# Assignment 4 - Markov Decision Processes

Michael Lukacsko
*mlukacsko3@gatect.edu*

***Abstract*— This assignment explores the application of reinforcement learning algorithms to solve two Markov Decision Processes - Frozen Lake and Cliff Walking. The algorithms used include Policy Iteration, Value Iteration, and Q-Learning. These algorithms are used to train agents to make decisions in these environments, with the ultimate goal of finding the optimal policy for each environment. The results and performance of each algorithm are discussed and compared, providing insights into the strengths and limitations of each approach. Overall, this work demonstrates the effectiveness of reinforcement learning for solving complex decision-making problems in dynamic environments.**

***Keywords—Policy Iteration, Value Iteration, Q-Learning, Markov Decision Processes, Frozen Lake, Cliff Walking.***

## I. The Environments

### A. Frozen Lake

Frozen Lake is a grid world environment in which an agent is placed on a frozen lake. The goal of the agents is to navigate to the other side of the lake without falling into any of the holes and incurring a negative reward. The agent can take steps to move in four directions (up, down, left, or right), but each step results in a small penalty and is not always deterministic. This is because there is a chance that the agent will "slip" in a different direction than intended because of the slippery ice. If the agent makes it to the goal state, it will receive a reward. The environment is characterized by its stochasticity, which makes it a challenging environment for the agent to navigate. Lastly, to expose differences in the learner's performance, and the effect grid size has on convergence, two different sized grids are implemented. Figure 1 below portrays the 8x8 and 20x20 Frozen Lake environment the agent will navigate.

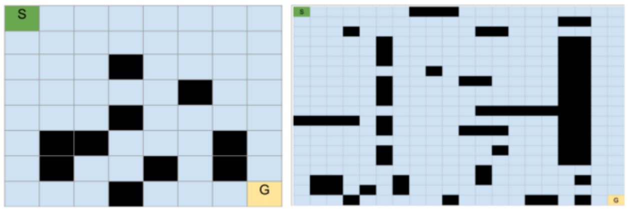


Figure 1 – Frozen Lake MDP. Left, 8x8 Grid. Right, 20x20 Grid

### B. Cliff Walking

Cliff Walking is another grid world problem in which an agent is placed on a grid consisting of a starting cell and a destination cell. The cells in the bottom row of the grid, between the starting and ending states, represent the cliff, which, if fallen into, results in a large negative reward. The agent can take the actions of moving up, down, left, or right, but a small negative reward for each movement is incurred in doing so. By reaching the goal state, the agent receives a large positive reward. As such, the agent's goal is to find an optimal policy that maximizes the expected cumulative reward over time.

Cliff Walking is a good example of the trade-off between exploration and exploitation. If the agent always takes the optimal action, it might fall off the cliff and receive a large negative reward. On the other hand, if the agent relies too heavily on exploring, it might waste time and not reach the destination cell. Figure 2 below shows the Cliff Walking environment.

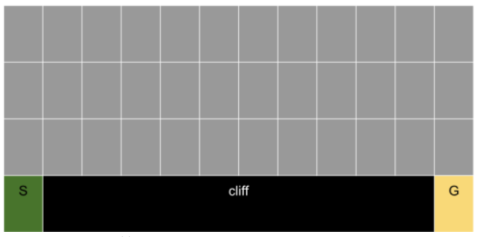


Figure 2. Cliff Walking MDP. 4x12 Grid

### C. Environment Correlation

Frozen Lake and Cliff Walking present different challenges for an agent to navigate and find an optimal policy. The two primary differences are each MDPs respective state space and action space, and the impact on finding an optimal policy.

#### 1) State Space

Both Frozen Lake and Cliff Walking have discrete state spaces. In Frozen Lake, the state space is defined by the position of the agent on a grid, where each cell can be either part of the frozen lake or a hole. Similarly, in Cliff Walking, the state space is defined by the position of the agent on a grid, where the agent's position can be any point on the grid. While both state spaces are discrete, Cliff Walking's state space is continuous along one axis, as the agent can move horizontally across the grid without any risk of running into an obstacle and incurring a negative reward and failing.

