CS 7641 Assignment 4

Introduction

This assignment explores the effectiveness of Policy Iteration, Value Iteration, and Q-Learning on two chosen Markov Decision Processes (MDPs). The two processes will both have stochastic state transitions but will differ in their number of states and mechanics. The algorithms will be used to solve the MDPs and the results, convergence rates, and timings will be compared. The impact of changing the parameters for each of the three algorithms will also be explored.

Markov Decision Processes

Frozen Lake

Frozen Lake is a grid world problem where the entire grid is a frozen lake and the goal of the agent to retrieve a frisbee, which is located in one particular grid cell. The frozen lake is slippery so moves may not result in their intended direction; the probability of going in the intended direction is 0.8 and the probability of slipping (to the left or the right) is 0.1 for each side. Additionally, there are holes in the frozen lake. If the agent goes into one of those grid cells, they will incur a -1 reward and the episode is immediately terminated. The rewards are -0.1 for each step, -1 for the hole, and 10 for the frisbee. Different sizes of the Frozen Lake problem were tested, but ultimately the largest one (15 by 15) was used in this report. This MDP was interesting because it's a staple grid world maze-like problem with a bit of uncertainty in its state transitions. This will help contrast the differences between some of the planning algorithms as well as the learning algorithm in how they determine values and policies.

Windy Cliff Walking

The Windy Cliff Walking is also a grid world problem. The goal is to cross from the bottom left side of the grid to the bottom right side but without falling off the cliff. The cliff is represented as the bottommost row of cells in the grid. Each time the agent falls off the cliff, a -1 reward is incurred and their position is reset to the start. Each step incurs a -0.1 reward and the episode terminates only if the goal is reached. There is also a wind component where, at each step, there is a 20% probability that the agent is pushed one or two steps towards the cliff. There are only two columns of the grid where there is a possibility of a two cell wind push. This problem is interesting because of the unpredictable factor that is the wind; the agent must be able to distinguish between the action and the random factor, which is essential in the learning algorithm case. Additionally, there is uncertainty about whether the wind pushes 1 cell or 2 cells, which adds an extra layer of complexity for this MDP. These factors combined make this a nice second problem for comparison.

Algorithms

Three algorithms were tested for the MDPs, two of which are planning algorithms (Policy Iteration and Value Iteration) and one is a reinforcement learning algorithm (Q-Learning). Planning algorithms assume the transition probabilities and rewards of each state are known and are then used to solve for the optimal policy and value functions of the MDP. This is usually done using dynamic programming, which is the case for both Policy Iteration and Value Iteration. A reinforcement learning algorithm on the other hand, assumes no such knowledge about the system and tries to find the optimal policy through interacting with the environment. The chosen reinforcement learning algorithm is Q-Learning, which is based on the state-action value function which is also called the Q function.

Policy Iteration

The Policy Iteration algorithm aims to find the optimal policy function by repeatedly evaluating and improving the agent's policy. It first starts off with an arbitrary policy and evaluates it. The evaluation is done by repeatedly updating the value function with the given policy. After the value function converges or another stopping condition is met, the policy is updated in accordance with the value function. The process is then repeated with the new policy. The algorithm continues to refine the policy until the policy no longer changes, at which point it will be the optimal policy. The main parameters of this algorithm are the stopping conditions for convergence, and the discount rate γ which is used to specify how short or long term oriented the agent is.

Value Iteration

The Value Iteration algorithm solves for the optimal value function by repeatedly improving the value function. The algorithm starts by arbitrarily initializing the value function. Then, it refines the value function by calculating the the value of each state given the optimal policy from the previous iteration's value function. This is repeated until the value function converges, at which point it is the optimal value function. The optimal policy is the extracted from the optimal value function. Value Iteration's main parameters are also the convergence criteria and the discount rate.

Q-Learning

In the case where the state transition probabilities and reward functions are unknown, the agent must learn through interacting with the environment. The agent can either try to estimate the state transition probabilities and reward functions, called Model-Based Learning, or try to derive the optimal policy / value directly, called Model-Free Learning. Q-Learning is a type of Mode-Free Learning algorithm. It attempts to find the optimal policy by repeatedly updating the Q function (the value for a given state and action) with information gained from the previous iteration's Q values. This is shown in the formulation below:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a))$$

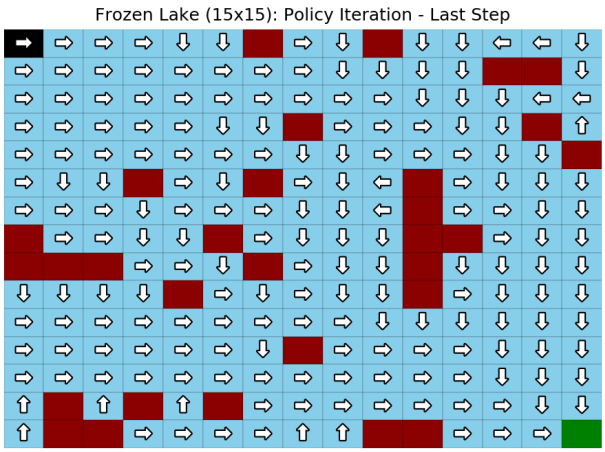
The α value represents the learning rate for the Q function, and the γ value represents the discount rate for future rewards. Another important factor is the mechanism for exploration in this algorithm, which is needed to ensure that unexplored paths are visited and that the agent isn't exclusively making greedy decision. The mechanism for exploration is through the use of an ϵ-greedy policy. At any given state, the policy will make a greedy choice with probability $1 - \epsilon$ and a non-greedy choice otherwise. However, as the number of episodes increases, the ϵ should decrease since the agent will be more confident in its Q values. As such, ϵ is set to decay exponentially. Another initial parameter is the way the Q values are first initialized; they could initialized to all zeroes or randomly. Various parameters were tested for the Q-learner for each of the experiments and the best ones are shown.

