A4. Markov Decision Processes

Jihoon ‘Jay’ Song
jsong350@gatech.edu

INTRODUCTION

In this section, I introduce the two Markov Decision Processes (MDPs) that I reference throughout the paper. I describe the MDPs and discuss what makes each of them unique and interesting from a machine learning perspective. Afterwards, I briefly go over the three algorithms which I apply to the MDPs. The objective of the assignment was to gain a comprehensive understanding of the three algorithms. This was accomplished through care observation of how each algorithm behaves when performed against the two MDPs which differ on state size (small/large) and problem structure (grid/non-grid).

MDP #1: Forest Management Problem (FMP) – I sourced the FMP from the hiivemdptoolbox [1]. In this problem, each possible state of the forest was represented by an array with dimensions of (2, 500, 500). Each cell represented the transition probabilities (action, state, state). There were 2 actions in FMP, which was ‘Wait’, indicated by 0 and ‘Cut’ indicated by 1. Reward was attained in one of two ways: One way was when “Wait” was performed while the forest was in its oldest state. The second way was when “Cut” was performed while the forest was in its oldest state. In the example I used, the first reward amounted to a value of 4, and the second to a value of 2. The main state space chosen for this problem was 500, but other sizes were also used to examine their effect on the three different algorithms. In summary, this problem was interesting primarily because it was set in a non-grid world and because of its large state size.

MDP #2: Frozen Lake Problem (FLP) – I sourced the FLP from Open AI Gym [2]. To convert the 4x4 FLP from Open AI Gym format into one which was compatible with the reinforcement learning algorithms in hiivemdptoolbox, I passed it through the built-in converter class called OpenAI_MDPToolbox [3]. Once converted, each possible state of the frozen lake was represented by an array with dimensions of (4, 16, 16). Each cell represented the transition probabilities (action, state, state). As shown by the 4 dimensions, there were 4 actions in FLP. The actions were 0, 1, 2 and 3, which indicated directional movement left, down, right, and up respectively. Reward value of 1 was attained if the agent landed on the goal, and 0 if the agent landed on a hole or a frozen space. The main state space chosen for this problem was 4x4, as noted above, but 8x8 was also explored to analyze the effect of size on the three different algorithms. In summary, this problem was interesting because it was set in a grid world and because of its small state size.

Algorithm #1: Value Iteration (VI) – In VI, the goal is to compute the optimal state value function. At each iteration, it calculates the resulting value using all possible next actions then selects the action resulting in the maximum utility. It does this until it finds an optimal value function and executes a single policy extraction [4]. A solution may be reached prematurely if the algorithm reaches the maximum number of iterations before finding an epsilon-optimal policy.

Algorithm #2: Policy Iteration (PI) – This algorithm is similar to VI in that it also iteratively improves. Unlike VI, however, instead of improving the value function, it improves the policy itself. It generates, evaluates, and improves the policy until the policy itself converges into an optimal one.

Algorithm #3: Q-Learning (QL) – unlike VI and PI, QL is not a model-based algorithm. In other words, it does not leverage the transition probability distribution provided by the MDP. Rather, it learns through execution of random actions that are not prescribed by a policy. A non-model-based algorithm was chosen intentionally to explore the differences between the performance of model-based and non-model-based algorithms on the two MDPs.

PART I: FOREST MANAGEMENT

In Part I, I explored the three algorithms, VI, PI, and QL by observing how they behaved when they were applied to MDP #1: Forest Management Problem (FMP). For each algorithm, I tuned their hyperparameters (HPs) in accordance with their convergence plots. Each algorithm’s convergence criterion are described in their respective sections.

Value Iteration (VI) – To start, I first performed VI on FMP using an arbitrary value of gamma=0.5 and default value of all other HPs. To identify patterns of convergence, I generated and analyzed the plot depicting the number of iterations vs. the rewards. I hypothesized that as the value of discount gets closer to 1, the reward would also increase.

To assess the effect of discount value (gamma), I changed the value of gamma while holding all other HPs constant. Through this experiment, I confirmed my hypothesis that increasing the discount value increased the maximum reward achieved via the resulting policy. Similarly, to assess the effect of the built-in stopping criterion given by mdptoolbox, I changed the value of epsilon while holding all other HPs constant. Although obvious, through this, I found that decreasing the stopping criterion increased the maximum reward achieved via the resulting policy. Finally, to validate that this relationship is maintained when both gamma and epsilon are both changed simultaneously, I plotted 5 lines – each depicting different values of epsilon on a graph which showed the relationship between reward and gamma. As shown by the blue line in *Figure 1*, epsilon=0.01 and gamma=0.8 resulted in the highest reward. This confirmed that the relationship held true independently, and that the two

HPs had completely opposite relationships in respect to reward.

To fine tune VI's HPs, I again made use of the plot, iterations vs. rewards. In addition, I also plotted iterations vs. delta of rewards and calculated the delta of the delta of the rewards to determine the index in which the algorithm first crosses the threshold. I selected the threshold to be the index where the delta between the delta of rewards is less than 0.01. This indicated the point in which the delta of rewards was no longer increasing at a significant rate, which was my convergence criteria.

To determine the optimal HPs using the convergence criteria, I determined the threshold index when the value of gamma was set extremely high (0.999), and epsilon was set extremely low (0.00001). Then I reversed the two HPs until the index at convergence matched the threshold indicated by the extreme example. As shown in *Figure 2*, this was achieved when epsilon=0.01 and gamma=0.97.

I do want to acknowledge that this convergence criteria was rather arbitrary (i.e., "extremely high" value was established to be 0.999). But this was necessary to find a threshold point. Regardless, to account for the bias, I take these assumptions into account when I compare the results of all algorithms in Part III.

Policy Iteration (PI) – Like VI, to begin, I performed PI on FMP using an arbitrary value of gamma=0.5 and default value of all other HPs. My first observation was that interestingly, the agent was able to reach convergence from very early on (iteration=2), at a lowly mean value of 1.325. Given the low value, I was curious to see if tuning the HPs would result in increase in value and iterations required for convergence. To test this, I again generated plots showing iterations vs. the rewards while tuning gamma. I hypothesized that for PI, just like with VI, as the discount value gets closer to 1, reward and iterations required for convergence would increase. As shown in *Figure 3,* results showed that my hypothesis was correct. As the value of discount got closer to 1, the reward also increased.