Because of these differences, for both Frozen Lake and Cliff Walking, certain states are more interesting than others because they influence the agent's behavior and the reward it receives. For example, in Frozen Lake, the agent will receive a negative reward for stepping on a hole or a positive reward for reaching the goal state. The agent's behavior should change in interesting ways when it approaches the edge of the lake or is surrounded by holes or walls. In Cliff Walking, the agent receives a negative reward for each movement and a large negative reward for falling off the cliff. Hence, the agent's behavior should change in interesting ways when it approaches the cliff or when it is close to the goal state. Moreover, because the agent can move continuously along one axis, it will be interesting whether the agent risks moving along the cliffs edge or moves away from the cliff to mitigate risk of the large negative reward.

#### 2) Action Space

The action space in Frozen Lake is a set of four possible actions (up, down, left, and right) that can be taken by the agent in each state. As such, the cardinality of the action space in Frozen Lake is relatively small and means that the agent's behavior is relatively simple and predictable, and the optimal policy should be learned relatively quickly. In contrast, the action space in Cliff Walking is continuous, meaning that the agent can move in any direction continuously along the horizontal axis. As a result, the cardinality of the action space in Cliff Walking is much larger due to the continuous nature of the actions. This should make learning an optimal policy in Cliff Walking more challenging, as there are many more possible actions to consider, and the agent will be forced to explore a larger portion of the state space to find the optimal behavior.

## II. REINFORCMENT LEARNING TECHNIQUES

### A. *Policy Iteration (PI)*

PI is a model-based algorithm for solving Markov Decision Processes and consists of two main steps: policy evaluation and policy improvement. During policy evaluation, the algorithm estimates the value function for a given policy using the Bellman equation. The value function represents the expected cumulative reward that the agent can obtain by starting in a particular state and following the policy thereafter. During policy improvement, the algorithm updates the policy based on the current value function estimate.

### B. *Value Iteration (VI)*

VI is another model-based algorithm for solving Markov Decision Processes and finding an optimal policy. Unlike policy iteration, the VI algorithm iteratively computes the optimal value function, which represents the maximum expected cumulative reward that can be obtained from each state. VI starts with an initial estimate of the value function and then repeatedly updates it using the Bellman equation until convergence. The Bellman equation expresses the optimal value function as the maximum expected reward of taking each action in the current state and then following the optimal policy thereafter. By repeatedly applying the Bellman equation, VI gradually refines its estimate of the optimal value function until it converges to the optimal value function.

### C. *Q-Learning (QL)*

With QL, an agent tries to learn the optimal action-value function that maps each state-action pair to its expected cumulative reward. QL starts with an initial estimate of the action-value function and then updates it after each action taken by the agent based on the observed reward and the estimated value of the next state-action pair.

Unlike PI and VI, QL does not require a model of the environment to learn the optimal policy. Instead, QL learns from experience by updating the action-value function based on the observed rewards and transitions. Being a model-free algorithm, QL uses an exploration vs exploitation strategy to balance between taking actions that are known to have high reward (exploitation) and taking actions that are unknown but may lead to better reward s(exploration). In this QL implementation, this is done by selecting actions based on an epsilon-greedy strategy, where the agent chooses the action with the highest Q-value with a probability of 1-epsilon and a random action with a probability of epsilon.

### D. *Expiriment Parameters*

The following reward, penalty, and unintended movement parameters are used for the Frozen Lake and Cliff Walking environments:

| | **Frozen Lake** | **Cliff Walking** |
|---|---|---|
| **End State Reward** | 1.0 | 100.0 |
| **Step Reward** | -0.1 | -1.0 |
| **Hole/Cliff Penalty** | -1.0 | -100.0 |
| **Slippage/Wind Probability** | 10% Left and 10% Right (20% Total) | 10% |

The following hyperparameters are tuned for the corresponding algorithms in both environments:

| | **Policy Iteration** | **Value Iteration** | **Q-Learning** |
|---|---|---|---|
| **Max Steps/Iterations** | 1000 | | 1000 |
| **Max Trials per Iteration** | 100 | | 1000 |
| **Theta** | 0.00001 | | |
| **Consecutive Theta to Convergence** | 10 | | |
| **Discount Factor/Gamma** | 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 | | |
| **Initial Q** | N/A | N/A | 0, Random |
| **Learning Rate/Alpha** | N/A | N/A | 0.1, 0.5, 0.9 |
| **Epsilon** | N/A | N/A | 0.1, 0.3, 0.5 |
| **Epsilon Decay** | N/A | N/A | 0.00001 |

