Eric Komperud – u0844210

A2 – CS4300 Lab Report – 9/14/17

An Agent With A* and Arc Consistency in Wumpus World

1. Introduction

Previously in Wumpus World, I reviewed the performance of an agent that took random actions in order to determine how effectively it could retrieve the gold. It was found to predictably bad. This week, I implemented two algorithms in the agent's logic to help it achieve the goal of finding gold and returning home safely: A* and Arc Consistency. A* was used once the agent found the gold to ensure that it could find its way home. In fact, by using A*, it's a given that the agent will find it's way home if it also manages to find the gold. Searching for the gold is the true challenge for the agent, and so this is where Arc Consistency comes into play. Using Arc Consistency in a problem means setting up constraint graph. A constraint graph can be described as:

- Any number of unique variables that are represented as nodes. These variables all initially share
- A Domain set consisting of some labels.
- These unique variables with their assigned labels are often connected/related by edges.
- When two variables are connected/related, they often carry a constraint. One common constraint is that two connected variables can't share the same label.

A very common Arc Consistency problem is that of a map where each state must be filled in with one of X colors, but no neighboring states may be the same color. In the Wumpus World problem, our consistency graph can be described as such:

- Unique Variables: Unique x/y coordinates.
- Domain Set of Labels: {**P**it, **C**lear, **B**reeze}. A space is either known to have a breeze or be clear, or assumed to have a pit.
- Edges are +/- directions in each x/y coordinate ([2,1] is connected to [2,2]).
- Constraints are dictated by the Wumpus World rules.
    - Any space connected to a **P**it has a **B**reeze.
    - Any space not connected to a **P**it is **C**lear.

Because the subject of Arc Consistency is so versatile and useful, I needed to see how helpful it would be in Wumpus World. Thus, I aimed to answer the following questions:

1. Does the use of Arc Consistency compared to random actions improve the likelihood of an agent successfully finding gold in Wumpus World? This will be tested by comparing random searching to searching that makes use of Arc Consistency
2. Is there a point of diminished returns for the maximum number of steps allowed? This will be tested by increasing the max steps allowed across trials.
3. If Arc Consistency does significantly improve the gold-pickup rate for an agent, is there a pit density that significantly reduces the effectiveness of Arc Consistency as well? This will be tested by increasing the pit density of generated Wumpus maps across trials.

To my knowledge, Arc Consistency only gives limited knowledge of surrounding nodes and so I anticipate increased success with a low density of pits, but substantially worse performance with a high density of pits.

2. Method

As the agent navigates Wumpus World, it will continuously apply the Arc Consistency restrictions on the locations it discovers. For example, when it comes upon a location and receives no percepts, it will mark all connected locations as **C**lear. When it comes upon a location and receives a breeze percept, it will mark all unknown connected locations as **P**its. In order to navigate Wumpus World safely, the agent will also make use of A* to travel to known **C**lear locations. In the case that there are no known **C**lear locations, it will instead choose a random action from *forward, rotate_right,* and *rotate_left* with equal likelihood. Because the nature of non-random algorithms makes them operate the exact same every trial if the environment is the same, I will be generating random Wumpus maps for every run of the code. Each trial will consist of 2000 runs of code, using generated Wumpus maps that have a constant pit density.

In order to test question 1, I will perform 5 trials of 1000 runs each for both an agent that uses random searching and for an agent that uses Arc Consistency. The mean gold-pickup success rate of each agent will be compared against each other. All other variables (max steps, pit density) will remain constant.

In order to test question 2, I will perform 10 trials of 2000 runs for an agent using Arc Consistency. The max steps will start at 20 and increment by 5 for each trial. Pit density will remain constant (0.20).

In order to test question 3, I will perform 10 trials of 2000 runs for an agent using Arc Consistency. The pit density will start at 0.2 and increment by 0.05 for each trial. Max steps will remain constant (30).

3. Verification

To verify that my AC algorithm works correctly, I chose 3 static maps to run it against. All of these maps contain pits that the agent will encounter and try to avoid, but none of them contains pits such that the agent will have to make a random choice and possibly die. The trace of actions (prioritizing moving right) was written beneath each figure, each leading to a success. My agent was run through all three of these maps in Matlab and succeeded in all of them. Though my agent uses a more arbitrary prioritizing process, it was still able to complete all three maps successfully. The traceback of all three runs is provided below the three figures as well as a key for the actions. These figures are all on pages 4-5.

4. Data

Table 1.

Trial	Random Mean	AC Mean
1	0.139	0.193
2	0.123	0.176
3	0.131	0.194
4	0.127	0.172
5	0.146	0.183

Table 2.

Max Steps	Mean
20	0.194
25	0.246
30	0.255
35	0.264
40	0.291

Table 3.

Pit Density	Mean
0.20	0.226
0.225	0.205
0.250	0.205
0.275	0.194
0.30	0.180

5. Analysis

It seems that my 1st postulation was correct, but my 2nd and 3rd were off base. The AC agent was indeed much more well suited to Wumpus World than was the random agent. Even the lowest mean of the 5 AC trials was higher than the highest mean of the Random trials. The number of maximum steps seemed to grow somewhat linearly with the mean success rate instead of dropping off like I anticipated. The trend shows that it might even continue to grow linearly for some time beyond 40 max steps. The rising pit density seemed to correlate with a lower mean success rate as expected, but there was no significant drop off point. Instead, the mean success rate seemed to fall pretty linearly with an increasing pit density.

