An Analysis of Reinforcement Learning using Grid World as a Markov Decision Process

Will Said

Abstract

This document analyzes the strengths and weaknesses of different methods of forming optimal policies for Markov Decision Problems (MDP). The analysis will focus on a Grid World with a small number of states as well as a complex Grid World with a larger number of states. The analysis will conduct Value Iteration, Policy Iteration, and Q Learning reinforcement learning and compare the results on both Grid Worlds.

1 Background

1.1 Markov Decision Process

Markov Decision Problems seek to determine the best Policy π^* that will inform an agent of the best action to take at any given state in a world. Actions are stochastic, so an agent's desired action is only actually taken with a certain probability. This probability is represented as the transition function $T(s, a, s')$ which holds the probability table for being in a state $s \in S$, taking an action a, and resulting in the new state $s \in S$, where S is the set of all states.

Each state has an associated reward, and some states are sink-holes that terminate the process. Note that in an MDP, an agent's previous history plays no role in determining its policy going forward. An optimal policy π^* will give each state an estimated utility based off the rewards it can expect to receive in the future, discounted by time. This discount factor is known as gamma, γ.

1.2 Value Iteration

The goal of MDP's is to determine the optimal policy. Value Iteration determines the exact utility of each state and solves the policy based off that utility mapping. The utility of a state is counted as the current reward plus the discounted reward for all following states, assuming the optimal path is taken. Each Value Iteration gets the utilities closer to convergence, and upon convergence the optimal policy is found.

1.3 Policy Iteration

Policy Iteration iterates over policies, not values, and thus experiences faster convergence times than Value Iteration. This is because Value Iteration first seeks to converge specific values, which may occur long after the Policy has already settled. Policy Iteration starts with a random policy and repeatedly tweaks the utilities until the current policy converges and stops updating.

1.4 Q Learning

While Value Iteration and Policy Iteration require upfront domain knowledge about the world, such as all the states and their rewards, several real-world applications of reinforcement learning do not know the exact cause and effect relationship between their actions and rewards. Q Learning is one such reinforcement learning algorithm that balances Exploration and Exploitation; alternative algorithms include SARSA, an "online" method. Q Learning builds a Q Table that maps past experiences to observed rewards. It determines its policy based off Q Values for each state, which is its experiential approximation of the actual utility that may be found with complete knowledge and iteration. There are several approaches to Q Learning, but this paper will focus mainly on epsilon-greedy Q Learning, which balances the exploration-exploitation tradeoff with a ratio of ϵ. That is, the epsilon-greedy strategy explores a new action with a probability of ϵ, even if more lucrative actions exist in the Q Table.

1.5 Grid Worlds

This document will analyze two different Grid Worlds, a simple one with 25 states and a more complex Grid World with 121 states (See **Figure 1**).

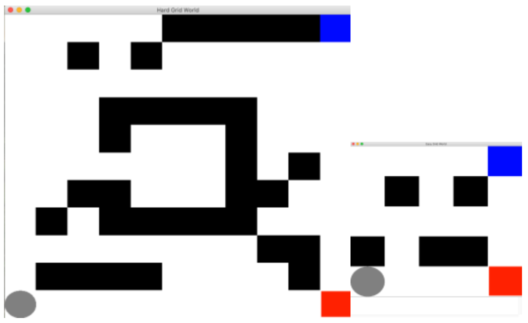

Figure 1: *Complex Grid World with 121 states (left) and Simple Grid World with 25 states (right)*

In a Grid World, the agent (shown in Gray above) seeks to maximize rewards. To do so usually involves reaching the goal state, which is shown in Blue in Figure 1 and Figure 2. The walls (shown in Black) are boundaries the agent cannot cross. The reward for any given state can vary, and in this project the reward is -1 for remaining in a state and +100 for reaching the goal. This strongly encourages the agent to quickly dash for the goal state, since any time wasted decreases the total utility. If the reward were positive, the agent might be less inclined to reach the goal state, since it could simply roam around in circles and accumulate an infinite reward.

