Isabelle Chalhoub - u0678302
Karla Kraiss - u0830999
September 13, 2017
CS 4300

A2 - Lab Report

1. Introduction

In this lab we created two agents: one that uses A* to find it's way out of the maze after it has randomly stumbled upon the gold, and another that uses arc consistency in conjunction with A* to both explore the unknown in search of gold and to get back home. We will measure the improvement in the following ways:

- Does arc consistency (AC) improve the A* performance in survival and success rate?
- Does AC within A* complete the task of acquiring gold and getting home faster than without AC?

2. Method

A*_Agent: This agent makes random choices on whether to move forward, left or right. If/when the agent finds gold, it uses the A* algorithm to find its way back home. The heuristic used for this problem is the Manhattan distance.
A*_AC_Agent: This agent uses arc consistency based on the labels of each cell {clear, pit, breeze} to make a safe decision on where to travel. It then uses the same A* algorithm described above to get to that cell. If no safe cell exists that is unvisited, it will make a random choice. Once the gold is found, the agent behaves the same as above and uses A* to find its way home.
AC3: This is our arc consistency algorithm that updates the labels as described above in order to remove bad choices from the search tree. Our predicate function only removes labels based on this matrix:

```
[1 0 1
 0 1 1
 1 1 0]
```

If there exists any label for the first vertex that is supported by a label from the second, we leave it. If not, it gets deleted from the first vertex set. Both this file and our AC agent use various helper methods to make these decisions and remove labels.

<u>Revise:</u> Compares the labels of two vertices from a given arc as described above and removes any inconsistencies.
<u>Driver:</u>

3. Verification of Program

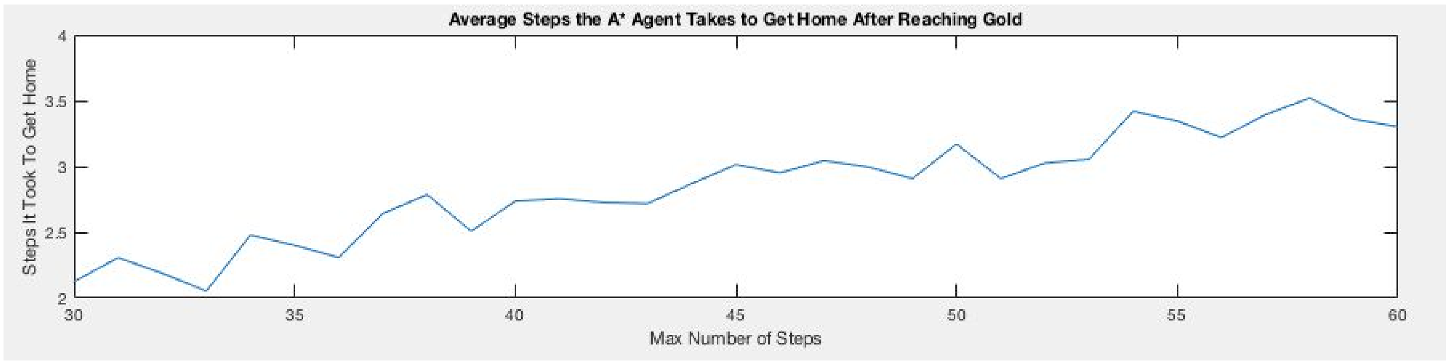
We tested our AC3/revise functions with the 4 queens problem. It behaved properly with the first two test cases but failed after that because of the problems we had with AC3 for the Wumpus agent. We also worked too much on making AC3 work for Wumpus that it isn't general enough to work for three queens. Our problems with AC3 stemmed from translation problems between the wumpus board, the matrix representation, and the D representation.

	Input G:	Input D:	Result:
1.	0 1 1 1	1 1 1 1	1 1 1 1
	1 0 1 1	1 1 1 1	1 1 1 1
	1 1 0 1	1 1 1 1	1 1 1 1
	1 1 1 0	1 1 1 1	1 1 1 1
2.	0 1 1 1	1 0 0 0	0 0 0 0
	1 0 1 1	1 1 1 1	0 0 0 0
	1 1 0 1	1 1 1 1	0 0 0 0
	1 1 1 0	1 1 1 1	0 0 0 0
3.	0 1 1 1	1 0 1 1	0 1 0 0
	1 0 1 1	1 1 1 1	1 1 1 1
	1 1 0 1	1 1 1 1	1 1 1 1
	1 1 1 0	1 1 1 1	1 1 1 1