To fine tune PI HPs, I adhered to an almost identical convergence criterion as the one I used to tune VI HPs. I determined the threshold index when the value of gamma set extremely high (0.999), which was iterations=20. Then I gradually reduced the value of gamma until the index at convergence matched the threshold indicated by the extreme example. As shown in *Figure 4 and 5,* this was achieved when gamma=0.999 resulting in reward=474.44.

Again, it's worth noting that although the core concept behind the convergence criteria for VI and PI was identical, given the mismatch of HPs available between the two algorithms, I recognize that it's not a true apple-to-apple comparison. This will be highlighted when I compare the results of all the algorithms.

Q-Learning (QL) – I first performed QL on FMP using the default value of all HPs across gamma between 0.05 and 1 in 0.05 intervals. Interestingly, the resulting reward values were significantly lower than those of VI and PI. I had 3 theories: 1) The QL HPs must be tuned significantly to solve this problem; 2) QL is not suited to solve a problem of this state size; 3) QL is not suited to solve this type of non-grid world problem; I hypothesized that it was most likely that theory #2 was correct. But to be certain, I explored all three. The convergence criteria used for Q-Learning was when the delta of delta rewards was less than 0.0001.

To test theory #1, I tuned all available HPs, gamma, epsilon, alpha, alpha decay, and epsilon decay and plotted their values against reward to determine their relationships (*Figures 6-10)*. As predicted, despite generating a policy using a combination of all optimal HPs, the reward at convergence was a mere 0.45. For the record, a convergence criterion using the same core concept used for VI and PI was used for QL. Therefore, I was able to prove that no matter how the HPs were tuned, QL was unable to generate an effective policy to solve this problem.

To test theory #2, I conducted a very simple experiment. While holding the HPs constant, I reduced the state size of the problem from 500 to 50. Afterwards, I found the convergence point using the same exact convergence criteria and the resulting reward. Surprisingly, my hypothesis was proven to be incorrect going from a state space of 500 to 50. The number of iterations it took the converge increased to 2131 and the reward was still very low, at 0.65. Therefore, I reduced the state space from 50 to 5. At this state space, the reward increased to 8.79, but the learner did not converge per my convergence requirement even when reaching the maximum number of iterations. The results were recorded in *Table 1.* Therefore, I was able to prove that a QL was able to generate a somewhat effective policy when the problem's state space was reduced significantly.

To test theory #3, I ran QL for both non-grid world and grid world problem then compared the results and observations between the two types of problem. Given Part I will only cover the non-grid world problem FMP, I will revisit this topic in Part II's Q-Learning section.

Evaluation of Three Reinforcement Learning Algos. On Forest Management (Values at convergence)					
Algo.	**State Space**	**Hyperparameters**	**Iter.**	**Reward**	**Time**
VI	500	gamma=0.970	32	37.27	0.0020
	50	epsilon=0.01	32	37.27	0.0020
PI	500	gamma=0.999	20	474.44	0.0010
	50		20	478.78	0.0010
QL	500	gamma=0.999	1440	0.45	0.2874
	50	epsilon=0.80	2131	0.65	0.1665
	5	alpha=0.45 alpha_decay=0.90 epsilon_decay=0.10	10000	8.79	0.1505

