Reinforcement Learning Analysis, Grid World Applications

Kunal Sharma

GTID: ksharma74, CS 4641 Machine Learning

Abstract

This paper explores two Markov decision process problems with varying state sizes. One has a relatively small number of states (>15) while the other has a relatively large number of states (>500). Throughout this paper, the following three reinforcement learning algorithms will be analyzed: Value iteration, policy iteration, and Q-Learning. Particular attention will be paid to understanding how each learning algorithm responds to an urgency and risk tradeoff. Where urgency is defined as a "living penalty" or a cost for any movement in the state space. In this context, "risk" is a path through the state space that has a high chance of acquiring a relatively large penalty. This paper will also analyze the runtime, number of steps to get to the goal, and reward accumulation for each of the reinforcement algorithms.

This paper uses the Grid World implementation from the Brown-UMBC Reinforcement Learning and Planning Java open source library. Microsoft Excel was used for data visualizations.

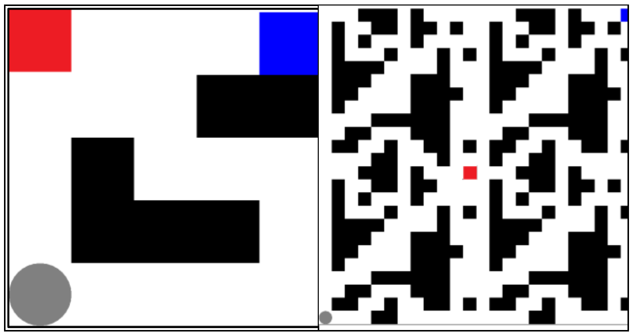
Section 1: Overview and Key Concepts

1.1 Problem Definition Overview

Markov Decision Processes is a mathematical framework that depends on the Markov assumption which explains that the probability of future states is not dependent on any information from past states, but rather solely dependent on the present state that an actor or decision maker is in. At a high level, MDP seeks to maximize the total reward by creating an optimal policy for an actor to navigate through a state space. In many applications, the actor must face a number of variables that are random in nature. MDP models are able to incorporate this stochastic element into their policy making.

This paper will present two Markov Decision problems that both fundamentally explore whether an actor resulting from various reinforcement algorithms may decide to take riskier paths through a state space given a greater urgency to reach the terminal state. In both problems, to get to the reward state there is a single shorter path and one or more longer paths. A penalty is placed adjacent but not directly on the shortest path. The images below represent the state spaces of the two problems. (Red Box: Penalty, Grey Circle: Actor, Black Box: Wall, Blue Box: Reward/Terminal State)

Easy Grid World	Hard Grid World

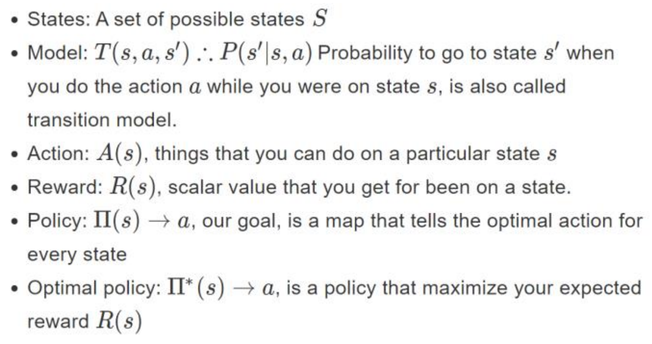


Since the environment is stochastic, specifically an actor has an 80% chance of taking its intended action and a 20% of taking an unintended action. For example, an actor may choose to go forward but will move right instead. This represents the "risk" for an agent to take the shorter path as it may slip into the penalty square and accrue a large penalty. The greater the penalty, the greater the risk. The "urgency" factor can also be a "living penalty", a penalty for every action that an actor takes. This penalty is intended to encourage the actor to reach a terminal state and helps reinforcement algorithms converge. The greater this living penalty, the faster an actor would wish to exit the state space.

