Assignment 2 – CS4300

A\* Search in a Wumpus World

Braden Scothern & Kyle Heaton

# 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?

# Method

**Description of algorithms**

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 ints 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.

If the state is valid 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.

**Are the new children nodes added to the tree immediately when a node is expanded from the frontier?**

Yes, as soon as a node has expanded and has its children they are added into the frontier according to the option given.

**When is a node checked if it is a solution?**

We check if a node is a solution as soon as it is pulled off the frontier. This takes place before any other calculations are run on the node.

**Produce results even with no solution to goal (include stats of search trees that fail)**

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.

**Other issues you deem important to understand the data**

We have made several assumptions throughout the process of this assignment that are important to understanding our data. We have assumed that the heuristic function given via h\_name in the A\* function always has the form of taking a 1x3 vector consisting of x, y, dir and a 1x3 vector that is the goal\_state. All of our data was computed with the Manhattan distance for this function. We have assumed that turning left or right is an action step. We have also assumed that a node’s g is equal to the amount of action steps to get to that point and that its h is always the results returned by the heuristic function created by calling str2func(h\_name).

# Verification of Program

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

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| w | pit | pit |  |
|  |  |  | gold |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | gold |
|  |  |  |  |
| pit |  |  |  |
|  | w |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| W |  |  |  |
| gold | pit |  |  |

1

2

3

1

1

left

right

2

1 node

right

4 nodes

3

forward

forward

left

right

left

right

right

right

forward

10 nodes

Results from Matlab

* Board 1 –
* Board 2 –
* Board 3 –

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

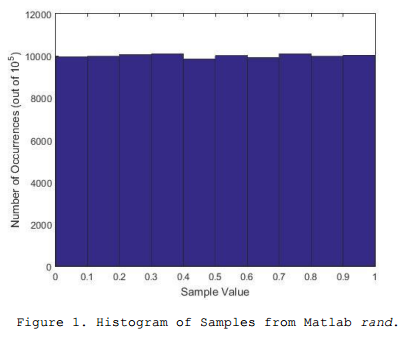
The maximum tree size: 14 cells (no pits, 1 gold cell, and 1 wumpus cell)

# Data and Analysis

Description of data and how it is used/organized for training and testing  
Results from experiments (graphs, tables, means, variance, confidence intervals)  
Describe relationships found or qualitative description of results  
Discuss implications of statistical results

Give 4x4 tables with the number of nodes in the search tree for each of these options when the gold is at (x.y)  
Plot actual size of individual trial results and a histogram

Figure 1 shows the data collected from Matlab rand.



# Interpretation

ANSWER QUESTIONS

Answer questions posed in into, use analysis to support conclusions  
Discuss future work and extensions

The results produced by rand are very close to the theoretical values of the mean and variance and the confidence interval at the 95% level is very short. Thus, the mean and variance results are close enough to the theoretical values to be acceptable in most applications.

# 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.

# Log

Braden Scothern

* 240 min – Writing and debugging code
* 60 min – Writing report

Kyle Heaton

* 57 min – Reviewing assignment, fleshing out requirements, outlining report
* 98 min – Writing helper functions and writing report

# 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