Assignment 4

Markov Decision Process

Kuan-E Chao

For this project, I decide to choose (explore) 2 Markov Decision Processes (MDP). One of the MDP is a simple one that has small number of states and the other one is the difficult one which has large number of states. We knew that value iteration and policy iteration would be used to solve for MDPs. Therefore, I am going to use these 2 algorithms to solve for MDP and compare their iterations to its step sizes as well as its running time. Theoretically, we know that policy iteration will take shorter iterations to converge and they should converge to same number of steps. In this experiment, I am going to discuss how these 2 algorithms are differ.

Besides these 2 algorithms, I would also use Q-learning on these 2 MDPs. I will run MDP with knowing the reward model.

Reason I choose Grid World for MDP

The reason I choose Grid World as my assignment is because it kind of simulates how robot is going to act in real life. We can see that how can robot react when they face obstacles. The way the robot chose the path can be explained the MDP well because we may see how the reward and transition functions changed based on different state. And Grid world in this project can explained how the error in transition function and reward function can affect the robot. In the real world we can use this kind of idea to make a robot such as sweepers, which to clean all the places with given limited power. Using MDP we can see how the robot sweepers would clean the sweepers and see how the robot react to obstacle also we can see what robot would do when it is malfunction, which is, not acting the direction it is suppose to acting.

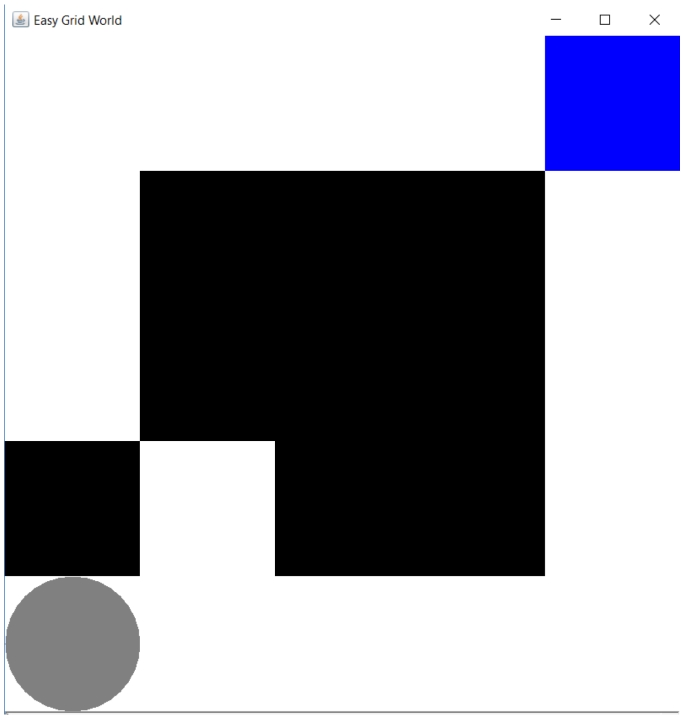
MDP problem explanation for grid world

For both problem, I would let the gray ball represent the current state of the agent, black block represent the obstacle which agent cannot passes through, if it is going to the black block it will stay in the current position. Blue box would represent goal. Agent will subtract one point if it is not moving to the right direction, and it will be out of 100.

I would use value iteration, policy iteration to solve these problems. I would also use Q-learning as a reinforcement learning algorithm to solve these 2 MDPs. I will talk about MDP on Q-learning later.

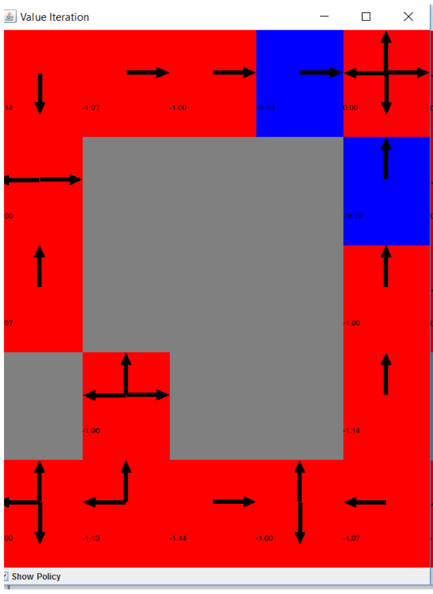
Small Grid World MDP

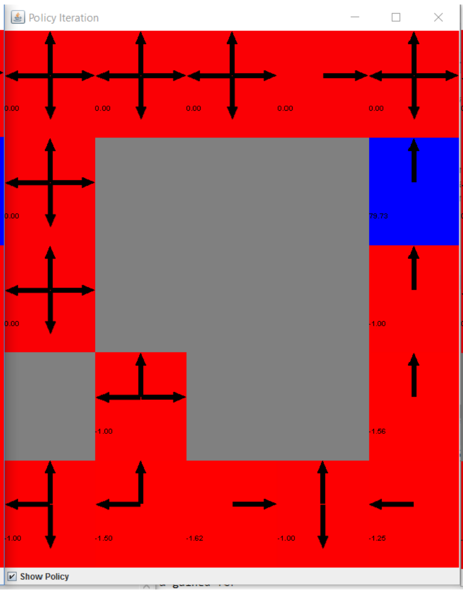
The first One I choose it is the small Grid World, which has simple version and small number of state. The reason I choose this MDP is because I want people to understand the importance of MDP and I want to demonstrate the importance of MDP can do in computer's perspective. I want to use this simple MDP to demonstrate the idea and importance of idea and policy. Therefore, I will try to use this MDP to explain the importance of MDP in small states.

