Markov Decision Making Process

Assignment 4

Er Li Suo
903458480
Nov. 25, 2018
esuo3@gatech.edu

DESCRIPTION

This assignment explores Markov Decision Process and Reinforcement Learning. The grid world problems are discussed to demonstrate the processes of the algorithms. As we learned in class, the grid world problem is very classic and extremely easy to visualize its progress in each iteration which can help in understanding the intuition of methods. The easy grid world problem is a 5X5 maze and the hard grid world problem is a 15X15 maze. For both problems, the entry is at the bottom left corner of the maze and the exit,with a reward of 100 points, is located at the top right corner of the maze. The immediate reward for each state is -1 except the exit state. For translation function, similar to what we used in class, there is 80% chances to take the desired action successfully and 6.67% of chances for the agent to land on the other three neighbor states. If the obstacles are in the next state, the agent will be bounce back and stay in current state.

Moving in the maze and finding the optimal way to reach a target is extremely useful in the real world applications. For example, in some video games, it can be used to design the AI of a monster chasing the player. If the game is simulating the real world situation, noises, errors and mistakes are possible. The reinforcement learning algorithms can effectively help in resolving the indeterministic issues.

ALGORITHMS

Thee algorithms are evaluated with the grid world problems, Value Iteration, Policy Iteration and Q-Value Iteration. The value iteration and policy iteration methods are based on Bellman equation to estimate and compare the rewards in current state as well as future states with the best actions taken in every state. Q-Value iteration method is different from the previous two methods. It requires no model. It assigns Q-Values randomly for all states to start with; as the method iterates, the Q-Value would be recalculated and updated to approach the true values. The discounting factor for both worlds are 0.99.

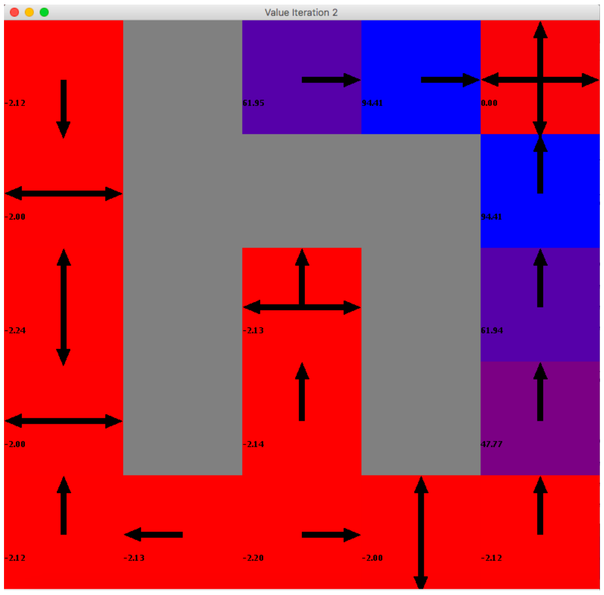
RESOURCES

The assignment code is modified based on formal classmate's work, Yan Cai, which is shared on GITHUB; the reinforcement library, Burlap is used with Jython to translate Java library to work with Python programs.

