Project4 Report - Markov Decision Processes and Reinforcement Learning

Kyle Grace
kgrace6@gatech.edu

***Abstract*—**Markov Decision Processes (MDP) are a framework for modeling decision making models which can be trained using Reinforcement Learning. They are discrete processes which include a state, a discrete number of actions, and state transitions based on actions. By adding a reward function to the states, a reinforcement learning model can develop a policy, or set of best actions for each state. Three mothods for developing this policy, Value Iteration, Policy Iteration, and Q Learning, will be discussed in this paper.

1 MDP PROBLEMS

The two MDP problems being used to test RL techniques are the Forest Management problem, and the Frozen Lake problem

1.1 Forest Management

The Forest Management scenario involves a forest which is managed by 2 actions, "wait" or "cut". Each year, an agent must decide whether to cut the forest, or wait. The objectives are to maintain the forest for wildlife, or make money on the cut wood. There is also a possibility that a fire burns the forest and is reset to an initial state. For this project, there are 625 states being used so show how large state spaces affect different RL techniques.

1.2 Frozen Lake

The Frozen Lake problem is a "grid-world" problem where the objective is to navigate over a frozen lake. The lake has a goal location, holes that the agent can fall in, and some probability that the action chosen by the agent will not succeed. For example, if the agent chooses to move left, there is also a random chance that they will instead move up or down. This problem includes a smaller number of states, but the agent must learn to navigate even with random mistakes if motion.

2 LEARNING ALGORITHMS

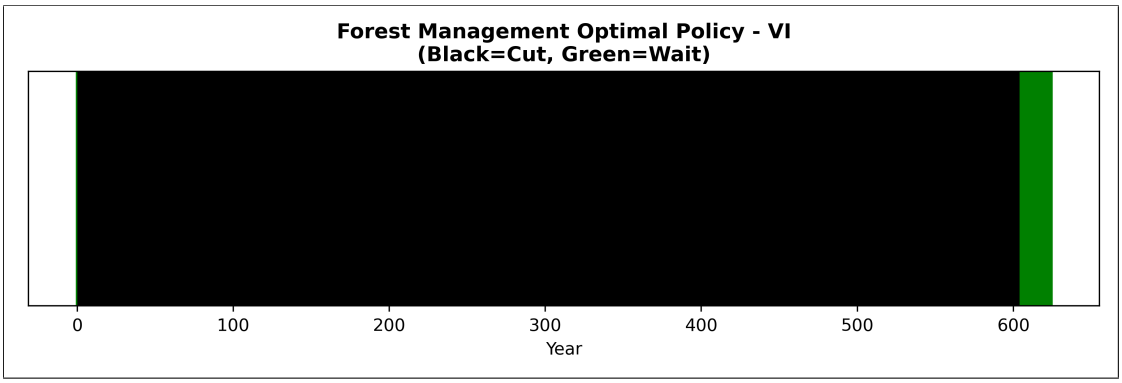
There are 3 learning techniques being evaluated: Value Iteration (VI), Policy Iteration (PI), and QLearning (QL). Each of these methods attempts to determine the optimal policy, or mapping of state to action that will result in the greatest reward.

VI attempts to find the optimal policy by determining the value of each state based on the reward and possible actions at each state. The utility at each state is determined directly using the Bellman equation. Convergence is defined as when the new Value function on some iteration improves by less than some chosen epsilon value.

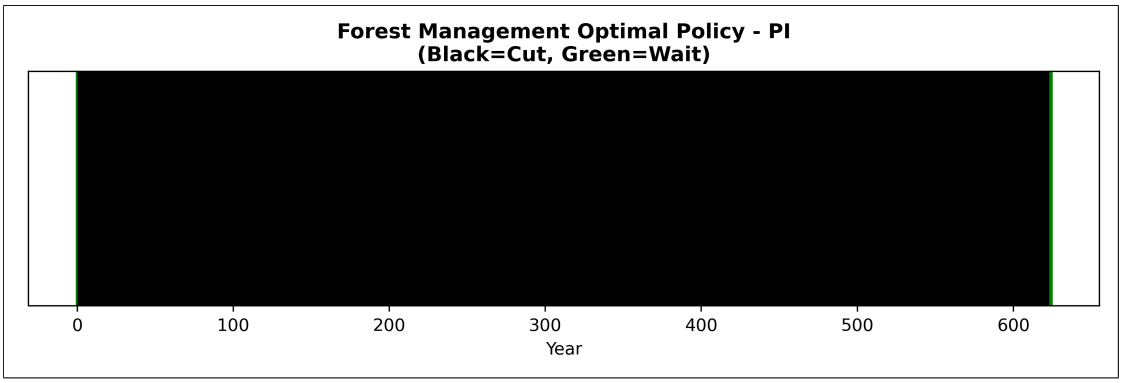
PI starts with a random policy, then that policy is iteratively evaluated and improved. Rather than iterating of the value of each state, the policy is evaluated and improved directly. Generally speaking, PI should converge faster and in fewer iterations than VI. Convergence is defined as when the policy update steps no longer results in any improvement.