TABLE 1. PROVIDED ARE METRICS OBSERVED DURING THE EVALUATION OF THREE REINFORCEMENT LEARNING ALGORITHMS

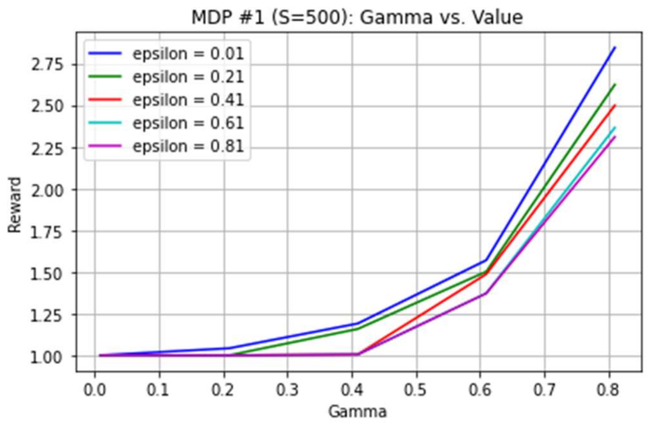

FIGURE 1. . VI - REWARD REACHED MAXIMUM WHEN THE VALUE OF EPSILON IS LOWEST, AND GAMMA IS HIGHEST

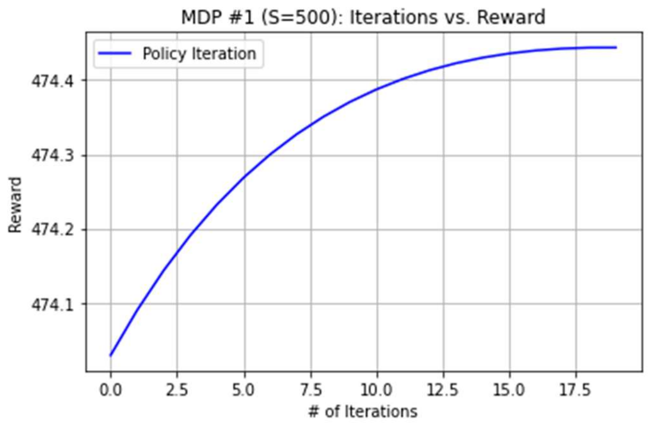

FIGURE 4. PI – REWARD CONVERGES TO 484.44 AT ITERATION=20

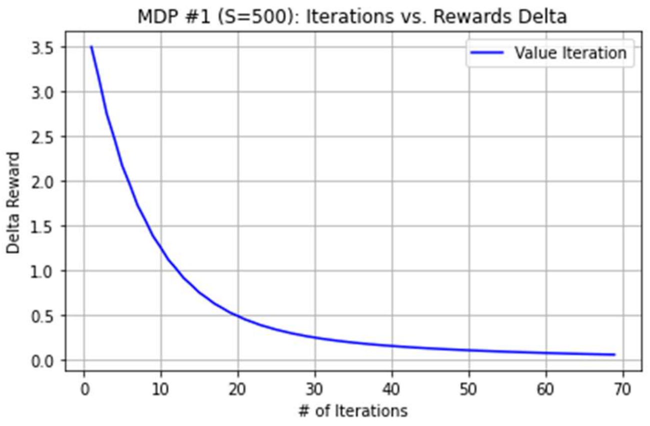

FIGURE 2. VI – DELTA REWARD THRESHOLD IS MET AT ITERATION=32

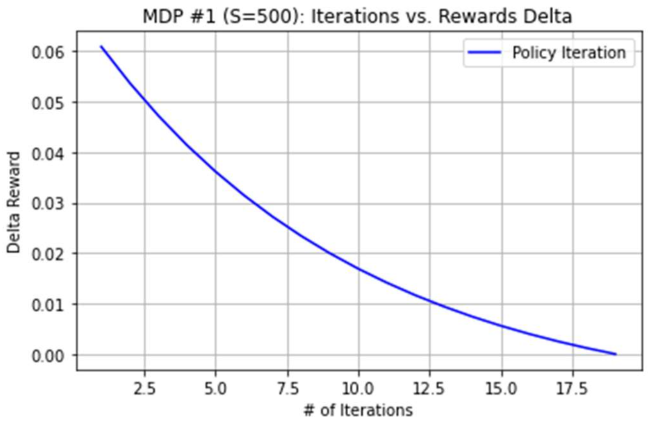

FIGURE 5. PI – DELTA REWARD CONVERGES TO 0.0001 AT ITERATION=20

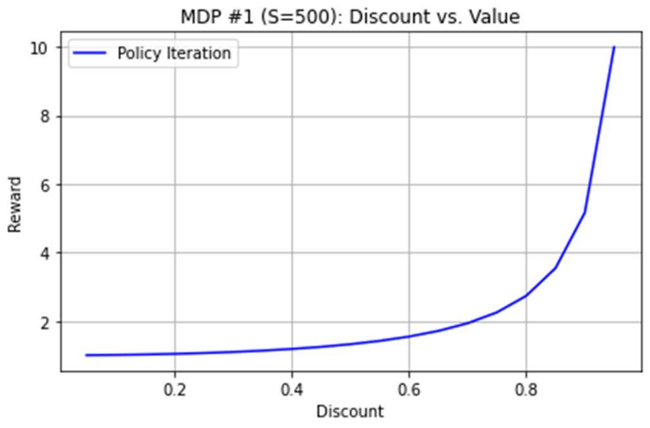

FIGURE 3. PI – REWARD INCREASES AS GAMMA GETS CLOSER TO 1

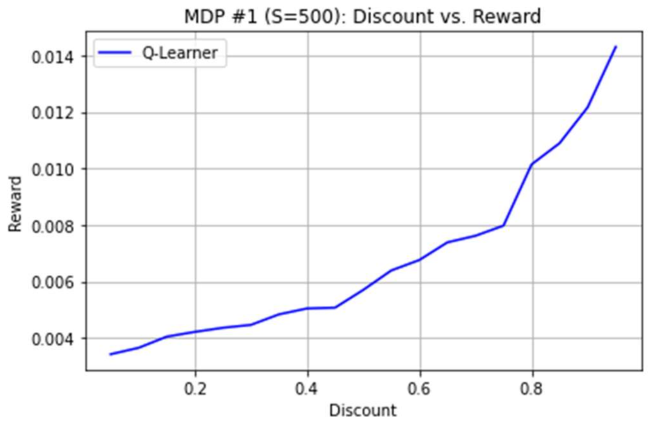

FIGURE 6. QL – REWARD INCREASES WITH GAMMA

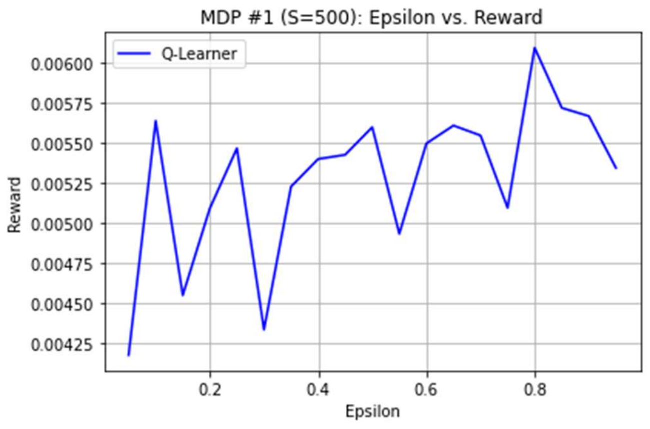

FIGURE 7. QL – REWARD INCREASES WITH EPSILON

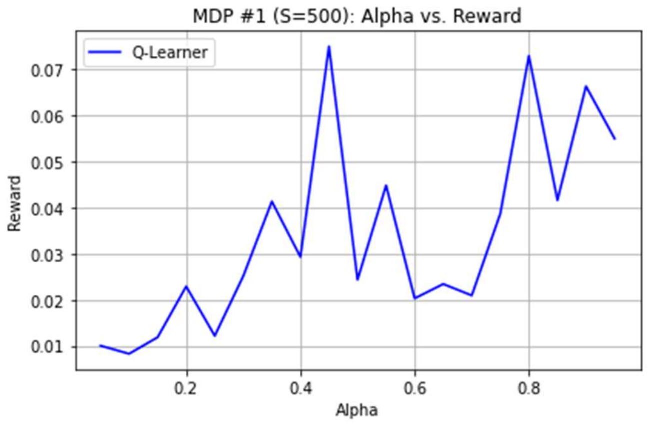

FIGURE 8. QL – REWARD INCREASES WITH ALPHA

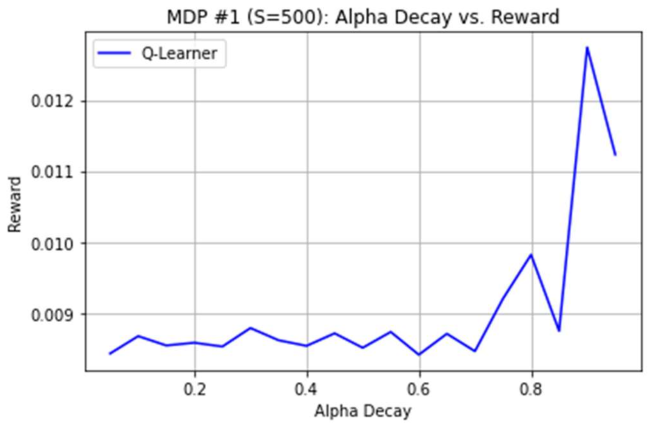

FIGURE 9. QL – REWARD INCREASES WITH ALPHA DECAY

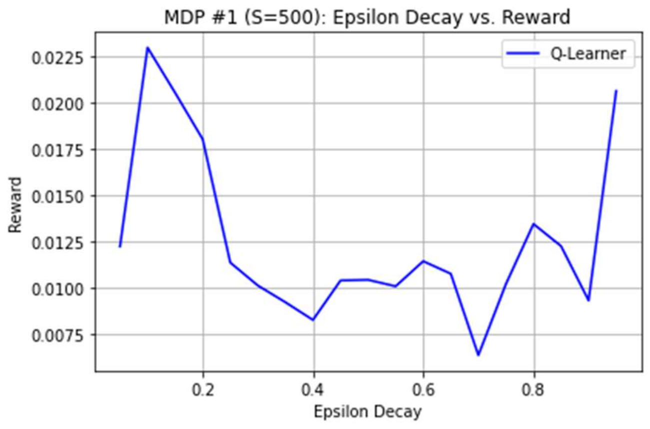

FIGURE 10. QL – REWARD INCREASES WITH VERY HIGH OR VERY LOW EPSILON DECAY

PART II: FROZEN LAKE

In Part II, I continued to explore the three algorithms, VI, PI, and QL. But this time I observed how they behaved when they were applied to MDP #2: Frozen Lake Problem (FLP). Like Part I, for each algorithm, I tuned their hyperparameters (HPs) in accordance with their convergence plots. Keeping consistent with Part I, each algorithm's convergence criterion are described in their respective sections.

Value Iteration (VI) – Like Part I, I first performed VI on FMP using an arbitrary value of gamma=0.5 and default value of all other HPs. And again, to identify patterns of convergence, I generated and analyzed the plot depicting the number of iterations vs. the rewards. Despite FLP's grid-world structure, once I thoroughly examined its state space after being converted to OpenAI_MDPToolbox, I found the differences were not as significant. Therefore, consistent with the behavior I observed with FMP, I hypothesized that as the value of discount gets closer to 1, and epsilon to 0, reward would increase.

To evaluate my hypothesis, I repeated the experiment from Part I. First, I changed the value of gamma while keeping all other HPs the same. Then, I changed value of epsilon while keeping all other HPs the same. Through these two experiments, I confirmed that my hypothesis was correct. Again, to ensure that the relationship is maintained when both gamma and epsilon are changed simultaneously, I plotted 5 lines, each depicting different values of epsilon on a graph which showed the relationship between reward and gamma. As shown in the blue line in *Figure 11*, epsilon=0.01 and gamma=0.8 resulted in one of the highest rewards. It is interesting to note that at gamma=0.8, in addition to epsilon=0.01, epsilon=0.21 and epsilon=0.41 also reached the same highest reward. This is likely due to FLP having a more confined state space, because of its grid-world structure. All in all, this set of experiments confirmed that gamma and epsilon independently had an inverse relationship in respect to reward, regardless of the size of the state space and grid-like structure of the problem.

To fine tune the VI HP's, I once again utilized the iterations vs. rewards convergence and iterations vs. delta of rewards convergence plots. But unlike Part I, due to the manageable number of iterations and obvious convergence, I did not need to plot iterations vs. delta of delta of rewards to determine the index in which the algorithm first crossed the threshold. Therefore, I selected the threshold to be the index where the delta of rewards is less than 0.05. It was interesting to note that due to the small size and confined structure of the problem, convergence occurred much faster, and much more obviously, as indicated by the delta reward value of 0.