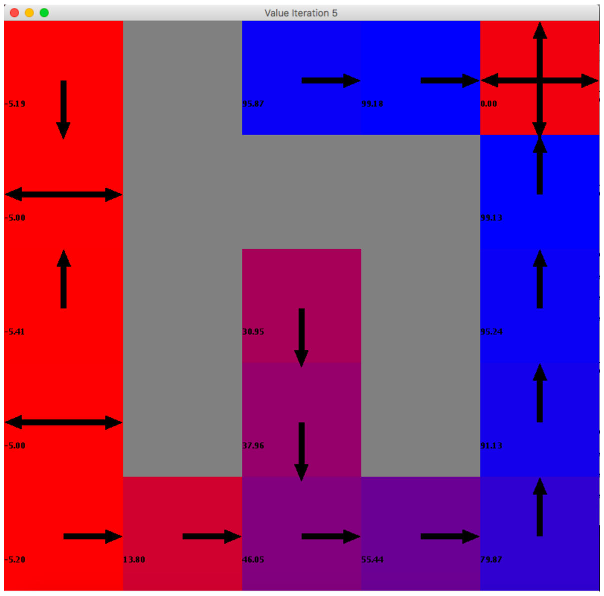
Easy Grid World

Value Iteration

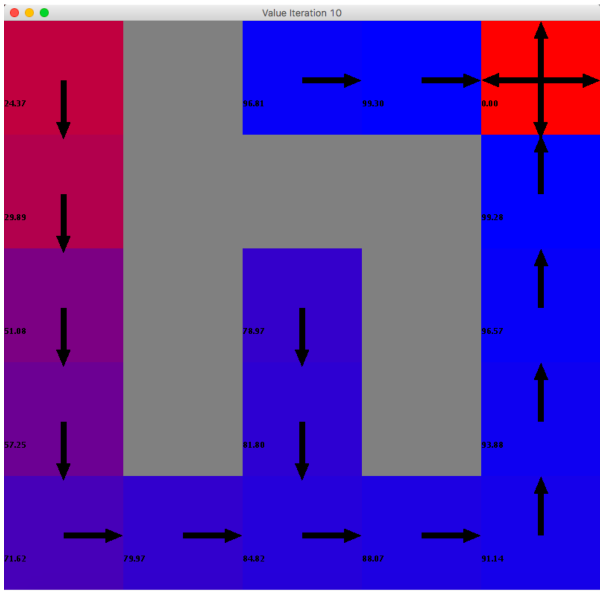
Iteration 2



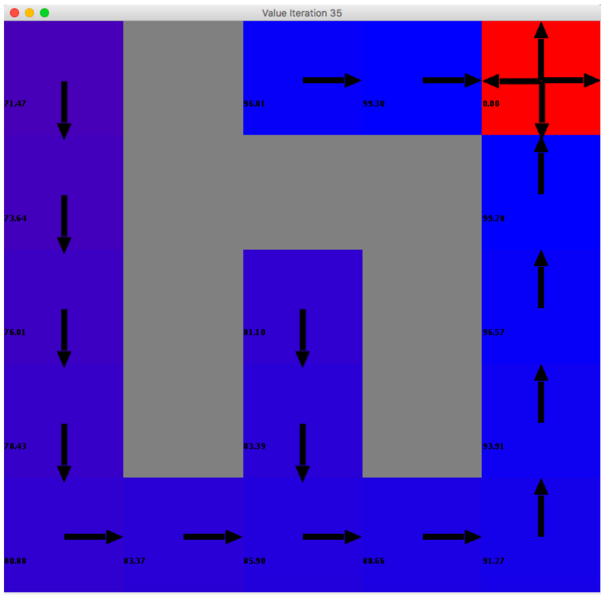
Iteration 5



Iteration 10



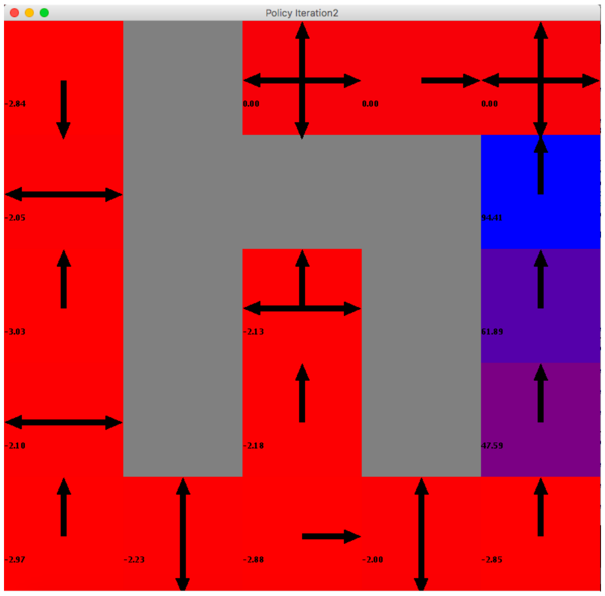
Iteration 35



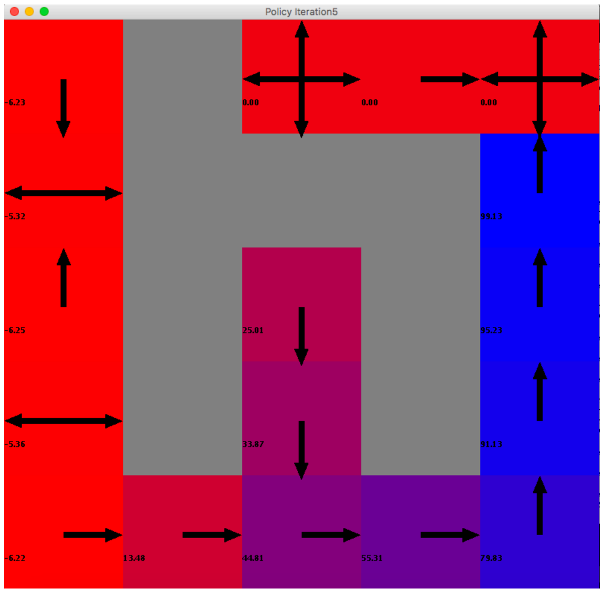
The graphs above demonstrate the process of Value Iteration method. As we can see, in iteration 2, most of the states are randomly taking actions because the immediately rewards are configured to be -1 for these states, such, there are no information given to make the good decision on choosing the best action. Except in the top right corner, the two states next to the exit state have information to make better choices; such, they are assigned with the correct actions and positive reward values. Iteration 5 and 10 show that more states are progressively influenced by the positive rewards given from their neighbor states. At iteration 35, the evaluation converges and the optimal policies are generated for all states.