While Grid World experiments might seem overtly abstract and overly simplified, it is because of this that they can be used to study various real world applications. Specifically, this Grid World experiment could be useful in transportation and delivery route planning. Suppose a delivery truck has to deliver a product purchased online to a customer by driving through the city. It can take the shorter route, but it would need to pass a busy set of train tracks. Most of the time trains are not passing by, but when they are the road is blocked and significant time can pass. If the product is not delivered on time, the company can face a monetary loss. The penalty in this case represents the duration that a train passes in the event that it does so. The urgency or living penalty represents cost to the company for late deliveries.

1.2 MDP

Markov Decision Process or MDP is a framework that follows the aforementioned Markov property and is used in order to help make decisions in a stochastic environment. Below are the key concepts essential to MDP problems.



- States: A set of possible states S
- Model: $T(s,a,s') \therefore P(s'|s,a)$ Probability to go to state s' when you do the action a while you were on state s, is also called transition model.
- Action: $A(s)$, things that you can do on a particular state s
- Reward: $R(s)$, scalar value that you get for been on a state.
- Policy: $\Pi(s) \rightarrow a$, our goal, is a map that tells the optimal action for every state
- Optimal policy: $\Pi^*(s) \rightarrow a$, is a policy that maximize your expected reward $R(s)$

Credit: Leonardo Araujo Santos [2]

MDP uses a concept of utility that is value of a state representing a cumulative delayed conditional reward that is typically calculated using some form of the Bellman equation.

1.3 Value Iteration

Value iteration assigns a utility value of every state. This value represents the extent to which the state is helpful in finding the optimal solution or policy. To find this utility, the immediate reward for the states with a reward is found. Neighboring state utilities are then found by applying a reward value with a particular discount value and assuming that the actor only makes the best choices in terms of state transitions from that state to the reward state.

More specifically, the Bellman equation is used to find the utility of all of the states in the state space. The equation is as follows:

$$\hat{U}(s)_{t+1} = R(s) + \gamma max_a \sum_{s'} T(s,a,s')\hat{U}_t(s')$$

Initially, all states are assigned random values. The utility for every state is updated until convergence is reached. Convergence is reached when the changes in the utilities are less than a selected threshold.

1.4 Policy Iteration

Value iteration is often very slow to converge since it is calculating the optimal policy indirectly. However, in policy iteration, policies are iterated over instead of states. First, a random policy is selected. Then, the utility for every state relative to the current policy is calculated. Finally, from the computed policies an optimal policy is chosen. The Bellman equation is used in policy iteration as well, the key difference lies over whether states or policies are iterated through to find the optimal policy.

1.5 Q-Learning

Value and policy iteration are similar in that they both assume the knowledge about all of the possible transition models and rewards in a state space. Very often however, this information is not present or easy modeled in real world applications.

Q-Learning is very different from value and policy iteration as it is only given information about the possible states and all actions that are possible at each state. Q-learning builds its intuition of the optimal policy from only this information and several trials of exploration through the state space.

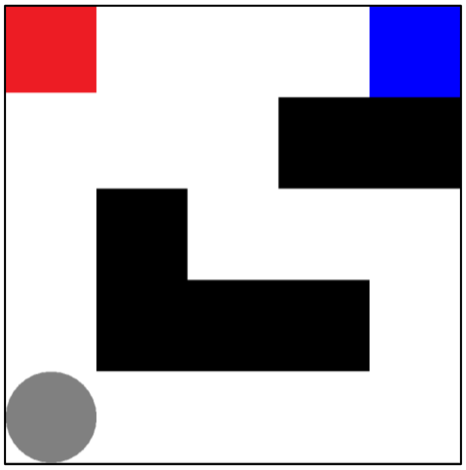
To do this, Q-Learning assigns to each state a value referred to as the Q-value. When a state is visited, a reward is given and the Q-value is changed. The Q-value attempts to leave each state in the most optimal manner as it visits new states, unless it is exploring new routes. Q-Learning builds its optimal policy as a result of its exploration. Q-value can be represented by the following equation:

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') max_{a'} Q(s',a')$$

Similar to value and policy iteration, the Q-learning algorithm continues to run until convergence is reached. While it is important to follow the best path to get closer to the solution, it is also important for the Q-learning algorithm to explore new areas in the state space in hopes that an even better solution would be found. This is known as the exploit-explore problem. The Q-learning algorithm does both in its search for the optimal policy.