4. Data and Analysis

For all of these the board looked like:

```
0 1 0 0
1 0 0 0
0 2 1 1
0 0 0 0
```



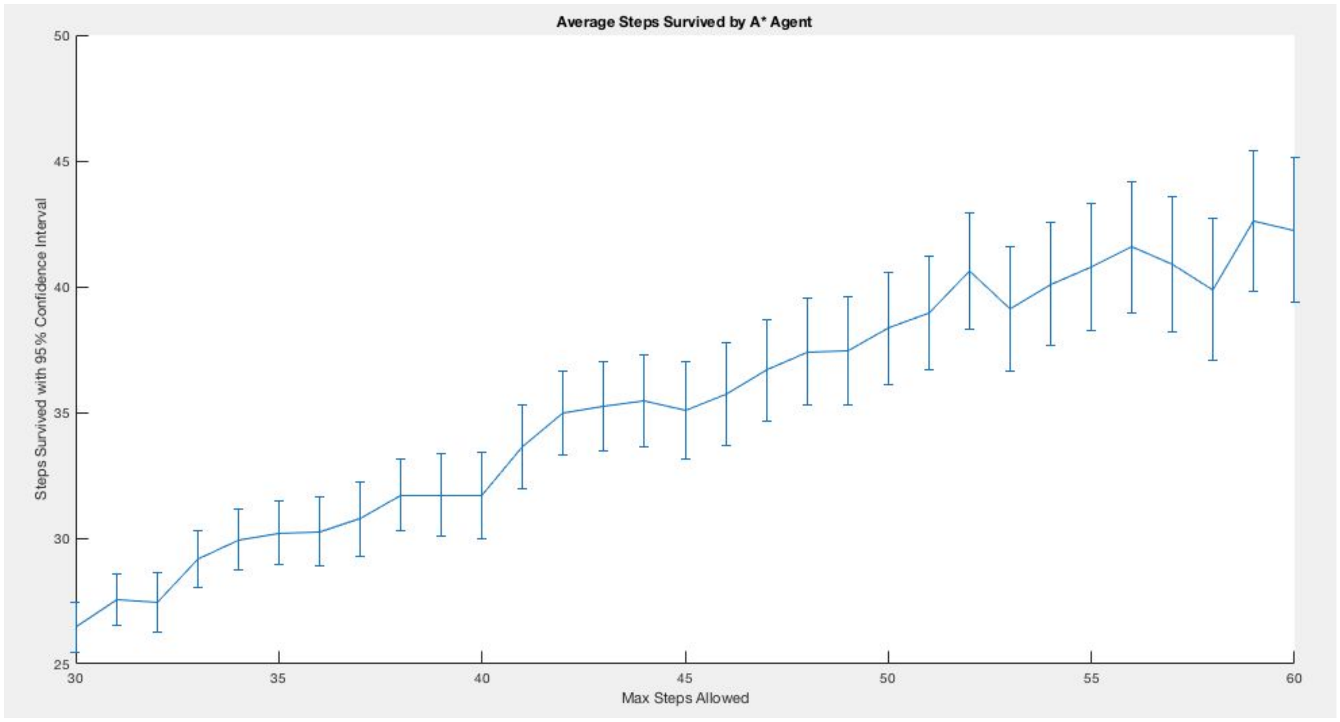
Snippet from Graph Above

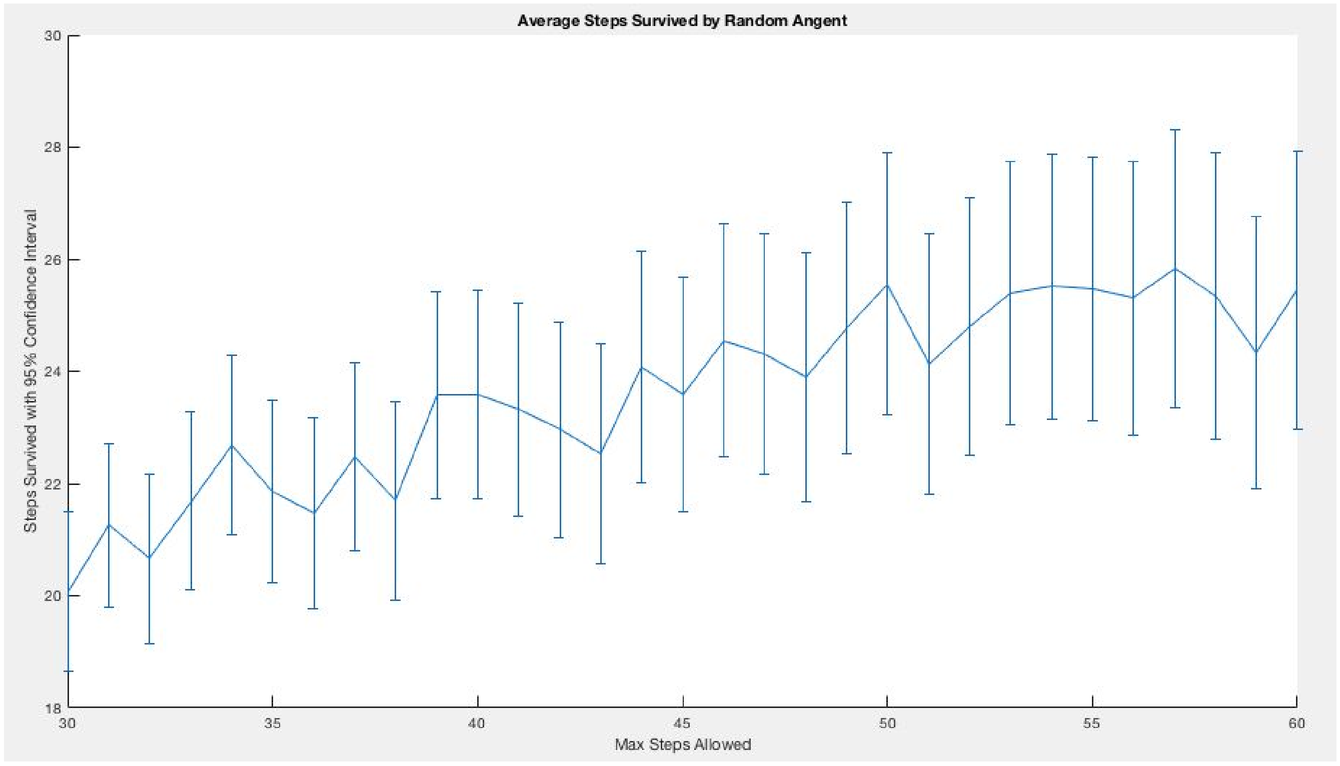
Step Count	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**
Average Steps Home	3.01	2.95	3.04	2.99	2.90	3.17	2.90	3.02	3.05	3.42	3.34	3.22	3.39	3.52

The steps it takes to get home averages out around 3. The slight upward trend is most likely due to the higher likelihood of the agent entering the gold cell at a different orientation and having to take more steps to turn itself correctly to get home.

We were unable to create any proper data/analysis for AC3 since we couldn’t finish it.

Instead here are some comparisons of steps survived by the A* agent compared to our previous random agent.





5. Interpretation

The A* Agent was able to survive longer on average because when it reached gold, it knew how to get home and wouldn’t accidentally die like the random agent would. Our confidence interval is also much tighter because we know that if the A* agent can at least make it to the gold randomly, it will be safe after that.

Unfortunately, we were unable to complete our AC solution with consistent results. My hypothesis is that the AC+A* algorithm survival rate should be much higher because we are running away from known threats. Our success rate should also be higher because we are very rarely backtracking, while when we were randomly walking we could often backtrack or spend many moves spinning in circles. With the AC improvement our agent will spin in circles much less (only when it has no guaranteed safe space) and therefore be much less likely to reach the step limit, therefore, increasing success rate. Another benefit of not spinning around as much is a faster average completion rate. Because we are not wasting time by spinning in circles at every potential move we should have a quicker average speed.