In Grid World, actions are stochastic. This means that if the agent attempts an action $a \epsilon A$, that action will only actually happen with a certain probability that is not 1. In this project, the probability of moving in the correct direction is 0.8, and the probability of moving in any other direction is $\frac{0.2}{3}$. This is what adds even more usefulness to real-world generalizations, since the world that we are agents in is likely stochastic, or at least appears to be so. The agent learns to not take risks. The discount factor is 0.99.

1.6 Implementation

We will conduct experiments thanks to Juan J. San Emeterio's work [1] with the Brown University BURLAP reinforcement learning library [2]. The bulk of focus with this library will involve a gamma of 0.99, a learning rate of 0.99, 100 maximum iterations, a discount factor of 0.99, and the movement probabilities mentioned in the section on Grid World above: $[\frac{8}{10}, \frac{0.2}{3}, \frac{0.2}{3}, \frac{0.2}{3}]$.

1.7 Motivation

The purpose of this paper is to form simple heuristics based upon simple Grid World tests that we can then use to solve real-world problems.

For example, trading stocks is an application of reinforcement learning that offers high profits. The goal of this paper's focus on Grid Worlds is to develop rules of thumb for traders and quants who can then have a greater domain knowledge to quickly select the best algorithm given a data set and domain. Any algorithm in trading will have trade-offs, such as whether the goal is to have the lowest time complexity (for the high frequency intraday trader), the most optimal policy (for the value investor), or the highest reward (for the swing trader). The simplicity of Grid World, with its small amount of states we can easily grasp, serves as great educational material for the algorithmic trader to then understand which algorithms to use in which situations. For example, a thorough understanding of Grid World could lead a trader to impose monetary amounts as the rewards, rather than being correct or incorrect; another wise use of reinforcement learning would let the actions be buy, sell, or hold, with custom timeframes as the termination states. Knowing the time complexity of each algorithm will also help the trader decide when to use it. For example, if a certain algorithm is highly accurate but takes several minutes to run, it will be better suited to swing trading over several days, while a faster, approximating algorithm may be better for intraday trades.

2 Simple Grid World

The Simple Grid World's optimal policy involved avoiding the negative sink hole in the bottom right corner, as well as reaching the goal in the top right as fast as possible. See **Figure 2** below for the optimal policies produced within 100 iterations by Q Learning, Value Iteration, and Policy Iteration.

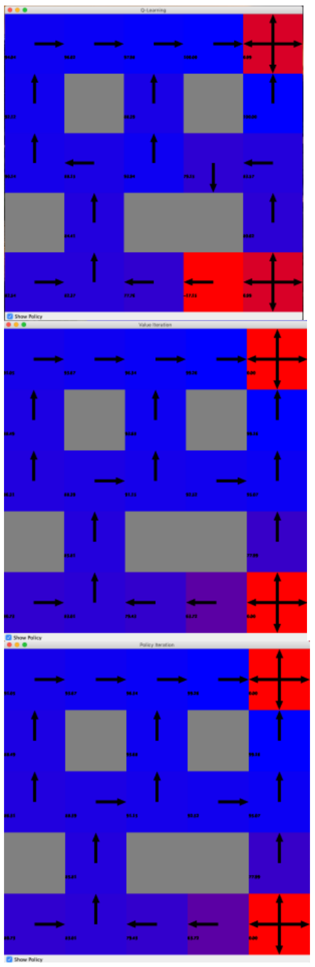

***Figure 2**: Simple Grid World policies generated after 100 iterations by Q Learning (top), Value Iteration (middle), and Policy Iteration (bottom).*

