# Solving Markov Decision Processes

Jacob Regan

## 1 Abstract

Markov decision processes (MDP) provide a mathematical representation of sequential decision problems for stochastic environments. In this paper, we examine the performance of two common MDP algorithms – value iteration and modified policy iteration – for two classic environments. We also evaluate the efficiency of model free approaches to solving MDPs, namely Q-learning and SARSA. In all cases, we are applying these solutions to obtain an optimal policy for our environment.

## 2 Introduction

MDPs are a useful way of representing problems pertaining making decisions for a stochastic environment. In practice, agents can leverage the structure of MDPs to develop an optimal policy for some sequential decision problem where utility depends on the set of choices the agent makes. An MDP framework can be utilize for a variety of problems ranging from simple cases such as navigating a simple maze to playing chess to complex tasks such as evaluating the impact of dynamic health policies towards mitigating an epidemic [1]. Here, we are concerned with solving simple MDPs for two classical environments: a grid world and a Wumpus world. We will discuss these environments in detail in the following section, along with a more formal definition of MDPs.

## 3 Methods

### 3.1 Markov Decision Process

As discussed, and MDP is a sequential decision problem for a stochastic environment. They are defined by $S$, the set of all possible states, $S$, with initial state $s_0$, a set $A(s)$ representing the actions at each state $s$, a transition model $P(s'|s, a)$, and a reward function $R(s)$. This definition comes from Chapter 17 of 'Artificial Intelligence: A Modern Approach' [2].

### 3.2 Environments

The environments used in this report were based on the implementations provided in the TUmasters GitHub repository [3].

### 3.2.1 Grid World

The grid world implementation has functions for generating a grid world with specified width and height. A base movement cost, -0.1 by default, can also be specified, as can the end terminals positions and associated reward values. Internal walls can also be added as desired to constrict movement. In this paper, the initial state will always occur at the position (0,0) on the environment. The implementation also implements the environment as an MDP, with support for all the components listed in section 3.1.

### 3.2.2 Wumpus World

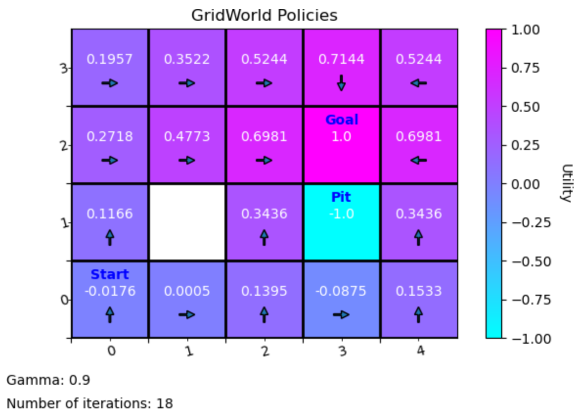
The Wumpus world implementation is, at a glance, very similar to the grid world implementation. They both share the same basic dimension features and MDP integration. Wumpus world, however, is slightly more complex, providing additional game objects including a Wumpus, pits for the agent to fall into, and two items the agent can collect: gold and immunity. If they pick up the gold, the agent will receive an additional reward, which can be specified. The immunity allows the avoid the penalty associated with both the Wumpus and a pit if they have the item when entering a square with either.