6. Interpretation

While it appears that AC is not the best algorithm for avoiding hazards in a dangerous environment, it does still have an improvement over random searching. In addition to that, the success that the algorithm does enjoy does not seem to drop off very quickly with higher stress as I thought it might. Further investigation into Path Consistency might be interesting to see.

7. Critique

The biggest critique I have of this experiment is the time I allotted myself to do it. I worked on it for around 15-20 hours and still feel like I've done very sloppy work. In fact, I'm not entirely convinced that AC's poor performance wasn't the algorithm's fault. Given more time, I would have spent more of it testing A* and AC to make sure that they were both very well oiled and free of bugs. Because of the lack of time, I also was not able to come up with very interesting questions or thorough testing methods. There is a lot of data that I did not collect that would normally be very valuable for detecting trends and patterns. All in all, there's simply too much to critique on this report, but I do now know how much effort a normal assignment takes compared to the first assignment. Lesson learned I suppose.

8. Log

Approximately 14 hours were spent analyzing, writing, and debugging code. Most of this was spent debugging. I've learned that it's probably best to write your own code if you don't know what sample code is doing. Another 3 hours were spent planning the lab report, along with collecting and analyzing data.

Eric Komperud

Fig 1

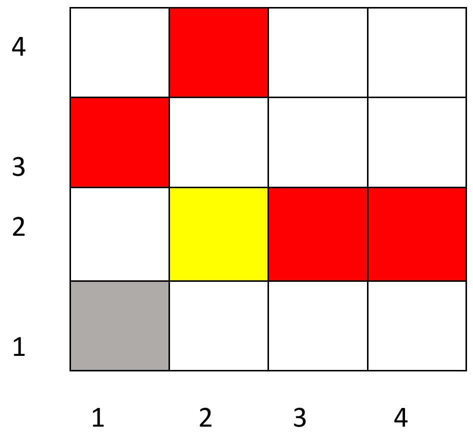


Table 4

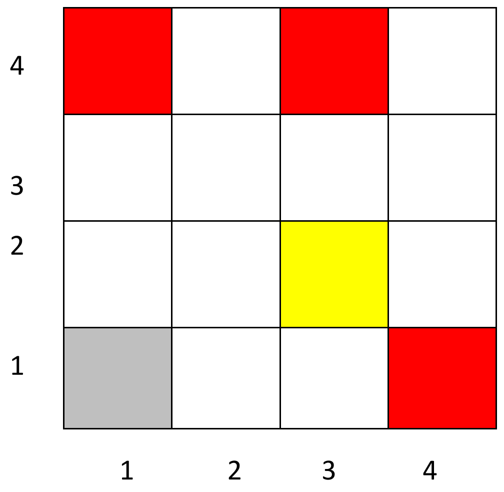
X	Y	Action
1	1	0
2	1	1
2	1	3
2	2	1
2	2	4
2	2	3
1	2	1
1	2	3
1	1	1
1	1	6

1. Start at (1,1). Choose from:
    a. (2,1). Choose from:
        i. (3,1). Breeze detected. Return
        ii. (2,2). Gold found. **Go Home**.
    b. (1,2). Not expanded.

Table 5

X	Y	Action
1	1	0
2	1	1
2	1	3
2	2	1
2	2	3
1	2	1
1	2	3
1	1	1
1	1	3
2	1	1
3	1	1
3	1	2
3	1	2
3	1	2
3	2	1
3	2	4
3	2	3
2	2	1
1	2	1
1	2	3
1	1	1
1	1	6

Fig 2



2. Start at (1,1). Choose from:
    a. (2,1). Choose from:
        i. (3,1). Breeze detected. Return
        ii. (2,2). Choose from:
            1. (3,2). Gold found. **Go Home.**
            2. (2,3). No expansion.

Fig 3

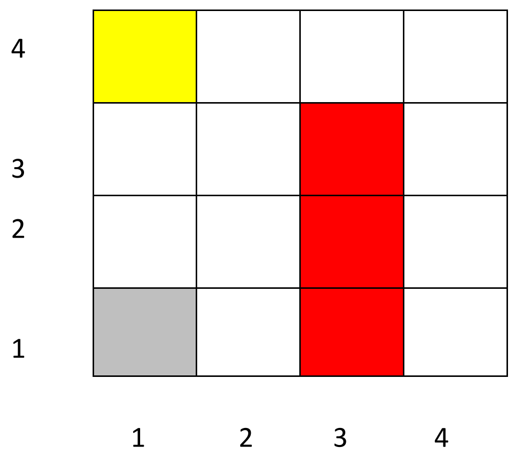


Table 6

X	Y	Action
1	1	0
2	1	1
2	1	2
2	1	2
1	1	1
1	1	2
1	2	1
1	2	2
1	2	2
1	2	3
2	2	1
2	2	2
2	2	2
1	2	1
1	2	2
1	3	1
1	3	2
1	3	2
1	2	1
1	2	3
2	2	1
2	2	3
2	3	1
2	3	3
1	3	1
1	3	2
1	3	2
1	3	3
1	4	1
1	4	4
1	4	2
1	4	2
1	3	1
1	2	1
1	1	1
1	1	6

1. Start at (1,1). Choose from:
    a. (2,1). Breeze detected. Return.
    b. (1,2). Choose from:
        i. (2,2). Breeze detected. Return.
        ii. (1,3). Choose from
            1. (2,3). Breeze detected. Return
            2. (1,4). Gold found. **Go Home.**

Action Key:

0	1	2	3	4	5	6
Nil	Fwd	Rotate Right	Rotate Left	Grab	Shot	Climb