Policy Iteration

Iteration 2

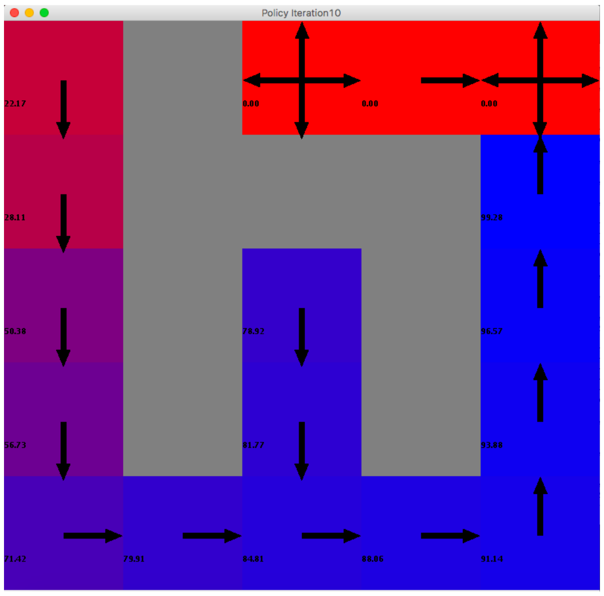


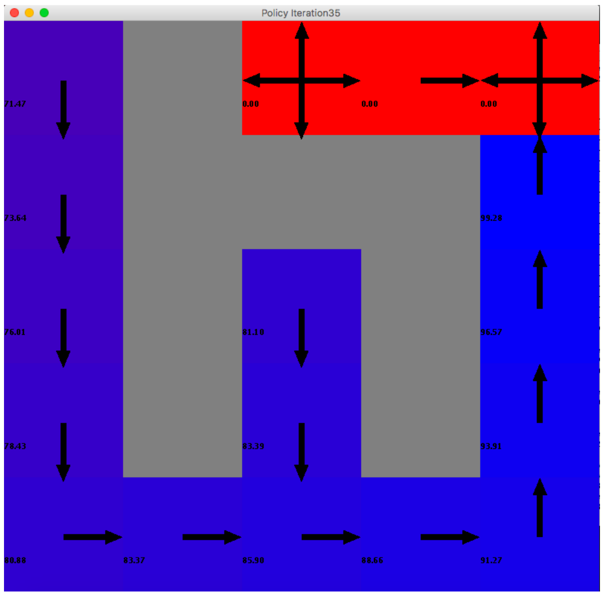
Iteration 5



Iteration 10

Iteration 35

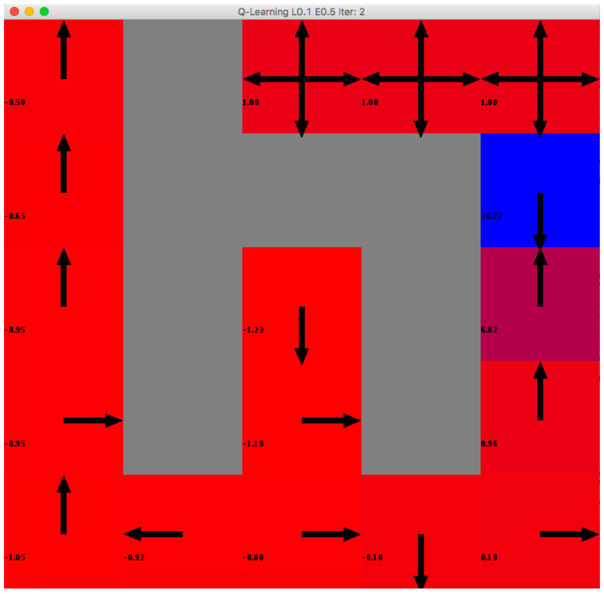




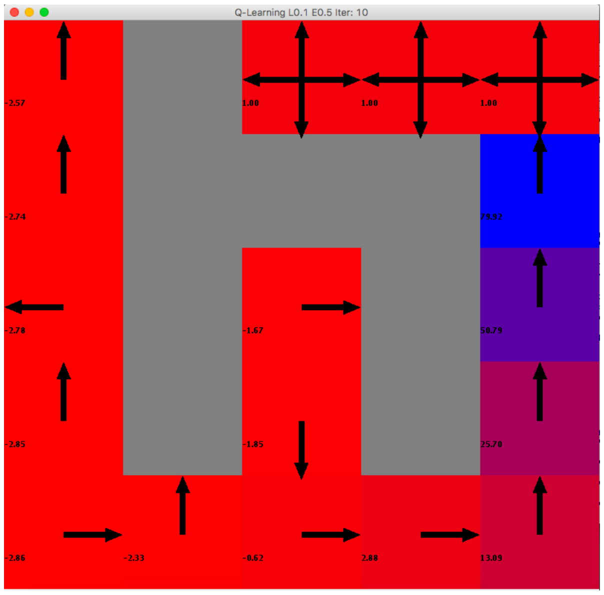
Similar to the Value Iteration method, the Policy Iteration method also slowly spreading out the influences from the top right corner (the exit state) and every states are assigned with the right action to take. Different from the Value Iteration method, there are 2 states on the left side of the exit state cannot be updated. Since the policy iteration method is based on the recursive call on Bellman equation with a valid action to take to reach the next state, there is no possible action to take from the starting state to reach the two isolated states; such, these two states are ignored.

