,0]
                [1, 2, 1]
                        [2, 2, 1]
        [1, 1, 1]
                [2, 1, 1]
        …       …
```

Clearly, from the above text, we see there are 4 nodes from [1, 1,0] to [2, 2, 1] which is also the answer found in table 1. This board with gold at (2,2) could also be used for hand verification. We take 3 cases and show solutions. We observe that the hand solution provided will be the same as the solution by the method solution. I.e

G : (2,2)	G:(3,4)	G:(4,1)
1 1 0 0	1 1 0 0	1 1 1 3
1 2 0 1	1 2 0 1	2 1 1 1
1 2 1 3	1 2 1 3	3 1 1 1
2 2 1 1	2 2 1 1	4 1 1 1
	3 2 1 1	
	3 2 0 2	
	3 3 0 1	
	3 4 0 1	

The minimum and maximum tree sizes shown by the experiment and by hand are around 5 to 60 nodes for both options 1 and 2. This can also be seen in the graphs for the next section.

4. DATA AND ANALYSIS: CONAN



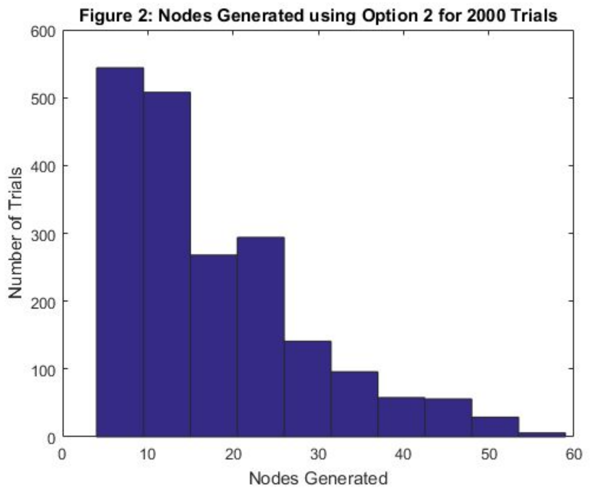
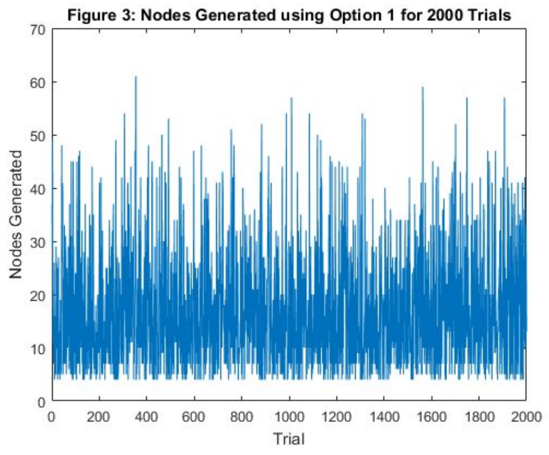


Figure 1: Histogram of Nodes Generated for Option 1

Figure 2: Histogram of Nodes Generated for Option 2

[Figure 1] shows a steep curve for the histogram of generated nodes for 2000 trial using option 1. [Figure 2] shows the same curve, but less steep since it is using option 2.



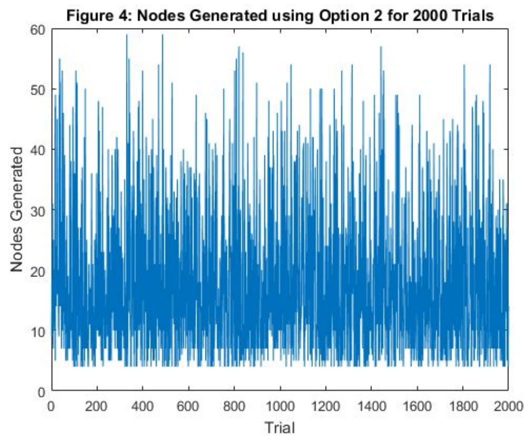


Figure 3: Plot of Nodes Generated for Option 1

Figure 3: Plot of Nodes Generated for Option 2

[Figure 3] shows the plot of all 2000 individual trials for option 1, while [Figure 4] does it for option 2. We can notice that option 2 has more spikes that reach the higher values of nodes generated.

Option 1: Data Generated for 2000 Trials			
Mean	Variance	95% Lower Bound	95% Upper Bound
15.3240	112.3072	14.8595	15.7885

Figure 5: Data Generated for 2000 Trials using Option 1

Option 2: Data Generated for 2000 Trials			
Mean	Variance	95% Lower Bound	95% Upper Bound
17.4530	130.3990	16.9525	17.9535

Figure 6: Data Generated for 2000 Trials using Option 2

[Figure 5] shows the actual mean, variance, and 95% confidence intervals for option 1 and [Figure 6] does the same for option 2.

5. INTERPRETATION: RAJUL

The above figures ([Figure 1] and [Figure 2]) represent the two options where [Figure 1] inserts before greater than or equal cost states, and [Figure 2] inserts after less than or equal cost states.

From the two graphs, we can clearly see that the graph for option 2 is shorter as well as much broader. This signifies that there are less number of trials and more number of nodes when compared to the one for option 1 . Looking up the first bar on both graphs, over 700 trials have less than 10 nodes in the tree for option 1 (figure 1), but only 500+ trials have less than 10 nodes for option 2.

This clearly helps to graphically prove that option 2 is less efficient.

Secondly, if we observe the data obtained for running 200 trials on option 1 and 2, we can also numerically prove the hypothesis of option 1 being 10% faster than option 2.

To obtain this result, we can observe that while the mean for option 1 is **15.3240**

And mean for option 2 is **17.4530** the percentage of their comparison is given by:

$$17.4530 - 15.3240 = 2.129$$

$$2.129/17.4530 = \mathbf{12.1\%}$$

Thus obtaining a result of 12.1% which might be different from the hypothesis due to the large variance. This is only 2.1% off of the estimate of the hypothesis.

6. CRITIQUE: CONAN

We gained a better understanding of how A* improves pathfinding from an agent that randomly selects a path. By using techniques such as optimizing using priority queues and heuristics, we have a better knowledge on how probability can increase the capabilities of artificial intelligence. Seeing that A* can be further improved by choosing how you insert into a priority queue, perhaps you can also increase the robustness of the pathfinding with different search methods. There are also several optimizations that can be made with the data structures we used, as many of them need multiple allocations to handle more insertions.

7. LOG: RAJUL & CONAN

Rajul: I spent 8 hours on Part 1, 1 hour for problem 3, 1 hour on problem 5, 3 hours on the Lab Report.

Conan: I spent 8 hours on problem 1, 1 hour on problem 2, 1 hour on problem 5, and 3 hours on the Lab Report.