QLearning is a model-free algorithm which seeks to learn the value of an action in a particular state. It differs from VI and PI in that the transition probabilities and rewards are not known, so rather than iterating through a known model, it must also discover the model in a sense. Since there is less information known to the learner, it is much slower and less effective than either PI or VI, but is a more realistic situation and more flexible solution. Convergence is defined as when the actual reward from the policy no longer improves.

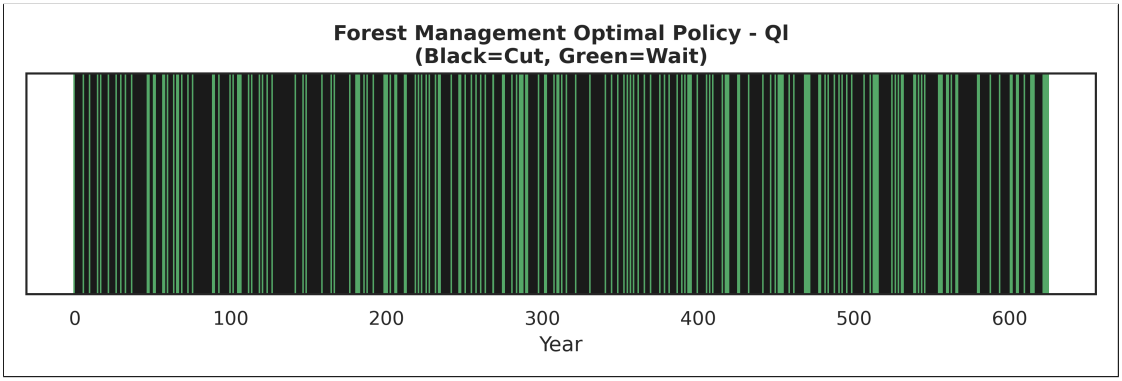
3 FOREST MANAGEMENT RESULTS



(a)



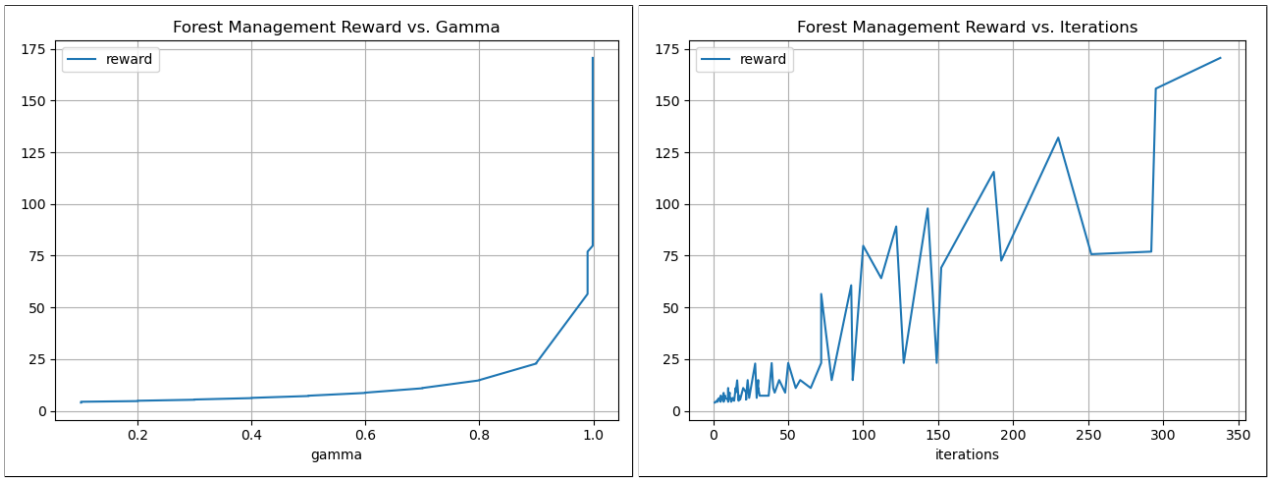
(b)