Q-Value

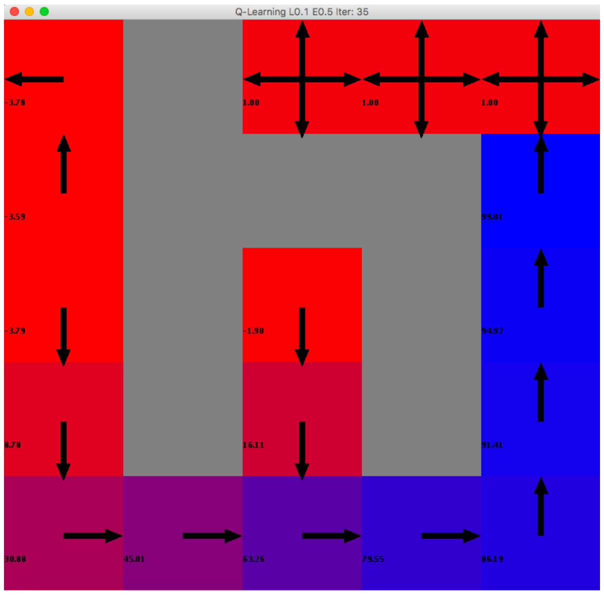
Iteration 2



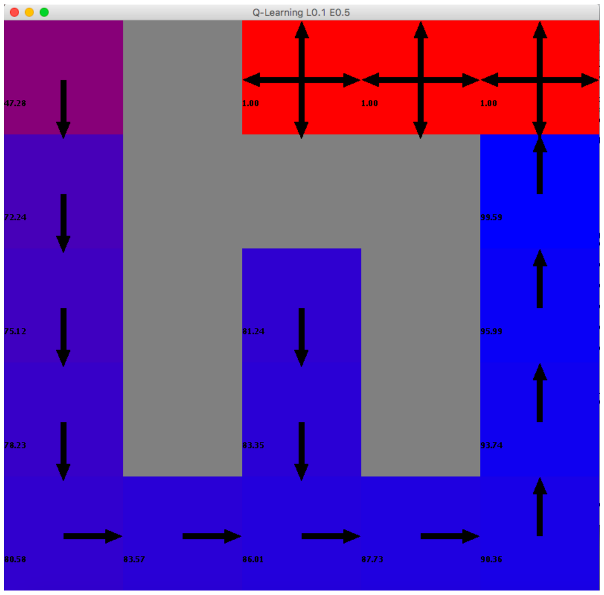
Iteration 10



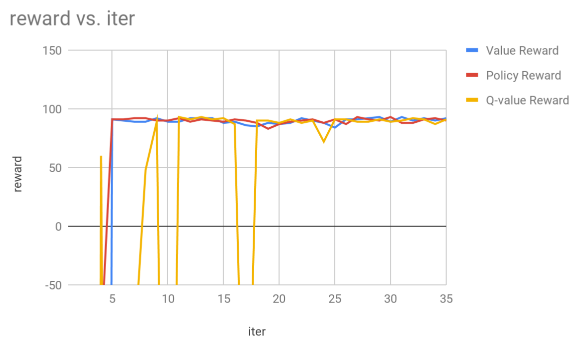
Iteration 35

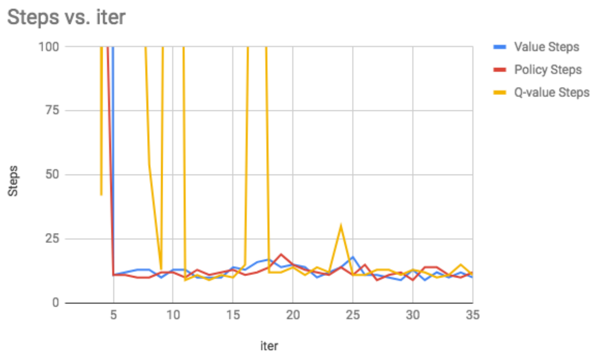


Iteration 1000

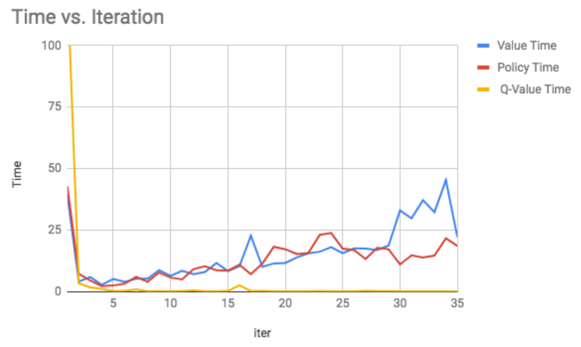


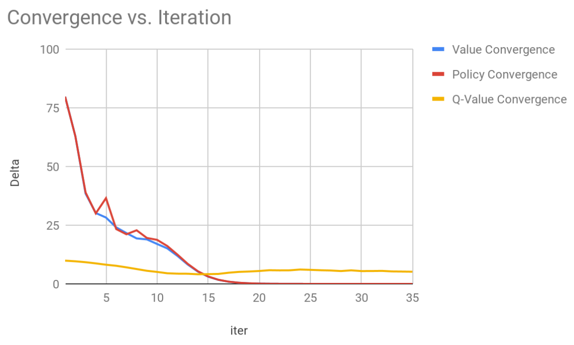
Q-Value iteration method converge much slower than the previous two methods in terms of iterations. One reason is that a small learning rate is given, in this case 0.1. At iteration 35, some of the states are still in "red" indicating the negative reward values. A total of 1000 iterations are executed. Similar to the Policy Iteration methods, the Q-value for the top right two states on the left hand of the exit state can never be reached.





