Isabelle Chalhoub - u0678302
Karla Kraiss - u0830999
CS4300
November 21, 2017

A7 Lab Report

1. Introduction

The purpose of this project is to create algorithms for Utility and Policy generation using the Markov Decision Problem. We are calculating utilities of cells on our Wumpus board using our scoring system as our reward policy and the actions:

A = {UP, LEFT, DOWN, RIGHT}

By calculating expected utilities for each state in our wumpus board:

0 0 0 G
0 0 P 0
0 0 W 0
0 0 P 0

We can then create a policy based on the expected utilities that will provide an agent with a path to the good terminal state (gold) and will avoid bad terminal states like pits or the wumpus.

- How effective is value iteration in finding a safe path policy?
- How does the effectiveness differ when given a varying range of error thresholds?

2. Method

CS4300_MDP_value_iteration:

This method runs the Markov Decision Problem. It finds the maximum utility for each state by checking that states' possible actions action against the surrounding states' rewards. The program continues this process until convergence is reached in accordance with the passed in gamma, or reaches the maximum allowed iterations.

CS4300_run_value_iteration:

This method sets up the Markov Decision Problem and then calls MDP_value_iteration. To set up the MDP each action a state can make must be recorded with the probability that action will be fulfilled. Each state has four actions (up, left, down, right). We hardcoded these probabilities. We also had to define the board size; 16 in the case of Wumpus World, 12 in the verification board.

CS4300_MDP_policy:

This method utilizes the max utility generated by MDP_value_iteration to decide the best action. It compares all the utilities of a given state's neighbors and chooses the highest one to move towards.

3. Verification of Program

To verify, our MDP algorithm, we ran it using the framework for a 4x3 board as explained in the textbook in figure 17.3 to see if we got matching expected utilities. Here are our results for this framework:

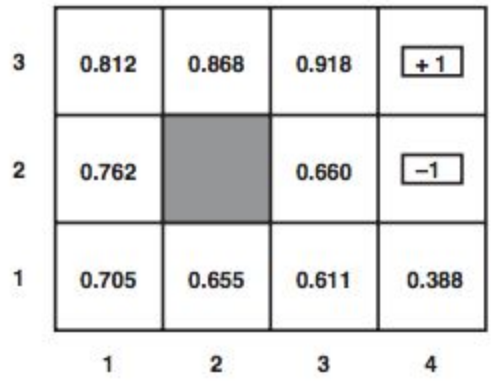
```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> U

U =

  Columns 1 through 7

    0.7053    0.6553    0.6114    0.3879    0.7616         0    0.6603

  Columns 8 through 12

   -1.0000    0.8116    0.8678    0.9178    1.0000
```

In a more readable format:

Our results:

.8116	.8678	.9178	1
.7616	0	.6603	-1
.7053	.6553	.6114	.3879

Book results:



Now to verify our MDP policy algorithm, we ran the same framework from above and then called policy to get the preferred actions.

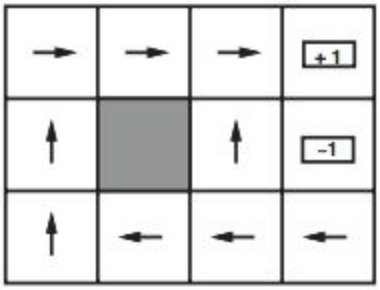
```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> policy = CS4300_MDP_policy(S,A,P,U)

policy =

     1     2     2     2     1     1     1     1     4     4     4     1
```

We are using 1-4 to signify [UP,LEFT,DOWN,RIGHT] to match the specs. In a more readable format:

Our board:

4 - RIGHT	4 - RIGHT	4 - RIGHT	
1 - UP		1 - UP	
1 - UP	2 - LEFT	2 - LEFT	2 - LEFT

Book's board:



The actions in states 6, 8, and 12 will be ignored on our board because they are terminal states.

Then we tested the same board with varying values for our reward as specified in figure 17.2 (b) in the book:

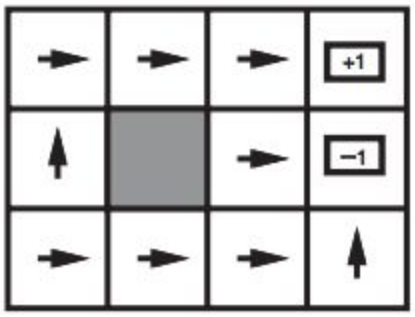
R(s) = -1.8

```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> R

R =

  Columns 1 through 7

   -1.8000   -1.8000   -1.8000   -1.8000   -1.8000         0   -1.8000

  Columns 8 through 12

   -1.0000   -1.8000   -1.8000   -1.8000    1.0000

>> policy = CS4300_MDP_policy(S,A,P,U)

policy =

     4     4     4     1     1     1     4     1     4     4     4     1
```