## III. FROZEN LAKE

### A. *Policy Iteration (PI)*

#### 1) *8x8 Grid*

Figure 3 below (left) depicts the optimal policy, or actions that the agent should take in each state to reach the goal state while avoiding the holes and maximizing the expected cumulative reward. What is interesting in the plot below is that the agent's first move is up. Hypothetically, this is most likely because the probability of hitting a hole, or moving closer to a neighboring hole, is 0 by moving up. If the agent slips to the right, the optimal policy directs the agent to the right side of the grid. Finally, after reaching the rightmost column, the agent is then directed down toward the goal. Taking into consideration the two holes closest to the goal, the agent is directed to move towards the wall (right) where, if the agent slips, the agent will avoid falling into the hole because it will move either up or down. By performing the steps plotted below, the agent can successfully navigate the environment and minimize the risk of falling into a hole.

Lastly, using delta and reward values as a criterion of convergence, the plot on the right of figure 1 below shows that the PI algorithm reaches a minimum delta value and maximum cumulative reward value after only 1 iteration and terminates after 2 iterations when using a gamma/discount factor of 0.8. Because of this gamma selection, future rewards are given some importance, however, immediate rewards are considered more important. As a result, the agent in this Frozen Lake environment is encouraged to take some risks to reach the goal faster, but not at the expense of immediate rewards. Hence, the algorithm reaches convergence quickly, and the optimal policy is found after 2 steps.

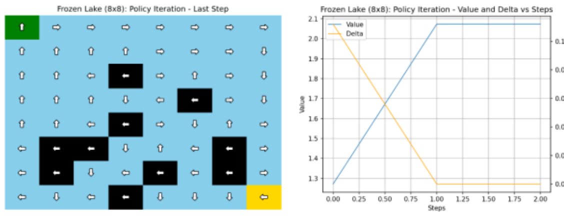


Figure 3. Frozen Lake – Policy Iteration (8x8). Optimal policy, Left. Delta vs Steps vs Value, Right.

### 2) 20x20 Grid

Referencing figure 4 below (right), the number of iterations needed to converge to an optimal policy using a discount factor/gamma of 0.9 is 6. Like the delta and reward value vs steps plot with a grid size of 8x8, the values of both delta and cumulative reward values plateau, this time around the second step, thus reaching the theta threshold and indicating convergence.

The graphical plot (left) showing the optimal policy is a bit different in that the agent's first step is down. Moreover, in this optimal policy plot, the agent continues to move down until reaching the hole located in row 12. After moving down, the agent eventually begins to move in the direction of the goal state in the 10th row.

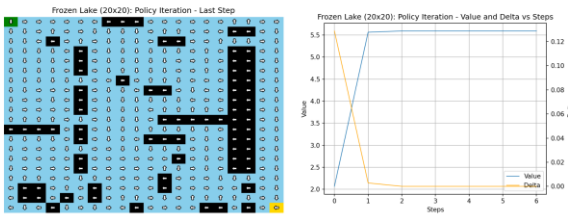


Figure 4. Frozen Lake – Policy Iteration (20x20). Optimal policy, Left. Delta vs Steps vs Value, Right.

### 3) Influence Of Tuning Gamma

For both the 8x8 grid and 20x20 grid, the discount factor/gamma plays an important role in the agents balance of exploration and exploitation, hence influencing the performance of the agent. This is observed when using a discount factor of 0.8 versus the other values of gamma on the 8x8 grid. Using a gamma value of 0.8 was clearly the ideal choice as the highest cumulative reward is achieved in the fewest number of steps. As a comparison, using a gamma value of 0.7 took 11 steps to converge on an optimal policy and had a cumulative reward of 1.54. A gamma of 0.9 took 4 steps to converge on an optimal policy. This makes clear that there is no obvious correlation between the gamma values and the number of steps. However, when a lower discount factor is used, the agent will place more of an emphasis on the immediate reward. In this case, the result of a lower gamma is a more myopic policy, which requires more exploration to find the optimal policy and results in a longer convergence time.