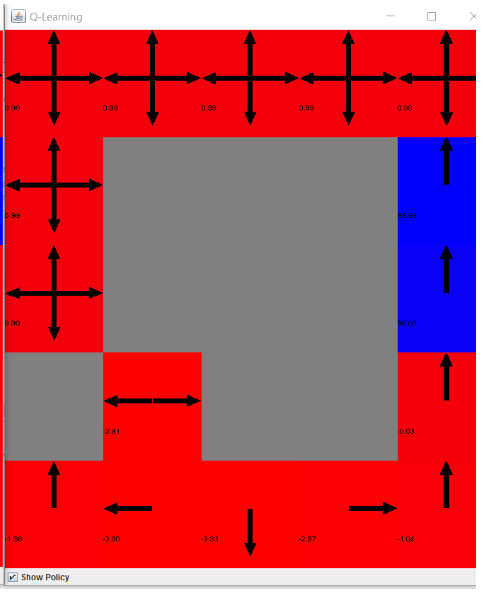


For this problem, my state would be the location of the agent, which is a 5 by 5 Grid World with non-shaded cells. My action would be north, east, south, west. My transitions would be 0.8 moving towards the direction, 0.2 moving at other 3 directions. Reward is 1 if reaches the goal, 0 otherwise. For any move that is not moving toward the expected direction, 1 point will be deducted. For this easy grid world one, I have 16 states, which is considered a small number of states.

Value Iteration (1 iteration)   Policy Iteration (1 iteration)   Q-learning Algortihm(1-iteration)



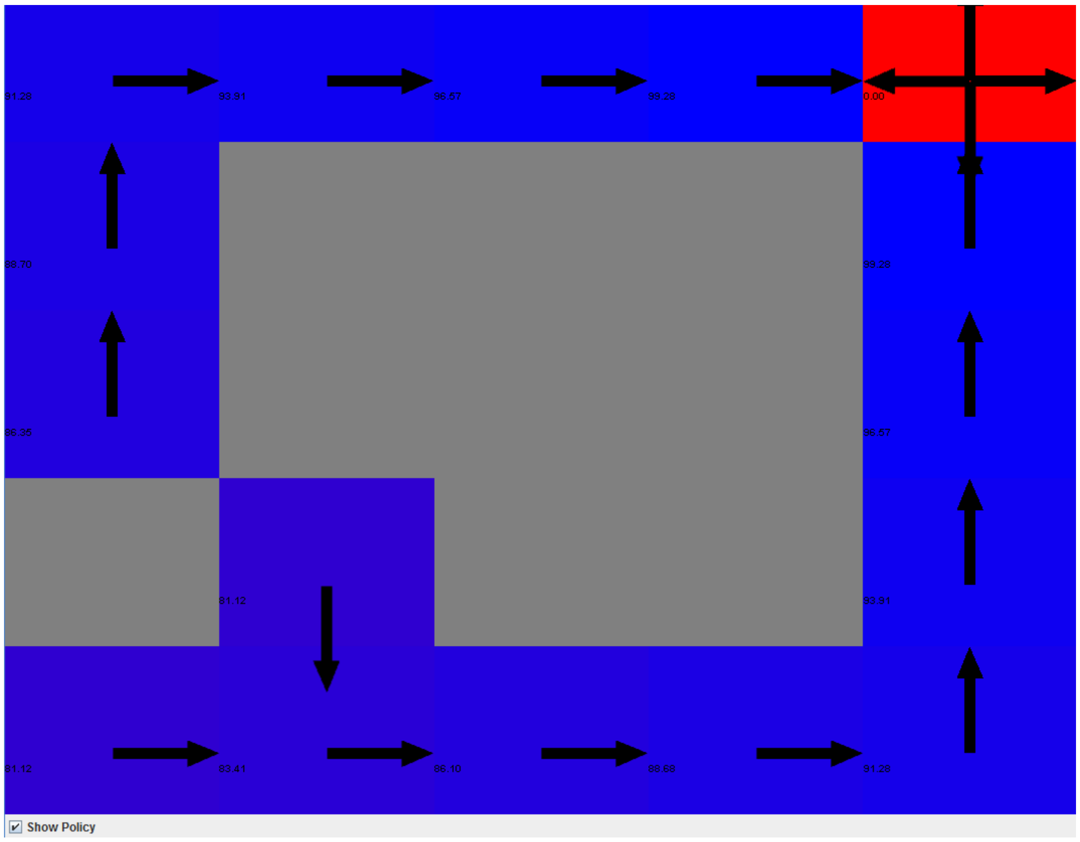




First thing I do is to run one iteration for value iteration, policy iteration and Q-learning. We can see that these algorithms are all going towards the goal when it is close to the goal. However, we can see that there is insufficient iterations to find out the optimal policy. Therefore, I decided to see if I can obtain the optimal policy by increasing the number of iterations.