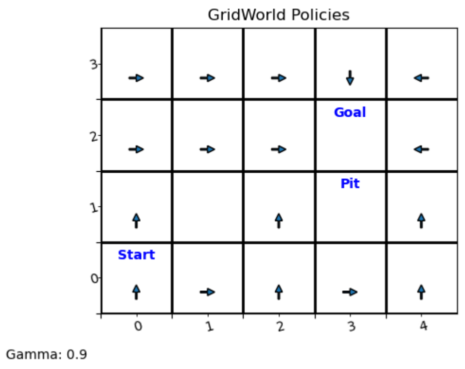
# 4 Results

## 4.1 Known Worlds

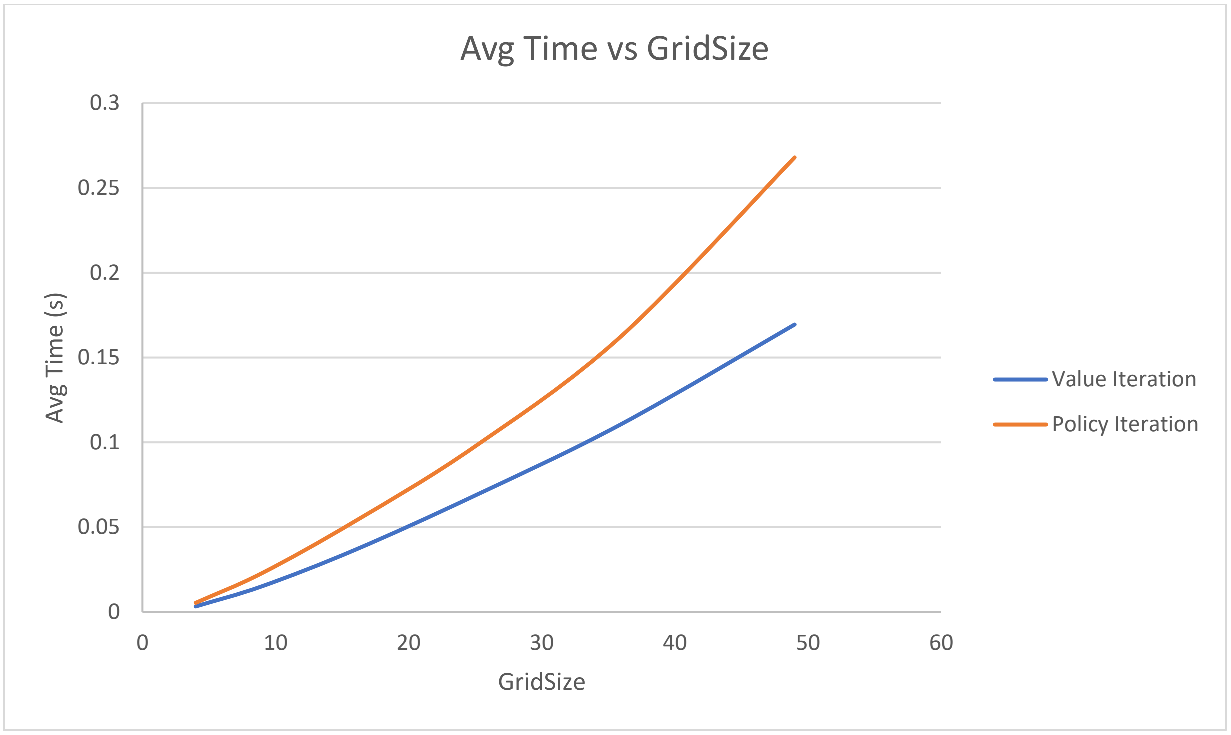
### 4.1.1 Grid World Comparisons

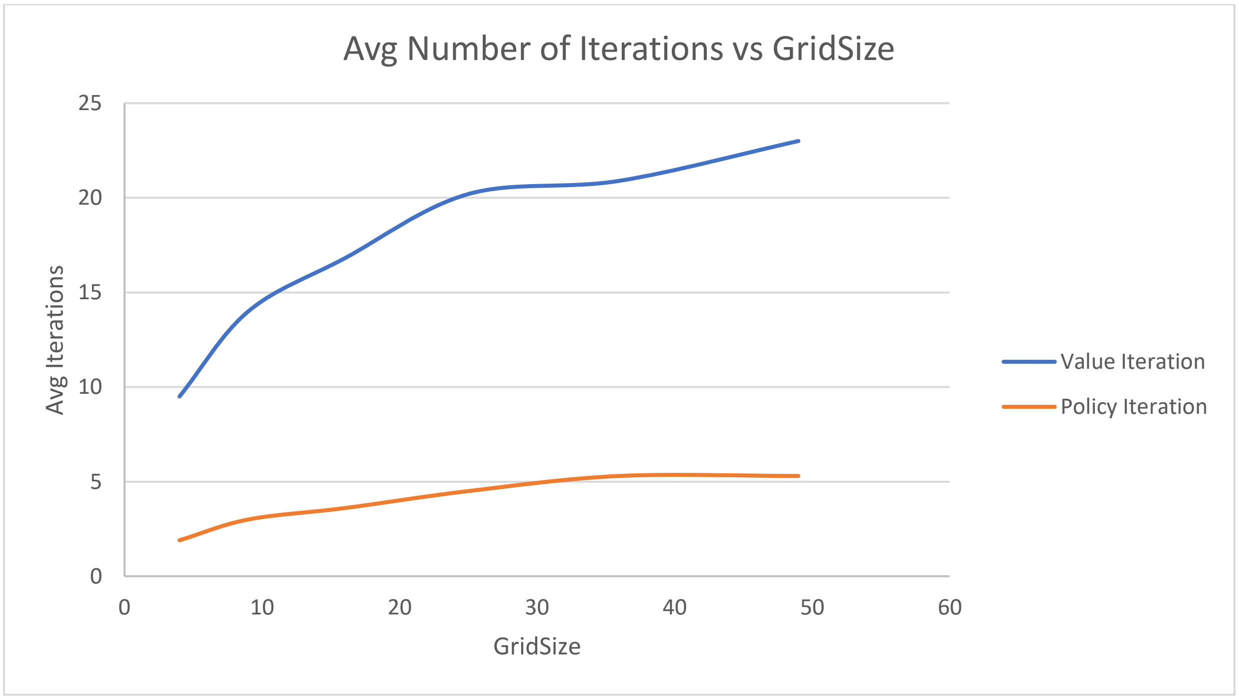
In terms of result, both the value iteration and modified policy iteration algorithms produced good policies for various grid worlds. The figures below show the results for the value iteration modified policy iteration on the same 5 by 4 gridWorld environment. In both cases, the discount factor λ=0.9.