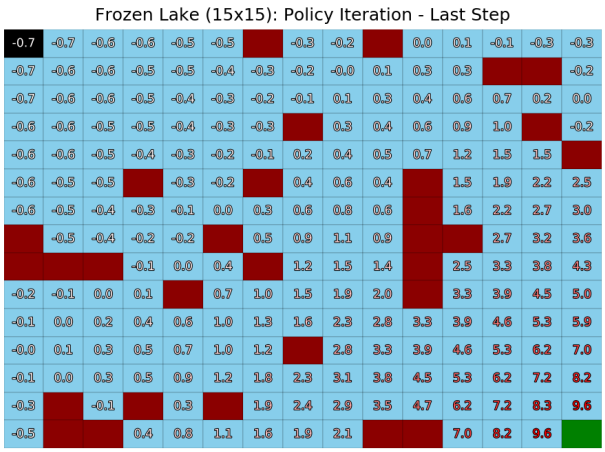
Experiments

Frozen Lake

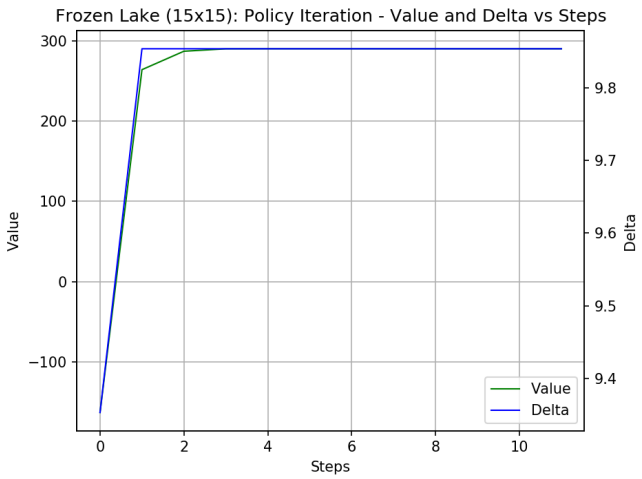
Policy Iteration

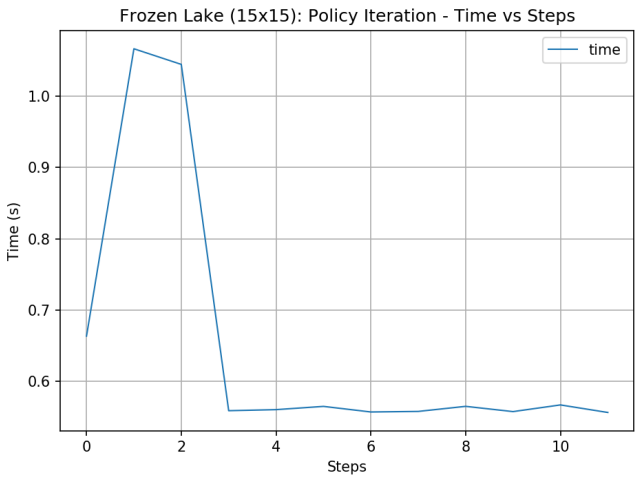
The charts below show the policy and value function at the last step of the policy iteration on a 15 by 15 Frozen Lake problem. The final policy looks quite reasonable with the policy directing the agent towards the goal and generally avoiding holes where possible. There are no loops in the policy function which may be attributed to the fact that there aren't many narrow corridors and the slippage probability is quite low at 20%. The optimal γ parameter for this MDP and this algorithm is 0.9, meaning future rewards are quite heavily discounted.





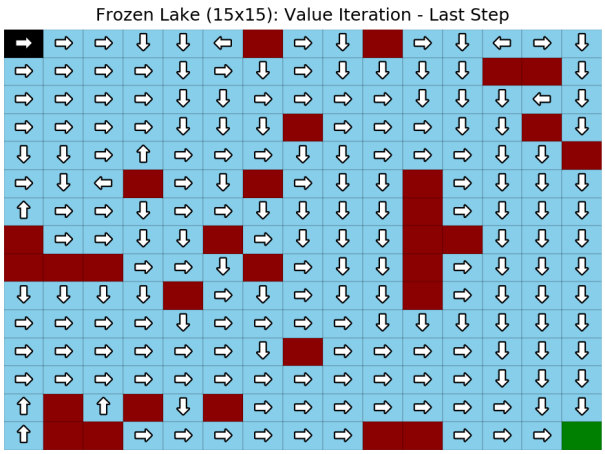
The convergence rate is fast in terms of number of iterations because for every iteration, there is an inner loop that evaluates that given policy until convergence. The value function is near optimal after the very first iteration. This does not always happen, especially for more complex MDPs, but for this straightforward problem it does. The time spent per iteration starts off very high as the policy is being revised quite a bit after each step; however, as the policy stabilizes, the time spent decreases drastically.