The next thing I did is run the value iteration. Which is, running the expected value of the agent at the given state. Below are the graph when I run 100 iterations.

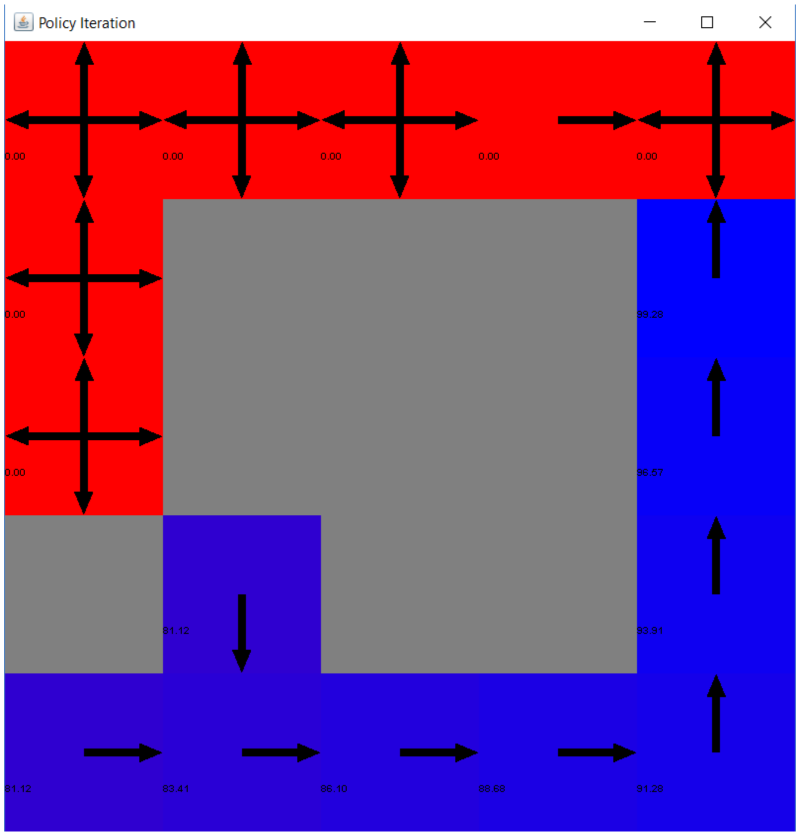
And then I run the value iteration:



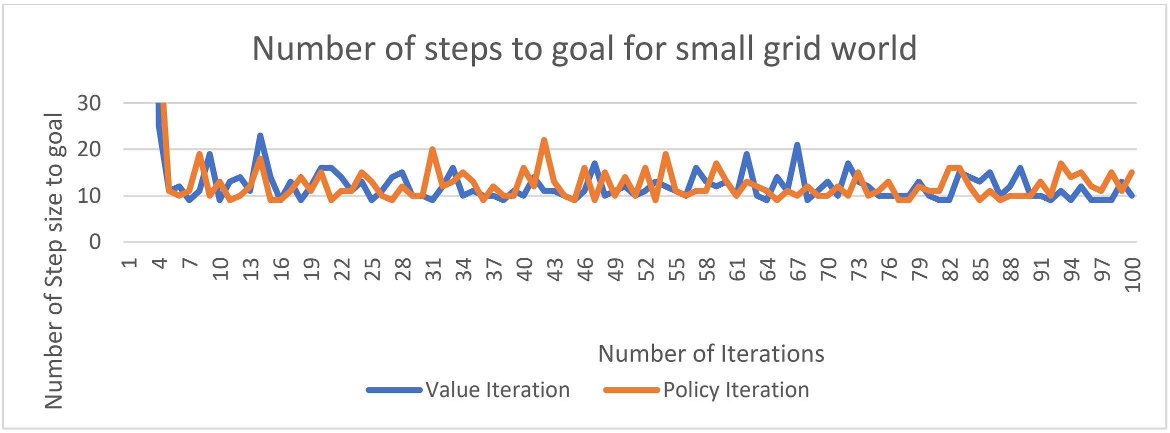
We can see that there is a big chance that it is going to converge using value iteration if it moves the correct direction. Notice that there is still some chance that it is not going to win with 100 points, which is, subtract 1 point due to moving in the wrong positions. However, it appears that all the state can reach the goal with high scores. We can see that value iteration for small states can find the optimal policy in 100 iterations.

Now I am going to use policy iteration to run the policy iterations, which is, solving this MDP by finding the optimal policy. Theoretically policy iteration should converge faster than value iteration and it should converge to the same number of step as value iteration.

And then I run the policy iteration for 100 iterations:



We can see that the red boxes do not converge because it doesn't start from those points, therefore we cannot calculate the policy from those states. However, this policy iteration gives us optimal policy for the rest of the states. Compare with value iteration, we can see that for the small number of states, using policy iteration and value iteration will not be too much differences. The only difference would be that policy iteration cannot calculate the state that cannot be reached from initial state.