This is not the case when looking at the effects of varying gamma on the 20x20 grid. Noting that a gamma of 0.9 converged to the optimal policy, this discount factor value put even more emphasis on exploring and produced a cumulative reward of just over 5.51 in 6 steps. A discount factor value of 0.7 produced a cumulative reward of 1.95, but it did so in only 5 steps. This gap in cumulative rewards makes clear the tradeoff of exploring and exploiting the environment to find the greatest cumulative reward. That is, gamma clearly affects the agents need to try different actions to explore the environment and gain new information, while also taking actions that it believes will lead to the highest immediate reward.

## B. Value Iteration (VI)

### 1) 8x8 Grid

Despite the difference in PI and VI implementation details and convergence properties, the VI optimal policy looks the same as referenced in figure 1 above. This is because both PI and VI converged to the same optimal policy. Moreover, the plot on the right of figure 5 looks very similar to the corresponding plot in figure 3, however, the number of steps required to reach convergence is over 50 (52), whereas PI converged to an optimal policy in 2 steps.

Regarding the optimal policy of this Frozen Lake environment solved using VI, the optimal policy shows the agent taking penalties for steps over risking a step that results in failure from falling into a hole. Knowing that the VI value function estimates the agents expected total reward achieved by taking a specific action in a specific state, the agent here determined that taking a step towards the goal with a penalty leads to a higher expected total reward than falling into a hole and receiving a large negative reward. Using a gamma/discount factor of 0.9, the optimal policy is plotted below on the left.

With respect to the cumulative reward values and delta plotted below (right), both delta and cumulative reward values plateau around the 30th step. Yet it takes an additional 20 steps before the difference in delta and reward value reaches the theta threshold, indicating that the estimated value function converged to the optimal value function.

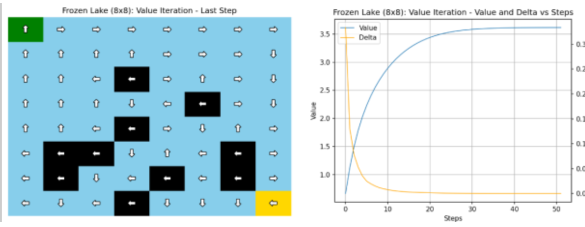


Figure 5. Frozen Lake – Value Iteration (8x8). Optimal policy, Left. Delta vs Steps vs Value, Right.

### 2) 20x20 Grid

Using the same discount factor/gamma of 0.9, the optimal policy produced by VI is identical compared to the optimal policy produced using PI. This is a sign that VI, as well as PI, is able to find the optimal policy by iteratively improving the estimate of the value function.

Referencing the delta vs steps vs value plot below (right), it is surprising that VI took 53 steps to converge to the optimal policy, which is one more step compared to when the environment is 8x8. Despite the larger grid in the problem having significantly more states and transitions, which would make the optimization problem more complex, one might expect that VI would require significantly more steps to solve the problem than a much smaller grid. However, this is not the case.

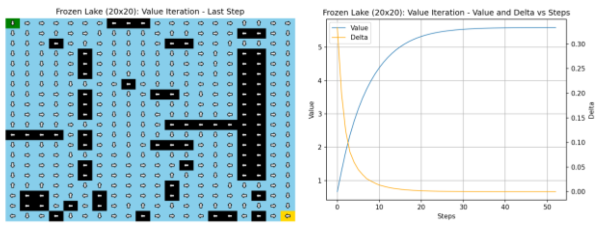


Figure 6. Frozen Lake – Value Iteration (20x20). Optimal policy, Left. Delta vs Steps vs Value, Right.

### 3) *Influence of Tuning Gamma (Discount Factor)*

The influence of gamma using VI is a little different than what is seen with policy iteration. On the 8x8 grid, the optimal policy using a gamma of 0.9 had the highest cumulative reward as well as the greatest number of steps. Here, a cumulative reward of 3.62 was achieved in 51 steps. However, a gamma of 0.8 converged to an optimal policy in 27 steps and achieved a cumulative reward of 2.07 and a gamma of 0.6 found an optimal policy in 13 steps with a cumulative reward of 1.25. As such, it can be said that where VI has a lower gamma value, less emphasis is placed on future rewards and more emphasis on immediate rewards. This tradeoff of immediate vs future reward leads the agent to prioritize exploration of the environment to find immediate rewards using less steps. Alternatively, a higher gamma value being used places more emphasis on future rewards, which leads the agent to prioritize exploitation of actions that lead to higher long-term rewards, even if it requires more steps.