(c)

Figure 1—The optimal policy obtained by each method. Each step is represented by a black like for "cut" or green line for "wait". (a) Value Iteration (b) Policy Iteration (c) Q Learning

The Forest Management problem was solved very easily by both VI and PI, but QLearning struggled to converge on a decent solution. Figure 1 shows the policies of each, where black means to cut, and green means wait for that year. Interestingly, PI and VI had the exact same solutions in previous iterations, but show slight differences in the final outputs, but I was not able to determine what changed that caused the policies to be different. Either way, the policy is nearly the same, with the general idea being that one should cut every year in the beginning and wait at the end. QL did not ever converge on a clean policy



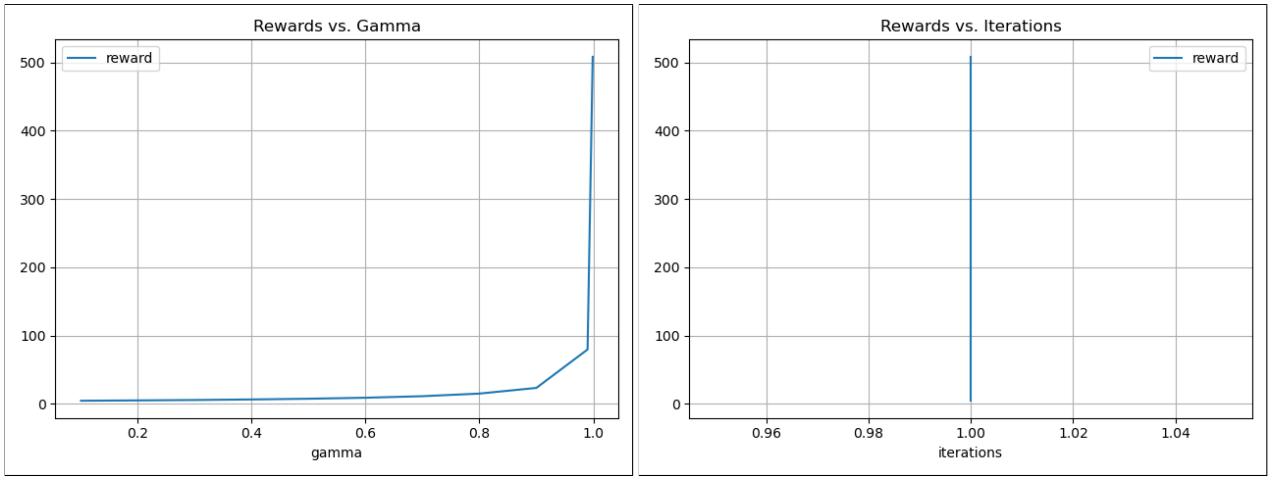
(a) (b)

Figure 2—(a) Reward vs Gamma (b) Reward vs Number of Iterations

3.1 Value Iteration

The optimal VI model occurred with a gamma value of 0.999 and epsilon of 1e-12. This also required a total of 338 iterations to converge, which happens to be the largest required for any of the parameter settings. The runtime for this run was 0.23 seconds. The total reward of the model increases with both higher gamma and number of iterations. Higher gamma means that a higher weight is placed on long-term gain, which makes sense considering the total reward includes all states and not just the current one.

3.2 Policy Iteration



(a) (b)

Figure 3—(a) Reward vs Gamma (b) Reward vs Number of Iterations

The optimal PI model occured with a gamma of 0.999. For reasons I was not able to determine, it only took a single iteration. In fact, it didn't matter what gamma value was used, it always took just a single iteration to reach the optimal policy. After trying to figure out why for some time, nothing changed, so I can only assume that this just happens to be a problem that is very well suited for Policy Iteration. While it only took a single iteration, that iteration took 4.89 seconds for the optimal run. On average, the runtime was much faster for PI than VI, but was much slower for very high gamma values.