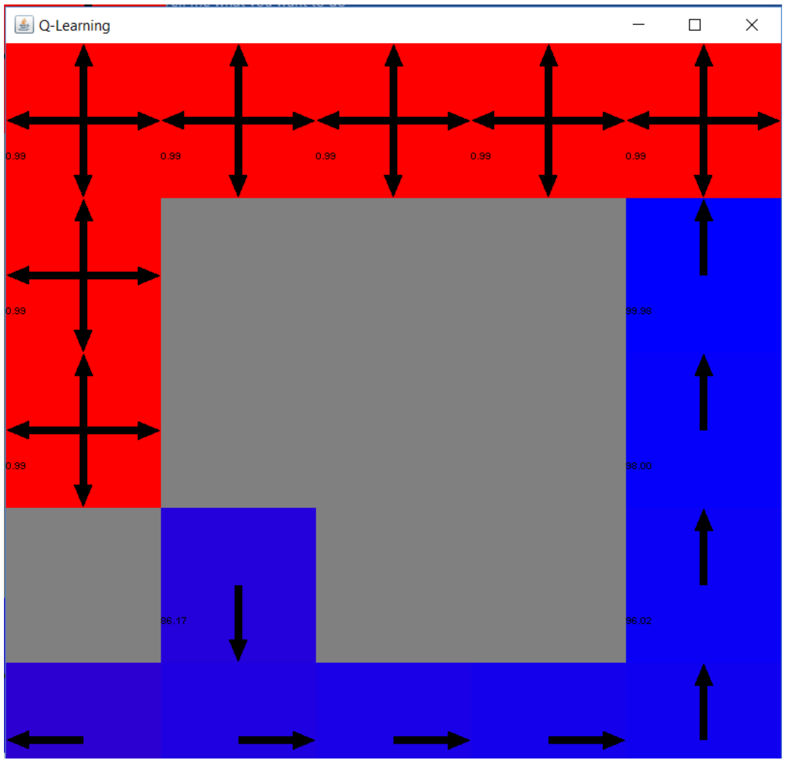


Iterations to converge for both policy iteration and value iteration:

Now I am going to compare these 2 methods and see the number of iteration of these 2 data to converge:

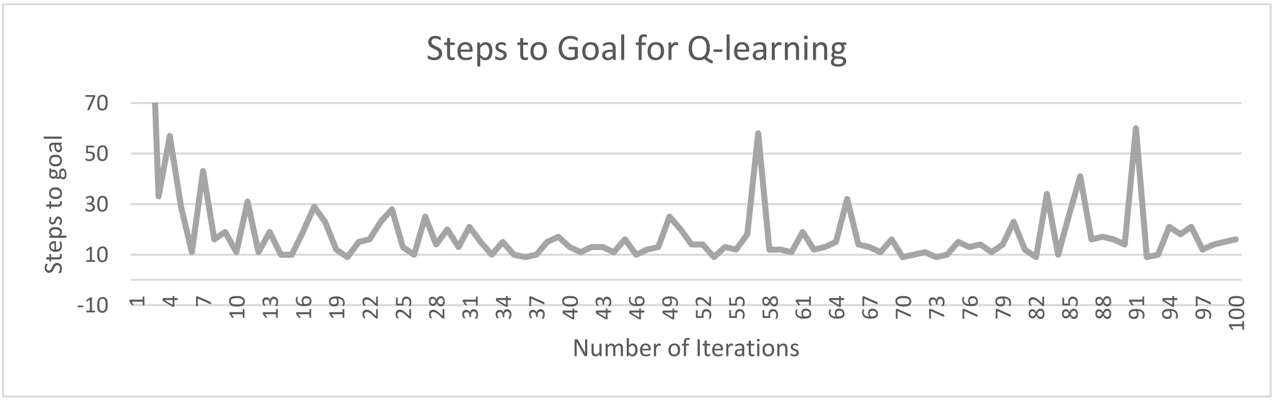
The statistics for policy iterations are on the right on the top. We can see that, there is no significant difference between policy iteration and value iteration from 5 to 100 iterations, they both converge to the same number of steps after 4 iterations.

Now I run the Q-learning with knowing its rewards and model:



We can see that Q-learning also able to find optimal policy after 100 iterations. However, Q-learning will end with lower reward scores due to more steps required to reach the goal.

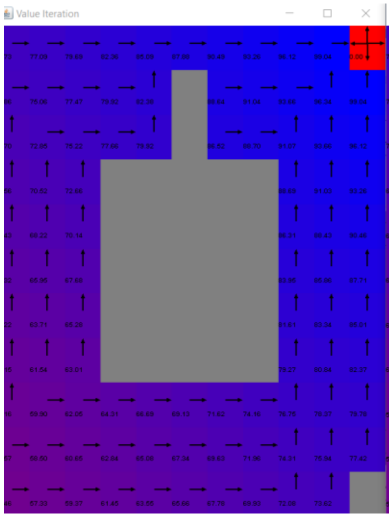
When we are trying to see the number of steps for the agent to reach the goal for Q-learning, I realized that Q-learning does not converge as fast as other 2 algorithms. We can see that as iterations goes high, except for few spikes which makes steps go very high, we are able to see that most of the iterations converges to somewhere around 15, which is slightly more than the number of steps which value iteration and policy iteration needs to reach the goal. A good reason would be that Q-learning does not explore the entire map as other 2 algorithms, therefore it requires more steps to reach the goal.