Section 2: Grid World Easy Problem

2.1 Problem Overview



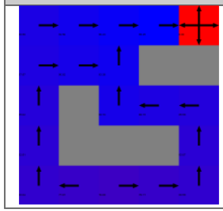
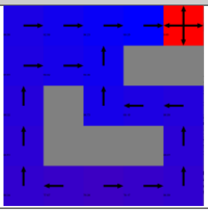
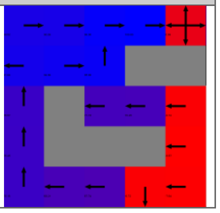
The first MDP problem is relatively small, a 5 by 5 gird. The agent is represented by the grey circle on the bottom left corner. The red square represents the penalty and the blue square represents a reward of 100 as well as the terminal state. The black squares represent walls, the agent cannot move onto these locations.

The short path in this problem is the one straight in front of the agent (8 steps to reward). The longer path is to the right of the agent (12 steps to reward). The agent only risks the penalty if it takes the shorter path. To understand how the reinforcement algorithms tradeoff risk and urgency we can test various values for risk and urgency listed in the quadrant below.

	Low Risk	**High Risk**
Low Urgency	Step Cost = -1 Center = -10	Step Cost = -1 Penalty = -150
High Urgency	Step Cost = -5 Penalty = -150	Step Cost = -5 Penalty = -150

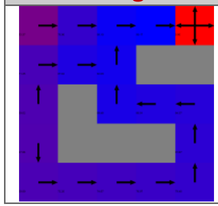
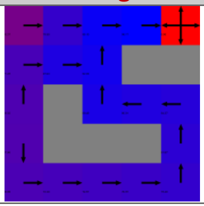
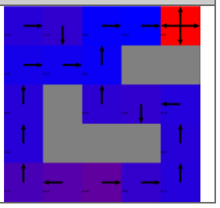
2.2 Behavior Analysis

a. Low Urgency and Low Risk

Policy Iteration	**Value Iteration**	**Q-Learning**
Short	Short	Short
		

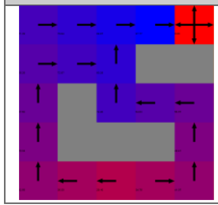
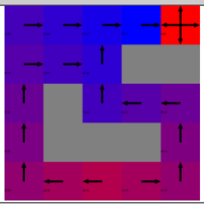
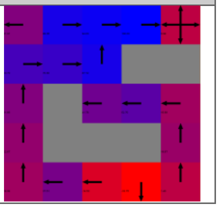
As expected, given low urgency and low risk, all of the reinforcement algorithms made the rational choice of taking the shorter route. While value and policy iteration will go right if in the middle of the bottom row, the Q-Learning algorithm insists on going left quite possibly because it is biased by its initial exploration.

b. Low Urgency and High Risk

Policy Iteration	**Value Iteration**	**Q-Learning**
Long	Long	Short
		

Given low urgency and high risk (Penalty = -150), all of the reinforcement algorithms made the rational choice of taking the longer route except for the Q-Learning algorithm. This may because the Q-Learning algorithm does not converge in 200 iteration like policy and value iteration, as can be seen in section 2.3.c. With more iterations or a higher exploration rate, it is likely Q-learning would prefer to go right.