The effect of gamma on exploration and exploitation, and the prioritization of immediate rewards vs future rewards, is very similar on the 20x20 grid. Here, a gamma of 0.9 converged to an optimal policy in 52 steps with a cumulative reward of 5.59. A gamma of 0.8 converged to an optimal policy in 28 steps, however, the cumulative reward was only 2.86. Using a gamma of 0.7 converged to an optimal policy in 18 steps and a gamma of 0.6 in 13 steps. As such, it appears there is a correlation between gamma, the tradeoff of immediate vs future reward via exploring and exploiting the environment, and the number of steps taken to converge on an optimal policy.

## C. *Q-Learning (QL)*

### 1) *8x8 Grid*

Implementing this QL with varying initial Q-values, learning rates (alpha), exploration rates (epsilon), and epsilon decay did not yield a solution that converged to an optimal policy. Unsurprisingly, the policy plotted below does not look like the optimal policies produced using VI or PI. The inability of this QL algorithm to converge to an optimal policy is most likely because the Frozen Lake problem has a high degree of stochasticity, meaning that the outcome of each action is not entirely predictable, and the reward function is sparse. As such, this QL implementation would more than likely need additional steps per iteration and/or more iterations to guarantee that all state-action pairs are explored infinitely often, thus resulting in a guarantee to converge to the optimal Q-value.

The suboptimal policy is plotted below on the left in figure 7. It can be seen, outlined in red, there are several states where the agent selects to move toward a hole, resulting in failure and large negative reward.

Supporting this is the plot on the right where the delta values remain at 0 except for a few spikes where the value reaches around 0.3, 0.8, and 1.0. This could be an indication that the exploration rate, or epsilon, is too low and causes the agent to get stuck in a suboptimal policy. To improve the convergence of this QL algorithm in future iterations, increasing the exploration rate or decreasing the learning rate, in addition to the point made regarding iterations, might also help reach convergence to an optimal policy more quickly.

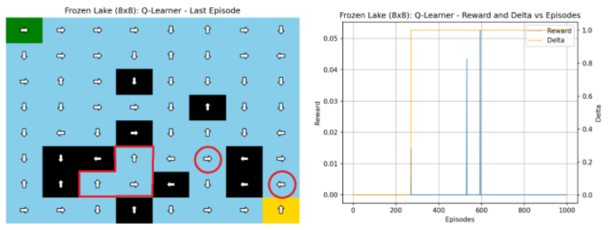


Figure 7. Frozen Lake – Q-Learning (8x8). Optimal policy, Left. Delta vs Episodes vs Reward, Right.

### 2) *20x20 Grid*

Like the smaller 8x8 grid, QL was unable to converge on the larger 20x20 plot. Referencing figure 8 below, QL did not successfully learn the optimal policy as the plot shows the agent always moving left. Moreover, the plot on the right in figure 8 below shows that the value of delta never goes above 0.0. As a result of figure 8, it is doubtful QL was able to learn much about the environment at all. If it did, it would be expected to some moves toward the goal state in the graphical plot, as well as some value of delta and/or cumulative reward value on the rewards vs delta vs episodes plot.

Again, to guarantee that all state-action pairs are explored infinitely often, thus resulting in a guarantee to converge to the optimal Q-value, there needed to be significantly more steps per iteration and/or more iterations.

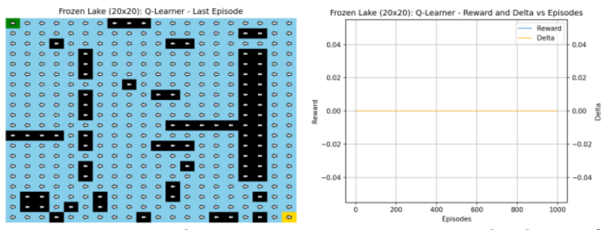


Figure 8. Frozen Lake – Q-Learning (20x20). Optimal policy, Left. Delta vs Episodes vs Reward, Right.

