Assignment 2 – CS4300
A* Search in a Wumpus World

Braden Scothern & Kyle Heaton

1. Introduction

To further continue to explore the Wumpus World we measured the complexities of variations on an A* search. Option 1, we insert new states before states rated to be equal or greater. Option 2, we insert after states rated equal or lesser. A Manhattan distance heuristic to measure the distance between any given state and the goal state to rank different states against each other. We ran 2000 randomly generated boards, each with a 20% change of having an empty cell be a pit, to answer the following questions

- What is the mean number of search nodes produced by A* for options 1 and 2?
- Is option 1 10% better than option 2?

2. Method

A lot of the work for the A* algorithm is split out into helper functions; most of the body of the actual function is verification code and the manipulation of arrays. The first helper function that is used is CS4300_A2_Expand_States(). It takes a current state and returns a 3x3 array of integers with the first row being the forward action, the second being the right turn, and the third being the left turn. We then loop over each row and validate that the state is in bounds, not a duplicate and that it not a death state. We verify that a state isn't a duplicate with the function CS4300_State_Is_Duplicate() which takes the entire tree of nodes and makes sure that the potentially new node cannot be found in it.

When a node is pulled from the frontier it is immediately checked for a goal state. If it is a valid solution that solution is returned. Otherwise the node is expanded and all valid children are immediately put on the tree. If the state is valid but not the goal then we will add it as a new node and make it a child of the main node we are currently looking at. Once we have all of these new children we add them to the frontier according to the logic needed to satisfy the option code given to the A* function. All of this logic is looped over as long as the current node hasn't reached a goal state and as long as we have more nodes in the frontier for us to search.

When it isn't possible to find a path to the gold, our A* function will return an empty solution path and a set of nodes that cover all reachable locations on the board.

The method used here is simply to generate a large number of samples and compute the mean, variance and confidence of the result. An alternative would be to run a large number of trials where each trial would get a fixed number of samples from rand, then compute the mean and variance of each trial, and then compute the mean and variance over all those trials. This latter approach was not implemented.

3. Verification of Program

To test out that program works we will be comparing our results to hand calculations of the following boards