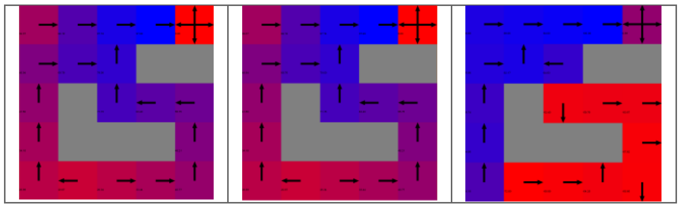
c. High Urgency and Low Risk

Policy Iteration	**Value Iteration**	**Q-Learning**
Short	Short	Short
		

Given low urgency and high risk (Step Cost= -5, Penalty = -10), all of the reinforcement algorithms made the rational choice of taking the shorter route. Q-Learning has a few off directional arrows including the one in the center and to the left of the bottom right corner. This again is likely because it has not yet converged.

d. High Urgency and High Risk

Policy Iteration	**Value Iteration**	**Q-Learning**
Short	Short	Short



Given high urgency and high risk (Step Cost= -5, Penalty = -150), all of the reinforcement algorithms chose to take the risk of a high penalty this time instead of avoiding it. Q-Learning has a several off directional arrows on the bottom right quadrant that would cause the agent to accrue a greater step penalty.

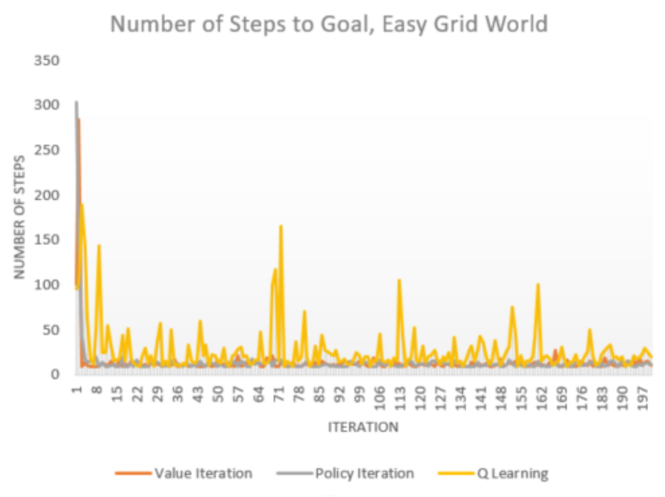
e. Analysis

	Low Risk, P = -10	High Risk, P = -150
Low Urgency, SC = -1	Policy: Short Value: Short Q-Learning: Short	Policy: Long Value: Long Q-Learning: Short
High Urgency, SC = -5	Policy: Short Value: Short Q-Learning: Short	Policy: Short Value: Short Q-Learning: Short

High urgency or step penalty increases the agent's likelihood of taking on a riskier route. Typically, agents are risk averse and will opt for the shortest path. Further analysis that tests more values for step costs and penalty would be key to getting a clearer understanding of the relationship between risk and urgency.