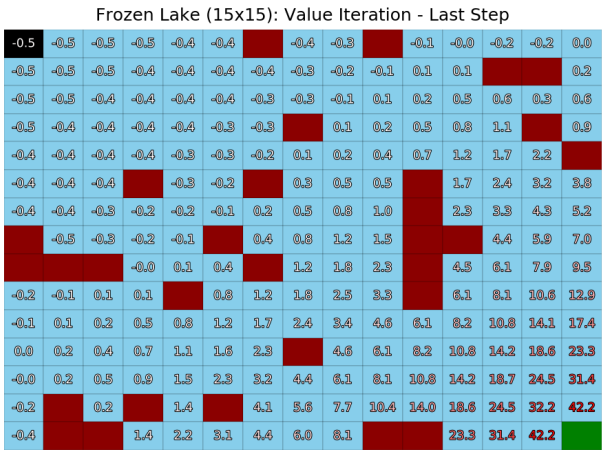




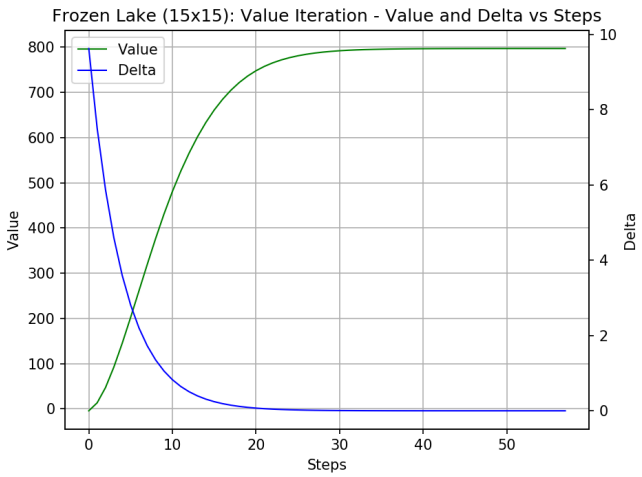
Value Iteration

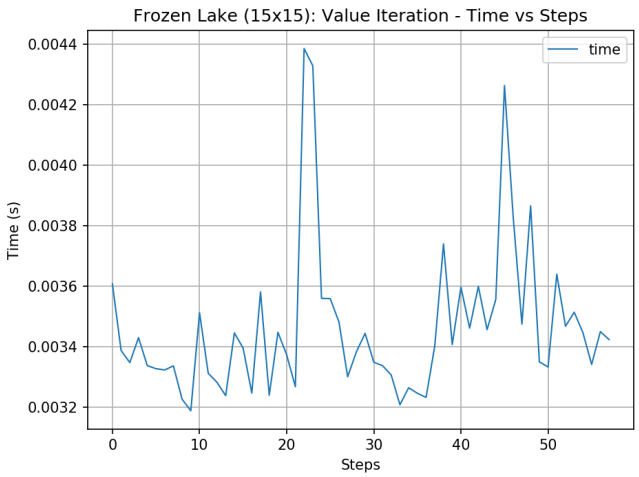
Value Iteration finds a slightly different policy and value function but it looks quite similar for the most part. This is expected since both algorithms should be able to converge on and solve the MDP given enough iterations. The optimal γ for this experiment was 0.8, which is a smaller discount rate than for Policy Iteration but still very punitive for future rewards. There are few moves in the policy that seem a bit unreasonable, for example, in cell (15, 9) and cell (4, 15), the policy suggests the agent walk into a hole. This might be caused by the fact that perhaps it believes going through the hole / resetting is more beneficial than actually going around and continuing to pursue the goal. Another reason could be that the convergence criteria is too lax and more iterations are required to find the optimal value function.





The convergence and timing charts are shown below. The value function has a smoother increase than Policy Iteration but it took substantially more steps to converge (~50 iterations for value iteration vs. 3-6 iterations for policy iteration). However, the timing chart shows that the time it takes per step is order of magnitudes lower (~0.0035 seconds vs. 0.5-1.1 seconds). This is because each step in the Value Iteration does only one pass through each state whereas the Policy Iteration does multiple passes as it needs to fully evaluate the policy at each step. In terms of total time until convergence, value iteration is much faster than policy iteration (~0.2 seconds vs. ~50 seconds).