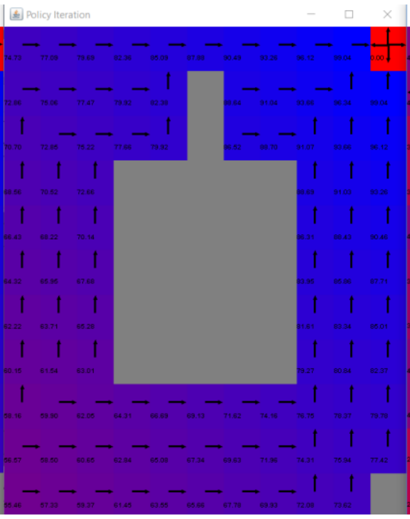
Large Grid World MDP

Now let’s explore the larger grid world. The reason I choose this dataset is to show how these iterations differ in the larger states. The figure below shows the policy when 100 iterations has run. This problem has 94 states (grids), transition probability functions and reward functions act as the same as MDP for small Grid World.

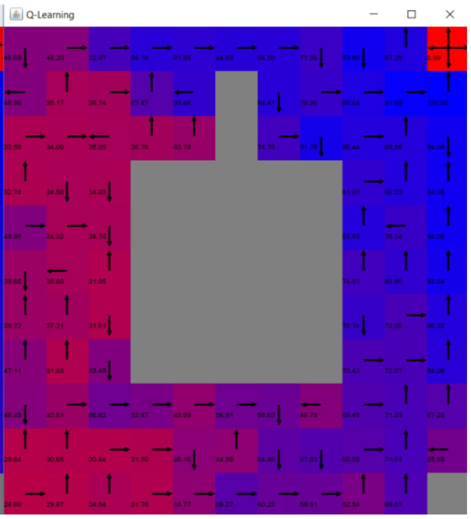
Value Iteration



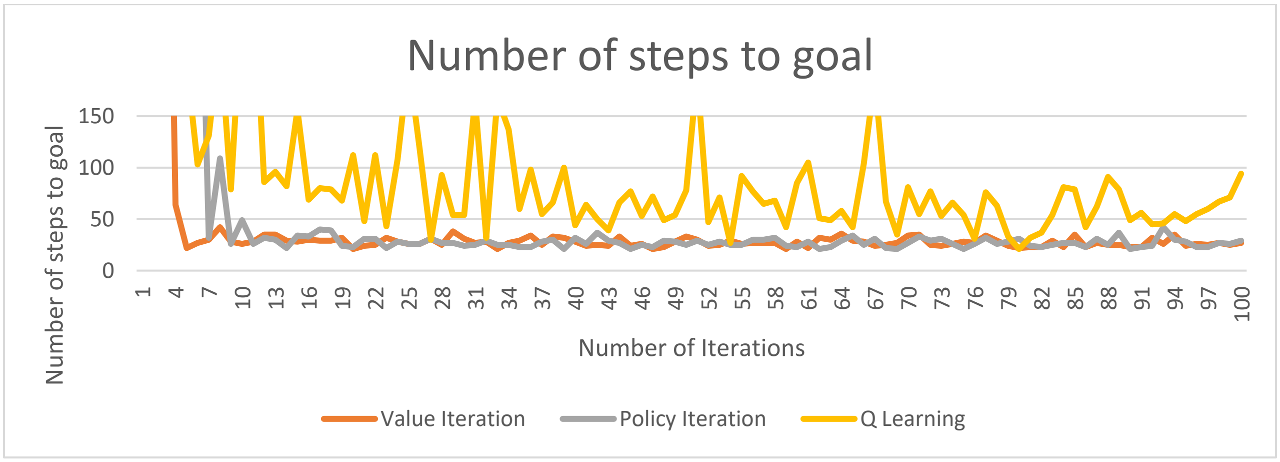
Policy Iteration

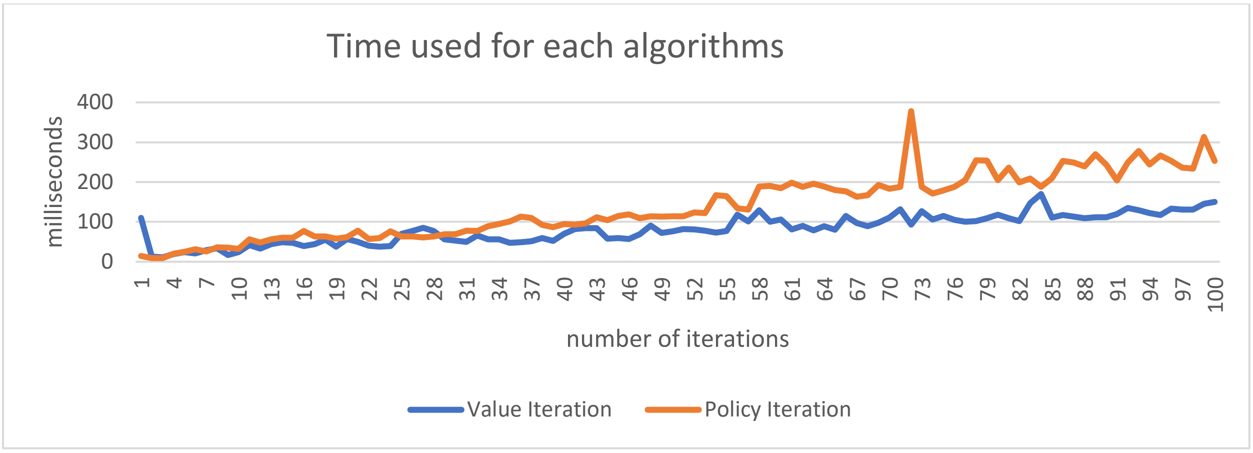


Q-Learning



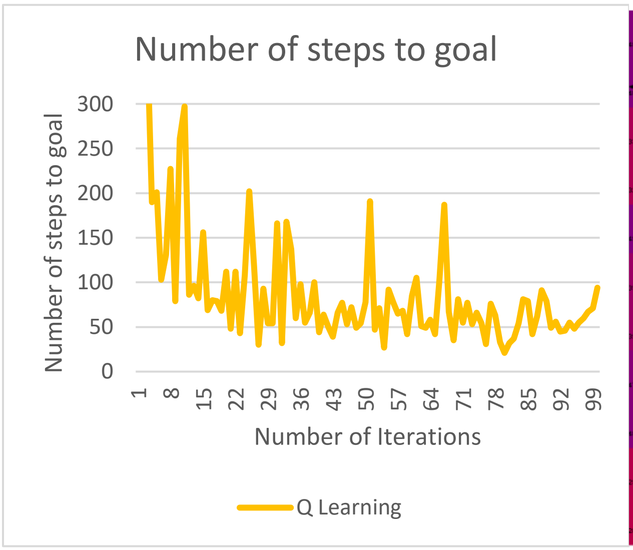
From these graph, we may see that value iteration and policy iteration also converges, and it also find the optimal policy. However, when we run the Q-learning, we can see that this is not the optimal policy. Not all the arrows are pointing toward the goal. This is due to increase of number of states. Large number of states makes Q-learning not been able to find the optimal policy when the number of iteration is not large enough. Therefore, from this difficult Grid World, we can see that it is hard for Q-learning to converge without knowing any information.