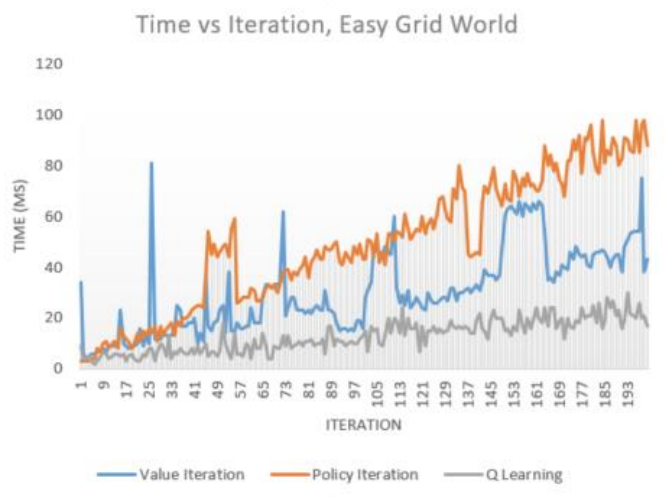
2.3 Performance Evaluation

a. Steps to Goal



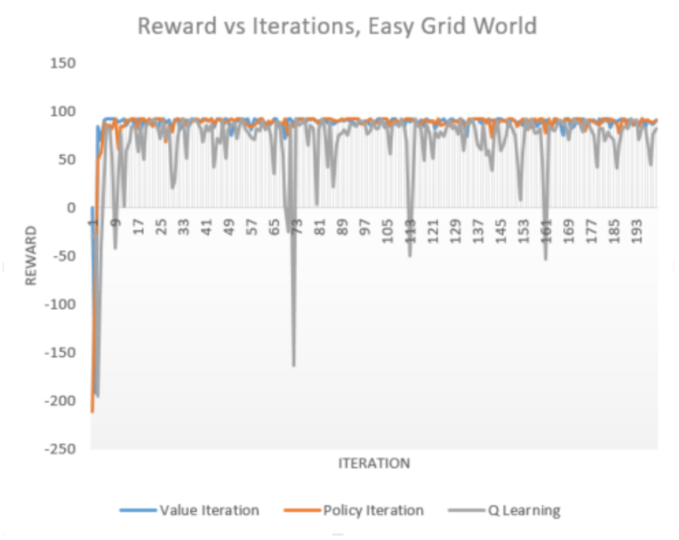
Policy iteration and value iteration both converge within 10 iterations. Q-Learning has several spikes that seems to get smaller but it does not seem to definitely converge within the first 200 iterations. The spikes likely represent the Q-learning algorithm visiting unexplored states instead of following its existing optimal plan strictly. This would explain why the spikes seem to get smaller over time.

b. Time



The graph above displays the runtime for the three reinforcement algorithms. Policy iteration takes the longest time, likely because it must permute through policies to find the optimal one requiring a substantial number of calculations.

c. Reward

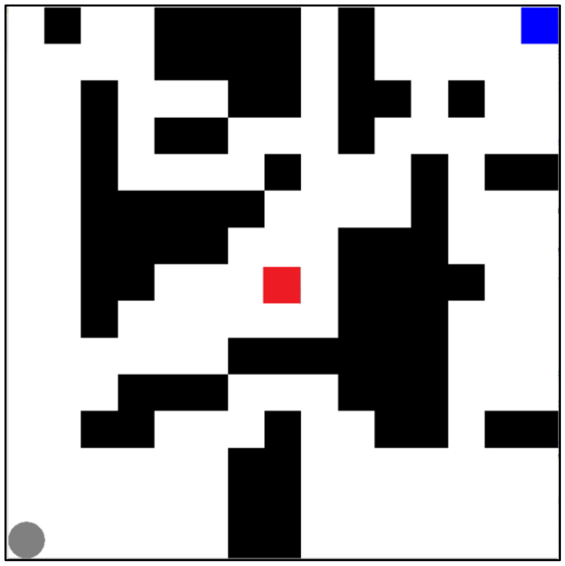


All three algorithms obtain very negative reward in the first 5 iterations. After this, value and policy iteration seem to converge to a relatively consist reward range while the reward for Q-learning continues to fluctuate. It is important to note that the spikes get smaller over time, likely representing the improved performance from state space exploration.

Overall, value and policy iteration have a similar performance on this Easy Grid World problem which is not a surprise as they operate on a very similar mathematical framework. Value iteration would be the best for this problem as it performs in slightly more than half the time as policy iteration.

Section 3: GridWorld Hard Problem

3.1 Problem Overview



The second MDP problem is much larger than the last, a 18 by 18 gird. The blue square still represents a reward of 100 as well as the terminal state.

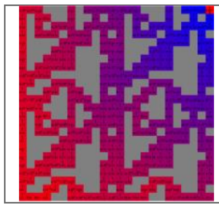
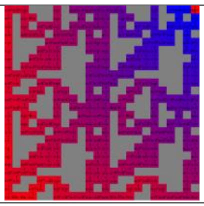
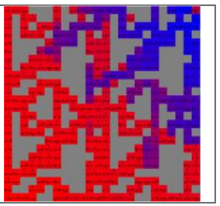
The short path in this problem is the one that crosses the diagonal. There are two other longer paths that go from the top left or the bottom right corner. Again, the agent only risks the penalty if it takes the shorter path. To understand how the reinforcement algorithms tradeoff risk and urgency in this Grid World, we can test the same values for risk and urgency as we did in the last problem.