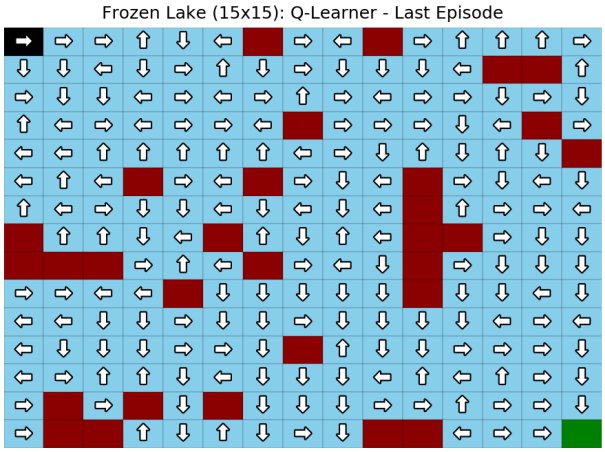


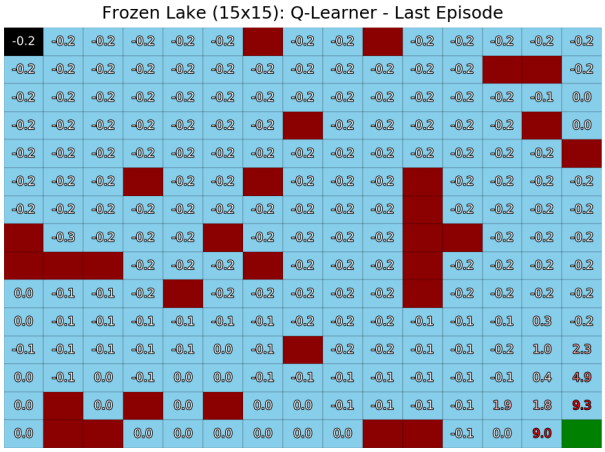


Q-Learning

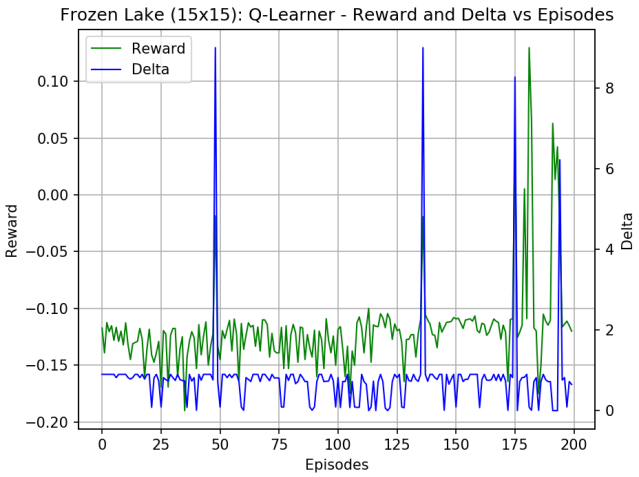
The Q-Learning algorithm assumes no information about the reward and transition functions ahead of time, which means it will take longer and more steps to converge on a policy. The best parameters for the experiment are as follows: $\alpha = 0.9$, $\epsilon_0 = 0.1$, and $\gamma = 0.5$. The large learning rate means the Q values are updated almost entirely with each iteration. A moderate discount rate of 0.5 is lower than what was seen as best for Policy and Value Iteration. The final

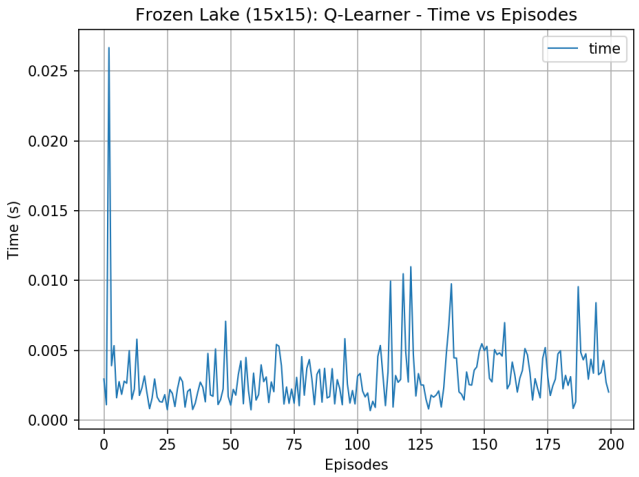
policy produced by the Q-Learning problem does not seem to be optimal but does show a general strategy that is consistent with the previous two outputs. Due to time constraints, the Q-Learner was not able to run until full completion on this large grid. The low initial ϵ means a relatively greedy strategy (i.e. low exploration) for the most part is fine for the limited training budget the learner was allotted. With more training time and iterations, a higher ϵ might be better as there is more leeway for exploration. The best Q value initialization for this MDP was random initialization.



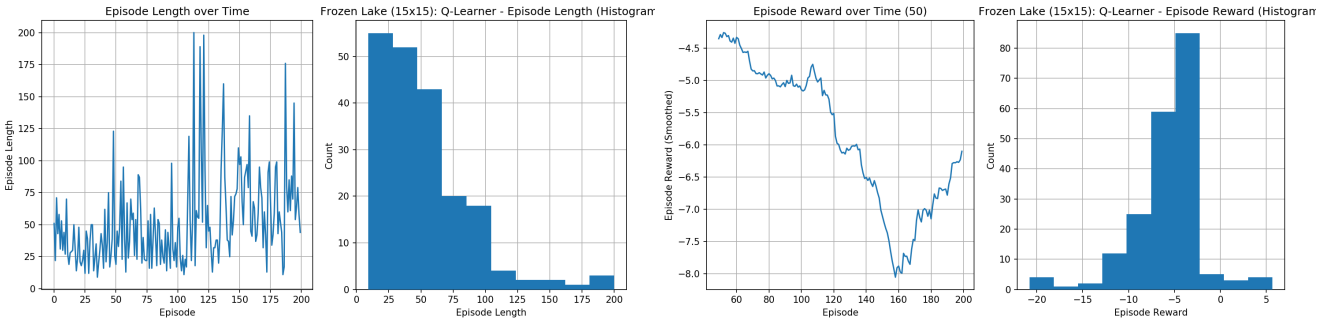


The reward chart below shows that the Q-Learner does improve over time. However, the rate of which the reward is increasing is very slow, since it is sampling just one episode from the environment at every step. The amount of time spent per episode is very short (~0.004 seconds) since it does not perform any iterative computations on each episode; the agent just needs to follow a set of actions until it reaches a terminal state.