The graphs above show the expected rewards and steps to exit for each method in 35 iterations. As we can see, Value Iteration and Policy Iteration methods quickly reaches the expected reward, 90 points, within 5 iterations. The Q-Value iteration method reaches the similar value at about iteration 20. Similar pattern can be observed from the expected steps as well with a expected number of step equal to 10.





In terms of execution time, Q-Value iteration method is much faster than the other two methods. The advantage of model free property is well illustrated. When facing big data, Q-value iteration method is more adaptable, we can observe the effect more clear in the hard grid world problem below . We can notice that around iteration 1, the computation time for all methods are extremely high; it can be ignored because it could come from the object initialization time required by the program which is independent from the algorithms.

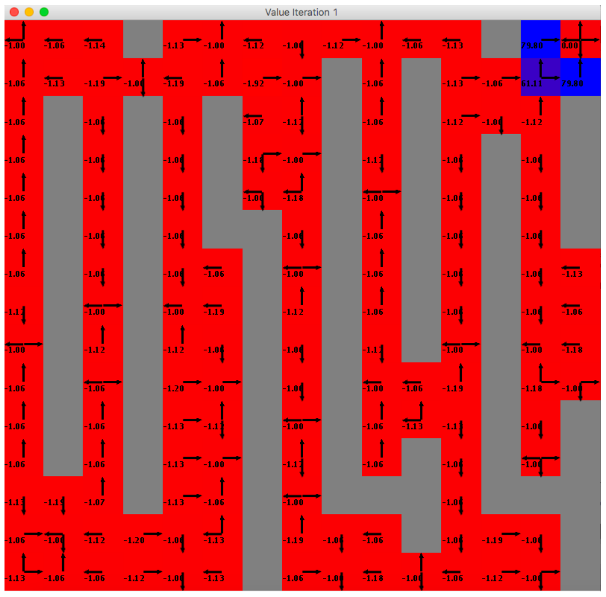
As shown in the graph at the right side, Q-Value iteration and Policy Iteration quickly converged to zero around iteration 17. However, Q-Value iteration method seems converging to a constant value. One reason could be a fixed learning rate is used in the problem. For each iteration, we purposely generated some error based on the learning rate. If a decay learning rate is used, the delta should be converged to zero as other two methods.