Keeping consistent with the general concept behind the convergence criteria from Part I, I determined the index when the value of gamma (0.99999), and epsilon (0. 00001) was set towards their extremes. Then I reversed the two HPs until the index at convergence matched the threshold indicated by the extreme example. As shown in *Figure 12,* this was achieved when epsilon=0.01 and gamma=0.96.

Policy Iteration (PI) – Once again, to kick off the experiment, I performed PI on FML using an arbitrary value of gamma=0.5 and default value of all other HPs. Interestingly, the convergence plot looked almost identical to that of VI. Upon further investigation, even the raw values of the rewards at each iteration was almost identical. My initial theory was that this was again due to the simplicity and rigidity of the FLP. Through this initial observation, I hypothesized that just like VI, as the discount value gets closer to 1, reward and iterations required for convergence would increase. Due to how similar the initial convergence graph for PI behaved compared to that of VI, I was very confident in my hypothesis. As shown in *Figure 13,* results showed that my hypothesis was correct. As the value of discount approached 1, the reward also increased.

To maintain the apples-to-apples comparison, I wanted to adhere to the same convergence criterion. However, without the ability to tune the stopping criterion (epsilon) and due to the simplicity of FLP, by the time convergence occurred, gamma value became too small, resulting in a significantly reduced reward. As a result, instead of determining gamma by finding the convergence point using iterations, I used another method. To fine tune PI HPs, I generated two convergence plots. The first mapped rewards vs. gamma. The second mapped delta rewards vs. gamma as shown in *Figure 14.* To determine exactly how close to 1.0 gamma could approach before reward converges, the x-axis indicated the number of the 9's after 0.9. In other words, the x-axis indicated [0.9, 0.99, 0.999, 0.9999, 0.99999 ...]. Convergence was achieved when gamma=0.99999, resulting in reward=0.687. Using the results, I generated another convergence plot which mapped delta reward vs. iterations. As shown in *Figure 15,* the convergence plot indicated that convergence occurred at iteration=5.

Interestingly, despite using totally different convergence criteria, as shown in *Table 2,* the resulting reward and iterations required for convergence between VI and PI was remarkably similar. Given neither 500 nor 50 state space FMP produced similar values in respect to iterations or rewards (*Table 1*), it is likely that this interesting phenomenon is due to the simplicity of the problem type. Further supporting this explanation is the observation that increasing the difficulty by changing the state space from 4x4 to 8x8, led to a significant increase in the iterations required for convergence and the increase in rewards, especially for PI. Note that unlike VI, since PI utilized a convergence criterion based on reward, by design, its reward is a more accurate estimation of maximum reward compared to that of PI. In other words, due to FLP's strict adherence to a grid world problem design, the problem was made much more rigid. This reduced the choices available to the agent.