As expected, Value Iteration and Policy Iteration resulted in the same optimal policies. We expect the utilities of the states computed by Value Iteration to be more precise than Policy Iteration, although 100 iterations is apparently enough for both to converge. To test this, I ran 1000 iterations for both Iterations, which computed the exact same optimal policy. Value Iteration is more precise in its utility attributions than Policy Iteration but should takes longer to converge as a result. All three algorithms will converge given enough iterations, although Q Learning can be premature, yet 100 iterations was not enough for Q Learning to converge even with the simple Grid World. As can be seen in the figure above, the policy produced by Q Learning is far from ideal: it even decides the best action in some states is to ram its head into the wall. Keep in mind that there is a 20% chance of going in any 3 of the wrong directions. The Q learning model deviates from the optimal policy in three states, all of which are on the third row. These states are important since they are halfway between the high reward goal state in the top right and the penalty sink hole in the bottom right. It is interesting how Q Learning actually seems more risk averse than the optimal policy. Notice how it decides to be as far away from the sinkhole as possible, taking the long route to the opposite corner. Another example is how it labels the state adjacent to the sink hole as having a negative utility (-17.15), while the optimal policies display a much larger, positive, utility of 63.72.

The steps taken to reach the goal, per iteration, are shown in **Figure 3**. As expected, Q Learning's lack of a priori knowledge leads it to not converge within 100 iterations, even in the simple Grid World. Value Iterations and Policy Iterations were able to approach convergence by the 5th iteration, reaching average steps of around 12. Q Learning's iterations continually spiked above 50 steps to reach the end goal in the upper right corner, never solidifying underneath 15. It is worth noting how Policy Iterations, and especially Value Iterations, spike on the first and second iteration, far above the very steady Q Learner. This is due to the nature of the iteration algorithms, which do the majority of the necessary work upfront and only have to tweak it slightly afterwards.

Even 1000 iterations, ten times more than previously, was not enough for Q Learning to converge, given the same hyperparameters of a learning rate and discount factor of 0.99. See **Figure 4** for the lack of convergence. Recall that only 15 iterations were needed for Value and Policy Iteration to converge.

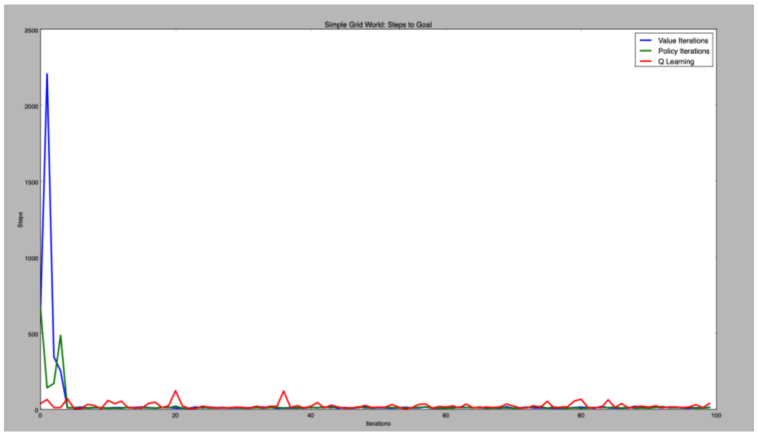

Figure 3*: Simple Grid World steps to reach goal, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

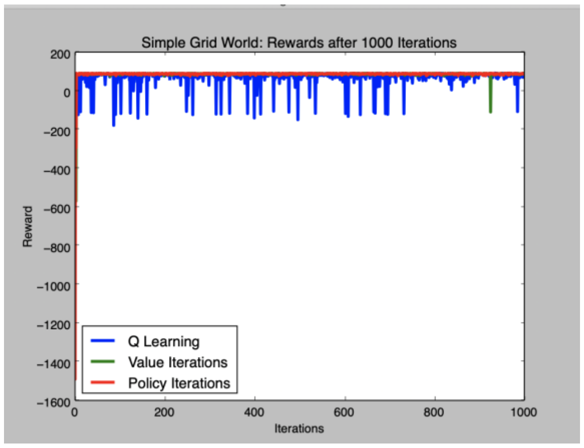

Figure 4*: Simple Grid World rewards after 1000 iterations, 10x the normal sample size. Q Learning is Blue, in the background, while Iteration is in front.*

