**Markov Decision Processes & Reinforcement Learning**

Anurag Sahoo  
[asahoo41@gatech.edu](mailto:asahoo41@gatech.edu)

GT ID: 90368005

**Abstract**— This paper deals with solving two MDP problems of different sizes using Value Iteration, Policy Iteration and Q Learning. I divided this paper into 3 different sections. First section delves into the two MDP problems including a grid and non-grid world. In the second section, we go deep into the algorithms we used to solve these problems. Third, we use Value Iteration, Policy Iteration and Q Learning to solve Frozen Lake of different sizes. Fourth, we use those three techniques to solve Forest Management problem of different sizes.

1. **Introduction to Problems**:

Frozen Lake: Frozen Lake is a grid world problem where an agent moves from start to goal without falling into holes by walking over the frozen block. There lies an element of stochasticity as the agent may not always move in the desired direction because of the slippery nature of frozen lake. The agent can take either of UP, DOWN, LEFT and RIGHT actions. Because of the stochasticity nature of the environment, the agent moves in the intended direction with a probability of 0.33 and ends up on either side with a probability of 0.33 each. The reward for reaching the goal is 1. I have used 8x8 and 15x15 map for small and large grid worlds respectively.

Forest Management: It is a non-grid problem where the forest grows up to maximum and the agent has the option to wait or cut. The action is decided with an objective to maintain an old forest for wildlife and to make money selling cut wood. There are certain rewards associated with each action. Also, there is a probability associated with the problem that the forest might catch fire. So, the agent must take a call if it wants to wait and go to final state for larger rewards with the risk of forest catching fire. I used two sets for this problem. The smaller one has 20 states, with reward of 10 for Wait when the forest is in its oldest state, reward of 2 for Cut when the forest is in its oldest state and a probability of 0.1 of forest catching fire. The smaller one has 400 states, with reward of 25 for Wait when the forest is in its oldest state, reward of 8 for Cut when the forest is in its oldest state and a probability of 0.05 of forest catching fire.

Why are these problems interesting?

These problems are interesting as they represent real world situations. The grid world problem Frozen Lake can be thought of analogous to robot navigating to move the desired box from start to goal in a warehouse setup. A positive reward would be given if the robot successfully moves the boxes to destination. There would be negative reward if the robot bumps into an obstruction. These problems can automate a whole lot of tasks and are used predominantly in the logistics space. The non-grid problem (Forest Management) is a typical example of risk-reward trade off problem. It finds application in the areas of Stock Market Trading where the user takes a call on each trading days if he/she exits the position and realizes the gain/loss or holds the position with the hope to gain more rewards but there is a certain element of risk of losing more.

1. **Techniques to solve MDP Problems**

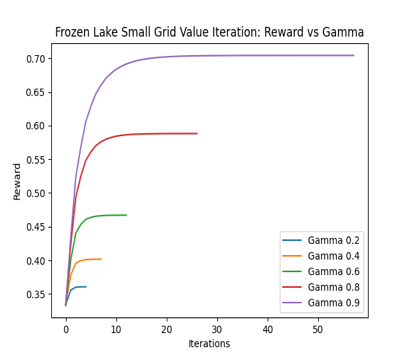
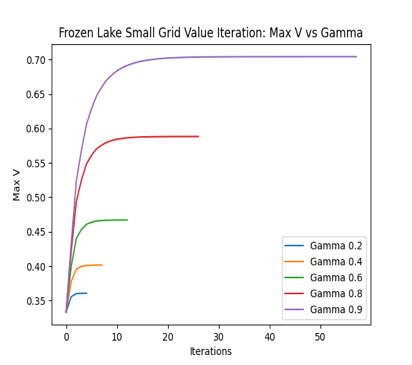
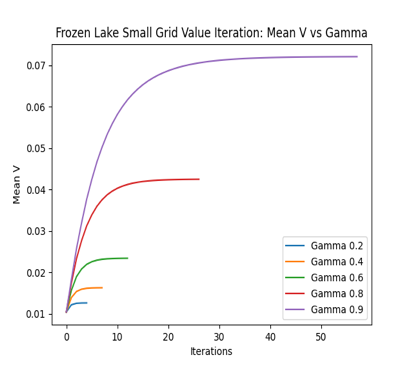
Value Iteration**:** Value Iteration is a model-based learner in which we start with a random value function and find an optimal value function using an iterative process. The optimal policy can be generated from the optimal value function using the Optimality Bellman Operator. Hence, Value Iteration includes finding optimal value function and one policy extraction. Value Iteration guarantees convergence.