Q-Learning (QL) – Please recall that I mentioned I would revisit the 3 theories I posed in Part I. The theories were as follows: 1) The QL HPs must be tuned significantly to solve this problem; 2) QL is not suited to solve a problem of this state size; 3) QL is not suited to solve this type of non-grid world problem. Assuming at least one of the three theories is correct, I should be able to generate a policy resulting in an acceptable reward value when performing QL on a grid-world problem such as FLP.

Like I did in Part I, I first performed QL on FMP using the default value of all HPs across gamma between 0.05 and 1 in 0.05 intervals. Then, just as I did in Part I, I tuned all available HPs, gamma, epsilon, alpha, alpha decay, and epsilon decay and plotted their values against reward to determine their relationships (*Figures 16-20)*. Given theory #1 was proven to be incorrect, and evidence supporting theory #2 was weak, I had high hopes that theory #3 was correct. For theory #3 to be correct, the resulting reward should have been comparable to that of VI and PI, which were 0.619 and 0.687 respectively for FMP with 4x4 state space. Fortunately, as shown in *Table 2,* QL resulted in a reward of 0.326. Therefore, although the reward value resulting from QL was still lower compared to those of VI and PI, it was still within reason.

To further validate test theory #2 using FLP, that the unexpected result was not due to the problem's state space, just as I did in Part I, I increased the state size from 4x4 to 8x8. Given the unimpressive reward when QL was performed on the smaller 4x4 problem, I hypothesized that increasing the state space to 8x8 likely would result in a higher reward, but lower than that of VI and PI's 8x8 counterpart. As predicted, as shown in *Figure 21,* increasing the state size from 4x4 to 8x8 resulted in a higher reward value of 0.401.

Evaluation of Three Reinforcement Learning Algos. On Frozen Lake (Values at convergence)					
Algo	**State Space**	**Hyperparameters**	**Iter.**	**Reward**	**Time (s)**
VI	4x4	gamma=0.96 epsilon=0.01	6	0.619	0.0010
	8x8		14	0.630	0.0050
PI	4x4	gamma=0.99999	5	0.687	0.0030
	8x8	gamma=0.99999	12	0.828	0.0080
QL	4x4	gamma=0.80 epsilon=0.65 alpha=0.50 alpha_decay=0.80 epsilon decay=0.956	14	0.326	0.0010
	8x8		54	0.401	0.0020

