Reinforcement Learning

Georgia Institute of Technology CS7641 Machine Learning Assignment 4

Yan Cai GTID: ycai87

Abstract

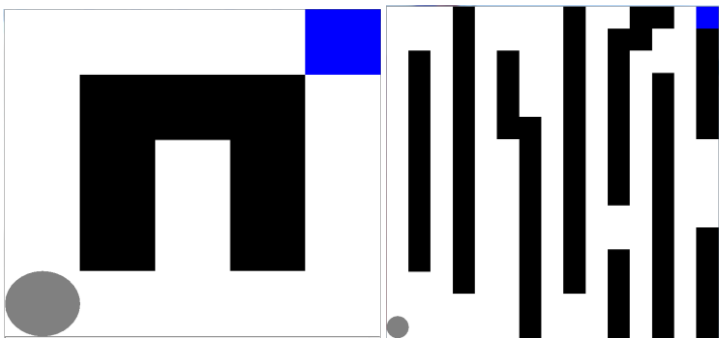
This paper explores three different reinforcement learning algorithms: value iteration, policy iteration and q learning and applies the algorithms to Markov Decision Processes (MDP) problems.

Problem Introduction

Grid world problems are explored as Markov Decision Process problems. The grid world problem is a classic problem for MDP because it is an abstraction of some practical problems. In the grid world, the agent must traverse a grid-like square from the start to the goal. In the traversal, the agent also needs to avoid the obstacles in the grid to reach the goal state. The agent starts from bottom left and goal state is in top right.

The two MDP problems are two grid world problems classified as easy and hard grid world problems. The easy grid world is a 5 * 5 grid world and the hard grid word is a 15 * 15 grid world. The agent tries to reach the terminal state in the least number of steps dictated by the reward function. In the grid world, the reward for each state other than obstacles and terminal state is -1. For the terminal state, the agent gets 100 as reward. We also have a transition matrix that sets the probability that the agent moves to the intended direction. We set 0.8 as the probability to move the target direction and 0.0667 as the probability to go to the other 3 directions other than the target direction.

The two grid worlds are shown as:



Easy Grid World vs Hard Grid World

Why Is The Problem Interesting?

The grid world is a great abstraction of some real-world applications. For example, the hard grid world can be seen as a maze for autonomous vehicle to try to reach the goal position. The vehicle needs to make a series of decisions that can reach the goal faster. The obstacles can be some real-world obstacles like walls or bricks. The vehicle acting as an agent needs to navigate itself and reach the goal. Another example for grid world

application can be route planning. We set the start location and exit and want to find out what the best route is to quickly arrive at the exit. The grid world is a implication of these real-world applications. It is an interesting problem to explore.

Reinforcement Learning Algorithms

In this paper, we use three different reinforcement algorithms to explore the grid world problems: value iteration, policy iteration and q learning.

Value iteration is using Bellman equation to evaluate the states till there is a converging solution. At the beginning, the unity is not known other than the immediate reward. However, the utility of nearby states can be calculated based on discounted award and it iterates until the utility values for all the states are calculated.

Policy iteration is creating arbitrary policies and calculating the utility values under the policy. This algorithm tries different policies and test if the new policy can increase the utility value. It iterates till there is no improvement for the utility value. The method also converges but it requires more calculation because it needs to solve equations when calculating the values for all the states.

Q – Learning is a model free algorithm because value iteration and policy iteration both use domain knowledge to calculate the transition function. Initially, we assign q values to all states. The agent visits each state and assigns new q values based on immediate and delayed rewards. Q learning also has a chance to randomly select the policy, but as it progresses, the q values will converge to its true values for each state.