Policy Iteration: Policy Iteration is a model-based learner in which we start with a random policy and find the value function of that policy (policy evaluation), and then find improved policy based on the previous value function, and so on. The value function of a policy can be generated using Bellman Operator. Policy Iteration involves policy evaluation and policy improvement. Policy Iteration guarantees convergence with lesser iterations compared to Value Iteration.

Q Learning: Q Learning is a model-free learner in which we do not know the transition probabilities and rewards beforehand. Agent learns through the experience of interacting with environment. It is an off-policy temporal difference control algorithm. Q Learning is a temporal difference algorithm meaning the predictions are re-evaluated after taking a step. It is an off-policy algorithm in which it estimates the state-action pairs based on the greedy policy, independent of agent’s actions. Q Learning converges only under certain conditions: learning rates approach zero and each state-action pair are visited infinitely.

1. **Frozen Lake**

Value Iteration, Policy Iteration and Q Learning were implemented in the Frozen Lake environment of size 8x8 (small grid) and 15x15 (large grid).

Chart

Description automatically generated

Fig 1: Value Iteration Convergence plots for Frozen Lake small grid

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

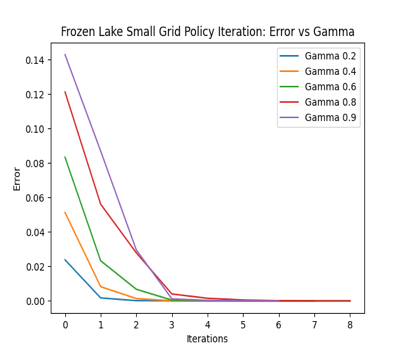
Description automatically generated

Fig 2: Policy Iteration Convergence plots for Frozen Lake small grid

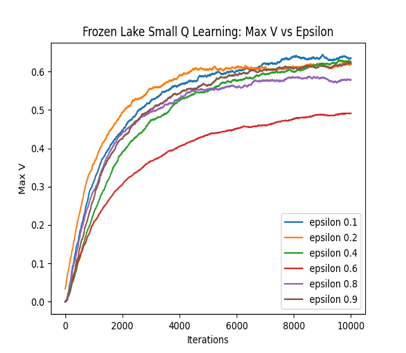
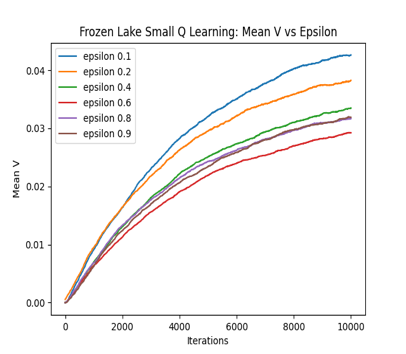
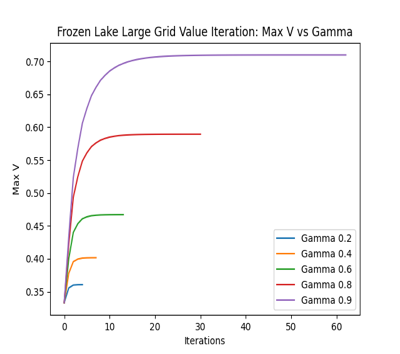


Fig 3: Q Learning convergence plots for Frozen Lake small grid

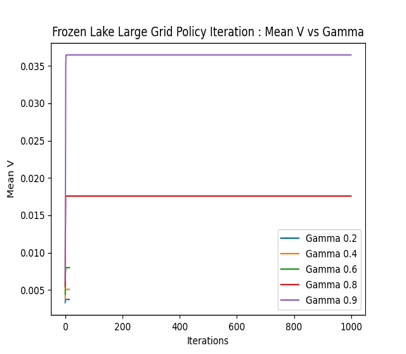
Chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

Fig 6: Value Iteration plots for Frozen Lake large grid size

Chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

Fig 7: Policy Iteration plots for Frozen Lake large grid size

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Fig 8: Q Learning plots for Frozen Lake large grid size

Value Iteration and Policy Iteration was run for different values of gamma. Lower gamma values put more emphasis on short term gains, higher gamma values put more emphasis on long term gains. Closer the gamma is to 1, closer the policy will be to the one that optimizes the gains over infinite time. Plots of Mean V, Max V, Rewards and Error vs Iterations for different values of gamma were generated. We would be using Max V plots and Mean plots primarily to demonstrate convergence.