TABLE 2. PROVIDED ARE METRICS OBSERVED DURING THE EVALUATION OF THREE REINFORCEMENT LEARNING ALGORITHMS

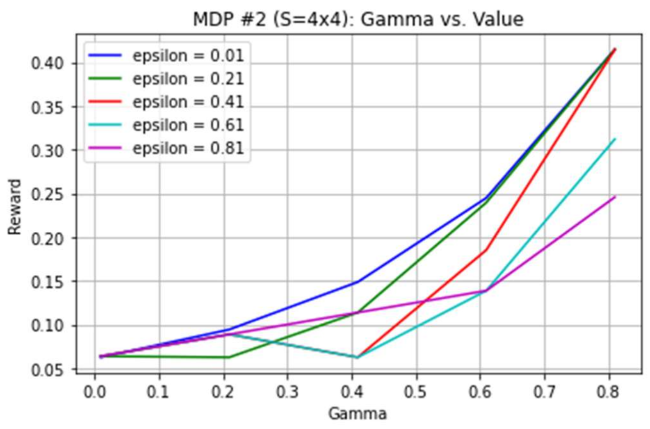

FIGURE 11. . VI - REWARD REACHED MAXIMUM WHEN THE VALUE OF EPSILON IS LOWEST, AND GAMMA IS HIGHEST

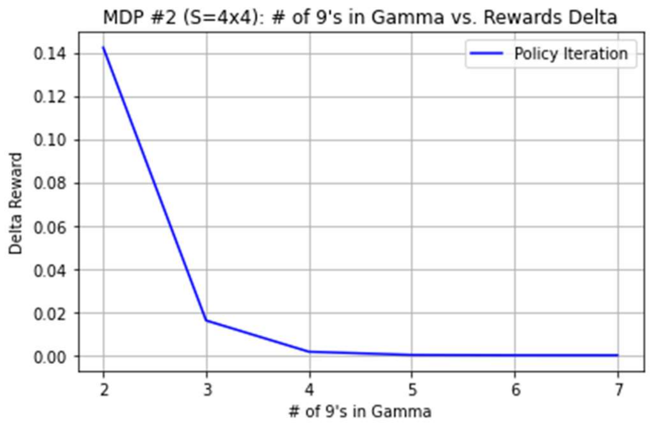

FIGURE 14. PI – DELTA REWARD CONVERGES AT GAMMA=0.99999

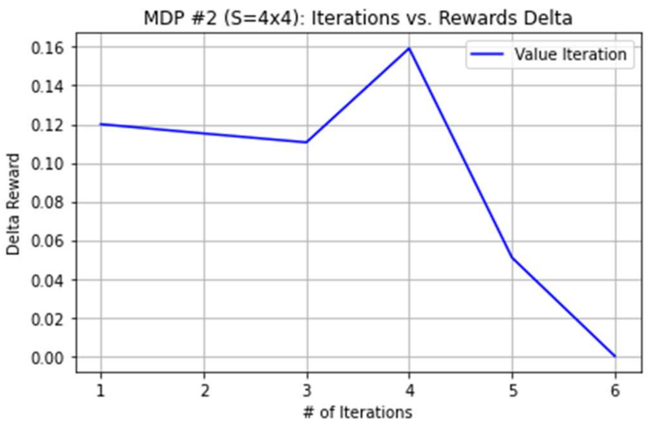

FIGURE 12. VI – DELTA REWARD THRESHOLD IS MET AT ITERATION=6

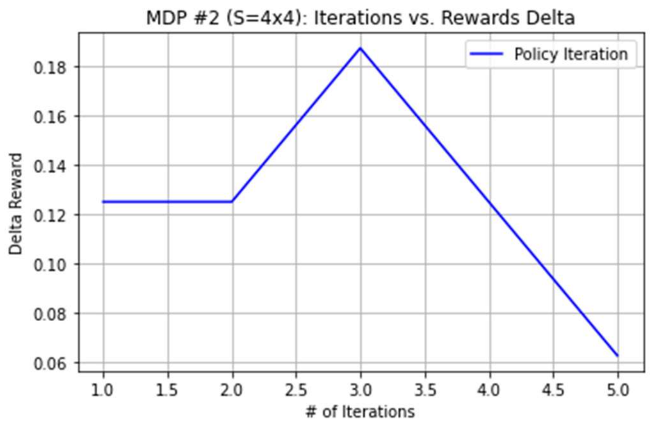

FIGURE 15. PI – GIVEN GAMMA-0.99999, DELTA REWARD CONVERGES AT ITERATIONS=5

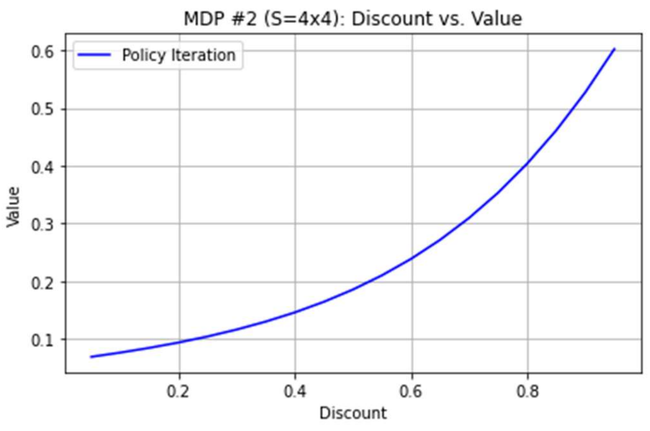

FIGURE 13. PI – REWARD INCREASES AS GAMMA GETS CLOSER TO 1

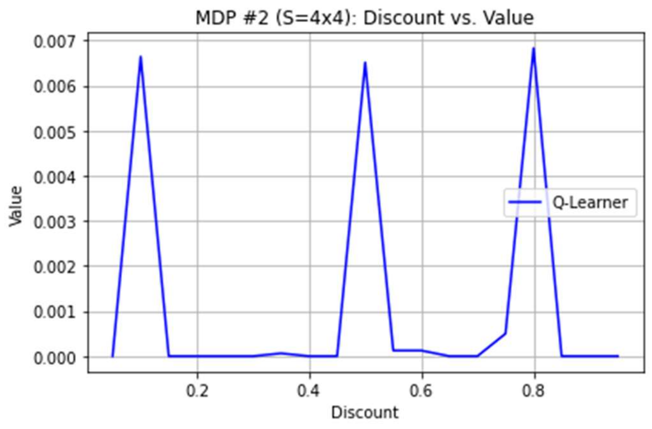

FIGURE 16. QL – REWARD IS HIGHEST AT GAMMA=0.8

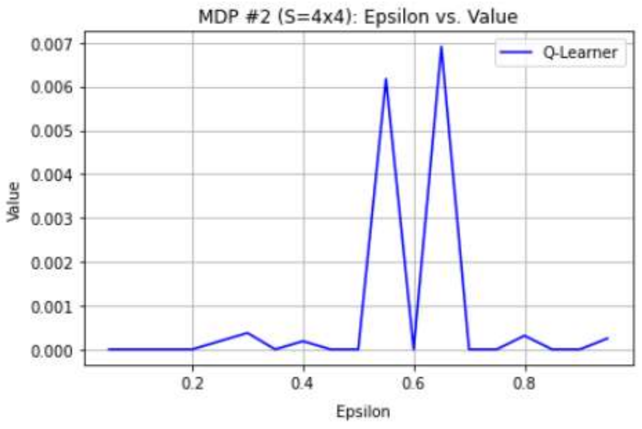

FIGURE 17. QL – REWARD IS HIGHEST AT EPSILON=0.65

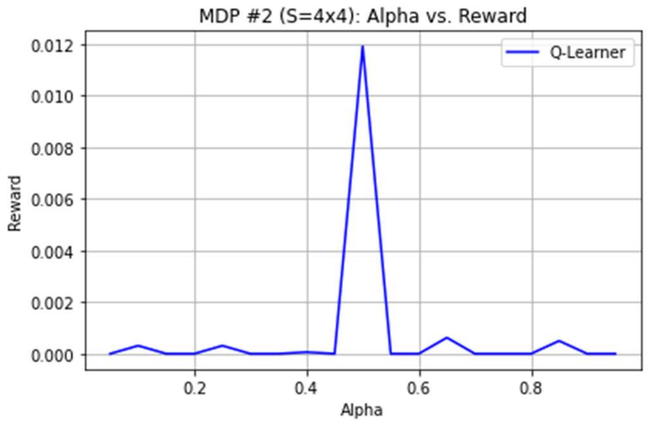

FIGURE 18. QL – REWARD IS HIGHEST AT ALPHA=0.5

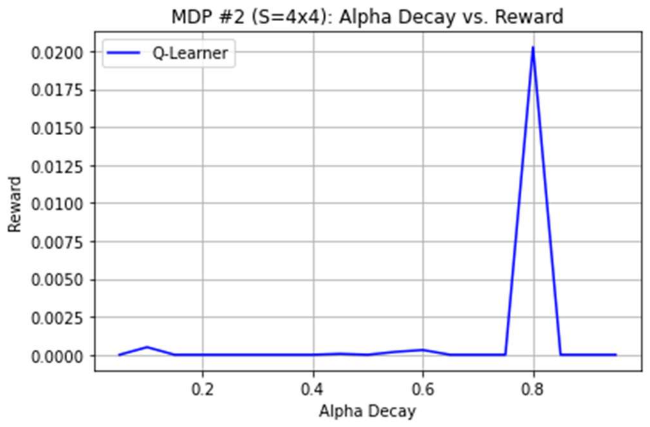

FIGURE 19. QL – REWARD IS HIGHEST AT ALPHA_DECAY=0.8

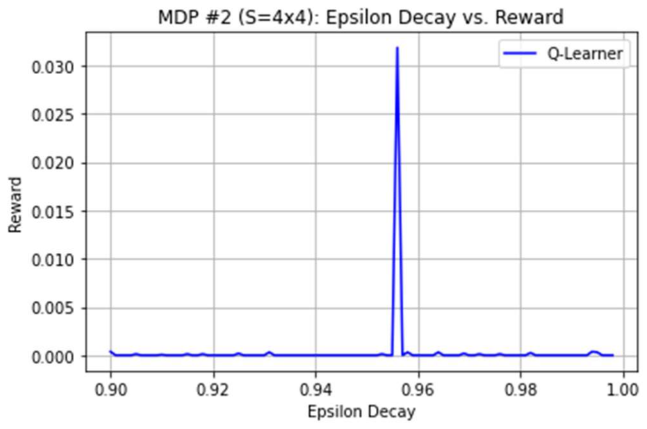

FIGURE 20. QL – REWARD IS HIGHEST AT EPSILON_DECAY=0.956

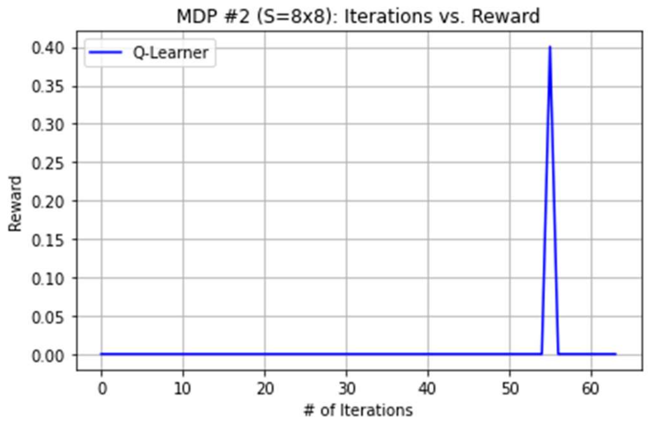

FIGURE 21. QL – REWARD IS HIGHEST AT ITERATION=54

PART III: COMPARISON & ANALYSIS

In Part III, I delve into my findings to analyze, make comparisons, and draw conclusions about the three algorithms: VI, PI and QL. I will also discuss how each were affected by the types and sizes of the two MDPs: FMP and FLP.

Performance – When performance was measured using final value/reward, policies generated by PI always scored the highest, followed by VI, then QL. It is worth noting that due to the convergence criteria that I chose, the value/reward listed in *Table 1* and *Table 2* cannot be compared at face value. In other words, they did not converge to the same answer due to the convergence criteria I chose, which factored in value/rewards, iterations, and discount value. However, the number of iterations and time can be considered, to make a fairer comparison.

For example, PI was able to attain a much higher reward value in far less iterations to solve both the non-grid and grid world problems. In addition, PI was able to converge much quicker in respect to wall-clock time when solving FMP. Although this was not the case for FLP, it can be argued that this was due to significantly higher gamma value assigned to PI. Since gamma was much closer to 1 in this case, with respect to the Bellman equation, the rewards would have a larger effect from the start and continue to have a larger effect throughout the process, which explains the higher wall-clock time. It was interesting to see that the similar reward values between VI and PI for FLP was very aptly reflected in their optimal policy visualizations (*Figure 25* and *Figure 26).*