3.3 Q Learning

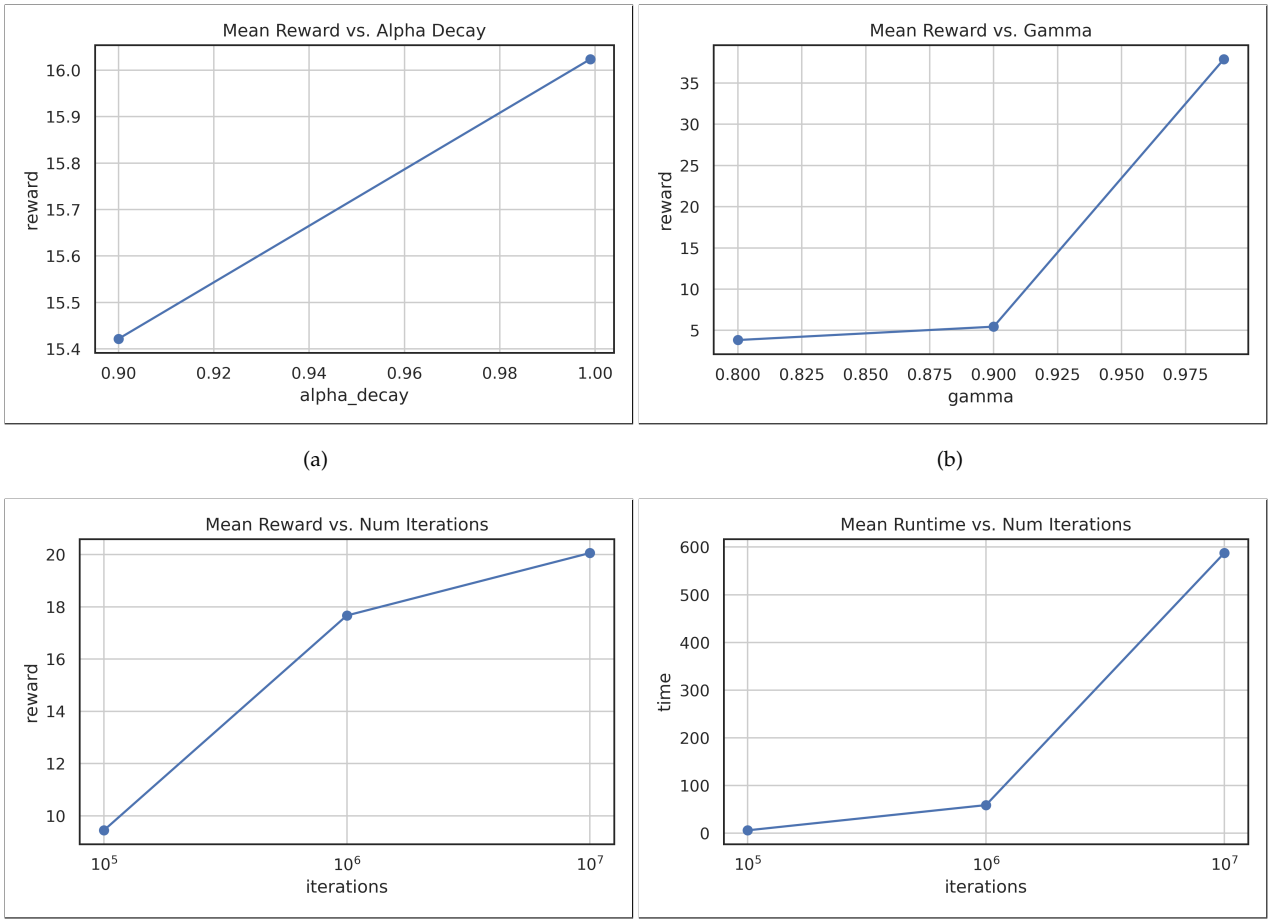


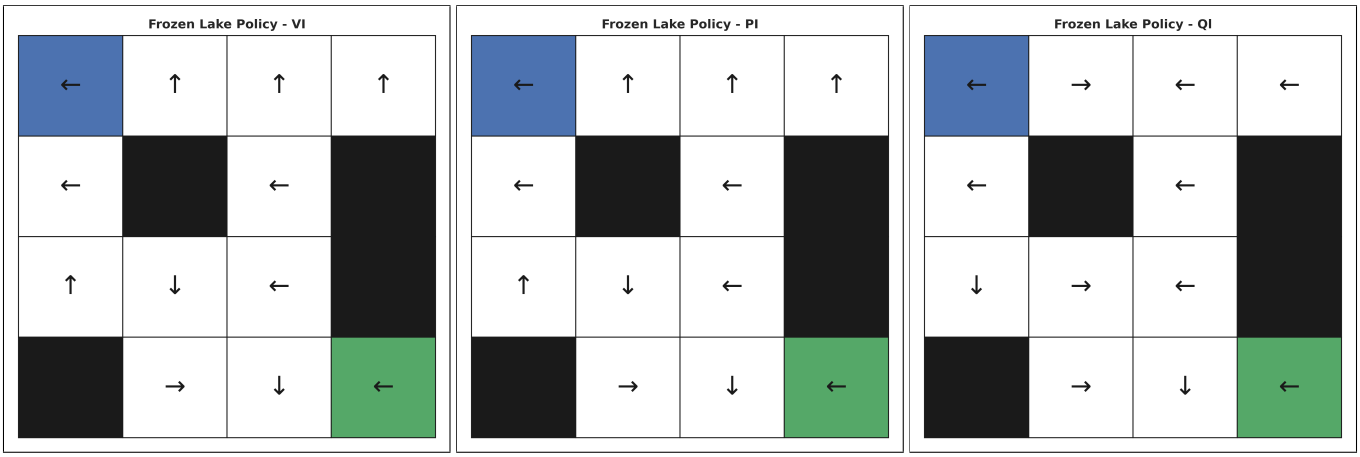
Figure 4—(a) Reward vs alpha decay (b) Reward vs gamma (c) Reward vs Number of Iterations (d) runtime vs number of iterations

As mentioned before, Q Learning was not able to converge on an optimal policy for this problem. The issue is that since the Q Learner is model agnostic, it would need to iterate through each state and each state transition in order to determine all possible rewards. Since this problem has 625 states, this becomes a massive problem.

The best solution found had a reward value of 47.68, and came when using gamma = 0.99, alpha = 0.01, alpha decay = 0.9, epsilon decay = 0.999, and 10000000 iterations. Since the reward is still trending upward with increased number of iterations, it's likely that a better solution could have been found with more iterations, but it would have taken longer than I was able to commit to the experiment. Suffice it to say, the Q Learner was able to find a decent policy, but the size of the state and transition space made it extremely challenging for the model to converge.

4 FROZEN LAKE RESULTS

The Frozen Lake problem was solved easily by both PI and VI, which is no surprise considering this was also true for the Forest Management problem which was far more complex. However, Q Learning fared much better this time than with the Forest problem. You can see in Figure 5 that both PI and VI have the same solution, and the QL solution is only different in a few cells. It is also worth noting that the cells which are different are at the upper edge where the agent is less likely to be located, and the desired action is a bit more ambiguous. As a result, it makes sense that those locations, if any, would be harder to reach optimality.