The total reward accumulated by each of the algorithms over 100 different iterations is shown in **Figure 5**. The figure shows the same results as the steps taken, just with the vertical axis flipped. Value Iterations and Policy Iterations both spike to the downside yet settle to a positive reward of +90 within five iterations, just like before. The Q Learner becomes mostly positive, approaching +90, by the 15th iteration, then continues to occasionally bounce to the downside quite often, long after the iterators converged. Again, due to the symmetry of the rewards, this chart is just the flipped version of **Figure 4** above. Interesting results would occur if there was no symmetry among the rewards; for example, I could charge a penalty for hitting a wall, or make the reward of +100 not equal to the sink hole of -100. In a real-world application, such as stocks, we would tend towards placing penalties that are much higher than end goal rewards to force the agent to stay within certain bounds. For example, the (negative) reward for losing 50% of a portfolio should equal twice as much as gaining 50%, since a 50% loss will require a 100% gain to return to the baseline.

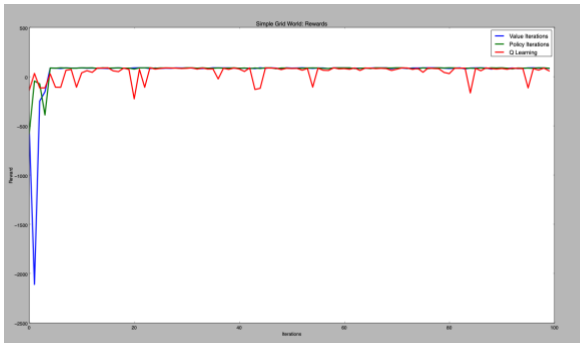

Figure 5*: Simple Grid World average reward accumulation, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

The time, in milliseconds, taken for Value Iterations versus Policy Iterations to compute the optimal policy given the number of iterations the algorithm was run is shown in **Figure 6**. Note the strongly linear increase in time needed to compute a policy as the number of iterations increased. This is expected since each additional iteration does not rely on past results, by the Markov Assumption, and each additional iteration only adds a constant time complexity to the total. Policy Iterations required more time to generate an optimal policy than Value Iterations, which is largely unexpected behavior since Value Iteration tends to converge its optimal policy long before it reaches value convergence. Value Iterations first seek to find the optimal value function, and then uses that to easily find the policy in one iteration. On the other hand, Policy Iteration concerns itself only with the policy, not the optimal utilities, so despite using the same Bellman Equation it tends to be much faster. As soon as another iteration doesn't yield a better policy, Policy Iteration ends. Further study is needed to determine if a statistically significant difference can be discerned for this theoretical difference in less than 100 iterations. Q Learning appears substantially faster than either Iteration, likely due to its approximation requiring less time complexity than a fully worked out exact policy.

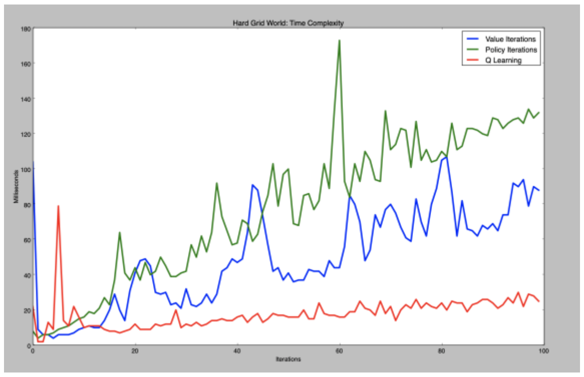

Figure 6*: Simple Grid World average time to calculate the policy in milliseconds, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

The learning rate throughout this project is 0.99, yet I still experimented with various learning rates to determine the ideal hyperparameter. Manipulating the learning rate between 0.10 and 0.99, Value Iteration and Policy Iteration were always able to converge within 100 iterations. Q Learning fared best (in terms of accumulated reward) with a learning rate around 0.50, as can be seen in **Figure 7**. Not including one massive peak—which can also be seen occasionally even in the Iterators—0.50 appears to be the best learning rate for this data set for Q.