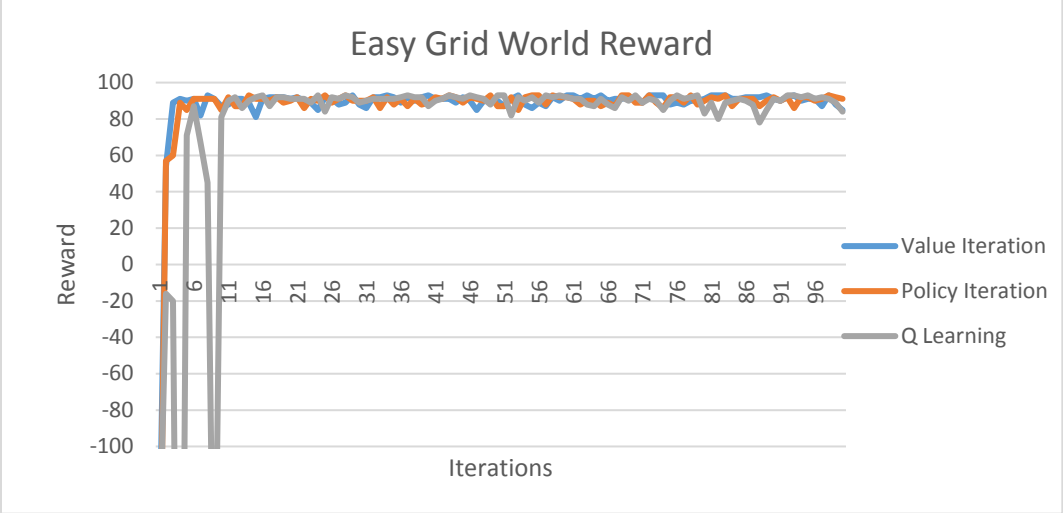
Implementation and Methodology

The 3 different algorithms and grid problems are implemented using BURLAP library. It is implemented in Java. There are parameters in the algorithms and we use the default values.

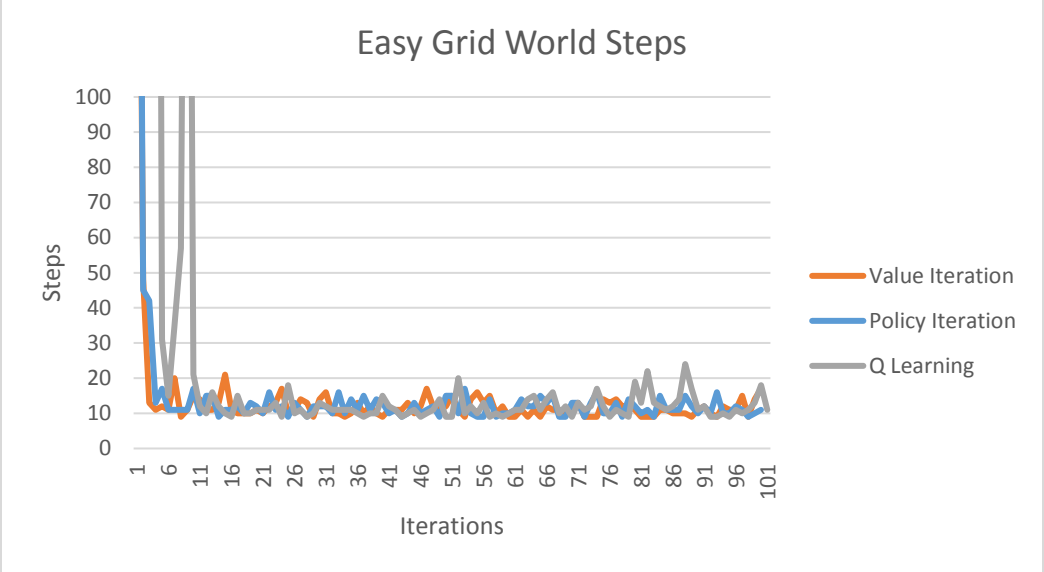
We run three different algorithms for both problems, for Q – Learning, we also tried on different learning rate and epsilon rate.

Easy Grid World

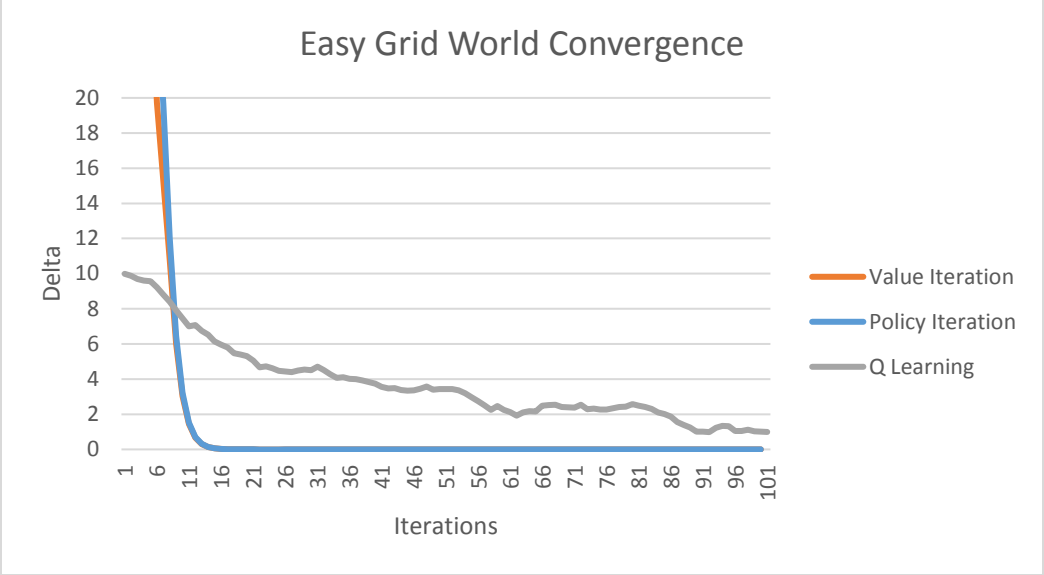
We ran the 3 different algorithms with iteration = 100 and for Q Learning, we also run iteration = 1000. For Q learning, we use learning rate = 0.1 and epsilon = 0.7.