(a) (b) (c)

Figure 5—The optimal policy obtained by each method. (a) Value Iteration (b) Policy Iteration (c) Q Learning

4.1 Value Iteration

The optimal settings for VI in the Frozen Lake problem are the same as the Forest problem, 0.999 for gamma and 1e-12 for epsilon. This makes sense considering the process is the same even though the problem is different. However, it took 1221 iterations to converge, and only 0.07 seconds. Just as with the Forest problem, higher gamma results in higher total reward. Since the reward is based mostly off of finding the goal and not falling into the pit, it makes sense that weighting future actions would result in a better total reward, as well as more steps. The policy that has the highest likelihood of reaching the goal is likely to be very conservative, which means there would be a lot of safe movements that may not be productive, but will not result in failure, wherase a more aggressive approach would get to the goal faster, but have a higher risk of accidentally failing.

4.2 Policy Iteration

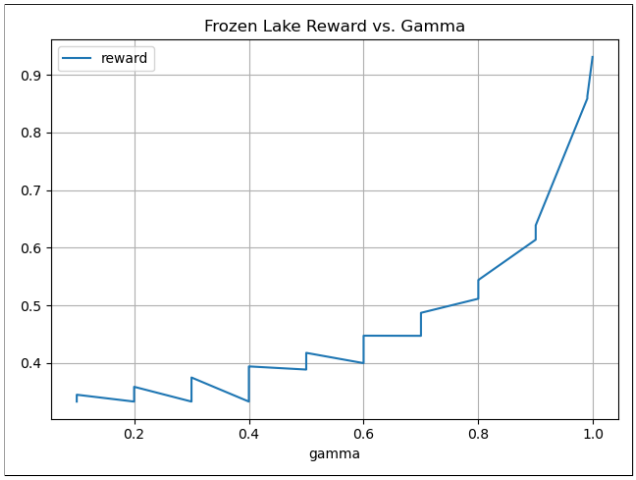
The optimal value for gamma was again 0.999, but this time there is some variation in iterations taken. The optimal run took 6 iterations and 0.01 seconds. This closer resembles my expectations from the beginning, that PI would require less time and fewer iterations than VI while still reaching the optimal policy.