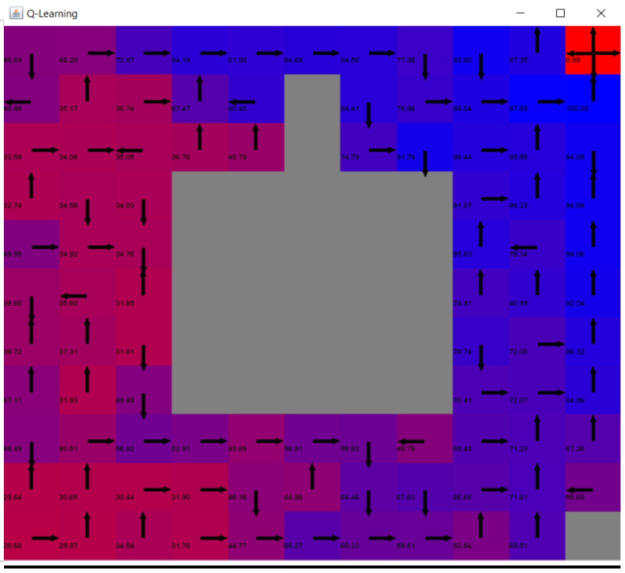




From this time-plot we can see that policy iteration and value iteration still converges to the same number. In practice, policy iteration supposed to converge very fast and quickly but takes longer time per iteration [1]. From the time vs. number of iterations graph, we may be able to see that policy iteration does take longer time per iteration. And it appears to be true when number of iterations getting larger.

Q-learning : Revisited





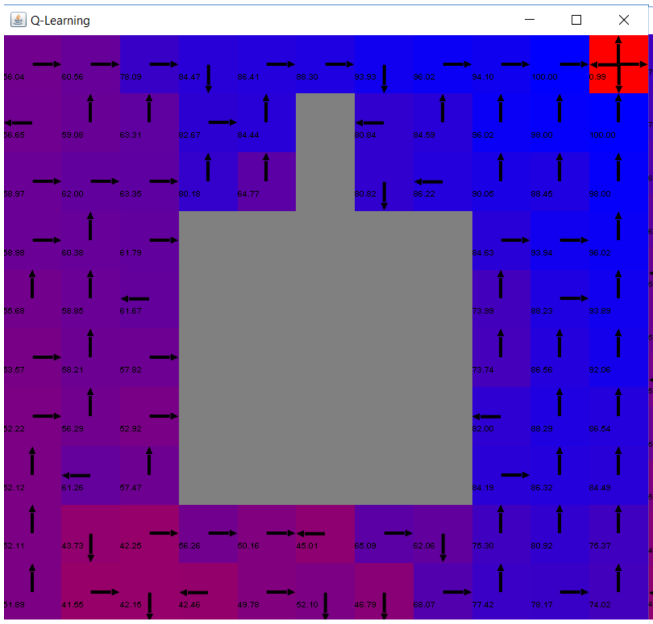
I am going to talk about the Q-learning algorithm for the large Grid World. It turns out that it does not find out the optimal policy after 100 iterations. This is due to larger number of states which makes it takes longer to converge. If we look at the number of steps to goal, we will see that it is not going to a fix number, this is because of number of iteration is not enough and it also does not give the optimal policy.