	Low Risk	**High Risk**
Low Urgency	Step Cost = -1 Center = -10	Step Cost = -1 Penalty = -150
High Urgency	Step Cost = -5 Penalty = -150	Step Cost = -5 Penalty = -150

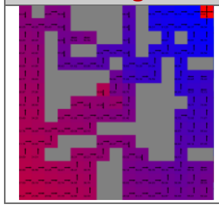
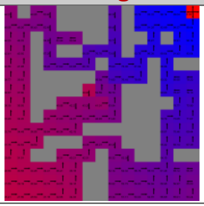
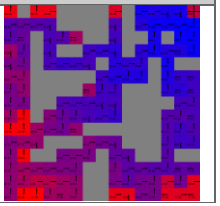
3.2 Behavior Analysis

a. Low Urgency and Low Risk

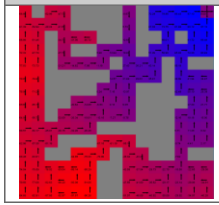
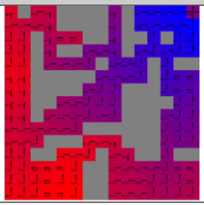
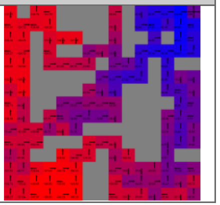
Policy Iteration	Value Iteration	Q-Learning
Short	Short	Short

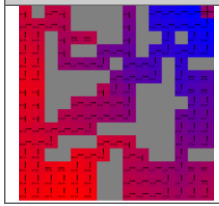
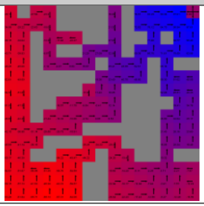
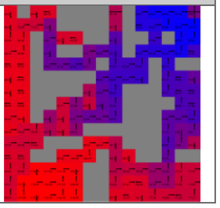
b. **Low Urgency and High Risk**

Policy Iteration	Value Iteration	Q-Learning
Long	Long	Short
		

c. **High Urgency and Low Risk**

Policy Iteration	Value Iteration	Q-Learning
Short	Short	Short
		

d. **High Urgency and High Risk**

Policy Iteration	Value Iteration	Q-Learning
Short	Short	Short
		

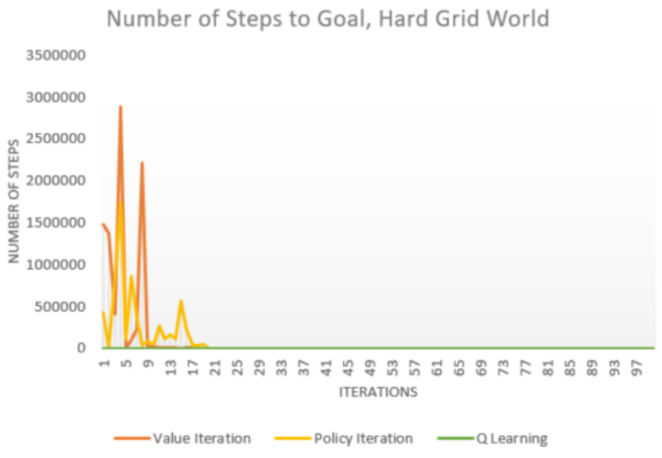
e. **Analysis**

	Low Risk, P = -10	**High Risk, P = -150**
Low Urgency, SC = -1	Policy: Short Value: Short Q-Learning: Short	Policy: Long Value: Long Q-Learning: Short
High Urgency, SC = -5	Policy: Short Value: Short Q-Learning: Short	Policy: Short Value: Short Q-Learning: Short

Even though the state space of this Grid World problem is nearly 13 times larger than the last and the orientation of the walls is completely different, the results are exactly the same. This seems to further support the notion that agents are typically risk averse and will opt for the shortest and safest path but will take risks when the urgency is higher. More experiments will need to be conducted on problems with shorter and riskier routes and longer but safer routes to identify this pattern with greater confidence.