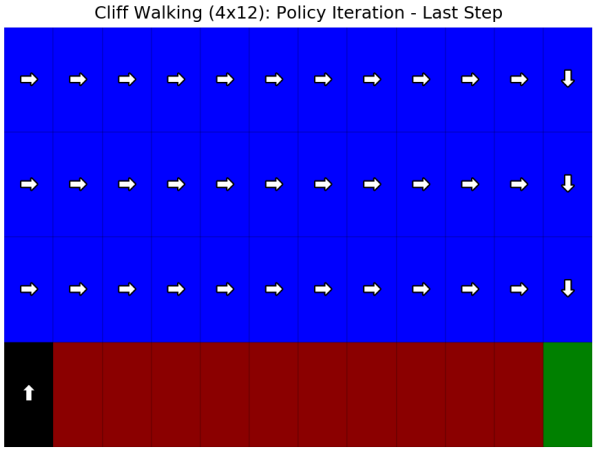
The charts below show the episode length and episode reward as a function of the number of episodes. There is a slight upward trend in the length of each episode, which is expected since the agent is likely to make it farther into the lake as it learns more about the terrain. Eventually the length will cap out at around the distance it takes from the start to the goal on the optimal path. The reward starts off as a downward trend because each move that the agent makes will make it incur a -0.1 reward, and falling into a hole also incurs a -1.0 reward. However, there is a turning point at around 160 episodes when the reward starts increasing because the agent has discovered a path to the frisbee and continue to discover shorter and shorter paths.

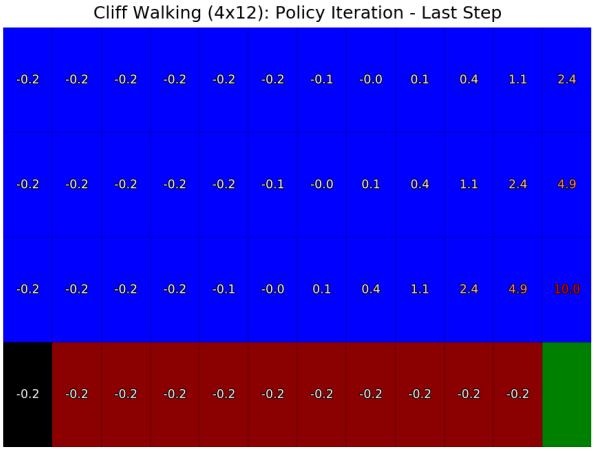


Windy Cliff Walking

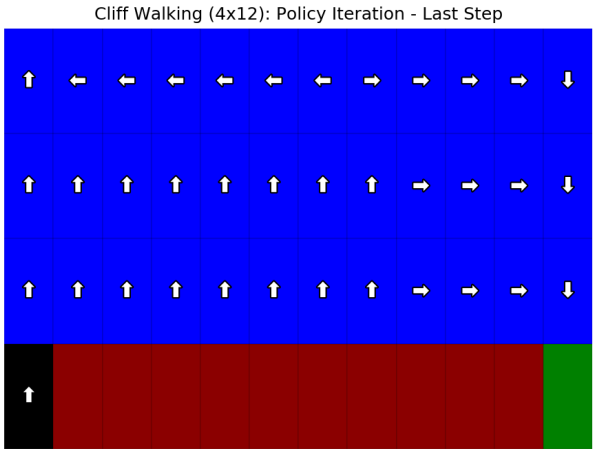
Policy Iteration

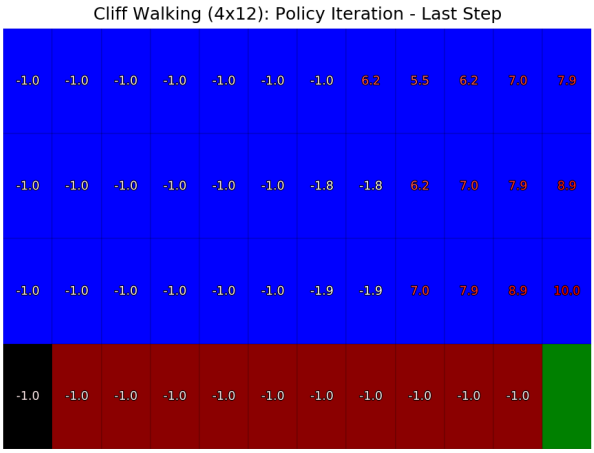
The policy iteration algorithm produced an interesting set of results for the Windy Cliff Walking problem. The ultimate policy depended heavily on the discount factor. The charts below show the policy and value functions for $\gamma = 0.5$. The policy suggests that, even with the -1.0 reward of falling down the cliff and restarting, it's still worthwhile to pursue a greedy strategy since the wind probability is reasonably low (20%).