As we can see, the optimal policy produced is the same, but there was a striking difference in performance.

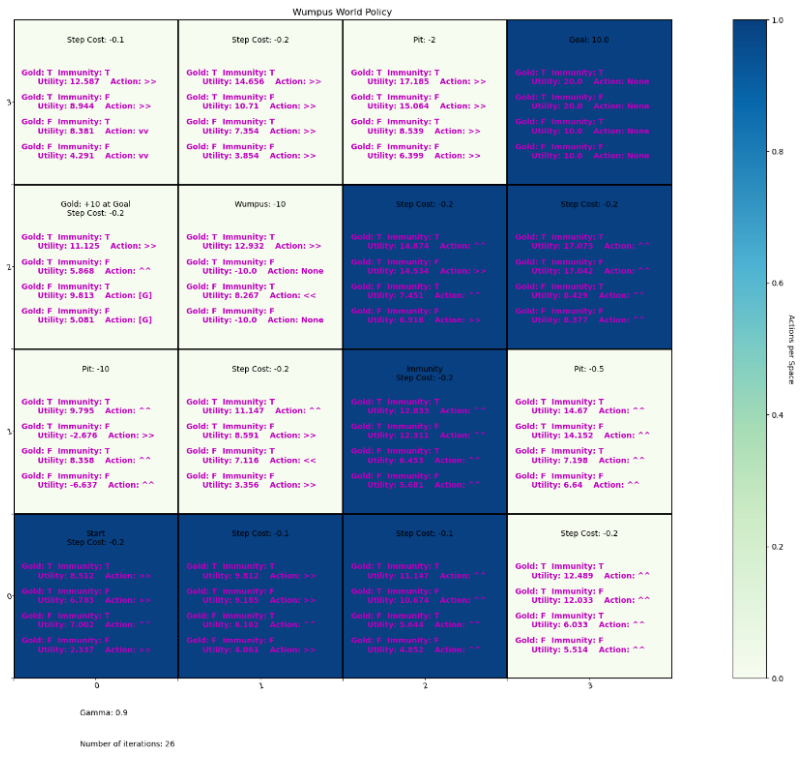




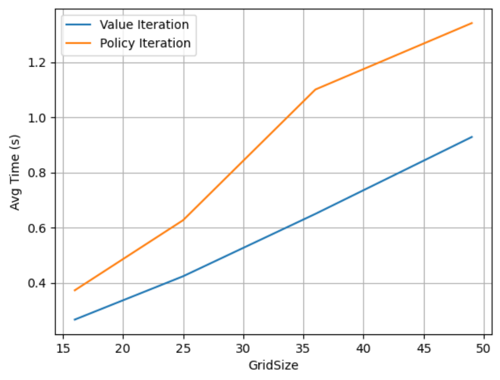
The above data was collected from the average of 50 samples for each gridSize. The position of the goal and a pit, with rewards 1 and -1 respectively, were randomly set for each sample trial. The movement cost was set to -0.1, and the agent had a 20% chance of not making the expected move, which was introduced as noise. By removing the noise, the avg time and the avg number of iterations was reduced by a factor of 2.