Therefore, according to the result of my experiment, PI outperformed both VI and QL in respect to value/rewards, iterations required to converge, and time required to execute the algorithm. QL, on the other hand, took significantly longer in respect to both iterations required to converge and time required to execute the algorithm. As shown by the results of the experiment, as well as the visualization of the optimal QL policies in *Figure 24* and *Figure* 27, it seems utilizing the probability distribution of the MDP gives both PI and VI a significant advantage over non-model-based algorithms such as Q-Learner [6].

Number of States – The effect of the number of states was explored at length when I tested theory #2. As shown in *Table 1* and *Table 2,* number of states of FMP and FLP had a noticeable effect on all three algorithms. When state space was decreased from 500 to 50 to 5 in FMP and increased from 4x4 to 8x8 in FLP, the number of iterations required to converge, and time required to execute the algorithm increased significantly. It was especially interesting to see that when the state space of FMP was reduced to 5, QL was able to generate a policy that performed significantly better than when the states were 50 and 500. As the number of states increased, the computational complexity of the algorithms increased, leading to worse performance across all three-performance metrics.

Type of Problem – The effect of the type of problem (grid vs. non-grid) was also explored extensively when I tested theory #3. As shown in *Table 1* and *Table 2,* the types of problem also had a significant effect, but only for QL. Regardless of problem type, VI and PI seemed unaffected as they both performed well. However, QL clearly struggled to generate a policy that could solve FMP. As noted previously, I believe this can be attributed to the simplicity of the FLP. The rigidity enforced by the grid-world nature of FLP makes the problem much more tractable.

Conclusion – In conclusion, through this assignment, I was able to witness first-hand, the power and utility of Markov Decision Processes. But as with any concepts in machine learning, there are always tradeoffs. Without being provided information such as probability distributions of the MDP, by default, the agent would have to make do with using non-model-based algorithms such as Q-Learner.

There does seem to be very little cost associated with Policy Iteration algorithm compared to Value Iteration algorithm. In most cases, PI outperformed VI in respect to total rewards gained, iterations required to converge and time. The exception at least from my experiment were cases when gamma (discount) value was extremely high. In that case, VI outperformed PI in respect to time.