Compare to value iteration and policy iterations, Q-learning has significantly lower running time compare to value iterations and policy iterations.

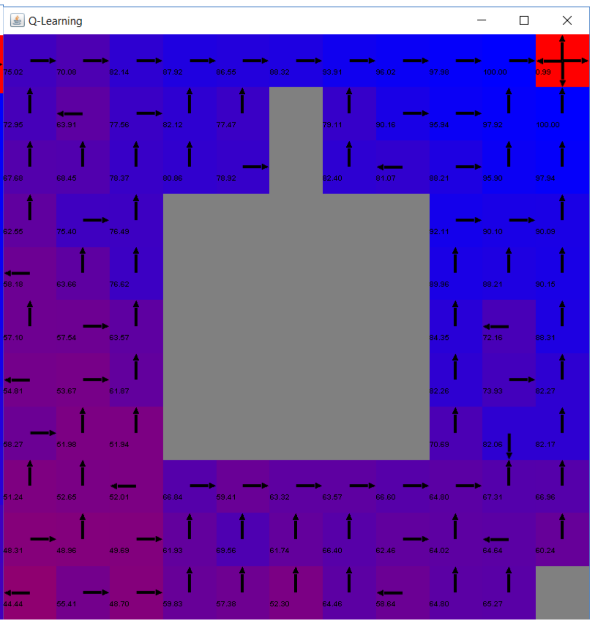
By looking at the graph and statistics for Q-learning, we can see that it converges slower than the other 2. Value iteration and policy iteration converges within 10 steps. The reason why Q-learning converges slowly is because the discount factor is not low (0.2) and the reward is delayed. Therefore, without enough iterations, agent does not necessary know all about the current state. This makes Q-learning does not converge as fast as the other 2.

To make the Q-learning perform better, I decided to make increase the number of iterations and see if Q-learning can find optimal policy. However, we can see that when I increased the number of iterations to 10000 and 50000. (see the graph on next page) The policy only gets a little bit better. Which means, the wrong direction decreases, but it is still not finding the optimal policy. We can see that the reward points have improve for each individual state, but it seems like we can improve this Q-learning algorithm by knowing the model.