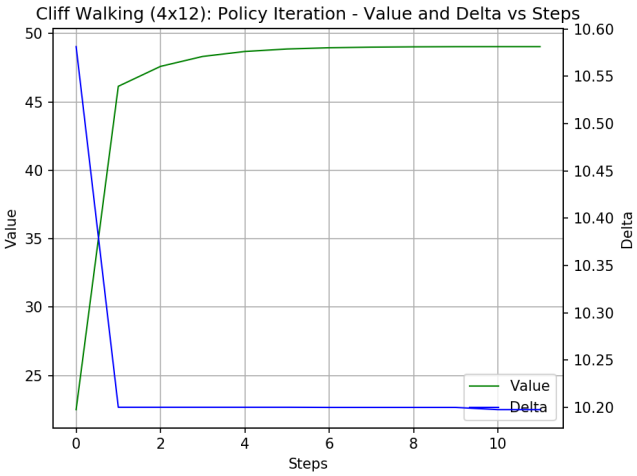


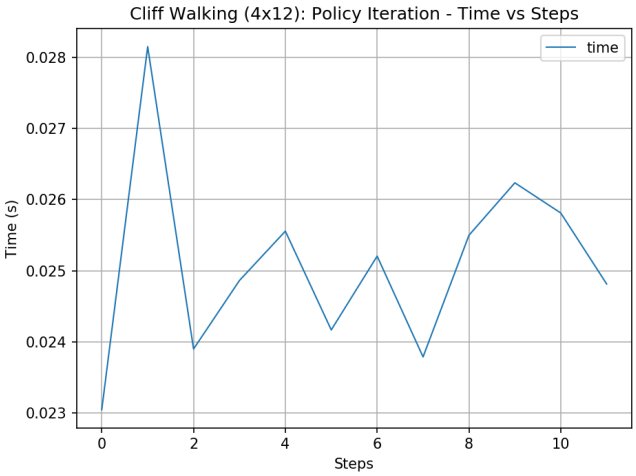
The charts below show the resulting policy and value functions for $\gamma = 0.9$. With a shorter term outlook, the agent tries very hard to avoid falling off the cliff, even if it means it won't ultimately reach the goal. The top left section of the policy function does not seem to make much sense as it's going backwards and away from the goal. These results demonstrate the difference in performance between this MDP and Frozen Lake when applying this algorithm, and the sensitivity of this algorithm to its parameters when the nature of the problem changes.





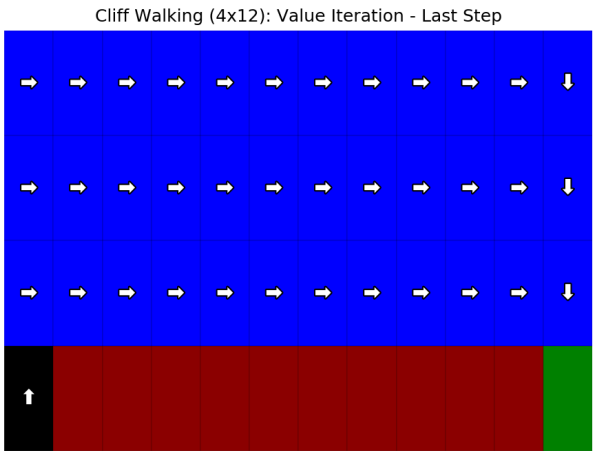
This MDP took much longer to converge compared to the Frozen Lake MDP (~10 steps vs. ~2 steps on average) despite having much fewer cells. The main reason for this is that the wind has varying strength and is completely outside the agent's control. As such, each step of the iteration will produce an updated policy that is likely to be different from the previous policy. The timing chart shows relatively stable time spent per step, which means the policy is still changing quite a bit even after 10 steps. Given more time it would be interesting to see if the timing reduces significantly when the policy becomes stable.

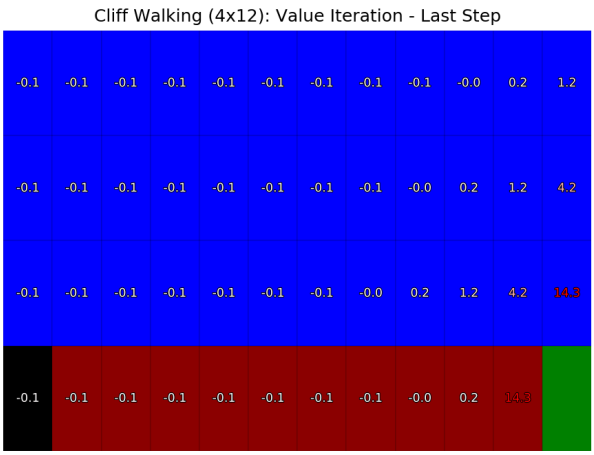




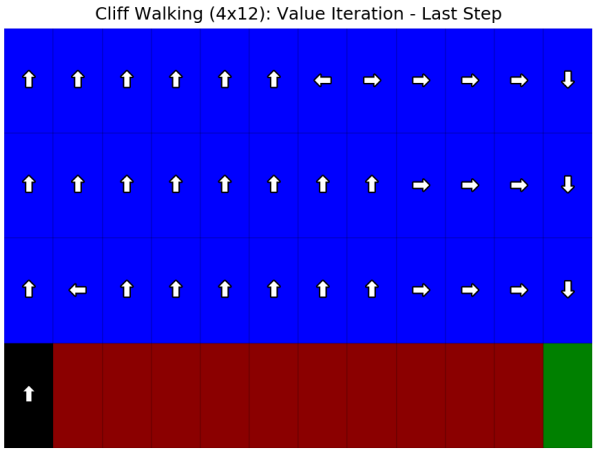
Value Iteration

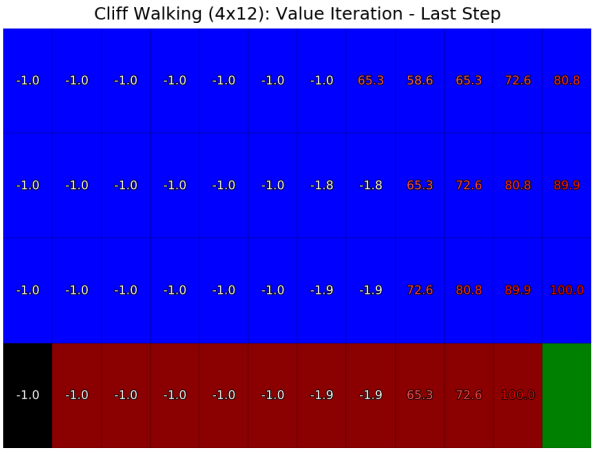
Similar to Policy Iteration, the results of this experiment are very much dependent on the discount factor. With a low discount factor of $\gamma = 0.3$, the strategy is greedy and the agent tries to reach the goal as soon as possible to reap the $+10$ reward. This happens to be the exact same policy as the Policy Iteration result with $\gamma = 0.5$.