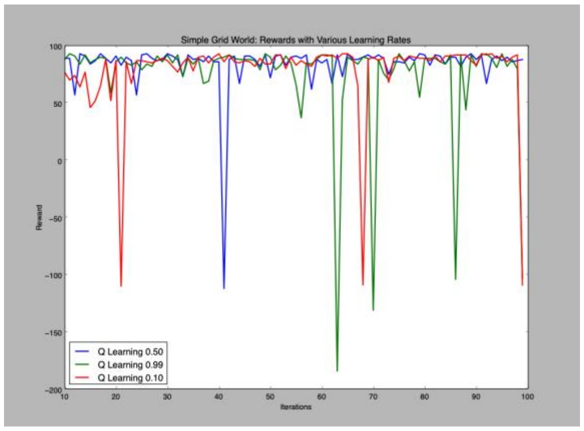

Figure 7*: Simple Grid World Q Learning Reward Accumulation on 1 to 100 Iterations, given different learning rates of 0.10, 0.50, and 0.99.*

Despite the new learning rate's ability to make Q Learning to appear to converge within 100 iterations, the policy is still not optimal and does not match the Iterators, as can be seen in **Figure 8**. Only one of the two differences is a true error, and only a slight one at that. More iterations would be needed to achieve optimality. This because a lower learning rate will eventually converge given enough iterations, while a high learning rate may reach a false convergence faster. However, even with 1000 iterations the optimal policy was not reached (See **Figure 9**).

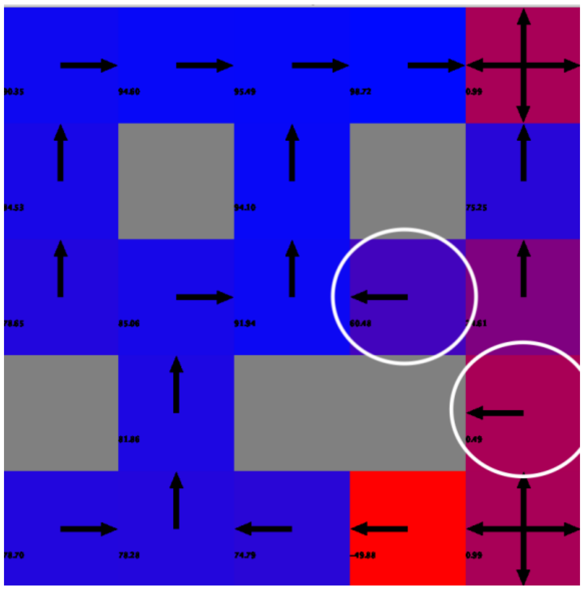

Figure 8*: Simple Grid World Q Learning Policy formed with a learning rate of 0.5 after 100 iterations, with non-optimal states circled.*

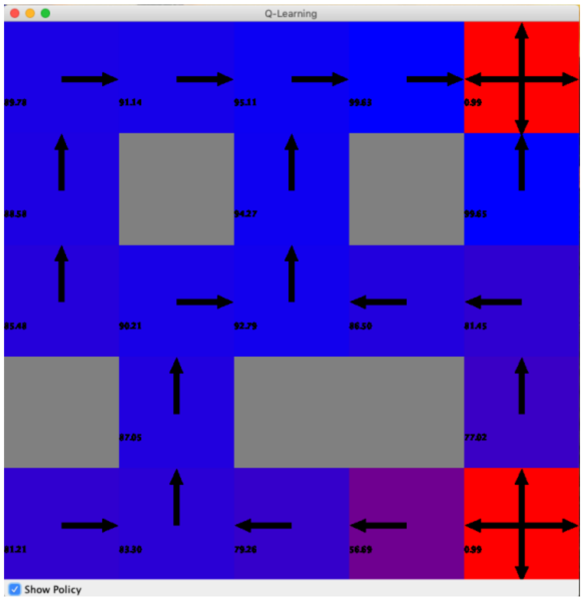

Figure 9*: Simple Grid World Q Learning Policy formed with a learning rate of 0.5 after 1000 iterations.*

The ideal discount factor for Q Learning reward accumulation appeared to be 0.90, as 0.99 (the default used throughout this exercise) is too immediate and 0.80 is too focused on the long-term outcome (in this case, lowering the expected utility too much for all states). See **Figure 10**.