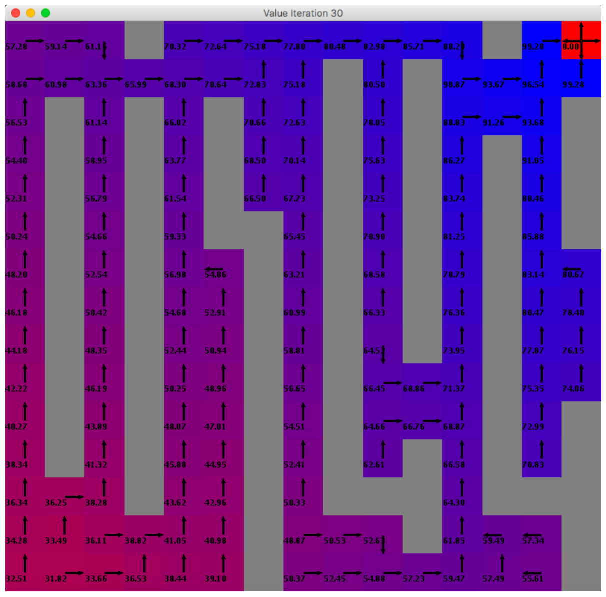
Hard Grid World

Value Iteration

Iteration 10

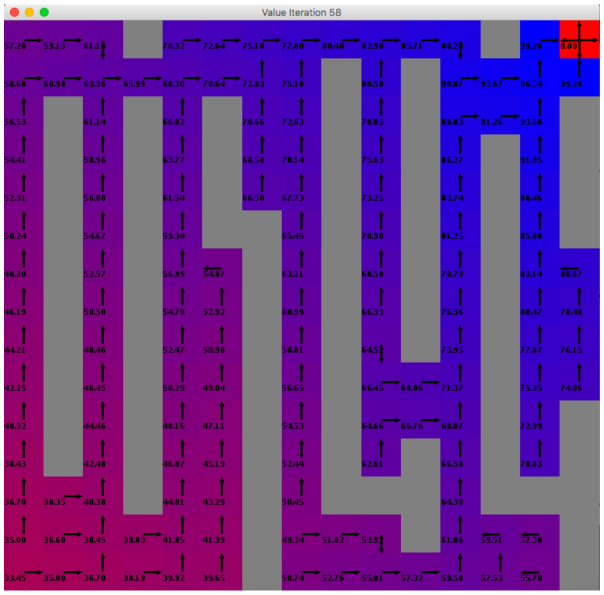
Iteration 30

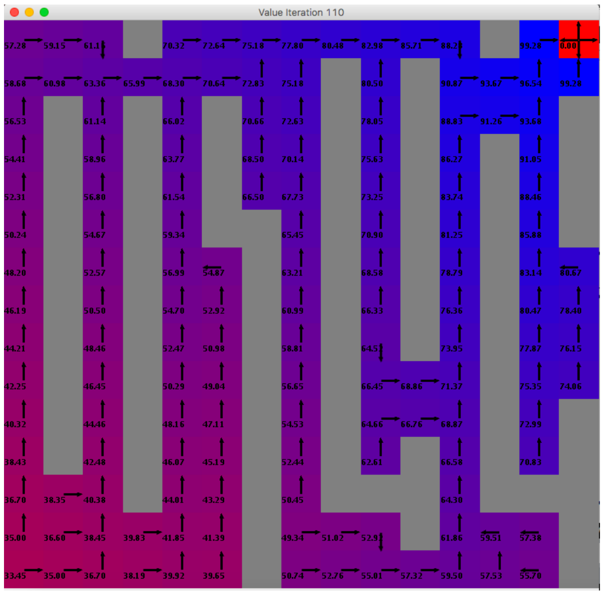




Iteration 58

Iteration 110



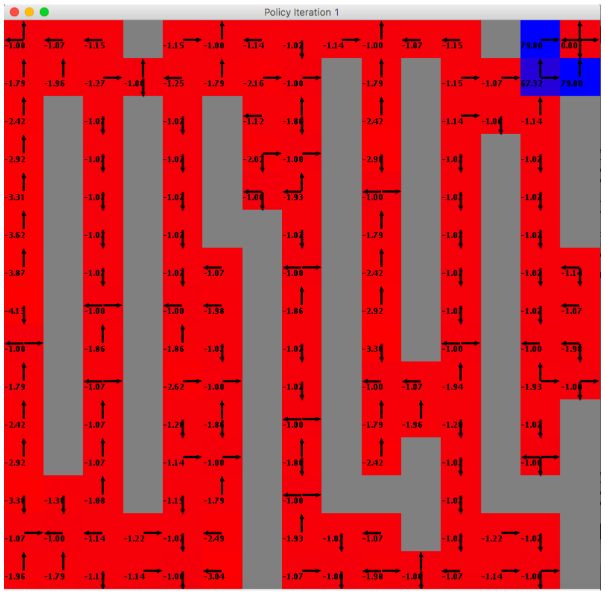


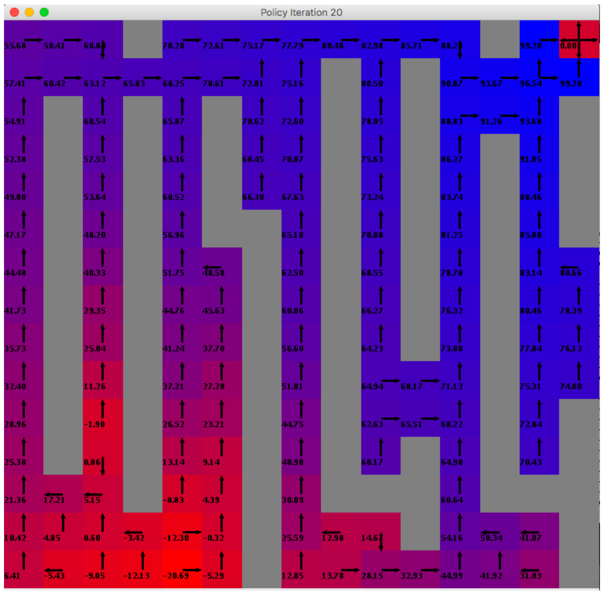
Similar to the easy grid world, every state is assigned with an random policy to start with due to the lack of the information for comparison. With more iterations, around iteration 58, the optimal policy is slowly generated. The algorithm converges at around iteration 110.