Our board:

4 - RIGHT	4 - RIGHT	4 - RIGHT	
1 - UP		4 - RIGHT	
4 - RIGHT	4 - RIGHT	4 - RIGHT	1 - UP

Book's board:



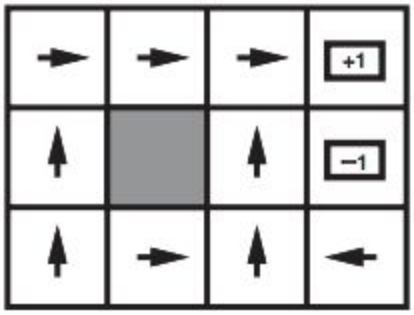
R(s) = -0.0850

```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> R

R =

  Columns 1 through 7

   -0.0850   -0.0850   -0.0850   -0.0850   -0.0850         0   -0.0850

  Columns 8 through 12

   -1.0000   -0.0850   -0.0850   -0.0850    1.0000

>> policy = CS4300_MDP_policy(S,A,P,U)

policy =

     1     4     1     2     1     1     1     1     4     4     4     1
```

Our board:

4 - RIGHT	4 - RIGHT	4 - RIGHT	
1 - UP		1 - UP	
1 - UP	4 - RIGHT	1 - UP	2 - LEFT

Book's board:



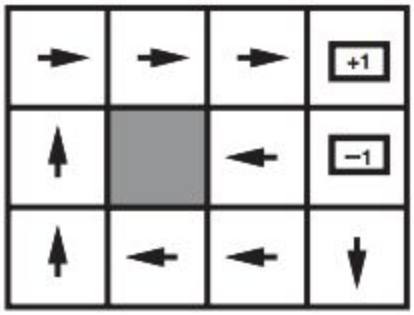
R(s) = -0.02

```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> R

R =

  Columns 1 through 7

   -0.0200   -0.0200   -0.0200   -0.0200   -0.0200         0   -0.0200

  Columns 8 through 12

   -1.0000   -0.0200   -0.0200   -0.0200    1.0000

>> policy = CS4300_MDP_policy(S,A,P,U)

policy =

     1     2     2     3     1     1     2     1     4     4     4     1
```

Our board:

4 - RIGHT	4 - RIGHT	4 - RIGHT	
1 - UP		2 - LEFT	
1 - UP	2 - LEFT	2 - LEFT	3 - DOWN

Book's board:



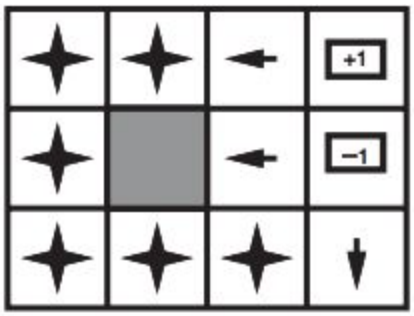
R(s) = 2

```
>> [S,A,R,P,U,Ut] = CS4300_run_3x4();
>> R

R =

     2     2     2     2     2     0     2    -1     2     2     2     1

>> policy = CS4300_MDP_policy(S,A,P,U)

policy =

     2     1     2     3     2     1     2     1     1     1     2     1
```

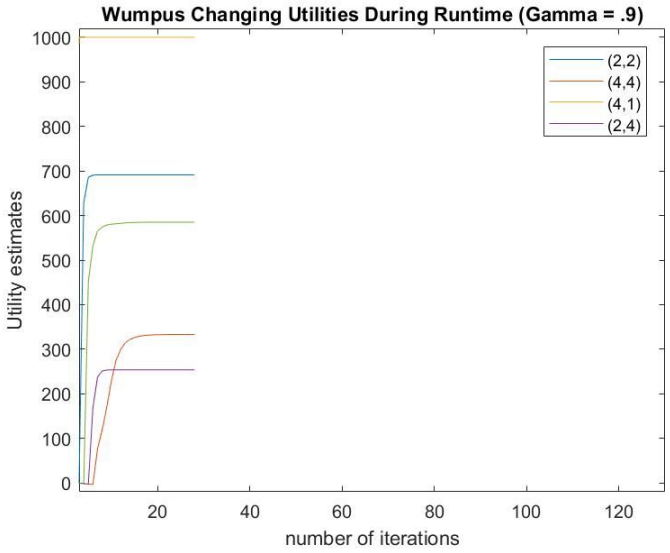
Our board:

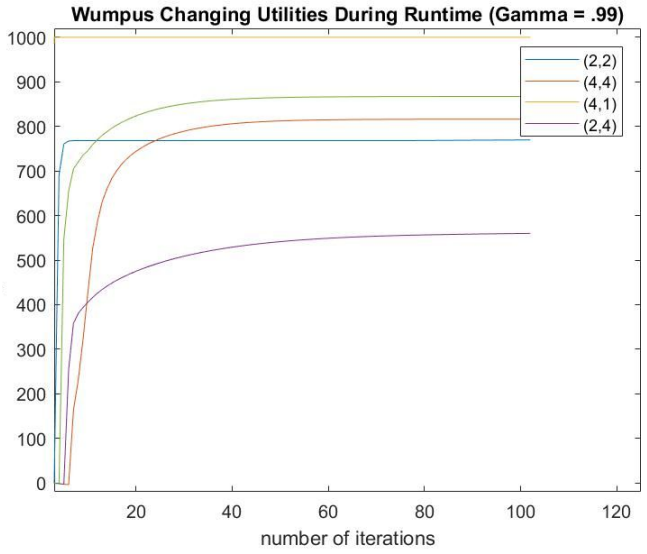
1 - UP	1 - UP	2 - LEFT	
2 - LEFT		2 - LEFT	
2 - LEFT	1 - UP	2 - LEFT	3 - DOWN

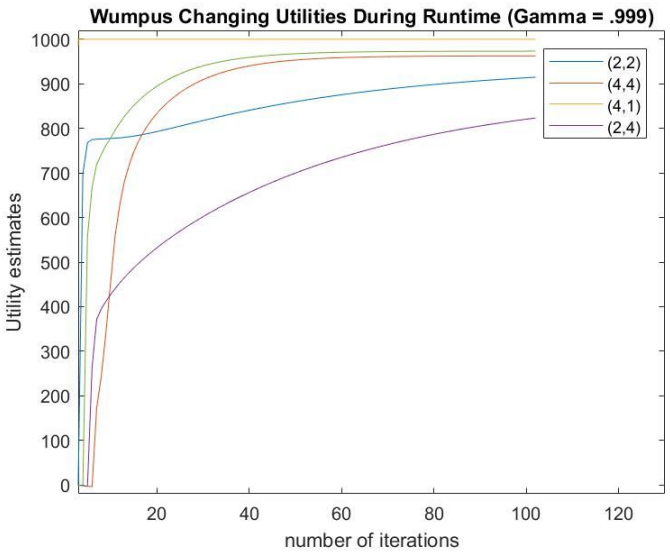
Book's board:

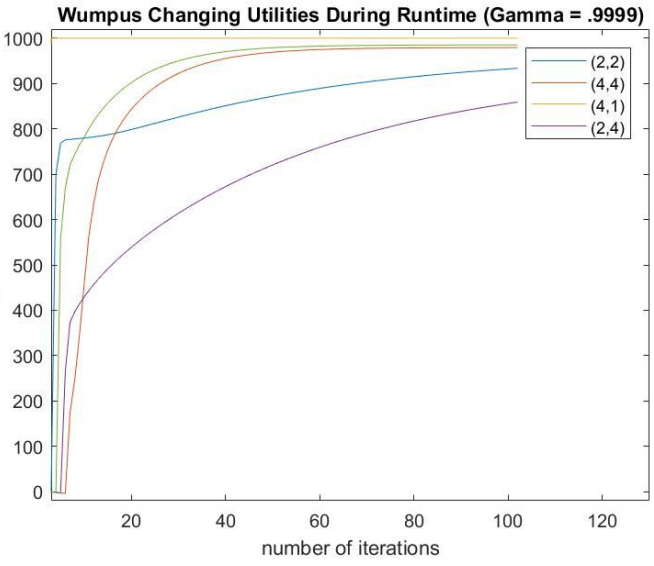


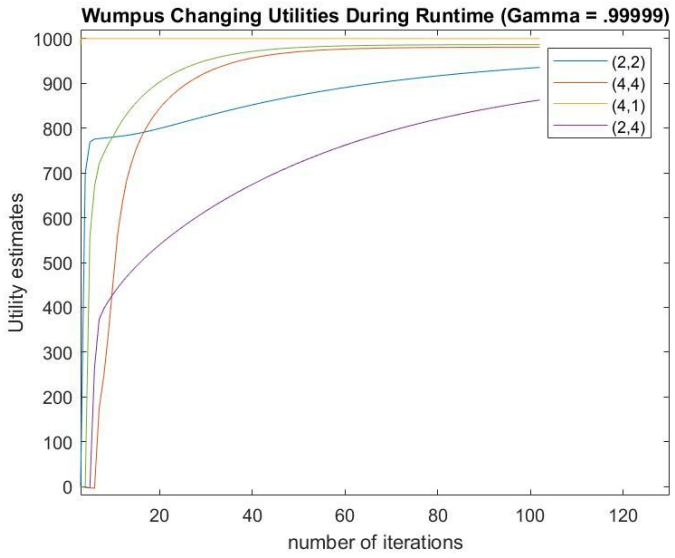
4. Data and Analysis

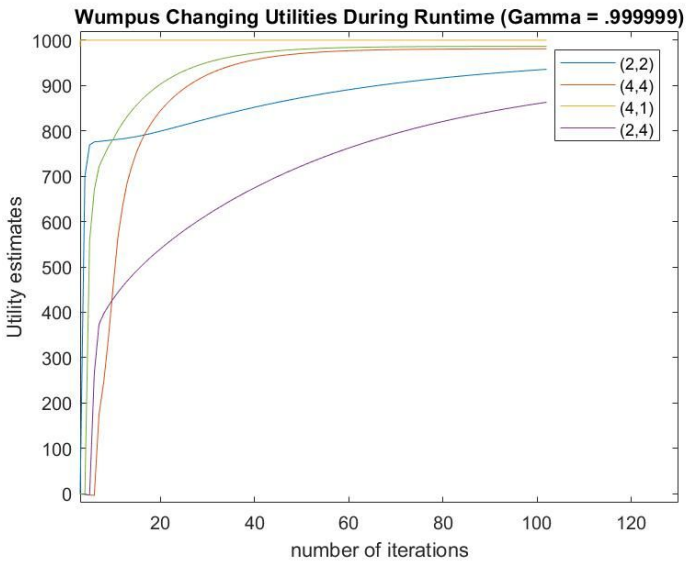












Our final MDP_Policy for the Wumpus board given in the intro:
This is for a gamma of .9 as well as a gamma for .999999!

RIGHT	RIGHT	RIGHT	X
UP	UP	X	UP
UP	UP	X	UP
UP	UP	X	UP

5. Interpretation

As the gamma gets more precise our Markov Decision algorithm requires more iterations to complete. In the later graphs one can observe that the lines always end at our max_iteration variable (set at 100). The first few graphs before even approaching the maximum iterations allowed. More precise gamma does allow for a much more precise utility. Comparing the graph with only .9 gamma to the .999999 gamma shows a vast difference in the resulting utility of the shown states. Such a disparity between utilities shows the importance of having at least three digits of precision. Comparing .999 with .999999 doesn't show nearly as big of a gap between the utilities.

Value iteration is extremely effective in finding a safe path to positive terminal states as shown by our policy boards above. It successfully finds the shortest path to the positive terminal states while avoiding the negative terminal states. This is based on the

rewards decided in the run_value_iteration method. For example, if the reward exceeds the bad terminal states, the best option is to head towards the terminal states as quickly as possible to avoid moving. On the other hand, if the reward framework is larger than the positive terminal states, then it will avoid those as well. Given our reward framework for Wumpus world, our MDP policy algorithm very successfully finds a quick path to the gold since it offers a large enough reward to encourage movement over staying in place.

It seems that the gamma value has a very clear effect on the utilities but little to no effect on the policy. When run for .9 and .999999 we were given the same policy. We believe that it is because even though the utilities are less precise, the ratio of their utilities is mainly unchanged for varying gamma values, leading to the same policy as a result.

6. Critique

In this assignment we had to figure out how to verify our implementation of the Markov Decision Problem without the use of an agent. An important concept we learned from this was how to visualize the correct response without the visual aid the agent provided. This assignment was much more conceptual than past assignments, and through that we learned to think of a bigger picture when working on assignments.

Something we could improve upon if doing this assignment again would be not hardcoding the neighbor probabilities. To avoid extra coding time and long debugging sessions, we opted to hard code the neighbor probabilities for each action in each cell. Because we hard coded this, we had to repeat the process for our 3x4 verification and Wumpus board

7. Log

Isabelle's Log: (Odd Sections)
I spent about 6 hours on this assignment.

Karla's Log: (Even Sections)
I spent about 3 hours programming and 3 hours working on the lab report. In total I spent 6 hours on this assignment.