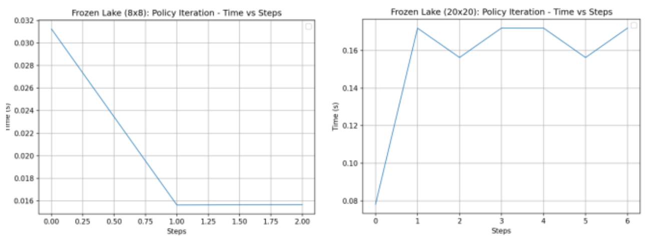
#### *3) Influence of Tuning Hyperparameters (Gamma, Initial Q, Alpha, Epsilon)*

The QL optimal parameters were difficult to verify in both the 8x8 and 20x20 environment grids. The plot on the right in figure 8 above is produced using 0.9 for alpha, 0.7 for gamma, an initial Q value that is random, and 0.5 for epsilon. These parameters did produce the greatest reward, however, because QL did produce a suboptimal policy, it is difficult to determine the effects of these hyperparameters on convergence. To further test hyperparameter tuning in both environments, the first move is to increase the number of iterations. Limiting the number of iterations to 1000 is clearly not sufficient.

### *D. Policy Iteration (PI), Value Iteration (VI), and Q-Learning (QL) Comparison*

The implementation of PI, VI, and QL on different sized Frozen Lake environments proves that updating the policy in each iteration is more effective than updating the value function. This approach allows PI to converge to the optimal policy faster than VI because it takes advantage of the fact that small changes in the policy can lead to large changes in the value function. In contrast, VI takes longer to converge to the optimal value function because of updating the value function in each iteration. This is especially true where the state space was larger. On the other hand, because QL is a model-free algorithm, meaning it does not require a model of the environment to learn the optimal policy, QL suffered from convergence issues because it updates the Q-values based on observed transitions and its reliance on exploration to learn the optimal policy.

Looking at the elapsed time for all three algorithms per iteration in figure 9 below, QL (bottom) is the slowest in terms of both overall time and time per iteration. Especially because QL ran all 1000 episodes, this algorithm took the most amount of time cumulatively. Evaluating VI and PI, PI took more time per step, however, it needed only 2 steps to solve the 8x8 grid and 6 steps to solve the 20x20 grid. VI, however, took more steps but each step was faster compared to PI. This tradeoff is due to how VI uses the Bellman equation to iteratively update the value function until it converges to an optimal policy.



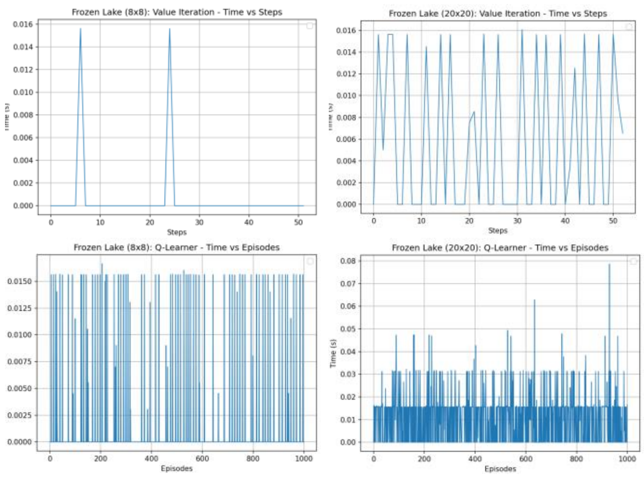


Figure 9. Time Per Step/Episode. Policy Iteration, top. Value Iteration, middle, Q-Learning, bottom.

## IV. Cliff Walking

### *A. Policy Iteration (PI)*

Using a discount factor/gamma of 0.7, figure 10 below shows PI converging to an optimal policy in 8 steps. Intriguingly, the value plot on the right shows a starting value at step 0 of about -1000. Thinking about this in terms of rewards and exploration, starting with a negative reward, or Q value, is representative of the worst-case scenario where the agent takes the worst possible action or actions and thus receives a large negative reward. Moreover, by setting the initial value function estimate to a negative value, it ensures that any positive rewards observed during the policy evaluation step will increase the estimated value function and any additional negative rewards will decrease it. This in turn helps ensure improvements to a better policy and faster convergence to an optimal policy. As a result, the negative reward value to start can be seen as a method to encourage exploration by the agent. This is proved by the fact that at the next step, step 1, the reward value jumps to over 2000. The jump in cumulative reward indicates that the agent is able to make significant improvements and move closer to the optimal policy, a direct result of the agent's exploration.