W			
gold	pit		

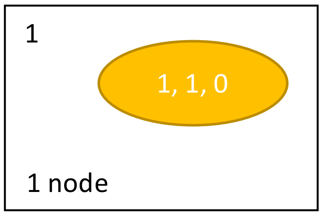
1

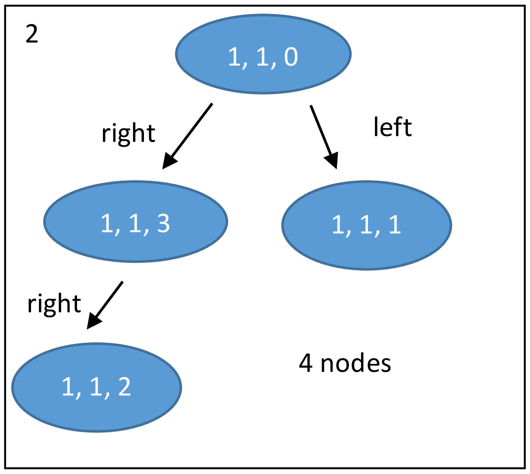
			gold
pit			
	w		

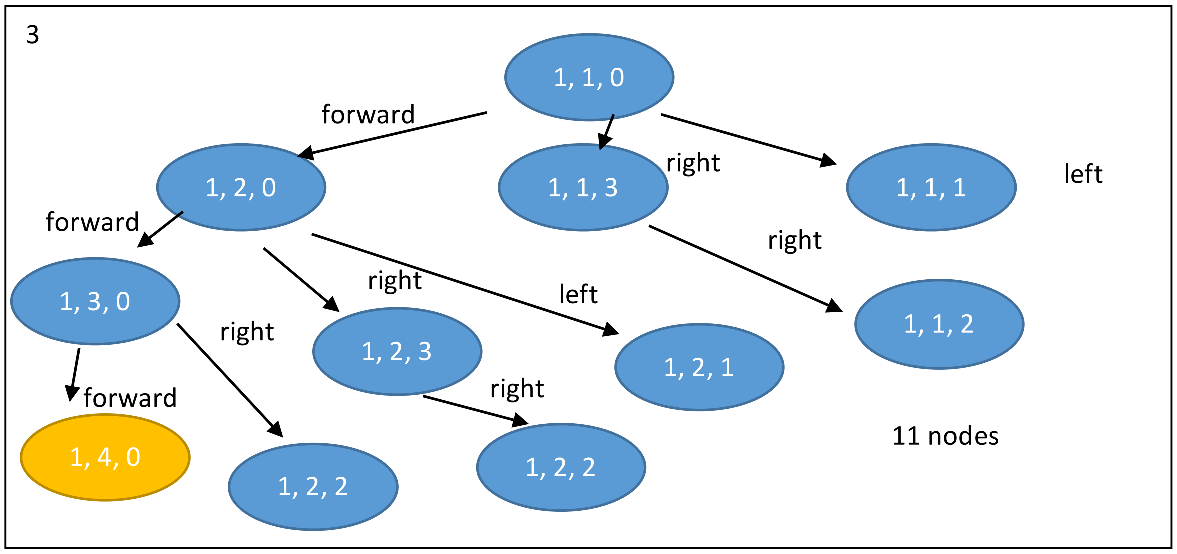
2

w	pit	pit	
			gold

3







Results from Matlab

- Board 1
  ```
  >> board1 = [2,1,0,0;3,0,0,0;0,0,0,0;0,0,0,0];
  >> [sol1, nodes1] = CS4300_Wumpus_A_star1(board1,[1,1,0],[1,1,0], '',1)
  ```
  Yields 1 node in the tree as predicted
- Board 2
  ```
  >> board2 = [0,3,0,0;1,0,0,0;0,0,0,0;0,0,0,2;];
  >> [sol2, nodes2] = CS4300_Wumpus_A_star1(board2,[1,1,0],[4,4,0], '',1);
  ```
  Yields 4 nodes as predicted
- Board 3
  ```
  >> board3 = [0,0,0,2;3,1,1,0;0,0,0,0;0,0,0,0];
  >> [sol3, nodes3] = CS4300_Wumpus_A_star1(board3,[1,1,0],[1,4,0], '',2);
  ```
  Yields 11 nodes as predicted

Minimum tree size: 1 node – if the gold is in the player spawn position.

The maximum tree size: 15 cells (no pits, gold and wumpus in the same cell)

$$15\ cells * 4\ states\ per\ cells = 60\ possible\ nodes$$

After running the simulation over the 2000 iterations we found the min and max to be 1 and 60 respectively. The theoretical min and max match up with our results perfectly.

4. Data and Analysis

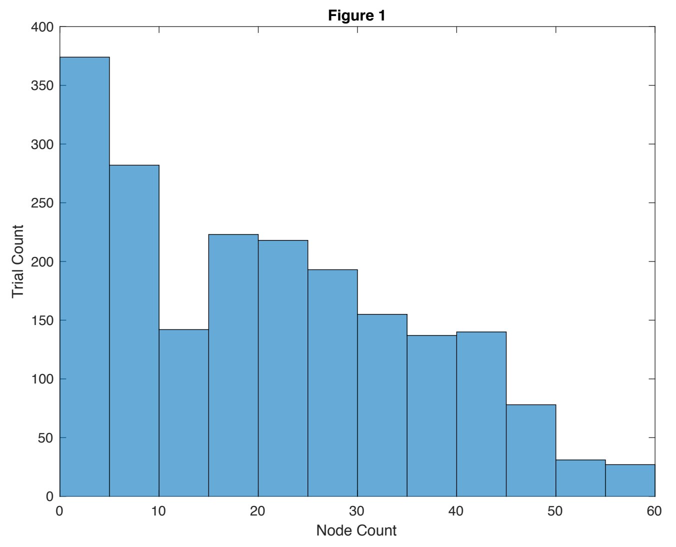
The figures below show the how frequently nodes are created. The data below also shows that our maximum solution path length for option 1 is 24 steps and for option 2 is 23 steps.

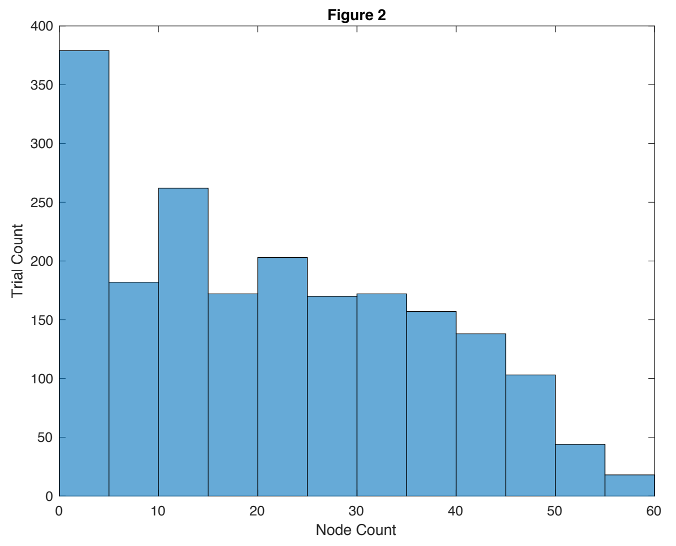
Figure 1 – This shows how frequently a node count was reached when using option 1 for A*.