The trends for gamma, number of iterations, and number of steps are essentially the same between PI and VI. The policy with the highest success rate is more conservative because it weights future actions more heavily, and as a result, takes a more conservative approach that is avoids holes in the lake.

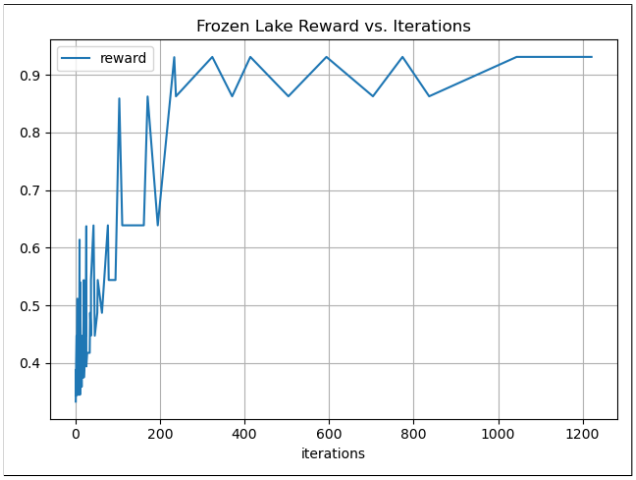
4.3 Q Learning

The best reward reached by the Q Learner was 0.57 compared to 0.93 for both PI and VI. Just as with the Forest case, the success rate is still increasing with higher iterations, and time was a limiting factor, so it's possible that an optimal policy could have been found with more iterations.

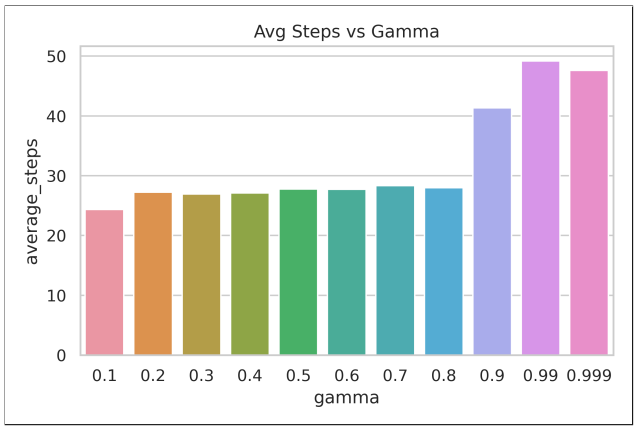
The best parameters for the Q Learner were gamma = 0.99, alpha = 0.1, alpha decay = 0.9, epsilon decay = 0.9, and 1000000 iterations, had a success rate of and it took about 34 seconds to complete. This problem seemed to be much more subject to randomness both in how well the model converged, and in how it performed in general. Even the "optimal" policy was only able to reach the goal successfully about 84% of the time, and it required many extra steps to do so. The best QLearning model was able to reach the goal 37% of the time, but did so in less than half the steps that the optimal policy took.



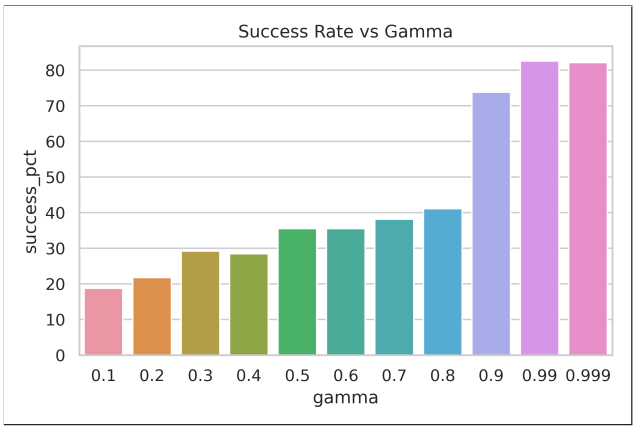
(a)



(b)