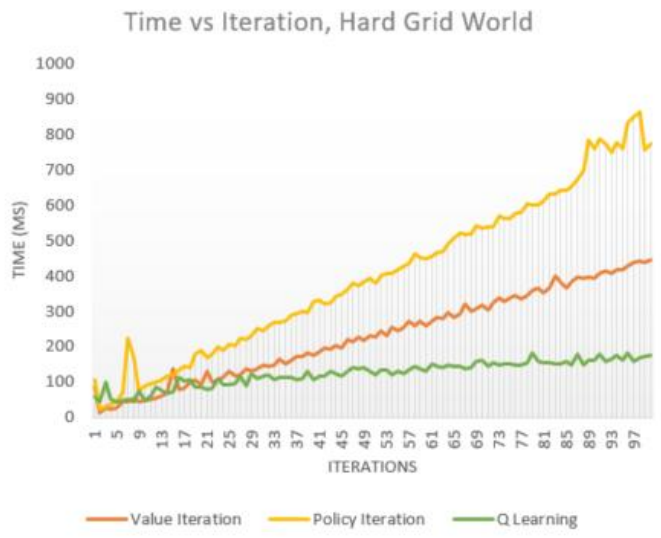
3.3 Performance Evaluation

a. **Steps to Goal**



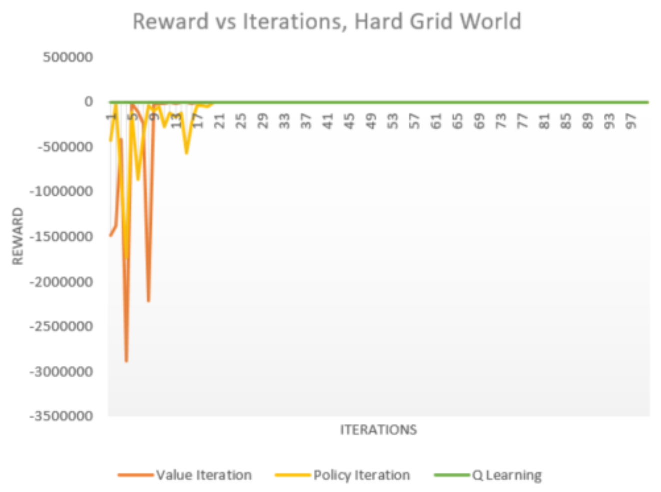
Value iteration and policy iteration take significantly longer to converge than Q learning. This is likely because since the state space is big, it would take several iterations before the discounted reward reaches the initial states at the opposite end of the board. Meanwhile, Q-leaning is able to improve quickly through exploration. While Q-learning converges faster, value and policy iteration converge to lower values.

b. Time



The performance in terms of time is very similar to the Easy Grid World problem. Q-learning takes the least amount of time likely because it is building up its calculations instead of calculating and updating the entire state space every iteration.

c. Reward



This graph closely follows the performance in terms of the number of steps that the reinforcement algorithms took in order to converge. Since value and policy iteration took a significantly larger number of steps to terminate in the first 20 iterations it is not surprising to see that the two algorithms have a proportionately low reward given the step cost.

For a problem with a larger state space, Q-Learning converges and performs faster and better than value or policy iteration. This is because the latter two algorithms need to update values for the entire state space every iteration while Q values are only updated as they are visited.

Section 4: Conclusion

In the smaller grid world problem we found that Value iteration had a slightly higher performance in terms of time than policy iteration. In the larger grid world problem, Q-learning converged the fastest as it need not calculate all state values every iteration.

After exploring the trade-off between urgency and risk in both small and large state spaces, we found that the behavior is the same. Agents are typically risk averse and will take the shortest but safest route except when urgency is high, agents are far more likely to take risks.

In a real world application this logical. If an Amazon delivery truck was to be fined a few hundred for a late delivery, it would likely rational to take the shorter route through the city and risk having to wait for a passing train.

It is important to note, that these models are highly abstracted and many more variables both human and environmental go into real world decision making.

References

1. “About.” BURLAP, Brown University, burlap.cs.brown.edu/.

2. “Markov Decision Processes.” Zeroassetsor,
www.cnblogs.com/hello-
yz/p/9252568.html.