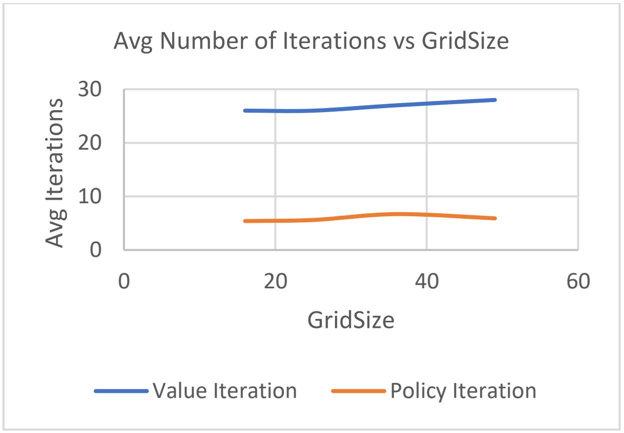
## 4.1.2 Wumpus World Comparison

For Wumpus world, the value and policy iteration algorithms had comparable performance to grid world, though the average execution time and average number of iterations was higher, since each space was represented by its 4 possible states. This is shown in the figure below.



These graphs show the performance of the algorithms on the Wumpus world, and they closely resemble the graphs for grid world.



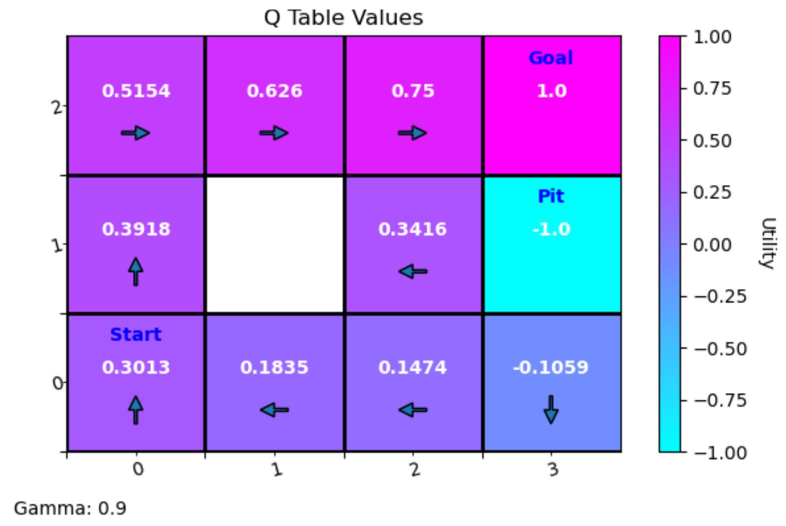


As the above table shows, the performance of value iteration and policy iteration on Wumpus world was consistent with the performance on grid world, though it took longer because there were 4 possible states per grid space, as opposed to one.

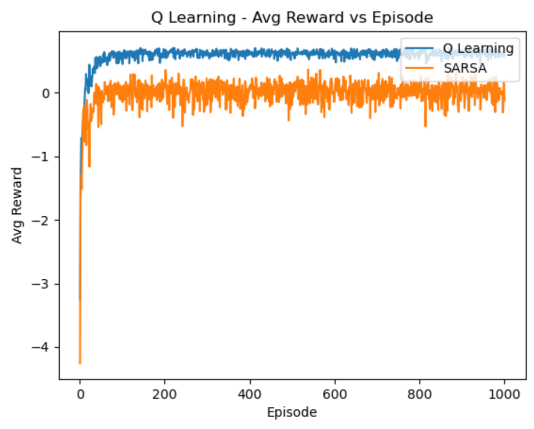
## 4.2 Unknown Worlds

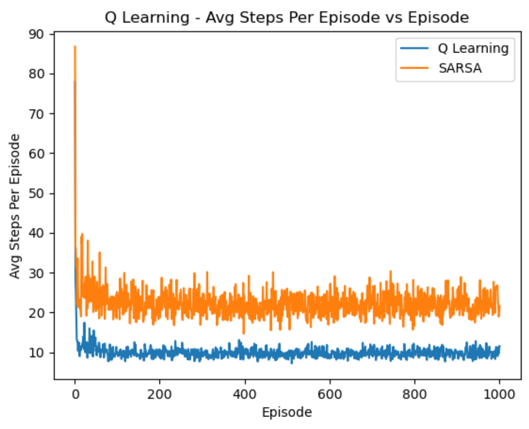
From exercise 21.5 in the textbook, we will test a Q learning and SARSA learning algorithm on a 4x3 and 10x10 grid world. For both environments, we used a penalty of -0.04, and a discount factor of 0.9. The results are show below:

Q Learning with 500 Episodes:



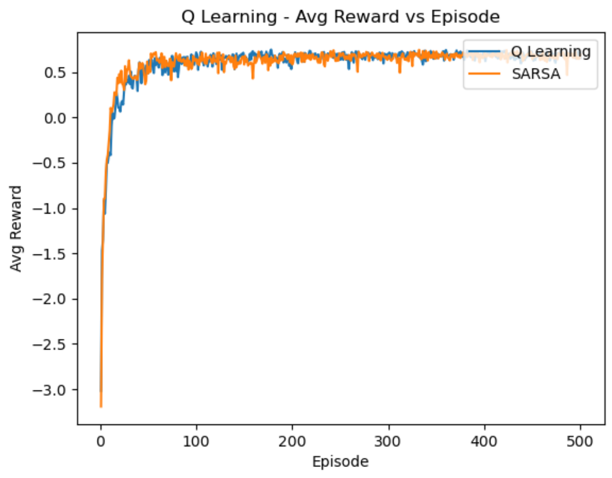
To compare the performance of Q Learning vs SARSA, we took the average over 50 trials, each 1000 episodes. The results of those trials are shown below.

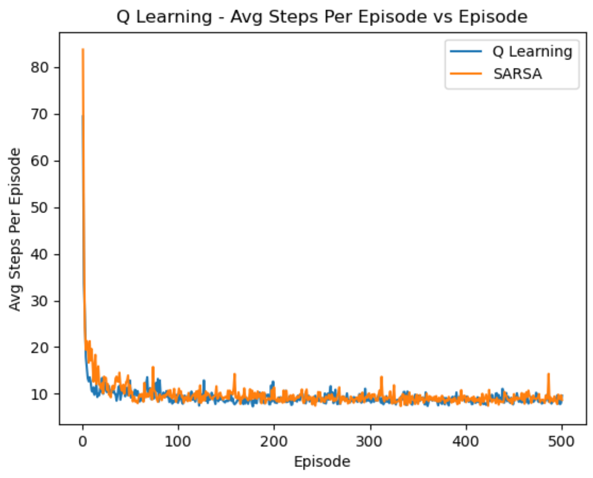




`

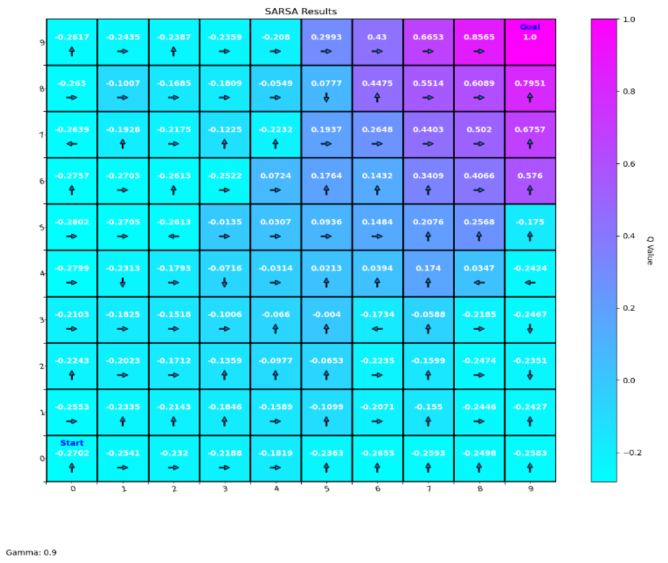
As we can see, for the grid world, 4x3 grid world environment, Q learning had both higher average rewards and lower average step count per episode, which shows better performance than SARSA for this case. The Reward graph also shows the convergence of the two algorithms were about the same and occurred by around the 70th episode. While the graphs have not been included here due to formatting, I did also compare the Q learning and SARSA implementations for noiseless environments, which showed much faster convergence, as well as significant initial variance before eventually converging to no variance. The last test I conducted was altering the learning rate from the function $\alpha(n) = \frac{60}{n+59}$ to α = 0.5. The results are show below.