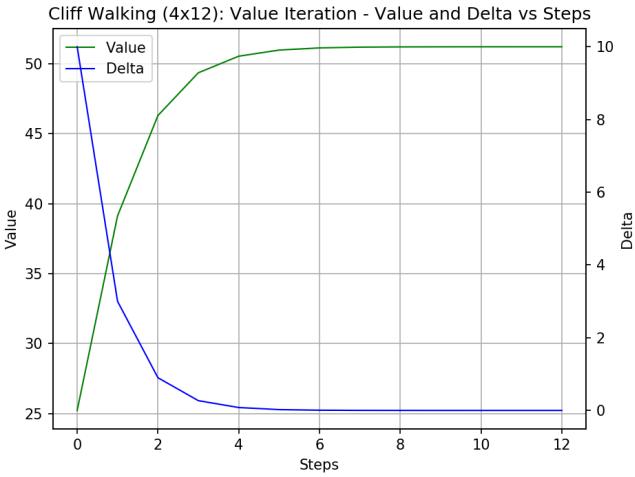


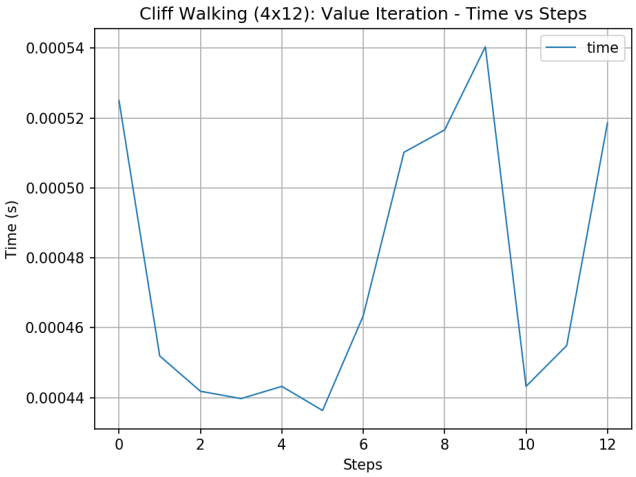
With a more short-sighted discount factor of 0.9, the policy becomes very conservative and avoids the cliff at all costs. The agent's approach seems to be to cling onto the top row and, if the agent somehow makes it halfway across the cliff, it then starts moving towards the goal. This policy seems a bit more reasonable than the high discount rate result from Policy Iteration, as it does not have as many reverse steps that go back toward the starting point.





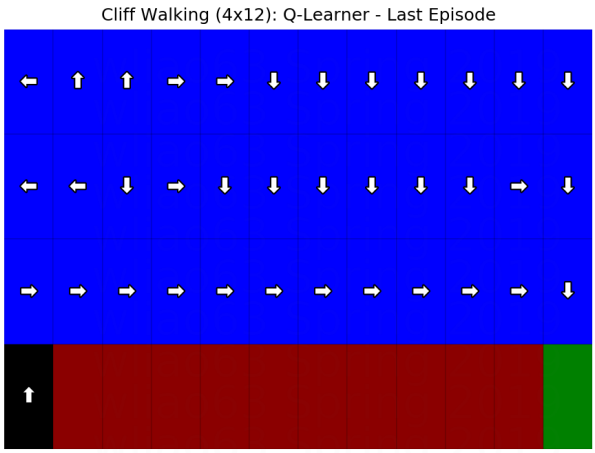
Value Iteration takes more steps to converge than Policy Iteration but does so in less time because each step is much quicker. This MDP seems to converge quite a bit faster than the Frozen Lake MDP. This is possibly because of the reduced state space and the smaller number of actions that need to be considered by the value function. The timing curve is fairly stable at approximately 0.00044 to 0.00054 seconds per iteration. Again, this is much faster than Frozen Lake's Value Iteration because there are much fewer states to consider.