Even though we were unable to complete the entire assignment, we were able to run some experiments. These experiments took a very long time to complete because of the high maximum number of steps and large amount of trials. We did this to ensure as accurate a number as possible but it took an exceedingly long time to complete each experiment. In the future we should explore finding a middle ground between accurate results and time spent.

6. Critique

In this assignment we struggled much more with Matlab than anything else. We couldn’t finish AC because we were having translation problems between the various board representations/coordinates. We have various pieces of our problem that work independently but not cohesively because of these translation problems. We have 3 states for our agent, smart decision, random, and go home. They work separately but we couldn’t seem to get them to work in unison. I think we mostly learned that we need to make a larger plan before we start so that everything will work together instead of trying to patch things together at the end.

If we had finished/had more time, I would have liked to run the experiments with different boards to see if there was a definite difference in board layout with success.

In the future I think this assignment should be split into two, or at the very least two phases. One assignment should be A*, and the other should be arc consistency. I think splitting the assignment will greatly increase comprehension and focus.

7. Log

Isabelle’s Log: (Even Sections)
In total around 22 hours of pair programming and 2 hours of solo programming. So 24 hours.

Karla’s Log: (Odd Sections)
We spent about 22 hours working on the code over the course of three days. The first seven hours were spent on A*, and the next 14 were spent on solving AC. I then spent 2 hours on the lab report.
In total I spent approximately 24 total hours working on this assignment.