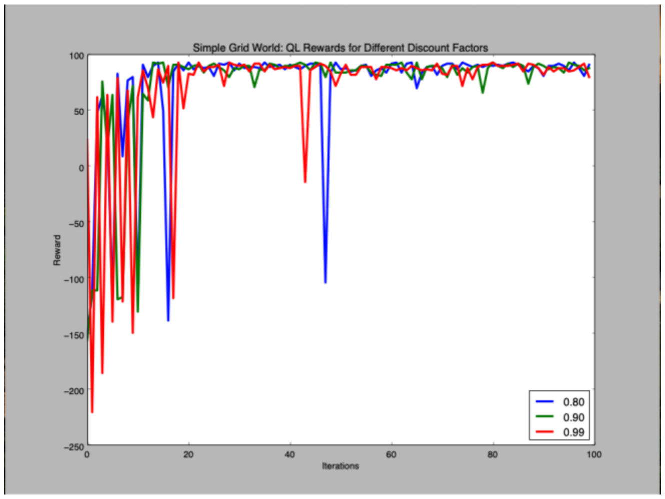

Figure 10*: Simple Grid World Q Learning Reward Accumulation among different discount factors of 0.80 (Blue), 0.90 (Green), and 0.99 (Red).*

Manipulating the Epsilon values (exploration-exploitation tradeoff) did not seem to have much weight on the outcome. Recall that the default Epsilon was 0.10 for this entire project. I experimented with Epsilons as shown in **Figure 11** and was not able to find any difference in rewards or policy optimizations based on epsilons for this data.

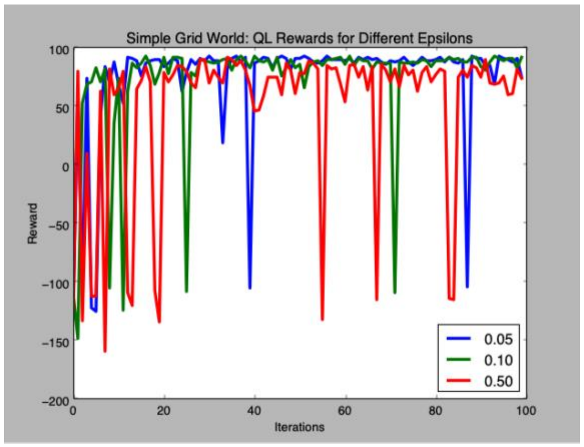

***Figure 11**: Simple Grid World Q Learning Reward Accumulation among different epsilon values of 0.05 (Blue), 0.10 (Green), and 0.50 (Red).*

3 Hard Grid World

The more Complex Grid World's optimal policy also involved avoiding the negative sink hole in the bottom right corner as well as reaching the goal in the top right as fast as possible. See **Figure 12** below for the optimal policies produced within 100 iterations by Q Learning, Value Iteration, and Policy Iteration.

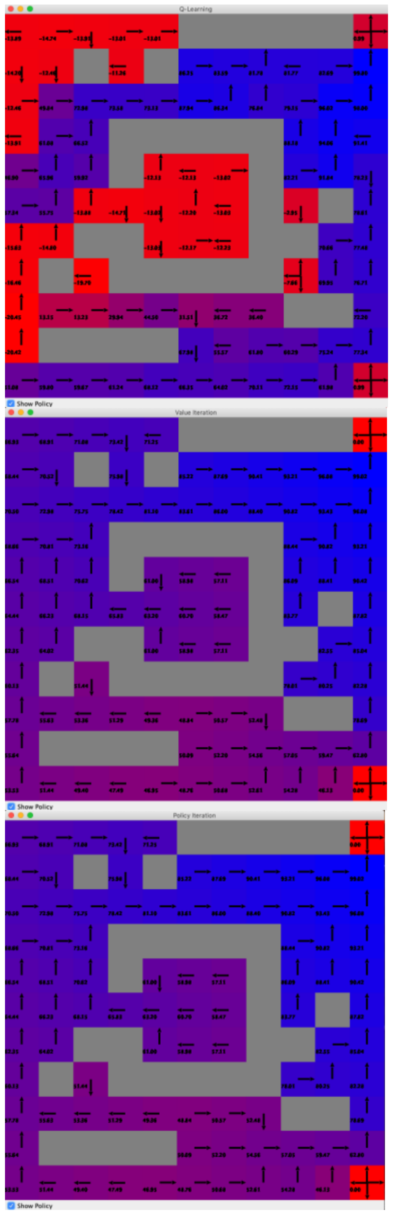

***Figure 12**: Complex Grid World policies generated after 100 iterations by Q Learning (top), Value Iteration (middle), and Policy Iteration (bottom).*