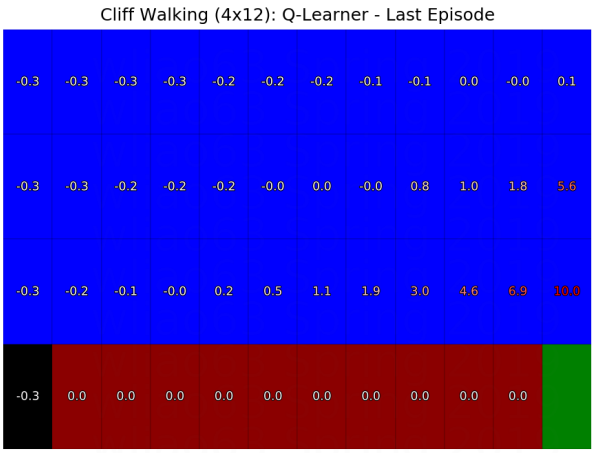




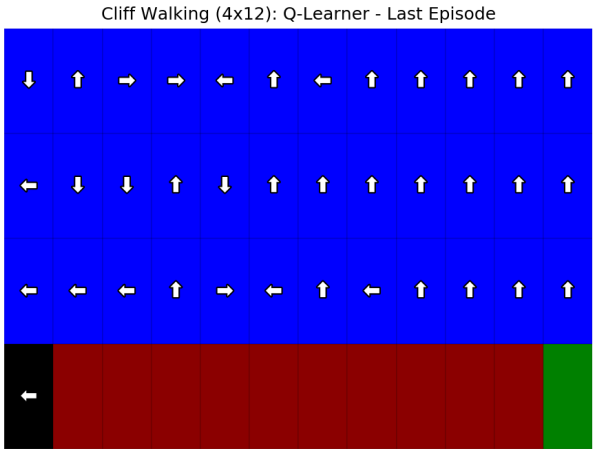
Q-Learning

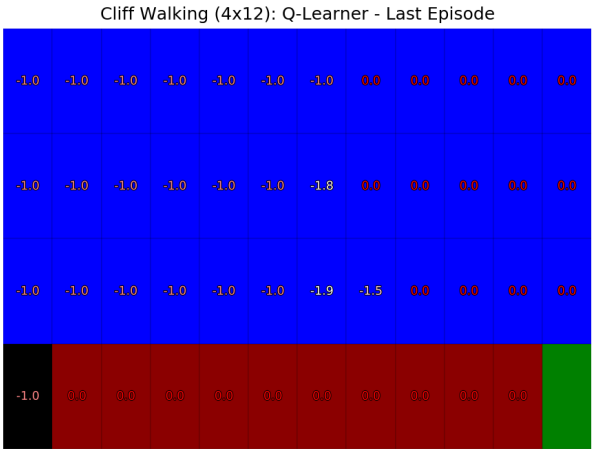
The Q-Learning results also varied based on the learning parameters. The best exploration parameter seemed to have been $\epsilon_0 = 0.5$ which is a good balance of exploration and exploitation. This is higher than the parameter for Frozen Lake ($\epsilon_0 = 0.1$) since this MDP has more complex mechanics that need repeated sampling to be fully understood. The best Q value initialization for this MDP was 0, which may have contributed to why most cells not near the goal have a Q value near 0. The policy and value functions below correspond to the parameters $\alpha = 0.1$ and $\gamma = 0.7$. The low α means the learning rate is quite slow and it takes many iterations for the Q values to move substantially. The resulting policy is very greedy and tries to reach the goal as fast as possible, which makes sense given the anchoring of the Q values and the moderate discount rate.



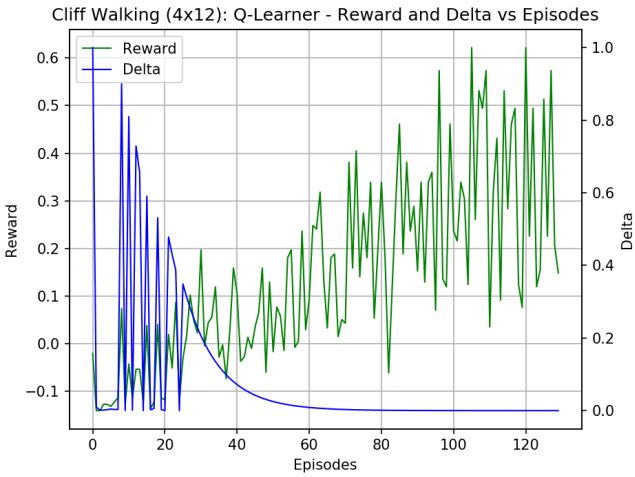


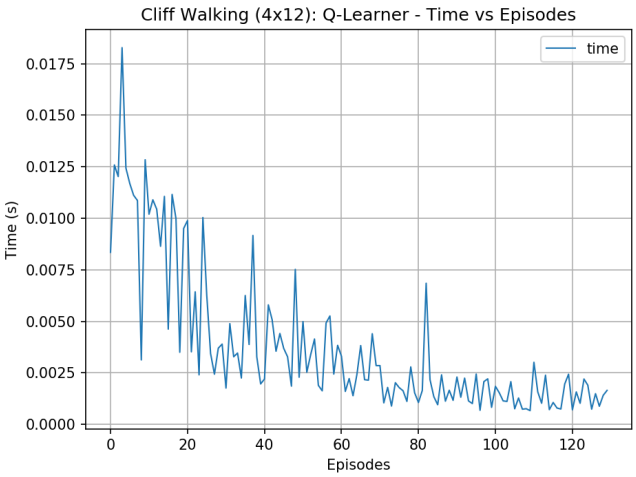
When run with $\alpha = 0.9$ and $\gamma = 0.9$, the policy becomes extremely conservative, even more conservative than the policies from Policy Iteration and Value Iteration. The agent has no desire to even go to the goal. This might be because the agent has not done enough exploration to have reached the goal yet, or have had to reset through the cliff so many times that it deems the goal as not even worth trying. The learner has not converged yet and still needs a substantial amount of training to produce a reasonable policy. This shows that without the information from the MDP / environment itself, the learning process is much slower. The learner needs a lot of time to navigate through and evaluate the environment.





The reward chart shows that the learner is definitely improving over time. However, as the number of episodes increase, the rate of learning slows down substantially. Perhaps a lower ϵ decay rate could benefit this MDP if given more training time. The time spent per episode decreases as the agent is falling off the cliff less and less (which resets the agent to the starting point) as it becomes more aware of the danger.





The episode length chart below corroborates the point about the agent becoming more aware of the cliff and falling off of it less frequently. As such, the total reward accumulated per episode becomes less negative and slowly increases towards the maximum reward possible. Eventually it will plateau to the optimal Q function but it likely has not done so yet. Given more training time, it would be interesting to see the performance of the Q-Learner with the various exploration parameters tested and determine the best set for fastest convergence.