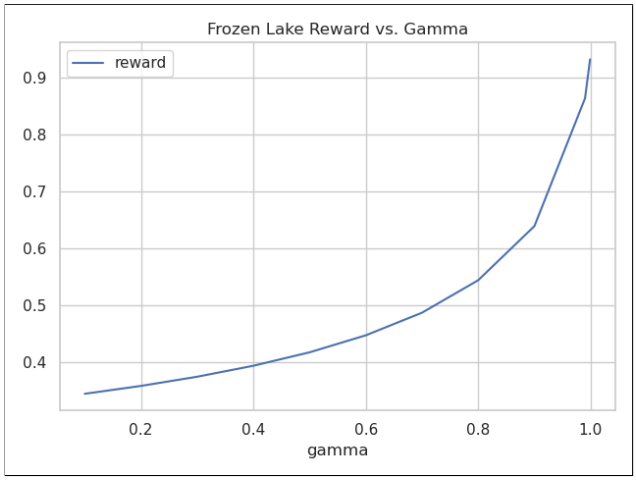


(c)

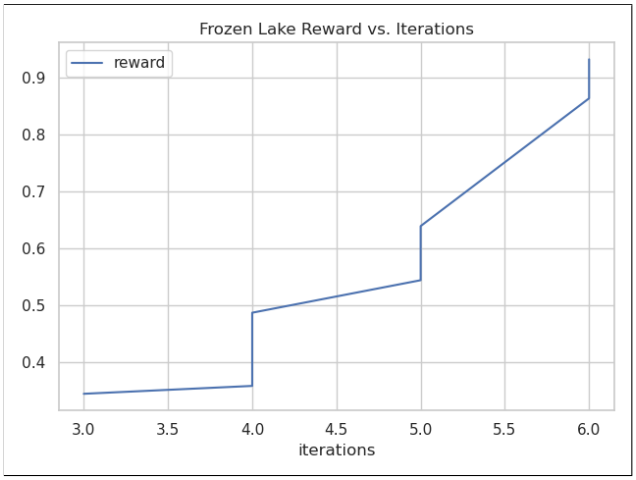


(d)

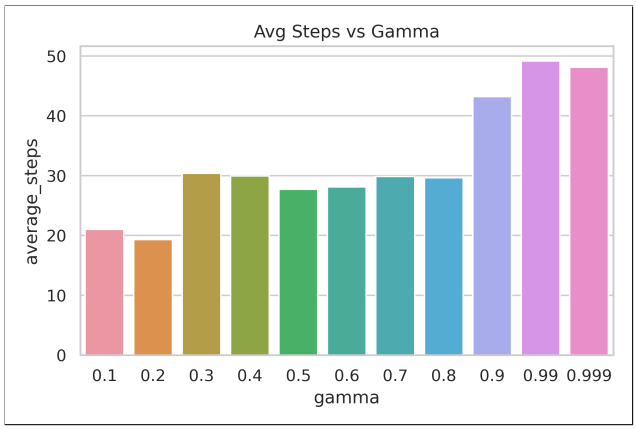
Figure 6—(a) Reward vs Gamma (b) Reward vs Number of Iterations (c) Gamma vs average number of steps (d) Gamma vs Success Rate



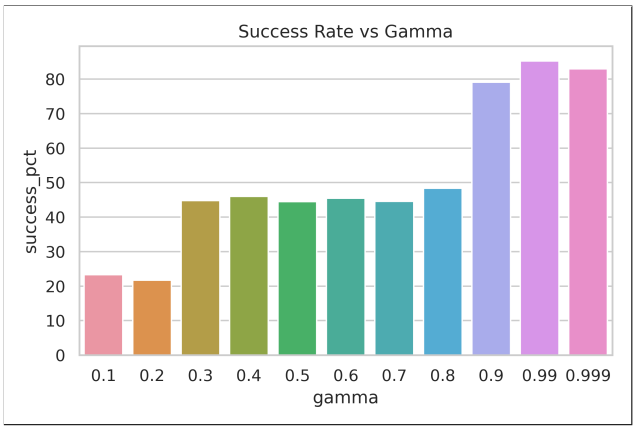
(a)



(b)



(c)



(d)

Figure 7—(a) Reward vs Gamma (b) Reward vs Number of Iterations (c) Gamma vs average number of steps (d) Gamma vs Success Rate