The optimal policies produced by Value Iteration and Policy Iteration are the same, as expected. This is because the process of Value Iteration, by definition, produces the most optimal policy upon convergence, and 100 iterations was more than enough for convergence to occur in the iterative algorithms.

Q Learning resulted in an awful policy after the 100 iterations. It recommends beelining straight for the sinkhole in the bottom right corner and dodging it at the last moment. Although this would result in the same outcome in the same number of steps in a deterministic world, Grid World is stochastic, and the chance of falling into the -100 sink hole in this policy is $\left(\frac{0.2}{3}\right) + \left(\frac{0.2}{3}\right) \approx 13\%$, while the optimal policy produced by iteration only has the chance of $\left(\frac{0.2}{3}\right) * \left(\frac{0.2}{3}\right) * \left(\frac{0.2}{3}\right) * \left(\frac{0.2}{3}\right) * (\dots) \approx 0$. Of course, this is because the Q Learner has not *experienced* enough mistakes in that corner for it to know that it is walking on thin ice. Given the fact that this algorithm learns only from observations, we cannot blame it too much for making wrong decisions; instead, we must learn to prefer Iteration when the information is available to us, and understand the limitations of Q Learning when the model is not available to us.

As with the simpler Grid World, Q Learning appears much more risk averse and pessimistic in its early iterations. It has many negative utility values, while the Iterations do not have any negative utility states. This is because the likelihood of obtaining the +100 reward at the goal state far outweighs the -1 penalty on the non-terminal states, on this 11x11 Grid World. The state furthest from the goal in the optimal policy, the state three spaces to the right of the initial state, is only 23 states away from the end goal, which explains why no negative states exist in the Iterations.

The steps taken to reach the goal state are shown in **Figure 13**. The same spike occurred in the first few iterations with Value and Policy, so I removed the first 10 altogether from the chart to compare the remaining 90 iterations, which are less noisy and have more to offer. As before with the simpler Grid World, Q Learning takes more steps and never quite converges to the optimal policy within 100 iterations. Policy Iterations and Value Iterations report similar results, since they both converged by the 10th iteration (shown as the start of the chart). This is because the convergence occurred by the 10th iteration, as can be seen in **Figure 14**.

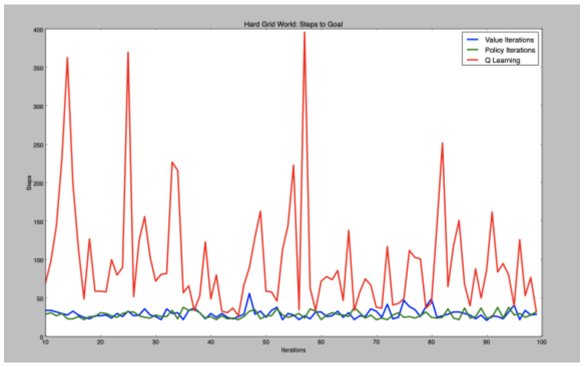

Figure 13*: Complex Grid World steps to reach goal, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

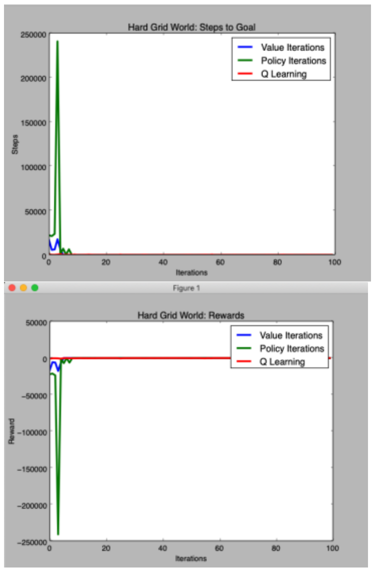

Figure 14*: Complex Grid World convergence appears within 10 iterations, as shown. These are zoomed-out versions of the exact same figures above.*

Figure 15 shows another interpretation of the same results that were shown above. This charts rewards versus iterations and depicts the same phenomenon on just a flipped vertical axis, as discussed above for the Simple Grid World.