Lastly, the optimal policy plot below on the left is interesting in the sense the agent moves across the grid on the edge of the cliff until reaching the halfway mark between the start and goal state. This makes clear that the agent does not find it necessary to use caution, and incur a small negative reward, by moving away from the cliff in the first couple of steps. Increasing the penalty from falling off the cliff, or lowering the penalty for each step, might persuade the agent to take a different route that could be considered more cautious.

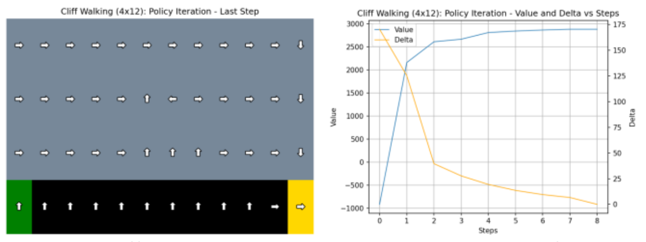


Figure 10. Cliff Walking – Policy Iteration. Optimal policy, Left. Delta vs Steps vs Value, Right.

The effects of gamma here are strictly reward related as gamma values of 0.5 to 0.9 all took 8 steps to converge on an optimal policy.

## B. *Value Iteration (VI)*

Unlike the large gap in steps to converge on an optimal policy for Frozen Lake using PI and VI, here VI was able to converge to an optimal policy in the same number of steps as PI using a discount factor/gamma of 0.6. However, unlike the solution using PI, VI's delta and cumulative reward value plateaued around the 4th step. This plateau is a sign that the algorithm has reached the optimal policy and is not making further progress in improving the value function. In this case, the delta and cumulative reward value plateauing indicate that the optimal policy has most likely been learned before terminating.

Finally, the optimal policy on the left in the figure below is the same as what is produced using PI. Again, the agent chooses to move along the edge of the cliff despite the risk of being blown off the cliff. Modifying the penalty for falling off the cliff and/or the penalty per step would most likely produce a different policy, one where the agent is motivated to move away from the edge of the cliff from the start.

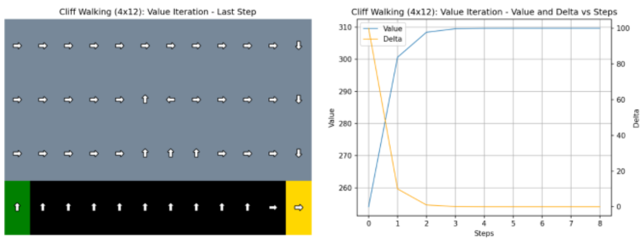


Figure 11. Cliff Walking – Value Iteration. Optimal policy, Left. Delta vs Steps vs Value, Right.

The impact of gamma on VI in this problem is much more profound. Here, a gamma of 0.1 converges to optimal policy in the least number of steps. Because a gamma value of 0.1 places more emphasis on immediate rewards and less on future rewards, the agent in this problem prioritizes avoiding the cliff over getting to the goal, as falling off the cliff incurs a large negative reward.

## C. *Q-Learning (QL)*

Like the Frozen Lake environment, QL produced a suboptimal policy. The fact that most of the arrows are moving either up or right toward the goal state is a good indication that this QL was at least able to explore the state-action space, but seeing arrows pointing to the left in the upper right corner and the on the top of the third column are signs that converging to an optimal policy did not happen. In this case, it may be necessary to further tune the hyperparameters, reward structure, or both, and to increase the number of episodes.

Lastly, the plot on the right in figure 12 below shows the agent's inability to achieve a cumulative reward over 6. Up until about the 800th episode, the agent often receives a negative reward meaning it must have failed to reach the end state. The fact that the cumulative reward is not negative after the 800th episode is another indication that QL was able to move in the direction of convergence, however, was unable to within the episode limitation of 1000. Again, further tuning hyperparameters like discount factor and exploration rate would encourage the agent to explore more state-action pairs. Similarly, increasing the number of episodes would enable the agent to have more time to explore.

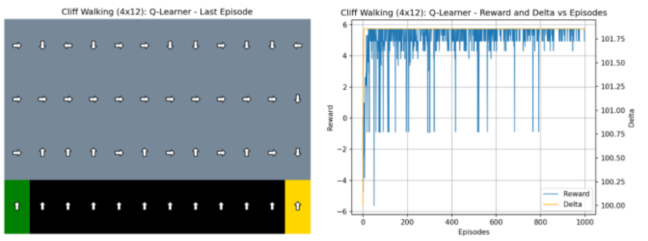


Figure 12. Cliff Walking – Q Learning. Optimal policy, Left. Delta vs Steps vs Reward, Right.