The flat lines in the Mean V and Max V plots signals convergence. From the charts in Fig 1 and Fig 2, it is evident that higher value of gamma (gamma of value 0.9) gives us the best reward as it is focused on long term gains and would not get trapped in local optima. Lower values of gamma are converging at much lower rewards indicating local optima. If we compare the Max V chart in Fig 1 and Fig 2, we can see that Value Iteration is converging at about 15-20 iterations whereas Policy Iteration is converging at 2-3 iterations. This is in line with what we had hypothesized earlier. Policy Iteration converges faster than Value Iteration as one policy can be represented by infinite number of value functions. So, in Policy Iteration when we move from one policy to an improved policy, we have jumped over multiple value functions.

For this problem, the Value Iteration and Policy Iteration converges at the same answer i.e., a reward of 0.7. The error plot supports the fact that the error for both VI and PI drops with each iteration and are converging with a minimal error value.

Unlike Value Iteration and Policy Iteration, Q Learning doesn’t know the model and rewards. I ran Q Learning algorithm with different values of epsilon by keeping the alpha (learning rate) and gamma constant. The epsilon parameter introduces randomness into the algorithm, forcing us to try different actions to avoid local optima. If epsilon tends to zero, we never explore but always exploit the knowledge we have. Epsilon value close to one force us to always take random actions and never use past knowledge. From the charts in Fig 3, we can see that a lower value of epsilon of 0.1 gives us better rewards than with higher epsilon. With such lower value of epsilon, the agent tries to learn from previous knowledge and acts based on that rather than taking random actions. It is to be noted that extremely lower value of epsilon might land our problem into a local optimum.

Because of the lower number of states, Q Learning finds it hard to converge. We can see from the charts in Fig 3, that the Max V plot is trying to consolidate but hasn’t fully converged. Since, it is hard to converge, we can assess if the variance of rewards in last few iterations is lower than certain threshold, if it is lower, we can declare convergence. It is evident that Q Learning is not performing as well as Value Iteration and Policy Iteration. Since, Q Learning doesn’t have access to the model and rewards, it works like an approximation algorithm and learns by visiting the state-action pair infinitely. Q Learning with an epsilon value of 0.1 is converging at a reward value of 0.6.

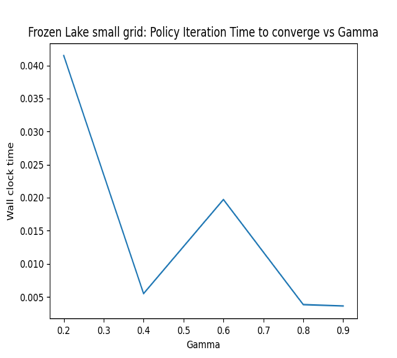
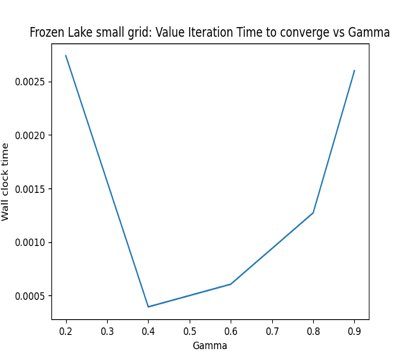
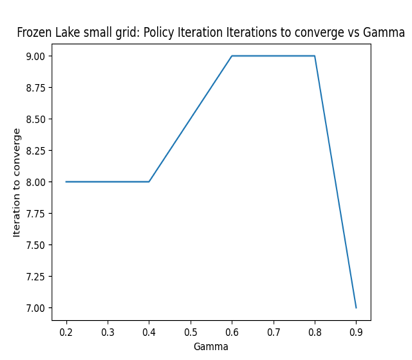
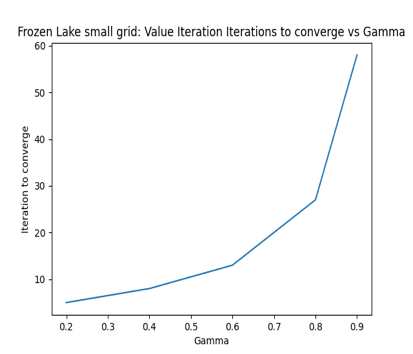
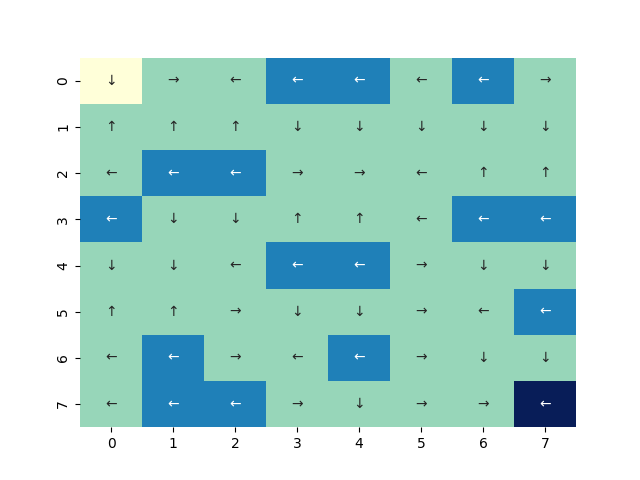