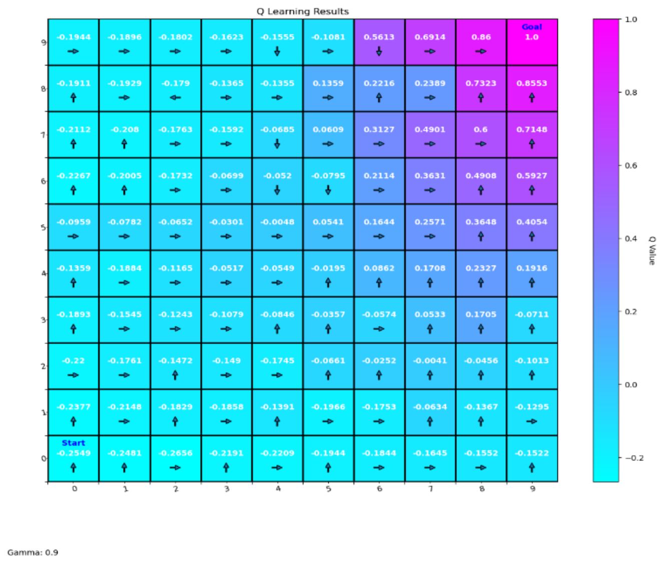




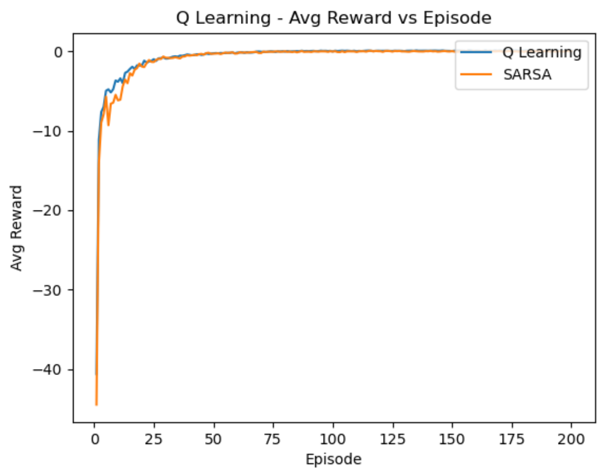
To my surprise, SARSA performed significantly better with the constant learning rate. Q learning was comparable to the functional learning rate.

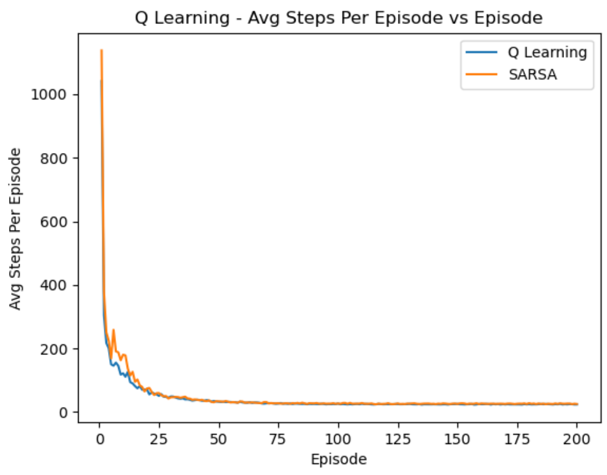
For the 10 by 10 environment, we got some different results. All other parameters were kept as for the 4x3, except the size, the internal wall, and the presence of a -1 weighted obstacle. The result of the Q learning and SARSA algorithms are shown below. The number of episodes were reduced to 200 for better viewing results, and the number of trials was left at 50.





Unlike with the 4x3 environments, the two algorithms had comparable performance on the 10 by 10 world. The Q Learning converged just a tad faster, but their results were basically identical, as shown by the following graphs.





Based on the Reward vs Episode graph on the left, the convergence occurred after 75 episodes for the 10 by 10 for both algorithms.

## 5 Conclusions

Based on the results for the first part of the experiment, policy iteration surprising took longer, though the average number of iterations were not only significantly but grew much slower with increasing grid size. It was increasing that value iteration performed more consistently on the Wumpus World problem than on the grid world, but the results were overall consistent. Of note, the time to reformat the result of value iteration into actual policy was not included in the execution time for those results, so the outcome of policy iteration was more useful in the sense that it produced an actual optimal policy.

I was surprised that SARSA seemed to perform worse on the 4x3 grid world when a functional learning rate was used, and that the constant learning rate performed better. Otherwise, the results were consistent with expectation.

## 6 References

[1] R. Yaesoubi and T. Cohen, "Dynamic Health Policies for Controlling the Spread of Emerging Infections: Influenza as an Example," *PLOS ONE*, vol. 6, no. 9, 2011.

[2] S. J. Russell and P. . Norvig, Artificial Intelligence: A Modern Approach, ed., vol. , , : Pearson Education, Inc., 2010, p. .

[3] C. Crawford, *Sample MDPs for Advanced AI*, TUmasters, 2018.