Graph after 10000 iterations

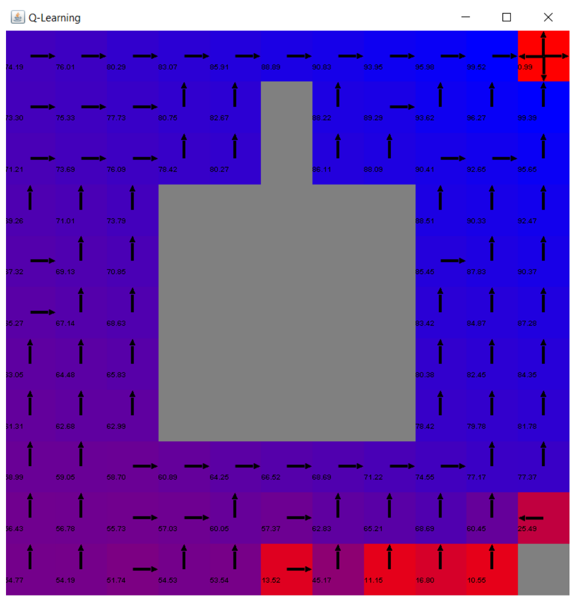


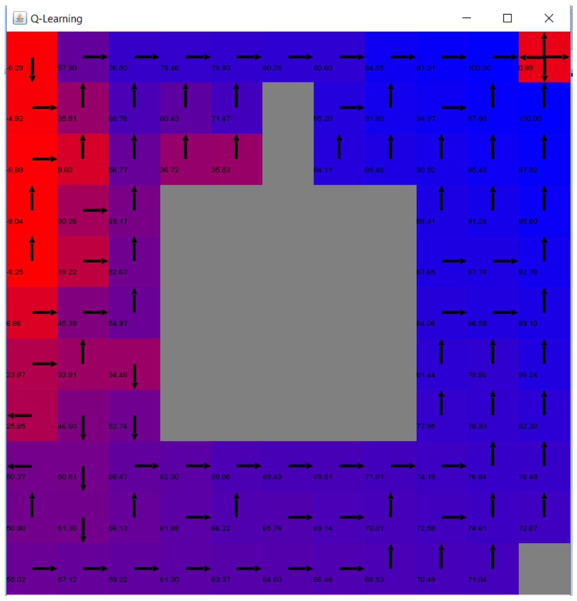
Graph after 50000 iterations

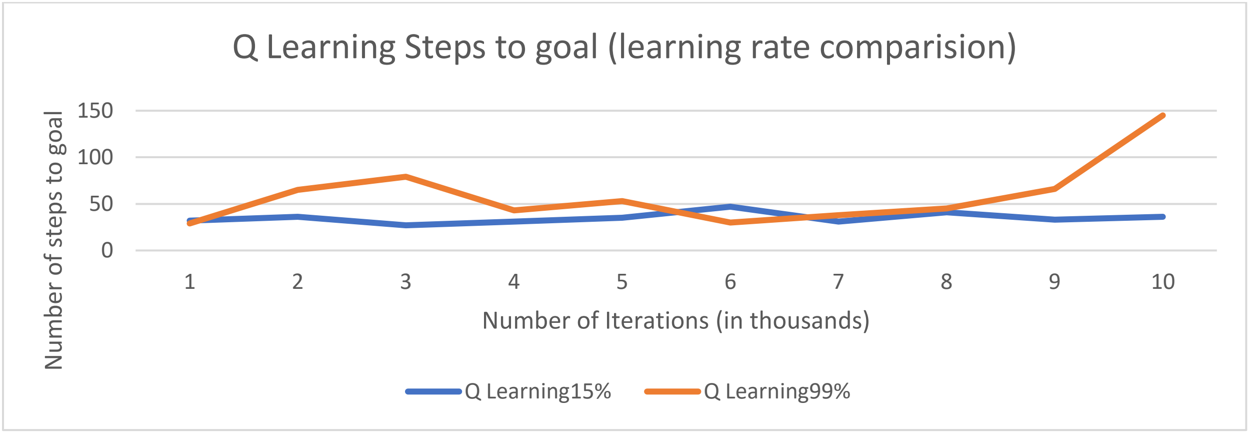


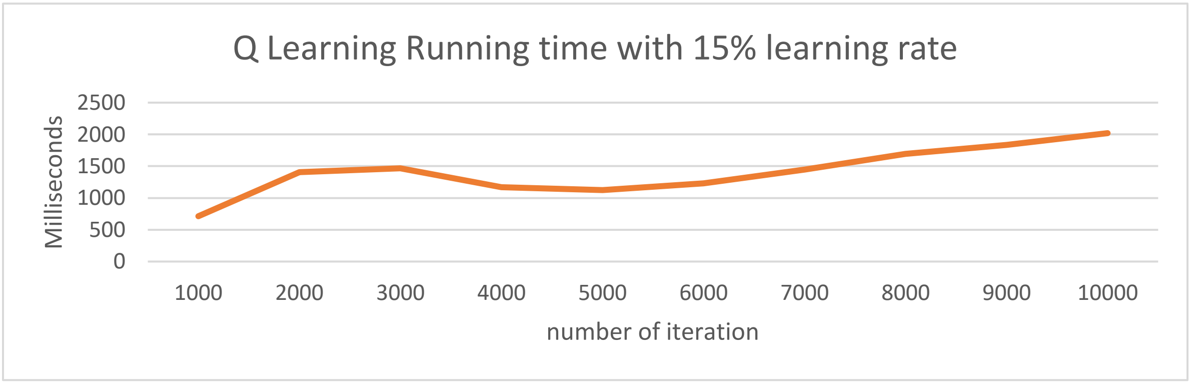
Therefore, I decide to do the way that would improve the Q-learning from the lecture: decrease the learning rate. (See the graph below). We can see that it helps to find the optimal policy by decreasing the learning rate. As we decrease our learning rate from 0.99 to 0.5, we can see that there are more arrows are pointing towards the goal, with some states on the left and left of top not pointing towards the goal. In addition, when I run Q-learning using the learning rate of 0.15, we can see that this graph is extremely close to the optimal policy, except for the state on the bottom left. One way to improve this would be making the learning rate lower, but not too low. We can see that increasing number of iterations (exploring more) or/and decreasing the learning rate can both help Q-learning increasing its performance by decreasing number of steps to goal.

Learning rate of 0.15 Learning rate of 0.5









When the number of iterations are above 1000, we can see that Q-learning has average of 35 steps to converge when I use learning rate of 15%, which is more stable than the original learning rate. We can also see that 35 steps to converge is almost about the same as the step of policy iteration and value iteration, which is very low and stable when we decrease the learning rate.

When I run the Q-learning algorithms, I also notice that Q-learning has linear running time, which is a good running algorithms because we can see that large number of iteration will not make Q-learning algorithm running too long, it is proportional to number of iterations.

Conclusion

From those 2 Grid World MDPs, one has a small number of state and another has large number of states, we can see that both of the MDPs converges pretty fast with value iteration and policy iteration. Small one converges with 4 iterations and the large ones converges in 9 iterations. Both of the algorithms converges to the same number. We can see that since these 2 MDPs have low discount rate, these 2 algorithms seem to converge with same number of iterations even though theoretically policy iteration should converge faster. However, we did statistically proven that policy iteration requires less iteration to converge when I run the MDP with larger number of states.

For the Q-learning, we can see that it seems to be able to find the optimal policy for MDP with small states. For MDP with large number of states, we can see that we need to slow down the learning rate in order to find the optimal policy. We are also able to see that Q-learning requires more steps to move toward the goal when we compare with value iteration and policy iteration. On the other side, Q-learning is a good reinforcement learning algorithm because it has low running time and it is very good to use it to approximate the optimal policy when the probability transition function and reward function are not available. However, if we know the transition probability function and reward function, we should solve using value iteration and policy iteration because it takes less iteration to converge and it requires less steps to reach the goal.

Citation:

1. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node21.html
2. Juan J. San Emeterio. https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4/blob/master/README.md
3. BURLAP, The Brown-UMBC Reinforcement Learning and Planning (BURLAP) java code library
4. https://inst.eecs.berkeley.edu/~cs188/sp12/slides/cs188%20lecture%2010%20and%2011%20--%20reinforcement%20learning%206PP.pdf