Fig 4: No. of iterations & Wall Clock time for VI and PI to converge at different gamma values for small grid Frozen Lake

For Value Iteration, the number of iterations taken to converge increases with increasing value of gamma. With higher value of gamma, it focuses more on long term gain and hence takes more iterations to converge. For Policy Iteration, the number of iterations to converge range from 7 to 9. However, Value Iteration takes less wall clock time to converge compared to Policy Iteration as Policy Iteration performs both policy evaluation and policy improvement in each iteration compared to Value Iteration performing just value improvement in each iteration. With lower value of gamma, Value Iteration find tough to converge because of the volatility of short-term rewards. With increasing gamma, they reach global optima and take more time to converge.

A picture containing shape

Description automatically generatedA picture containing diagram

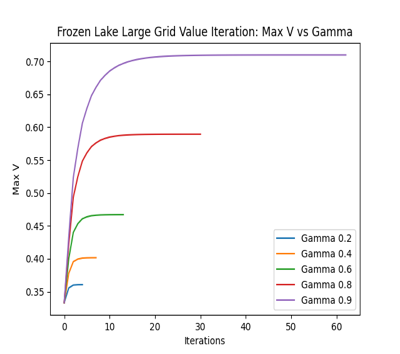
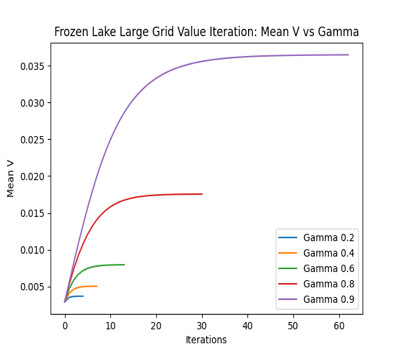
Description automatically generated

Fig 5: Policy Visual for Value Iteration, Policy Iteration and Q Learning from left to right

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Space for Policy discussion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Model-free algorithms like Q Learning relies on trial-and-error method. During these trials, agent must decide the action. Some of these actions have never been taken before and some of the actions have been performed earlier and the agent might know the outcome. This concept of exploiting what the agent already knows versus exploring a random action is called exploration-exploitation trade-off. Initially, as the agent has limited knowledge, it needs to do sufficient exploration and improve its knowledge base. If the agent exploits its knowledge initially without exploring much, it might end up in local optima. More exploration in the final stages may not lead to optima. So, we need to keep a balance between exploration and exploitation. Hence, a decaying epsilon value is recommended.

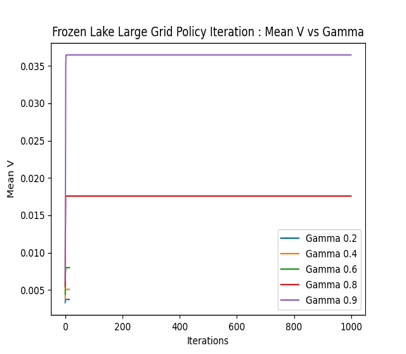
**Frozen Lake Large Grid**:

Chart

Description automatically generatedChart

Description automatically generated

Fig 6: Value Iteration plots for Frozen Lake large grid size

Chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

Fig 7: Policy Iteration plots for Frozen Lake large grid size

Chart, line chart

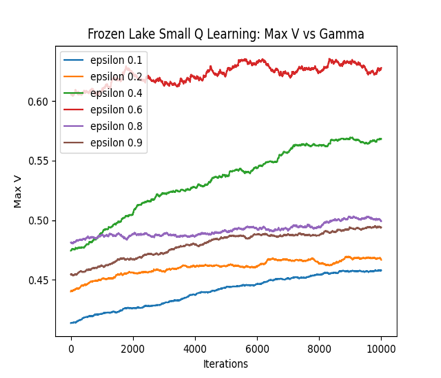
Description automatically generated

Fig 8: Q Learning plots for Frozen Lake large grid size

**References**:

1. <https://pymdptoolbox.readthedocs.io/en/latest/>
2. <https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration>
3. <https://stackoverflow.com/questions/37370015/what-is-the-difference-between-value-iteration-and-policy-iteration>
4. <https://www.baeldung.com/cs/epsilon-greedy-q-learning>