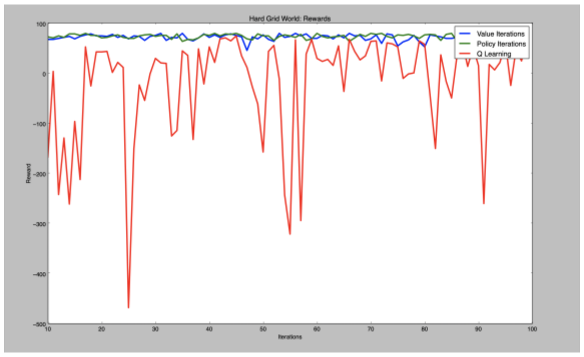

Figure 15*: Complex Grid World average reward accumulation, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

The time complexities (see **Figure 16**) showed the same overall outcome—Policy Iteration took the longest, while Q Learning was the fastest—although the differentiation was not as well defined. In the simpler world, the distinctions were clearly separated in a significant manner. Policy Iteration's spike was also observed in the simple Grid World.

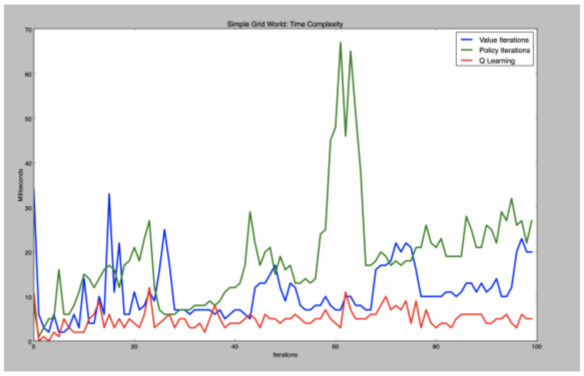

Figure 16*: Complex Grid World average time to calculate the policy in milliseconds, from 1 to 100 iterations. Q Learning is Red, Value Iteration is Blue, and Policy Iteration is Green.*

4 Conclusion

The motivation behind this research, as proposed in the introduction, is to explore various reinforcement learning techniques in order for real world applications, such as trading, to benefit from a broader understanding of the right algorithms to use given a particular data set. We now know that Value and Policy Iteration converge to an optimal policy much faster than Q Learning does, and it is therefore the better solution when the domain knowledge is given. When the domain knowledge is not given, Q Learning is an example of a reinforcement learning algorithm that can be used to build an approximation to the ideal policy, given no knowledge and only trial and error. As discussed above, the optimal Q Learning hyperparameters for this Grid World MDP are a discount factor of 0.90, a learning rate of 0.50, an epsilon of 0.10, and as many iterations as needed to converge; in this case, at least 1000.

The number of states had no noticeable impact on the relative performances of the algorithms; rather, the increase in complexity in Complex Grid World served to exaggerate the differences that were already observed in Simple Grid World.

Although I expected Policy Iteration to determine the optimal policy in the least amount of time, with Value Iteration second, it was quite the opposite in practice, with Q Learning being the fastest on the given data sets and Value Iteration coming second. As discussed, this was due to the specific ways the hyperparameters were tuned in this assignment, with the learning rate, discount rate, and epsilon all equal to 0.99.

Further examination of Markov Decision Problems must be analyzed specifically relating to a given area, such as stocks, before betting money on the results of this paper. Always consult a financial advisor before making trading or investment decisions.

5 References

[1]: See https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4 for his work.

[2]: See http://burlap.cs.brown.edu for BURLAP.