Again, because QL was unable to converge to an optimal policy, it is hard to determine the effect of tuning hyperparameters. The plot above is created using 0.9 for alpha, 0.7 for gamma, an initial Q value that is random, and 0.5 for epsilon. More iterations are needed to enable QL to converge to an optimal policy and only then would the impact of hyperparameters be known.

## D. *Policy Iteration (PI), Value Iteration (VI), and Q-Learning (QL) Comparison*

PI and VI perform similarly regarding the number of steps required to converge to an optimal policy, and they both produce the same optimal policy. This is interesting because VI updates its value function at every state in every iteration, whereas policy iteration updates its policy only after evaluating the value function. This difference in how each algorithms respective policy is updated often causes the number of steps required to converge to be greater for one or the other, which is seen in the Frozen Lake environment. However, in this case, VI and PI were each able to converge to an optimal plot in 8 steps. Moreover, this means that both algorithms were able to balance exploration and exploitation, unlike QL, and neither PI nor VI got stuck in a suboptimal policy because it didn't explore sufficiently.

Because PI and VI required the same number of steps, time per iteration is an important factor. Figure 13 below shows the time elapsed per step. Comparing the top two plots, PI and VI, it's clear that VI requires less time than PI to converge to an optimal policy. QL, on the other hand, takes a significant amount of time and produces a suboptimal policy.

Regarding QL not finding an optimal solution, there needs to be a better balance between exploration and exploitation. Having 1000 trials for 1000 episodes might have been sufficient for the agent to explore different paths to find a route that avoided the cliff, however, once a safe path was discovered, the agent did not have enough time to focus on exploiting this knowledge to reach the goal as quickly as possible. As a result, this failure to balance exploration and exploitation produced a suboptimal policy in a much greater amount of time when compared to PI and VI.

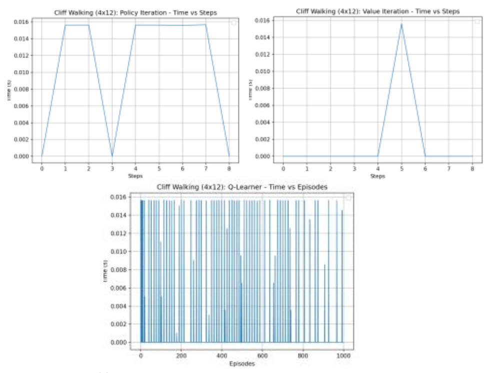


Figure 13. Cliff Walking – Time vs Episode. Policy Iteration, top left. Value Iteration, top right. Q-Learning, bottom.

## V. Conclusion

The three algorithms tested, Policy Iteration (PI), Value Iteration (VI) and Q-Learning (QL) exposed several strengths and weaknesses while solving the two reinforcement learning MDP problems, Cliff Walking and Frozen Lake. First, it made clear that the model-based algorithms, PI and VI, had a significant advantage over the model-free QL algorithm because of the way they interact with their environment. This is supported by the outcomes discussed in previous sections and is the result of the fact that the model-based algorithms explore the environment by using the learned model to simulate the consequences of different actions, allowing them to learn the optimal policy more quickly. In contrast, the model-free QL algorithm has to explore the environment by trial and error, which results in it being both slower and less efficient.

Second, PI and VI did a better job generalizing to new environments because they were both able to learn the underlying structure of the environment. This enabled PI and VI to adapt to the new environments more easily compared to QL. Moreover, PI and VI were equipped to handle uncertainty in the environment because they we able to use the learned model to make predictions about the future. On the other hand, QL had to rely on trial and error again and as outlined in previous sections, resulted in QL struggling in both environments where there existed high levels of uncertainty.

Finally, as it relates to the grid-world problems, PI was able to perform better on the Frozen Lake problem versus VI. As the varying state space size shows, it was clearly more difficult for VI to converge to an optimal value function since it requires updating the value function for every state in each iteration. However, because PI updates the value function only for the states that are visited during policy evaluation, it ended up being more effective for both the small and large state spaces. QL, on the other hand, struggled with both grid-world problems. This was true regardless of the number of state spaces.