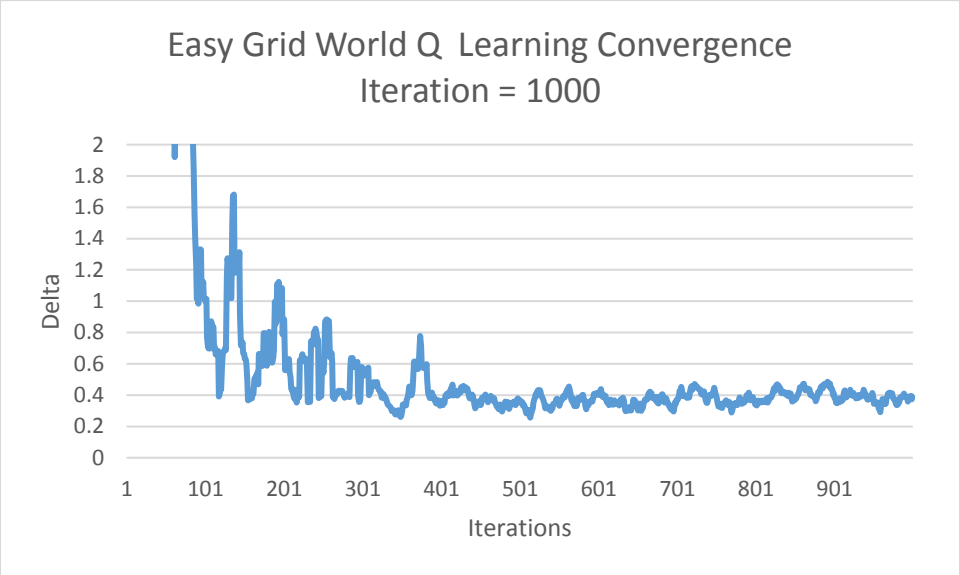


We can see for value and policy iteration, it quickly converges to ~90 with 3 to 4 iterations. For Q Learning, it oscillates until 12 iterations to reach the similar ~90 range. Also it tends to have larger variations of rewards in future iterations.

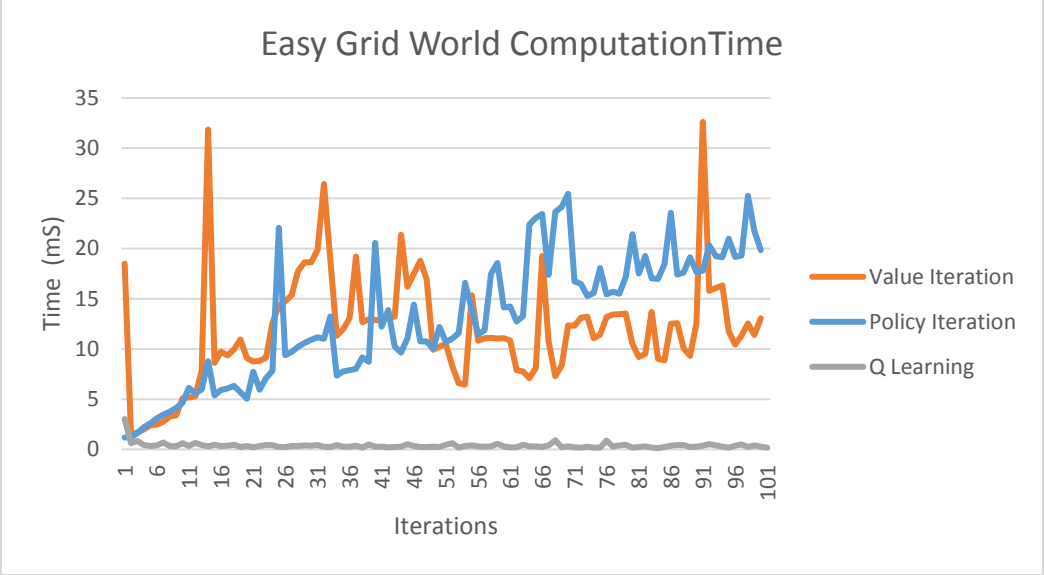


For steps, both value and policy iteration converges to ~10 steps very fast at first few 3 to 4 iterations while Q Learning waits till 11 iterations.