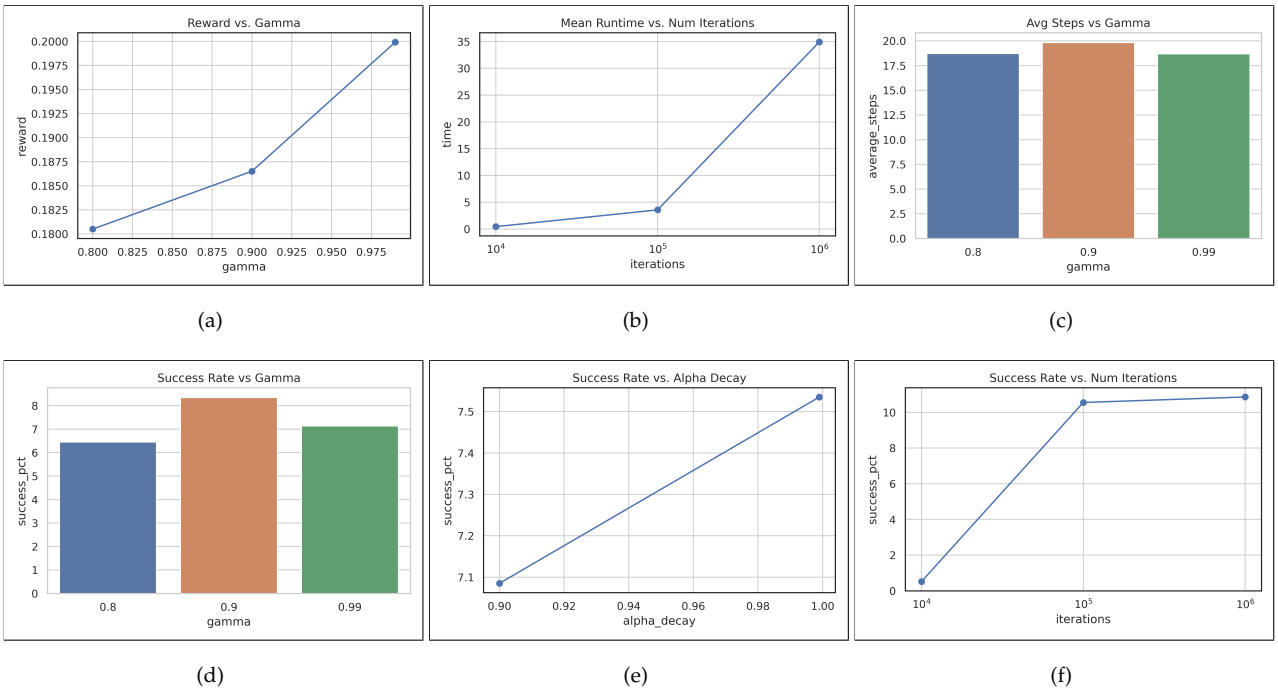


Figure 8—(a) Reward vs Gamma (b) Time vs number of iterations (c) Gamma vs Average number of steps (d) Gamma vs success rate (e) Alpha Decay vs Success Rate (f) Number of Iterations vs Success Rate

Most of the parameter trends make sense, but some are not entirely clear. Just as before, higher gamma and higher iterations result in higher reward (and longer runtimes and number of steps), but I would have thought there would have been a linear correlation between rewqard and success rate. However, increased reward from high gamma does not necessarily result in a higher success rate. Aside from randomness, I'm not sure why this would be the case.

5 CONCLUSION

Markov Decision Processes are a valuable tool in modeling real-world scenarios and training agents which are meant to act in the world. Policy Iteration and Value iteration both converge very quickly to optimal policies, but they require a knowlege of the environment that may not be available in the real world. PI is noticeably faster than VI in both runtime and total number of iterations needed to converge.

Q Learning is a more flexible method since it doesn't require any knowledge of the environment and can learn that as it is run. The downside is that there is some information that is much more challenging to learn, and very complex problem spaces cannot be traversed to the extent that would be needed to get a full understanding of the environment. As a result, Q Learning seems to be well suited for situations where the problem space is limited, or where a true optimal policy is not required. Since it's not realistic to traverse all possible states and transitions in some cases, Q Learning will miss information that PI and VI would capture. However, by adding sufficient iterations and randomness to the action choices within the Q Learner, usable policies can still be obtained.

REFERENCES

[1] Mitchell, Tom M. (2013). *Machine Learning*. McGraw-Hill.