Policy Iteration

Iteration 1

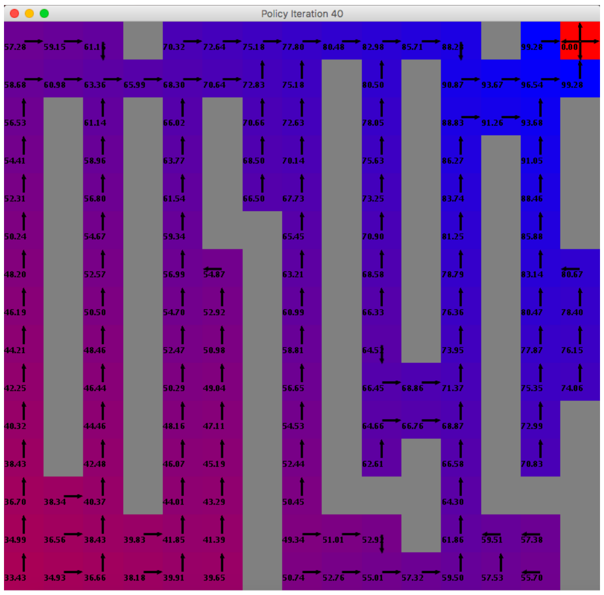
Iteration 20

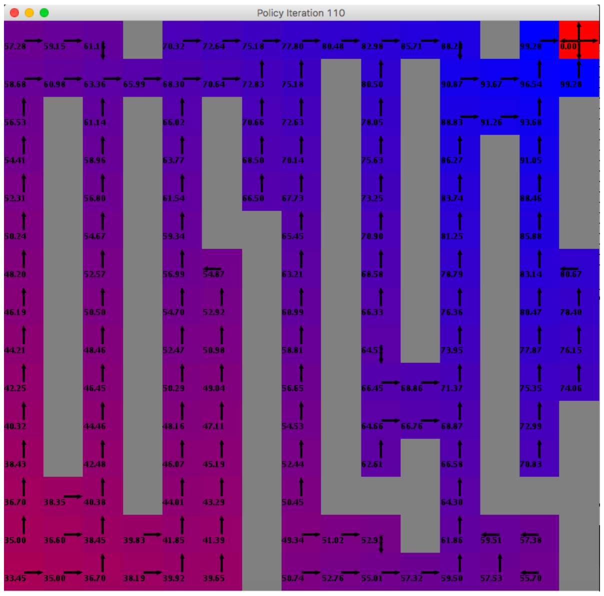




Iteration 40

Iteration 110

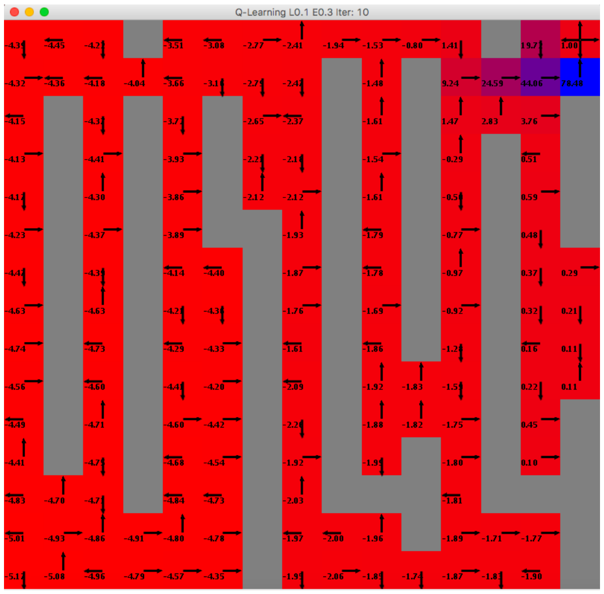


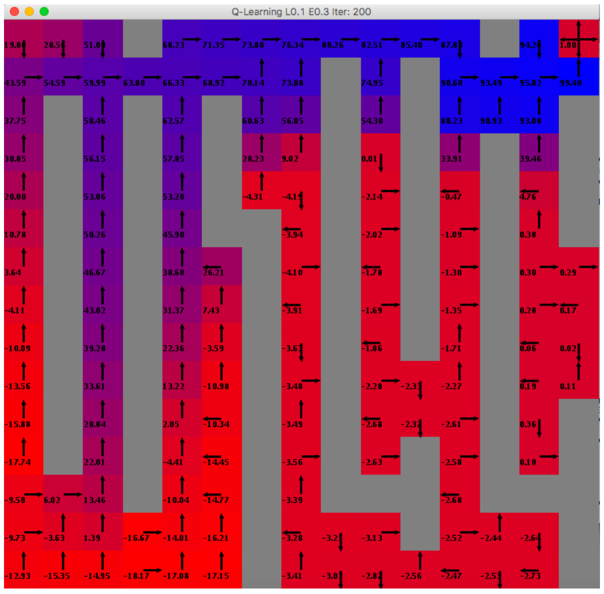


For hard grid world, policy iteration method reaches the optimal policy plan at around iteration 40 which is faster than the value iteration method.