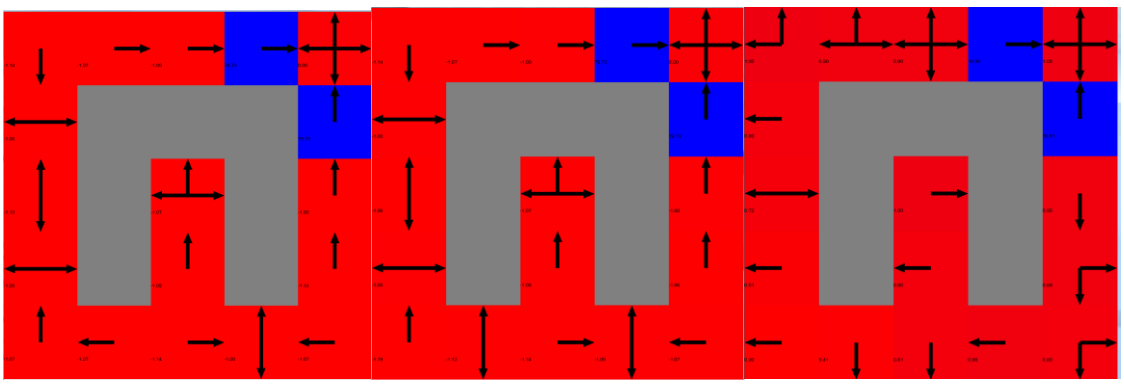
Both value and policy iterations converge fairly quickly and at about 13 iterations, both algorithms have convergence value < 0.5. For Q Learning, the convergence plot decreases a lot slower even at smaller convergence start. Since Q Learning does not know the model, it is expected to take longer time to converge. For better reference, we also run 1000 iterations for Q learning convergence.



Both value iteration and policy iteration take a lot longer time than Q Learning. This is expected because Q Learning runs at a constant speed because it just hashes the actions taken and rewards until a policy is calculated. For policy iteration, it needs to calculate the policy each iteration so it takes the longest time.

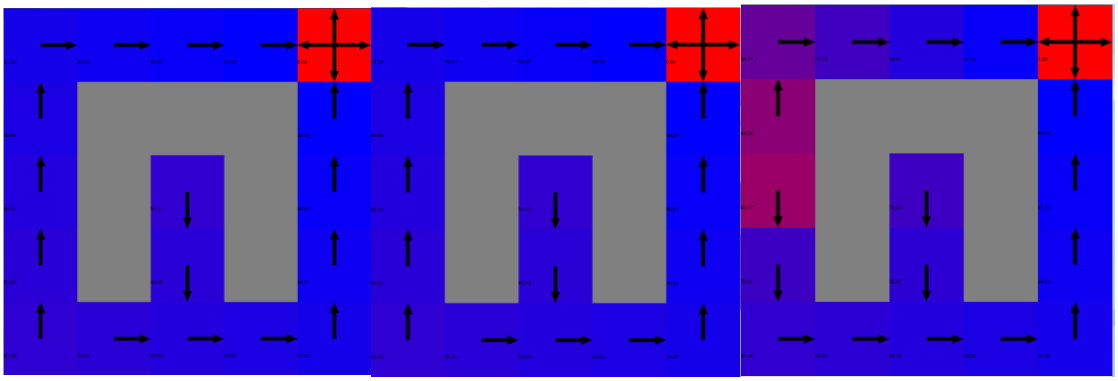
We also compare the policies evolution in the iterations.

When iteration = 1:



Value Iteration Policy Iteration Q Learning

When Iteration = 100



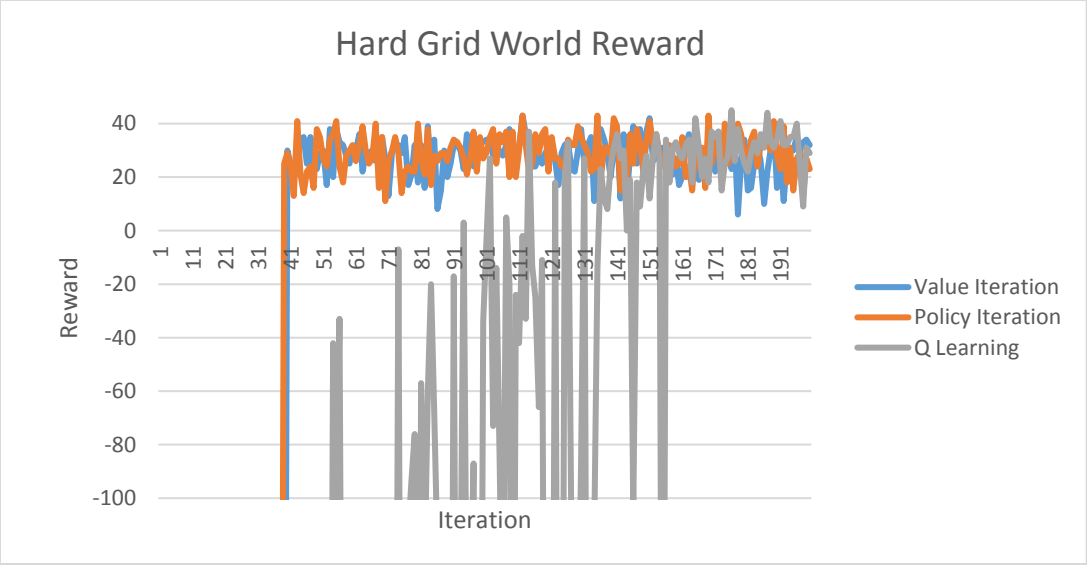
Value Iteration Policy Iteration Q Learning

We can see both value and policy iterations get the optimal policy map, while Q Learning at iteration 100 still does not have the optimal policies. This is expected because the delta at 100 iteration for Q learning is till larger than 0.5. We test that at iteration = 400, the Q Learning algorithm will constantly get optimum policies because its delta will be smaller than 0.5 mostly.

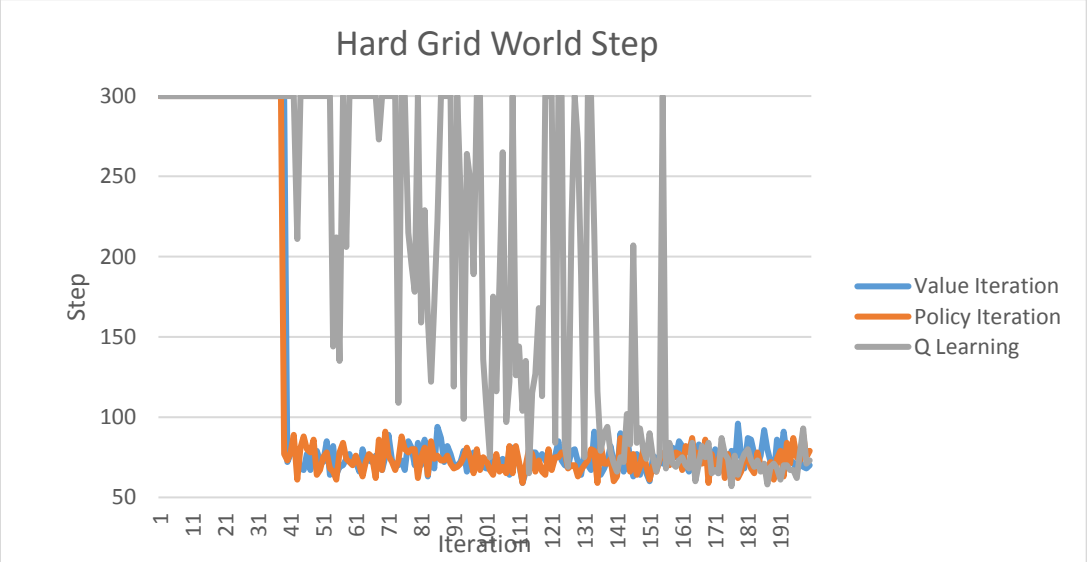
Based on the findings from 3 algorithms, the best reward for easy grid world is 93 and 9 steps in total.

Hard Grid World

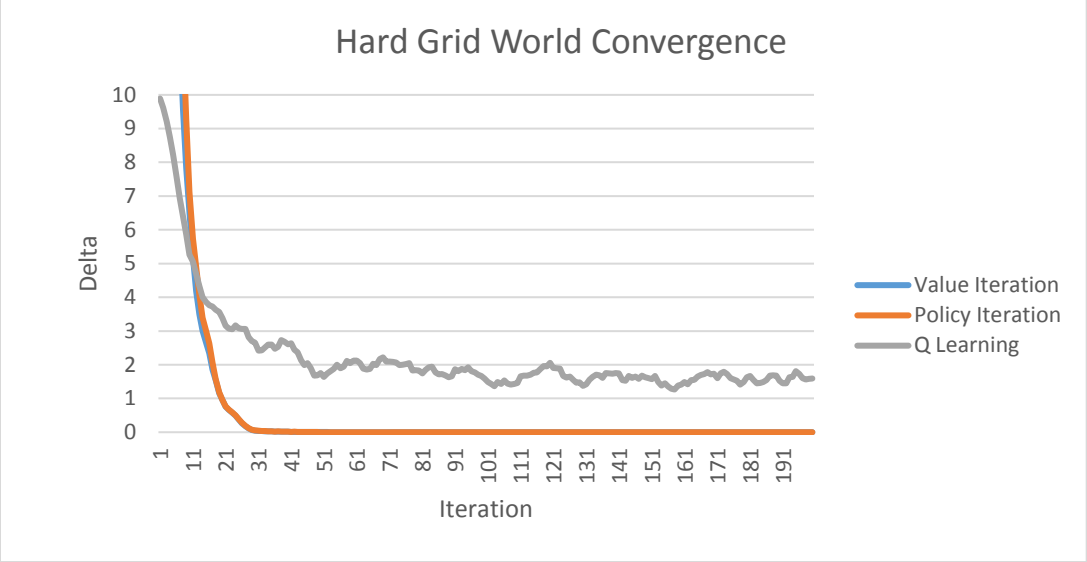
For the 15 * 15 hard grid world, we run 3 different algorithms with iteration = 100 and for Q Learning, we also run iteration = 100000. For Q learning, we use learning rate = 0.1 and epsilon = 0.7.



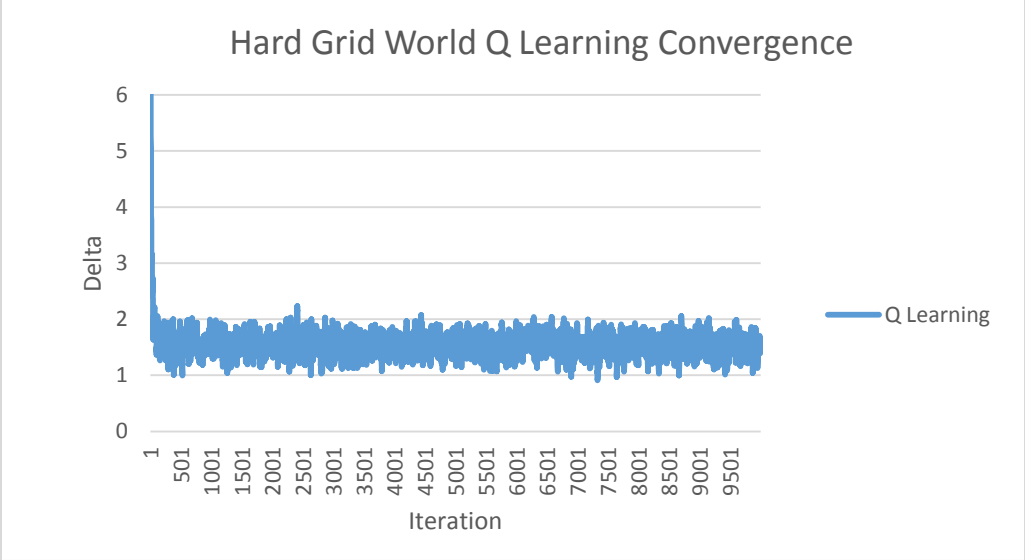
The reward for both Value and policy iteration quickly converge to 30 at about iteration = 40, while Q learning converges to the same range at iteration = 160. This is expected since Q learning is model free.



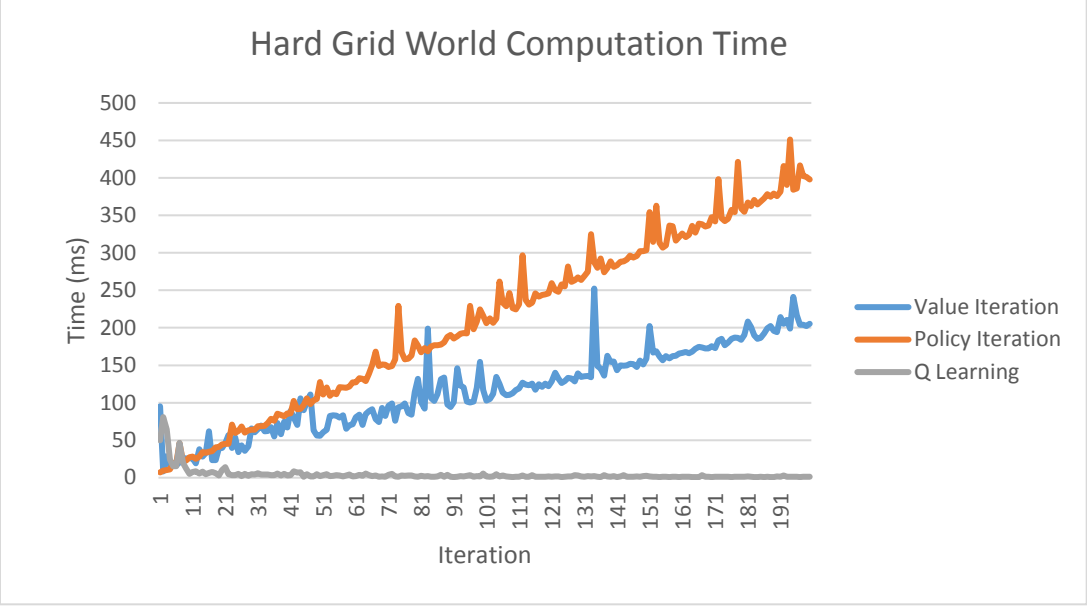
Same for steps, both value and policy iterations converge to 70 at iteration = 40 while Q learning takes longer time to. There are also spikes but Q Learning is able to get the fewest step.



The deltas for both value and policy iterations reach 0 at iteration = 103, while Q Learning delta is always around 1.3. For reference, we also run 10000 iterations for Q Learning.

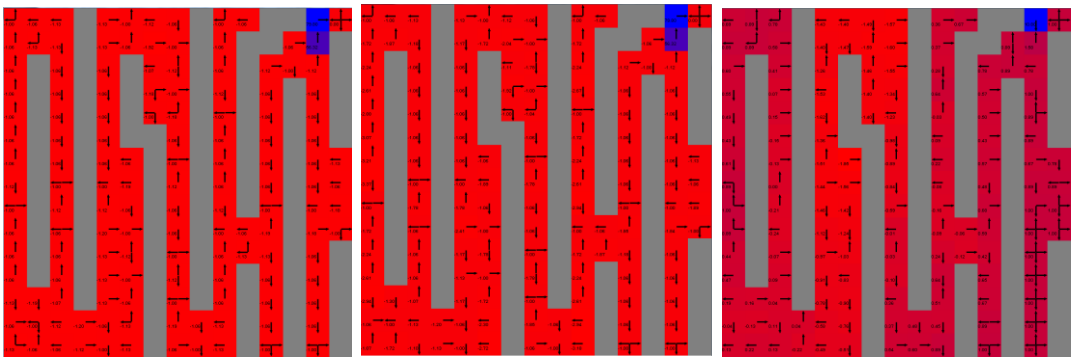


As we can see, even at iteration = 10000, the delta is always above 1. To have it converge to a lower value, it may take a lot more iterations.



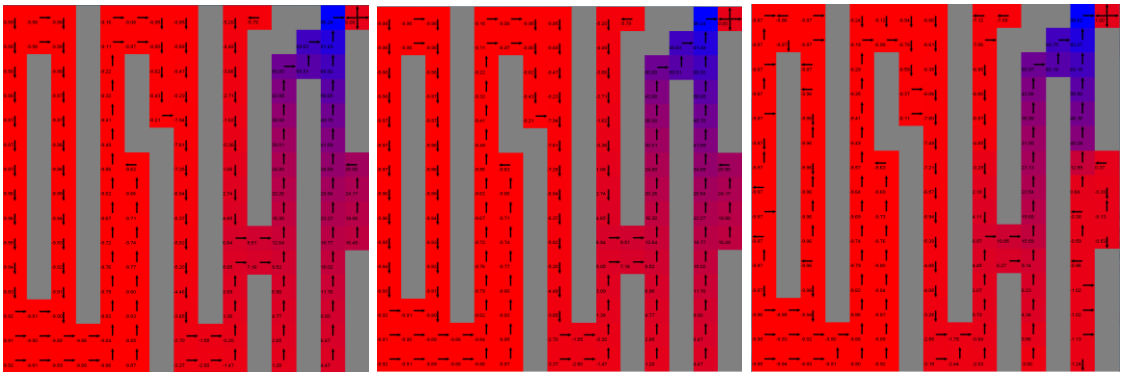
Policy iteration takes the most time as it needs to compute the policy in addition to values. Q Learning runs at constant time as it only has hash value access.

At iteration = 1:

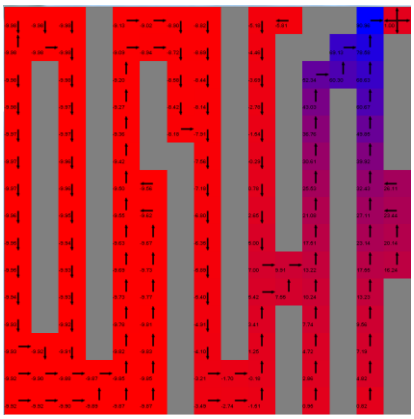


Value Iteration Policy Iteration Q Learning

At iteration = 200:



We can see both value and policy iteration show the optimal policy map while Q Learning still have a couple of states incorrect. This is expected as it is still far from converge at iteration 200.

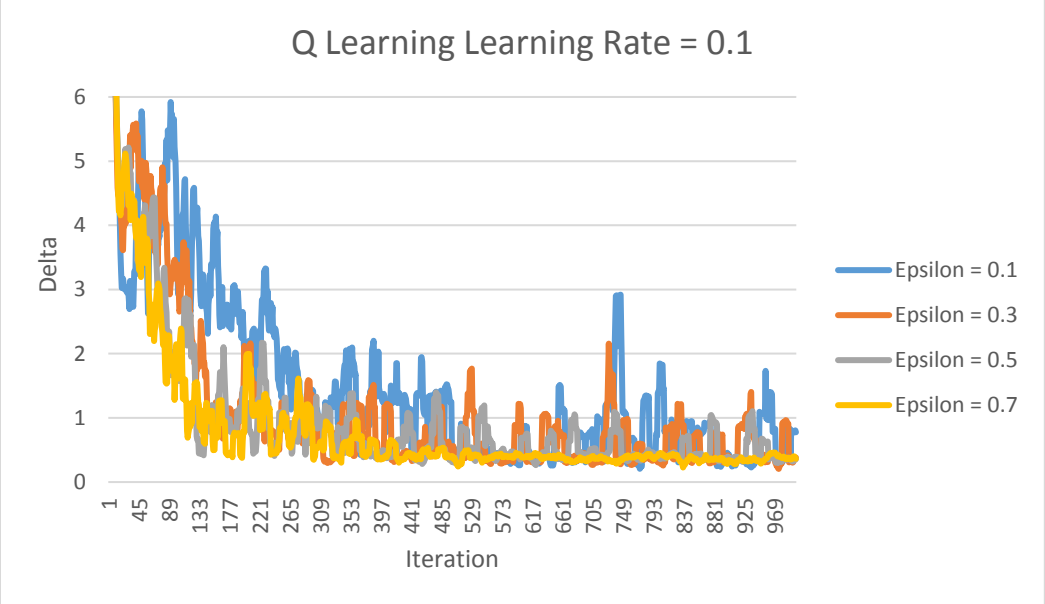


The figure above is Q Learning at iteration = 100000. As we can see, it has better policy map than iteration = 200, but still have a few states in correct.

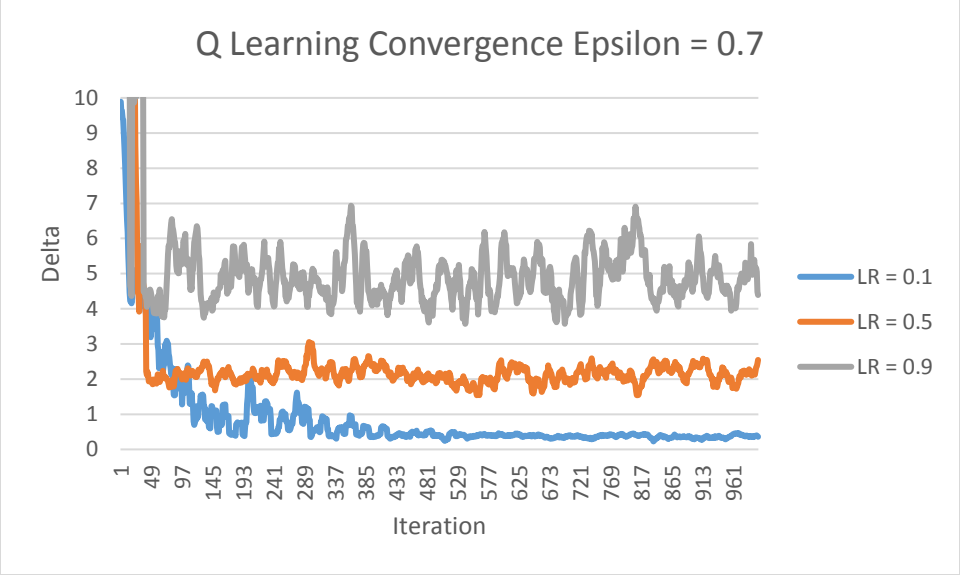
Q – Learning Learning Rate and Epsilon

In this section, we take the easy grid world as an example and explore the significance of learning rate and epsilon. The learning rate controls how important future utility value is with respect to the immediate reward. The epsilon controls whether the agent randomly chooses an action without even considering the utility value.

We fix either learning rate or epsilon and compare the convergence.



As we can see, with the same learning rate, higher epsilon value makes the Q Learning converge faster and also with fewer spikes. This shows that a higher possibility for the agent to randomly select actions actually helps the agent to explore better policies.



The figure above clearly shows that higher learning rate takes more time to converge with same iterations. This suggests that the new information at each iteration is not generating enough values, sticking with the immediate reward is the better judgement.

Conclusion

Both MDP problems shows that value iteration and policy iteration tends to converge a lot faster than Q learning. However, Q learning runs at a constant time while value and policy iterations takes more time as the iteration increases. Furthermore, policy iteration takes the most computation time. We also find out lower learning rate and higher epsilon value for Q Learning performs better. It is also clear that with higher number of states, Q learning takes an exponential increase of iterations to converge. If we know the model, value and policy iteration is able to calculate the optimum policy fairly quickly, otherwise Q Learning is the better algorithm to use.

References

1. Brown-UMBC Reinforcement Learning and Planning java library. http://burlap.cs.brown.edu/index.html

2. OMSCS: Machine Learning - Assignment 4. Juan J. San Emeterio. https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4