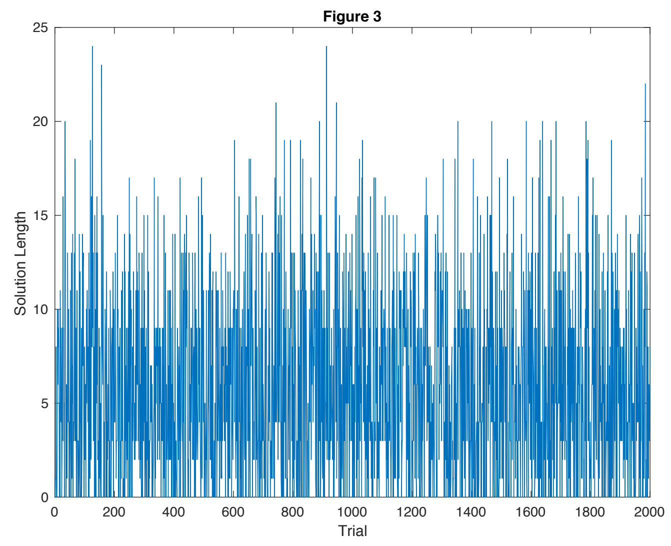
Figure 2 – This shows how frequently a node count was reached when using option 2 for A*.

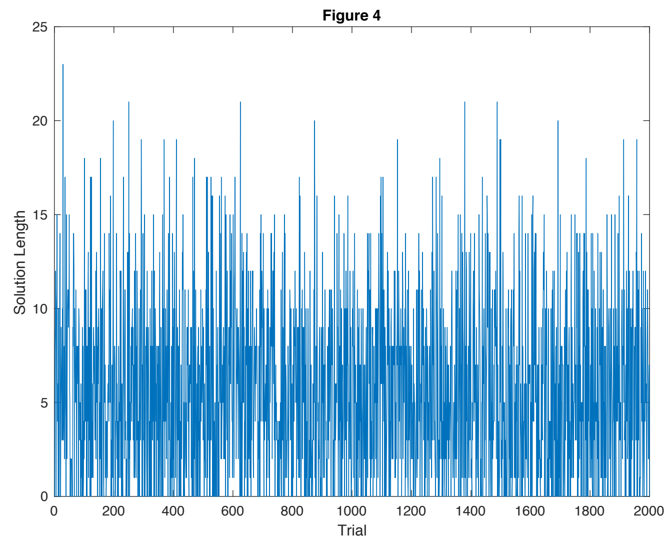
Figure 3 – This shows the number of steps required for each trial in order to reach the goal state when using option 1.

Figure 4 – This shows the number of steps required for each trial in order to reach the goal state when using option 2.









5. Interpretation

Mean option 1: 21.0335
Mean option 2: 21.2630

According to the results that we got option 1 is only 1.0793% better than option 2. Not the 10% hypothesized.

- What is the mean number of search nodes produced by A* for options 1 and 2?
- Is option 1 10% better than option 2?

6. Critique

This experiment is very effective at introducing a high level of complexity that comes with finding optimal solutions for even very simple problems. The concepts were all fairly straightforward and easy to understand as they were worked out by hand and in the process of writing code. The biggest suggestion to improve this experiment is to standardize the proper definition of the heuristic function in A* so multiple functions can be used without the potential of needing to modify the function call inside of the A* function.

7. Log

Braden Scothern

- 6 hours – Writing and debugging code
- 90 min – Writing report sections 2, 4, and 6

Kyle Heaton

- 57 min – Reviewing assignment, fleshing out requirements, outlining report
- 6 hours – Writing helper functions and writing report sections 1, 3, and 5

Appendix

MATLAB Code Files with Brief description:

- CS4300_A2_20percent_Pit_Board.m – Helper function to generate Wumpus World boards. The gold and wumpus are placed randomly, then every empty cell has a 20% change of having a pit.
- CS4300_ A2_Calculate_Tree_Size.m – Helper function to determine the size of the tree for tree node statistics
- CS4300_ A2_Expanded_States.m – Helper function to generate all of the possible reachable states from a state that has been passed in
- CS4300_ A2_Manhattan_Distance.m – Helper function to calculate the Manhattan distance for a state from the starting point
- CS4300_ A2_Wumpus_A_star1.m – An A* search algorithm to search a given wumpus board for a path to the gold
- CS4300_A2_Get_Stats.m – Function to encapsulate all of the tests to get a statistical data for our tests

- CS4300_A2_Run_Tests – Given a number of tests to run, it provides the number of nodes for option 1 and 2 generated in those tests