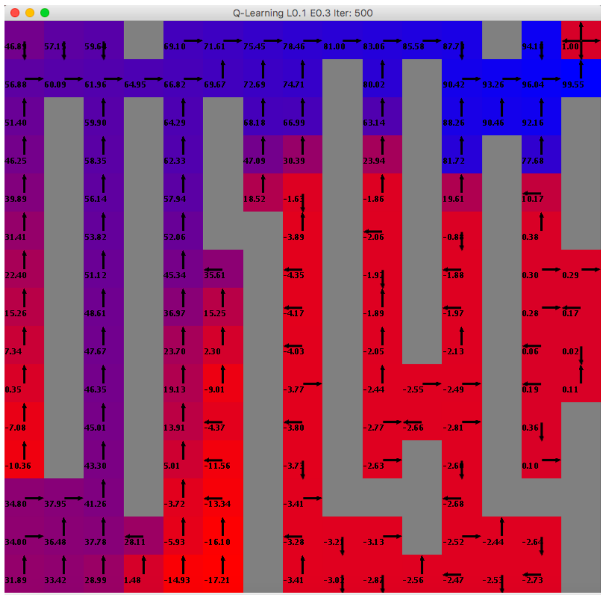
Q-Value Iteration

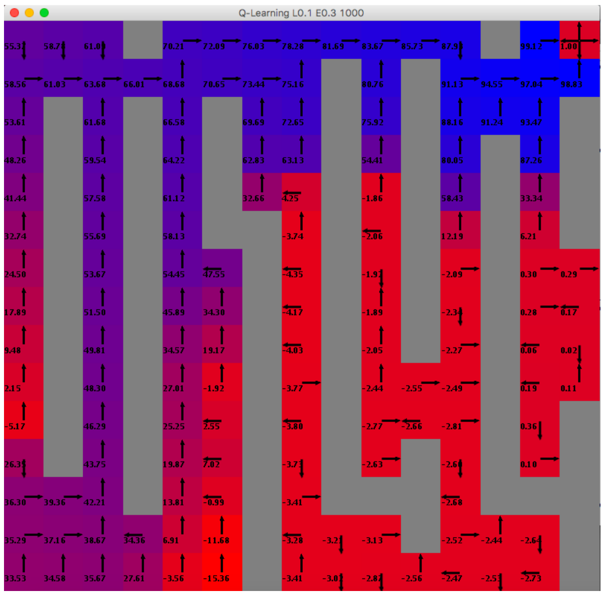
Iteration 10 Iteration 200



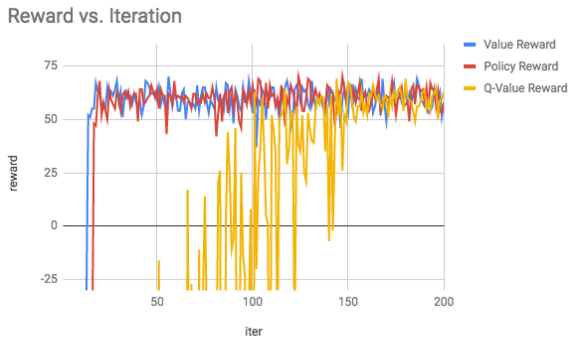


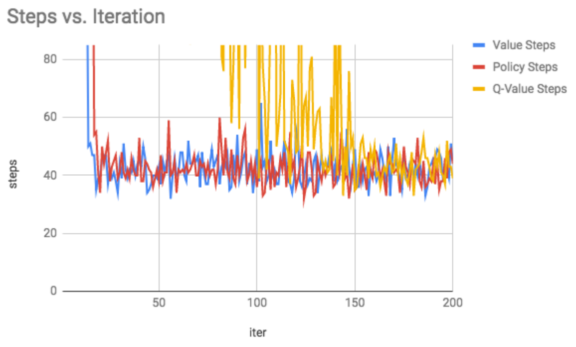
Iteration 500 Iteration 1000



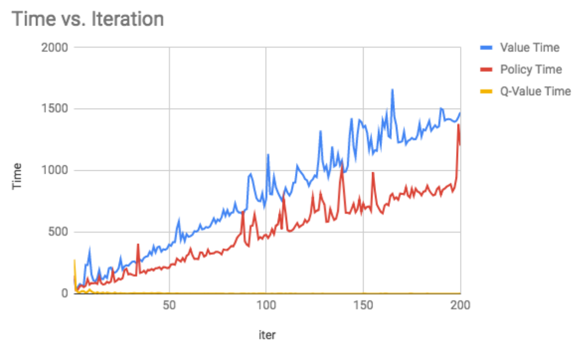


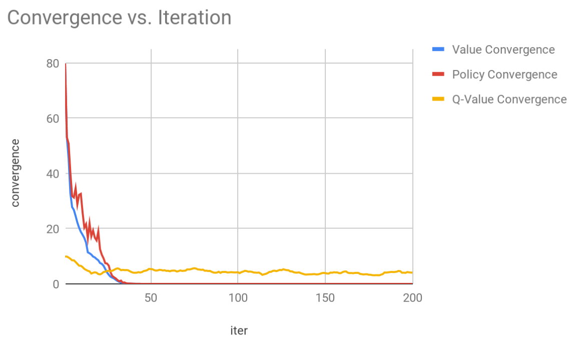
For the hard grid world, it is interested to observe that even after 1000 iteration, the bottom right region of the maze is still all colored red, which indicate that the negative rewards are assigned at these states. This explains why Q-Value iteration is much faster than the other methods because it does not take consideration of all the possible states in order to find the optimal policy plan to reach the exit state. Only the significant states are repeatedly visited and improved over iterations. In the above demonstration, a constant learning rate of 0.1 and the epsilon = 0.3 are used.





Similar to the observation made in easy grid world, Q-Value iteration requires more iterations to converge. All of the method eventually give the expected reward around 60. The value is smaller than the one in the easy grid world, which make perfect sense as more iterations the discounted factor makes the bigger impact on the rewards.
The hard grid world requires approximately 40 steps to exit the maze. As shown in the graph above at the right hand side, all 3 methods produce the similar result.





It is very important to realized that the computing time is grown linearly along with iteration counts for both value iteration and policy iteration methods. In contrast, the Q-Value method performs extraordinarily good with a nearly constant execution time. Even though more iterations are required for Q-Value methods to converge, the overall required execution time is significantly less than other methods. In the real world complicated problem with millions of states, the Q-Value method is definitely more applicable.

Similar to the easy grid world, the delta value converges to a constant value for Q-Value methods whereas for other two methods, the delta value converges to 0. Applying a decay learning rate could help with the issue.