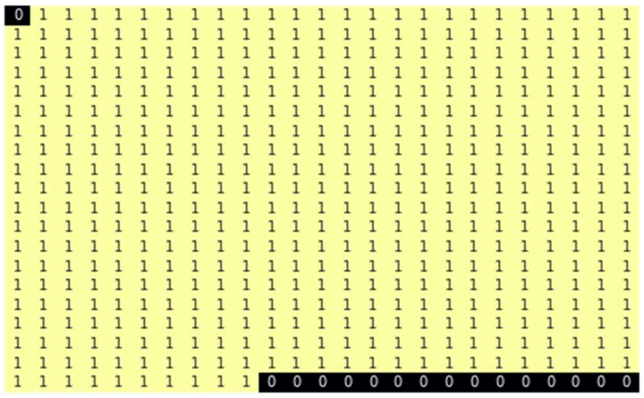
FIGURE 22. OPTIMAL VI POLICY: FMP(S=500)

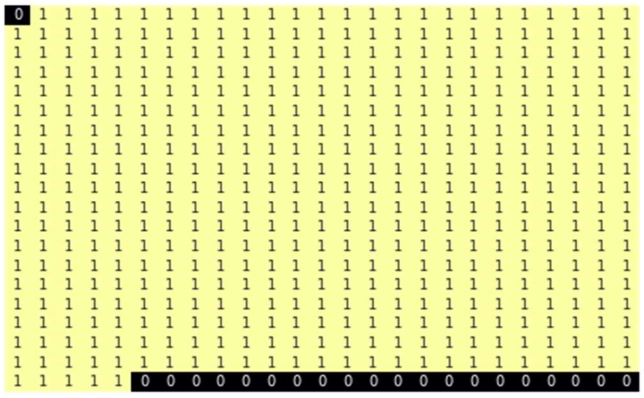
FIGURE 23. OPTIMAL PI POLICY: FMP(S=500)

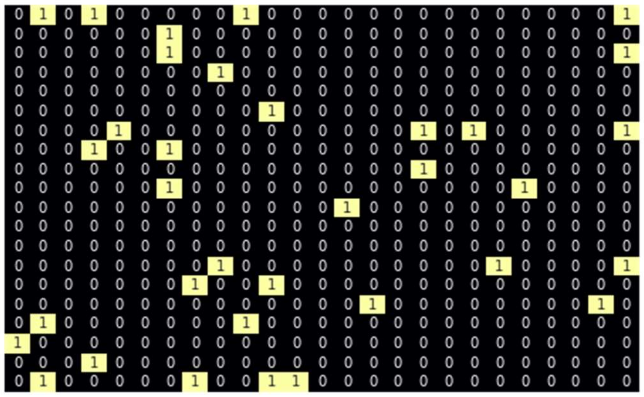
FIGURE 24. OPTIMAL QL POLICY: FMP(S=500)

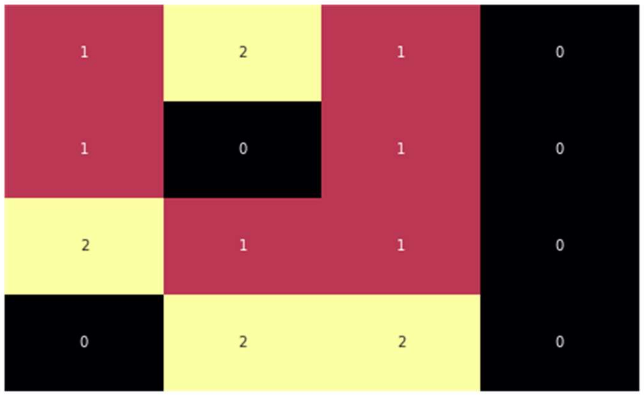

FIGURE 25. OPTIMAL VI POLICY: FLP(S=4X4)

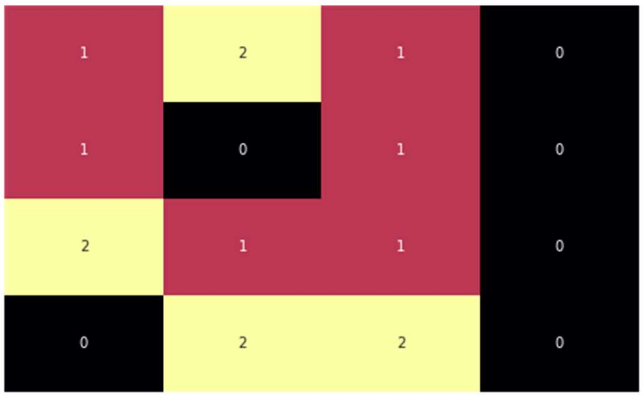

FIGURE 26. OPTIMAL PI POLICY: FLP(S=4X4)

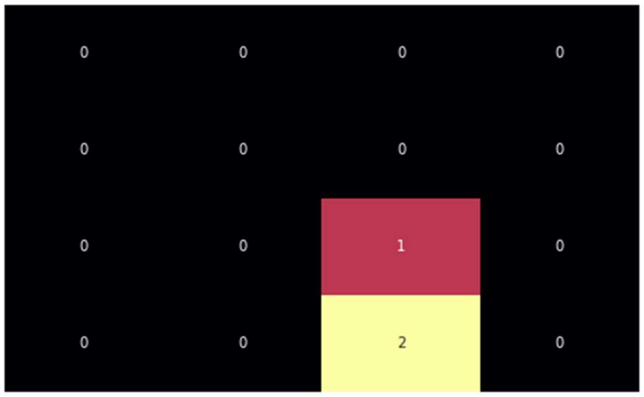

FIGURE 27. OPTIMAL QL POLICY: FLP(S=4X4)

REFERENCES

[1] Hiive. "Hiivemdptoolbox/Example.py at Master · Hiive/Hiivemdptoolbox." GitHub, 19 Apr. 2021, https://github.com/hiive/hiivemdptoolbox/blob/master/hiive/mdptoolbox/example.py.

[2] "Frozen Lake#." Frozen Lake - Gym Documentation, https://www.gymlibrary.dev/environments/toy_text/frozen_lake/.

[3] Hiive. "Hiivemdptoolbox/Openai.py at Master · Hiive/Hiivemdptoolbox." GitHub, 18 Apr. 2021, https://github.com/hiive/hiivemdptoolbox/blob/master/hiive/mdptoolbox/openai.py.

[4] "Source Code for Mdptoolbox.mdp." Mdptoolbox.mdp - Python Markov Decision Process Toolbox 4.0-b4 Documentation, https://pymdptoolbox.readthedocs.io/en/latest/_modules/mdptoolbox/mdp.html#ValueIteration.

[5] Arslán, et al. "What Is the Difference between Value Iteration and Policy Iteration?" Stack Overflow, 1 Aug. 1963, https://stackoverflow.com/questions/37370015/what-is-the-difference-between-value-iteration-and-policy-iteration.

[6] Tokuç, A. A. (2022, November 9). Value iteration vs. policy iteration in reinforcement learning. Baeldung on Computer Science. Retrieved November 19, 2022, from https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration