Markov Decision Process Problems

Zhilan Deng

Abstract- This paper focus on three Markov Decision Process (MDP) problems and applies value iteration and policy iteration on all three of them. Then this paper applies Q learning on these three problems and compare the results with the that in value iteration and policy iteration.

PART 1 VALUE ITERATION & POLICY ITERATION

I. INTRODUCTION

MDP PROBLEMS

This paper selects three MDP problems for this project: (1) Taxi problem, (2) Frozen-lake problem and (3) Gambler's problem.

Taxi problem is a grid-based environment, and the goal of this problem is to pick up the passengers at one location and drop them off at another location. The environment is fixed and there are 500 states in total. The pick up and drop off locations are all chosen randomly in the environment as well as the start point. There are 6 actions in total, 4 for moving in north, south, east, west directions and 1 for pick up, 1 for drop off. The rewards are: -10 if the taxi attempts to pick up a passenger when there is no passenger at the location, 20 if the taxi drop off the passenger at the correct drop off location, and otherwise -1 per time step. This paper selects this problem because of the increasingly speed of the development in self-driving car and Shanghai just conducted a road-test for the first self-driving taxi in this October. All these together makes the navigation problem interesting and worth to study with. However, this problem is a deterministic environment, so this paper also looks like another problem which has the similar environment but in a stochastic world.

Frozen lake is a grid-based environment but is in a stochastic world. The goal of this problem is to start from state S and goes to state G. The environment and states are user-defined, there are four types of states in this problem: (1) S for the starting points, (2) G for the goal point, (3) F for normal field that the agent can step on and (4) H for holes that the whole process will end without any reward when the agent falls into the holes . The rewards are: 1 for successfully reaching the goal state G and 0 otherwise. There are 4 actions: left, down, right and up in this problem but this world is stochastic and if the agent's action succeed only 1/3 of the time while the other 2/3 are split evenly between the two directions orthogonal to the intended direction. This paper selects this problem because of the similar reason of the taxi problem in the navigation of self-driving car and also this environment is stochastic so it might perform differently and more interesting compared to a deterministic problem like the taxi problem.

Gambler's problem is a non-grid problem, in this problem the agent places a bet on whether a coin flip will show heads. If the coin flip shows head, then they win the same amount of money the agent bet, otherwise they lose all the money the agent bet. The coin lands on heads 40% of the time and the game will end when the agent loses all the money or earn more than 100. This paper selects this problem because this problem has a very different settings with the grid world and the action space is always changing during the process: the agent cannot bet more than the money it has. And also, unlike the taxi problem and the frozen lake problem where the rewards of each step is only based on which state, the reward of gambler's problem does not relate to the state the agent is, and it is interesting to see how value iteration and policy iteration handle this problem differently than the two problems above.

II. POLICY ITERATION & VALUE ITERATION

POLICY ITERATION VS. VALUE ITERATION

Policy iteration and value iteration are similar to iterate through the whole process until converge or meet the maximum number of iterations. The difference is that policy iteration focused on policy converge to find the optimal policy while value iteration focuses on value function converge and to find the optimal value function.

In policy iteration, the algorithm first selects a random policy and then the algorithm does policy evaluation and finds the value function of that policy. The next step is policy improvement and is to improve the policy based on the value function that the algorithm found earlier. This algorithm then repeat policy evaluation and policy improvement until the policy converges and become stable.

In value iteration, the algorithm starts with a random value function and then iterate the process until the value function converges and optimal, the optimal policy will be derived from the optimal value function.

The application of these two algorithms on the three MDP problems will be presented in the following sections and this paper will compare the running time, number of iterations and the performance of these two algorithms in the three MDP problems.

TAXI PROBLEM

Policy Iteration

The result of policy iteration is shown in figure 1. Figure 1(a) shows how the policy converges to the optimal policy in the iterations. This paper set every time the algorithm improves a policy as one iteration and it shows that the policy converges very fast and only takes 10 iterations to find the optimal policy. Figure (b) shows the reward when performing the policy in a different randomly set environment and the optimal policy successfully solves the problem by pick up the passengers and drop them off at the correct location and earn 20 points reward. The running time of policy iteration is: 59.75s

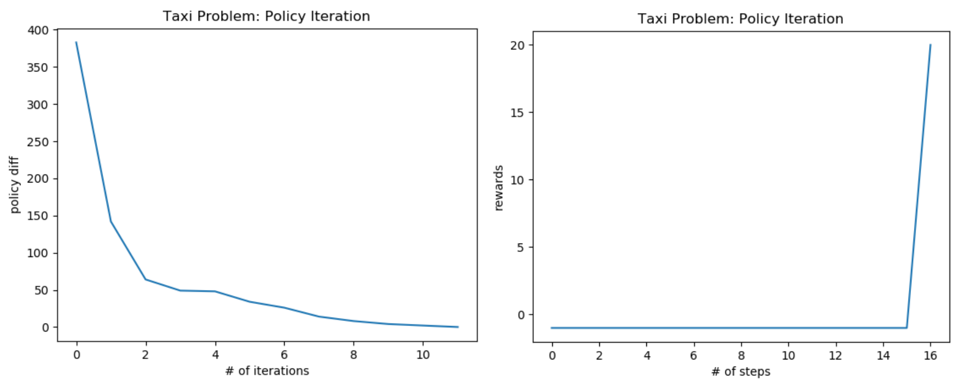

Figure 1. Policy iteration performance in taxi problem.

The results of policy iteration in different gamma values is shown in figure 2. Figure 2(a) shows the convergence of policy iteration in different gamma values. This paper only shows 20 iterations for the display purpose, but when gamma = 0.1, the policy didn't converge in the maximum number of iterations which is 1000. Figure2(a) also shows that the policy is not stable within 20 iterations when gamma = 0.1 while the policy convergence when gamma in other values larger than 0.1. The performance of gamma in other values are similar good, they all converge within 12 iterations. And if look at the difference between iteration 5 and iteration 7, when gamma value increases, the policy converges faster. Figure 2(b) supports the observation in figure 2(a). When gamma = 0.1, the algorithm failed to find the optimal solution and it takes 200 steps in the whole problem while the other policies takes only 16 steps to solve the problem.

The reason of the different performance under different gamma values lies in the equation of policy evaluation and policy improvement:

$$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\left[r+\gamma V(s')\right]$$

$$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r+\gamma V(s')\right]$$

When gamma is small, the dominant impact will be the direct reward of current state based on the policy, so the learning part from future states will be small, and the learning process will be slow, on the other hand, if gamma is large, the algorithm will learn very fast which is shown in figure 2(a) and large gamma value works good in Taxi problem.

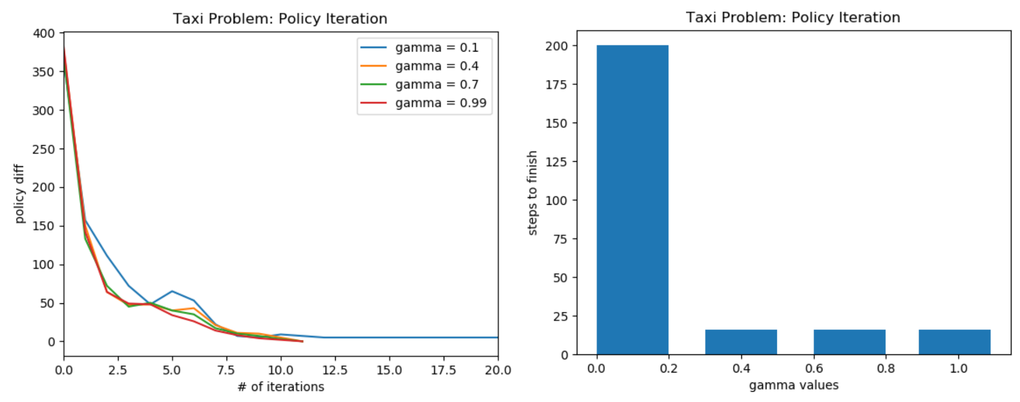

Figure 2. Policy iteration performance under different gamma values

Value Iteration

The result of value iteration in shown in figure 3. Figure 3(a) is the convergence of value iteration. This paper set each time the algorithm update the value function as one iteration. And the algorithm shows that the policy convergence around 250 iterations. Figure 3(b) shows the performance of the optimal policy found using value iteration and the agent successfully solve the problem in 16 steps. the running time of value iteration in taxi problem is: 5.97s.

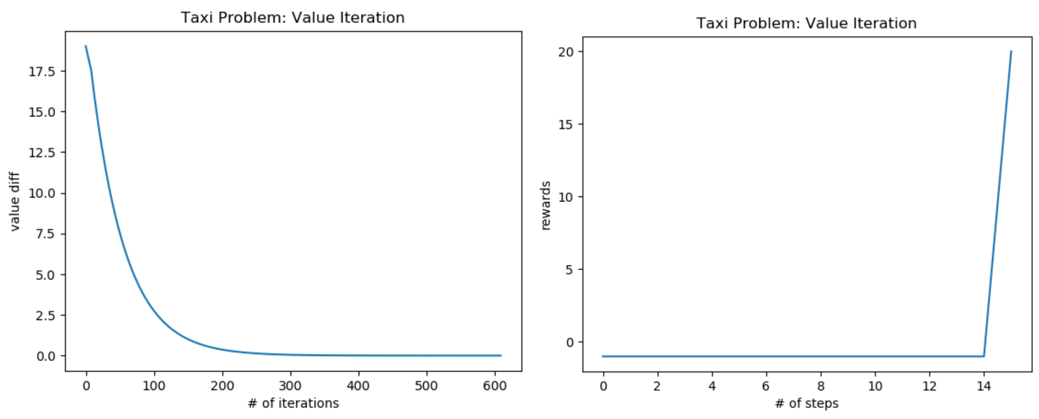

Figure 3. Value iteration performance in Taxi problem

The result of the performance of value iteration under different gamma values is shown in figure 4. For value iteration algorithm, different gamma values performance equally good.

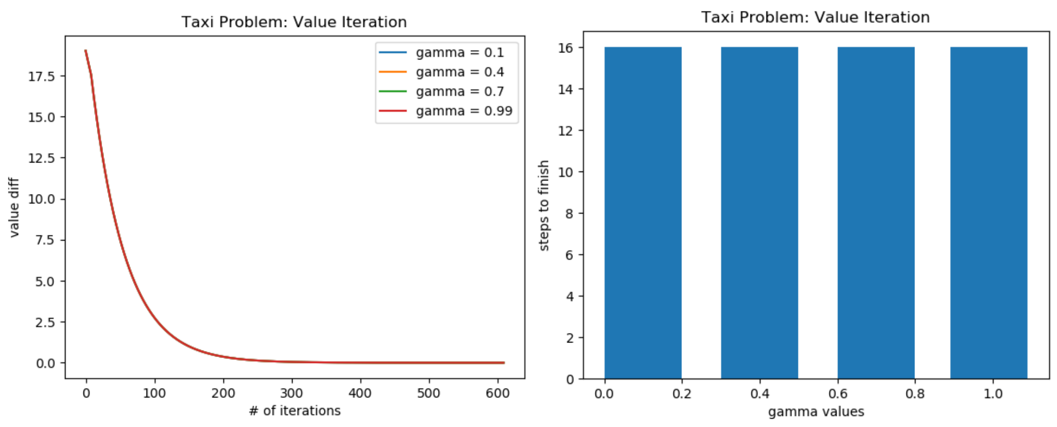

Figure 4. Performance of value iteration under different gamma values.

Comparison of policy iteration and value iteration

The comparison of policy iteration and value iteration is shown in figure 5. This paper compared the running time, number of iterations while convergence and the number of steps in successfully solving the problem in the optimal policy found by each algorithm. The running time is 59.75 and 5.97 respectively for policy iteration and value iteration. The number of iteration that the algorithm converges is 11 and 610 respectively, for display purpose, the figure below divided the number of iterations to converge by 10. Another thing to be noted here is that, although the number of iteration of policy iteration is less than the number of iteration of value iteration, the running time of one iteration in policy iteration is much longer than that in value iteration because in each iteration the value function has to converge in the current policy which makes another iteration process inside each policy iteration. The performance of the optimal policy is the the same and further more, the optimal policies these two algorithm found are the same as well.

In Taxi problem, the world is deterministic and simple, so in this situation value iteration performs much better than policy iteraton in terms of the running time and converge speed. The faster speed is because that value iteration didn't update the policy in the process, but only to update the value function and only extract the policy once from the final optimal value function.

So to conclude, value iteration is more suitable for Taxi problem which is deterministic and simple.

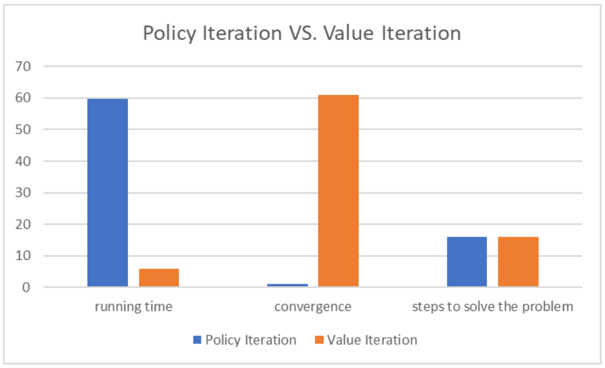

Figure 5. The performance of policy iteration and value iteration of taxi problem.

FROZEN-LAKE PROBLEM

Policy Iteration

The result of policy iteration in Frozen-Lake problem is shown in figure 6. Figure 6(a) shows the convergence of policy iteration in Frozen-Lake problem and figure 6(b) shows the performance of the optimal policy found by policy iteration. This figure shows that although the policy converges in only 4 iterations, but the optimal policy found didn't solve the problem, the agent ends the game with no reward which mean that the agent falls into the holes. The running time of this algorithm is 0.66s.

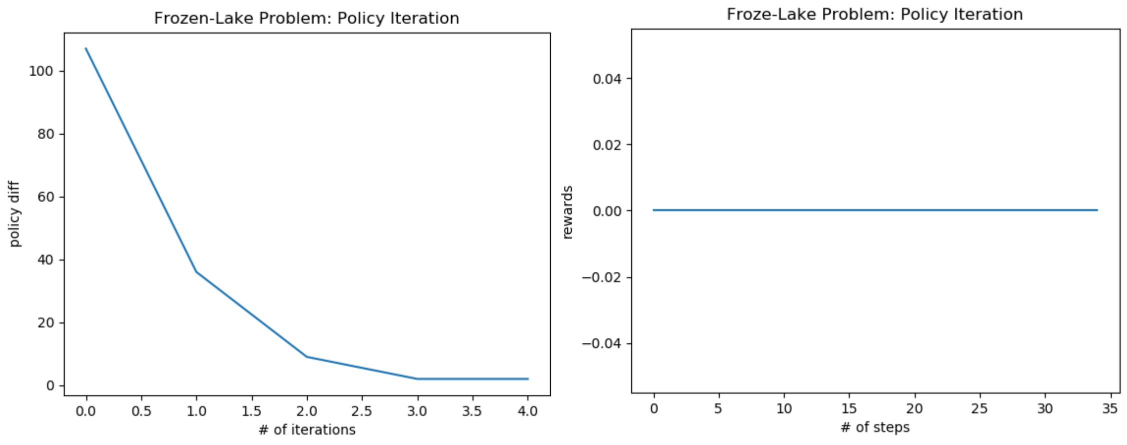

Figure 6. Performance of Policy Iteration in Frozen-Lake problem.

Value Iteration

The result of value iteration in Frozen-Lake problem is shown in figure 7. Figure 7(a) is the convergence of value iteration algorithm and figure 7(b) is the performance of the optimal policy found by value iteration. The value function converges around 1500 iterations and the optimal policy found by value iteration successfully solves the problem. The agent ends the game with reward 1 which means that the agent successfully reaches the goal state. The running time of value iteration algorithm is 0.27s.

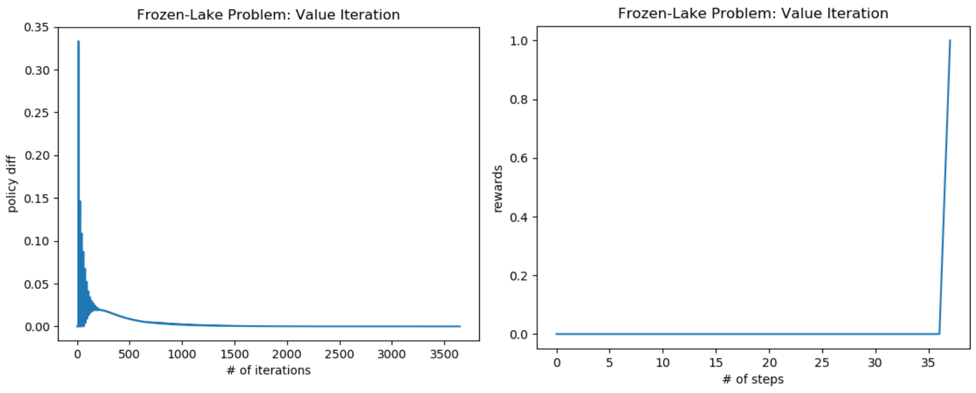

Figure 7. The performance of value iteration algorithm in Frozen-Lake problem.

Policy Iteration VS. Value Iteration

The comparison between policy iteration and value iteration in this problem is more obvious than that in the taxi problem. In this problem, policy iteration failed to find the optimal policy which can solve the problem while the value iteration successfully solves the problem, also the running time of value iteration is less than that of policy iteration. So for this specific Frozen-Lake problem where the environment is stochastic and the number of states is small, value iteration performs better than policy iteration.

This paper also did the same experiment on a Frozen-Lake problem with a user-defined map where there are 900 states, and this will be discussed in detail in the following sessions.

GAMBLER'S PROBLEM

Policy Iteration

The result of policy iteration is shown in figure 8. Figure8(a) shows the convergence of policy iteration in Gambler's problem and figure 8(b) shows the policy of each state. In this experiment, the state is the capital the agent has and the probability of heads is 0.4. The policy converges in 41 iterations and the running time of policy iteration in Gambler's problem is 0.2s. It is interesting to see how the policy changes dramatically between 20 and 80. The agent tend to bet all its money at some states, such as 40 and 50, but doesn't bet or just bet a little in the next states. The reason might be that the probability of head is 0.4, so the agent has less probability to win if it bets randomly, in this case, most of the states, the agent just doesn't bet or only bet a small amount of money to limit the loss when the probability of head is only 0.4. The policy might change when the probability of heads changes and the results are shown in figure 9.

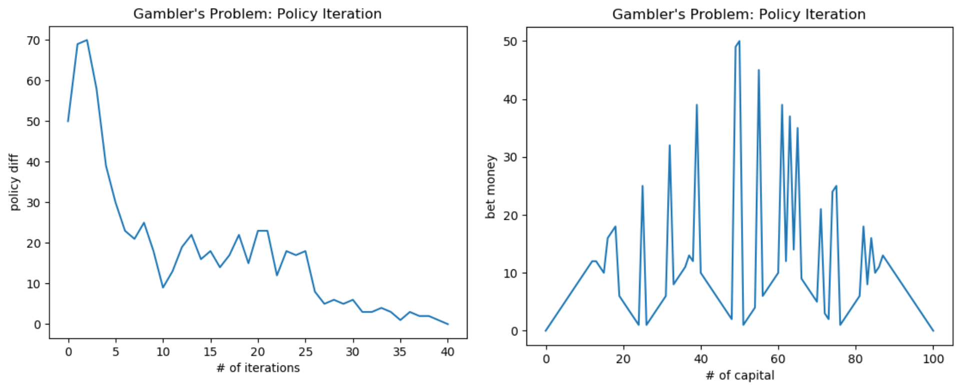

Figure 8. The performance of policy iteration in

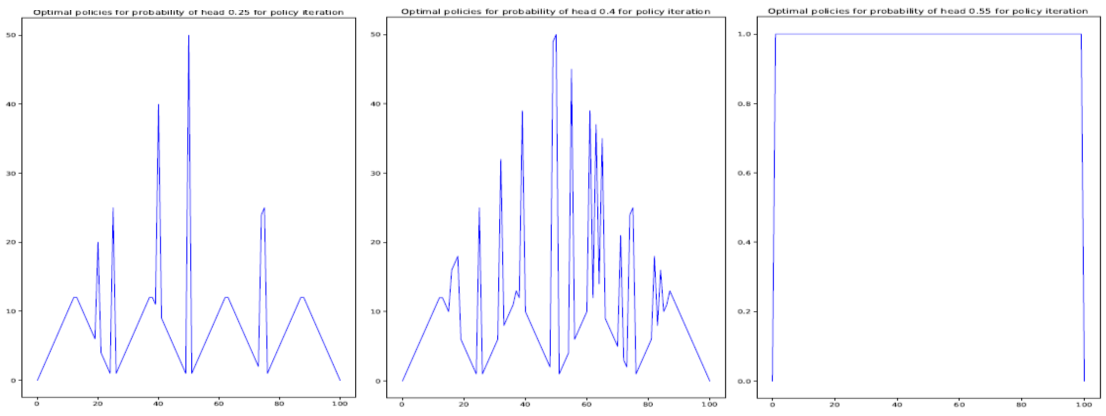

Figure 9. The policy under different probabilities of heads: (a) 0.25, (b) 0.4, (c) 0.55.

Figure 9 shows the different policy under different probabilities of heads. When the probability of heads is as low as 0.25, the agent tends to not bet or bet a little in most of the time to limit the loss, when the probability of head is 0.4, the agent jumps from bet all the money it has and bet a small amount of money it has. When the probability of head is as high as 0.55, the agent just bet its money from the beginning and it will win in any situations.

Value Iteration

The result of value iteration is shown in figure 10. Value iteration converges in 16 iterations and the optimal policy found by value iteration is almost the same to that found by policy iteration. The running time of value iteration in this problem is: 0.104s

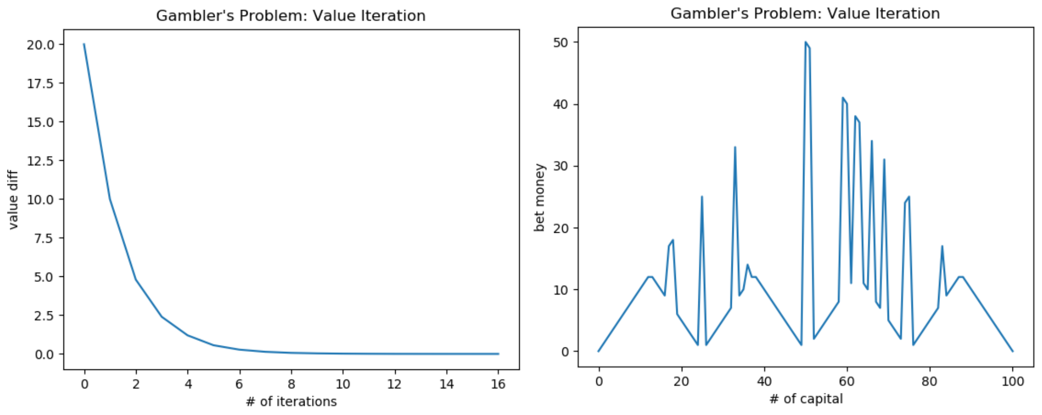

Figure 10. The performance of value iteration in Gambler's problem.

Policy Iteration VS. Value Iteration

The comparison of policy iteration and value iteration in Gambler's problem is shown in figure 11. Similar to Taxi problem and Frozen-Lake problem, the running time of value iteration is shorter than that in policy iteration. Contract to the previous two problems, the number of iterations in policy iteration is larger than that in value iteration. And since in each iteration, policy iteration would take longer time than value iteration. So in Gambler's problem, value iteration would be a better algorithm.

IMPACT OF THE NUMBER OF THE STATES

For this section, this paper selects Frozen-Lake problem to show the impact of different number of states. This paper will compare between the default 16 states Frozen-Lake problem and a user-defined 900 states Frozen-Lake problem. The results of the comparison is shown in figure 12.

Figure 12(a) shows that when the states space increases from 16 to 900, the running time increases significantly from 0.43 to 235.55, while the number of iterations until converge increases from 2 to 19 when using policy iteration. When using value iteration however, the running time doesn't increase as significantly as that in policy iteration, just from 0.27 to 15.88, but the number of iterations until converge increases dramatically from 3684 to 18100.

The significant impact of state space to policy iteration is because that in policy iteration, in each step, the algorithm need to loop through all states multiple times to get the value function of current policy, so when the space increases, the running time of looping through each state increases, so the overall running time increases dramatically. However in value iteration, there is no need to loop through each state, and the policy will be only extracted once after get the optimal value function, so the running time doesn't increase as much as that in policy iteration. The significant change in the number of iterations is that, when the number of states increases, the value function will become much harder to converge, which makes the number of iteration increases significantly with the increasing number of states.

To conclude, if a problem has a large states space, value iteration will be a better solution since the running time will not increase significantly while the state space increases.

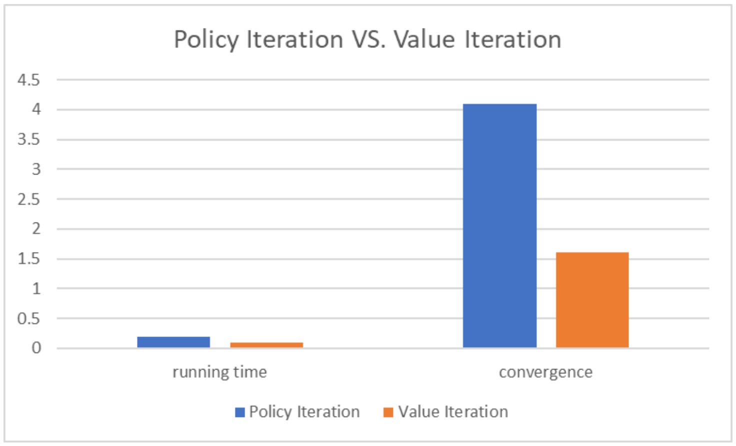

Figure 11. The comparison of policy iteration and value iteration in Gambler's problem.

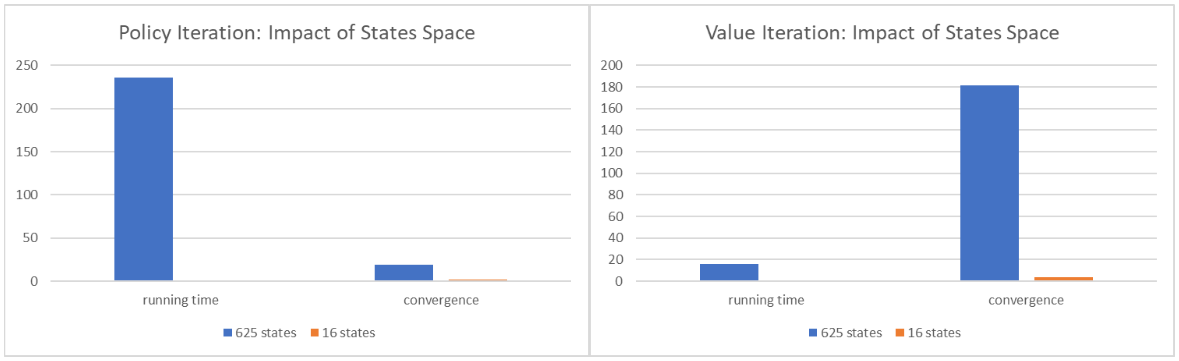

Figure 12. The impact of states space in Frozen-Lake problem.

DISCUSSION

After applying policy iteration and value iteration on these three problems, value iteration performs better than policy iteration in all three problems, with better performance and less running time. In taxi problem and froze-lake problem, policy iteration converges in less number of iterations than value iteration, but converges slower than value iteration since the running time of one iteration in policy iteration is much longer than that in value iteration. In Gambler's problem, policy iteration converges in both more iterations and more running time. Also, large state space has an significant impact on policy iteration algorithm too, when the number of states increases, the running time of policy iteration will also increase significantly.

To conclude, if there is no other reason, according to the experiments in this paper, value iteration would be a better choice for MDP problems.

PART 2 Q-LEARNING

I. INTRODUCTION

This part will apply Q-learning on Frozen-Lake problems with 16 states space and the Gambler's problem. The results of Q-learning will be compared with the results of value iteration above.

This paper selects Q-learning because Q-learning is relatively simple and straightforward to implement. It is an off-policy algorithm while it doesn't select action based on the policy but through greedy selection, and the goal of q-learning is to find the policy which maximize the total rewards. It creates a Q table which stores the [state, action] pair and in each step, the agent will either exploit by acting as the action which has the largest value in the [state, action] pairs in Q table or it will explore by acting randomly. After taking the action, the agent will update the Q table based on the equation below:

Q[state, action] = Q[state, action] + lr * (reward + gamma * max(Q[new_state, :]) — Q[state, action])

While the learning rate decides how fast the agent learns, large learning rate will make the new value dominant and will learn fast but lead to an unstable updates. Small learning rate will make the old value dominant and the agent will learn slowly during the process.

Another difference in Q-learning compared to value iteration and policy iteration is the stop criteria. For policy iteration and value iteration, the algorithm stops when the policy or value function converges, or it meet the maximum number of iteration pre-defined. However in Q-learning, the agent will stop learning when it meet the maximum episode that pre-defined, convergence is not a stop criteria in Q-learning.

II. APPLYING Q-LEARNING

FROZEN-LAKE PROBLEM

The result of Q-learning application in Frozen-Lake is shown in figure 13. The running time of Q-learning algorithm is 699.6. Q-learning successfully solved the Frozen-Lake problem which shows in figure 13, the agent ends the game with reward 1. The optimal policy found by Q-learning performs better than value-iteration or policy iteration, the agent can end the game within 9 steps while the optimal policy found by value iteration and policy iteration solves the problem in 14 steps. but since the running time of Q-learning is significantly longer than that of value iteration, so the user has to balance between the performance and the running time to decide which one is the best selection for a specific project.

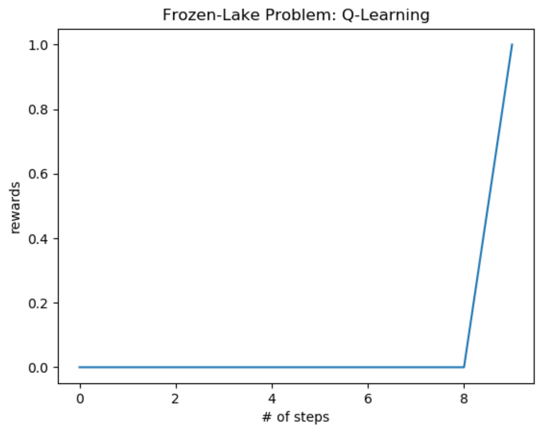

Figure 13. The performance of the optimal policy found by Q-learning in Frozen-Lake problem.

GAMBLER'S PROBLEM

The result of Q-learning application in gambler's problem is shown in figure 14. The optimal policy found by Q-learning is similar to that found by value iteration, the policy jumps between betting a large amount of money and betting no money or small amount of money. And the general trend of betting small amount of money on the two ends when the agent has less capital or large capital, but bet more money when the user has capital between 20 to 70. The running time of Q-learning is 0.45.

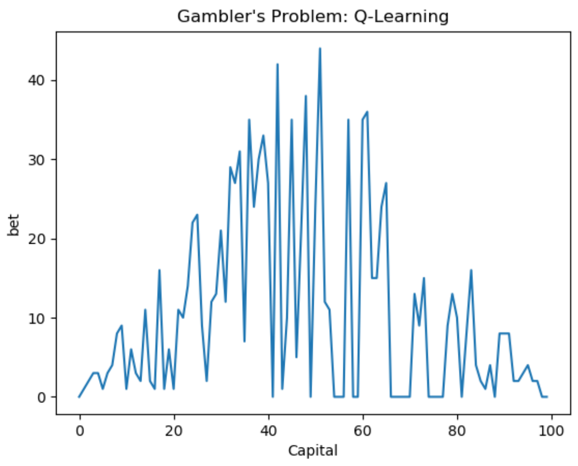

Figure 14. The result of Q-learning in Gambler's Problem.

DISCUSSION

To conclude the observations above, compared to value iteration and policy iteration, Q-learning does not require the policy or value function to converge and it is an off-policy algorithm which selects the action based on greedy method and the set up is simple and straightforward. The performance of Q-learning is as good as value iteration, but the running time increases significantly in the Frozen-Lake problem because Q-learning has to loop through many episodes to learn. The disadvantages of Q-learning is that, user has to decide the maximum number of episodes in learning, too large number of episodes will cause long and may unnecessarily training time as in the frozen-lake problem, while small number of episodes might cause the agent to not learn enough and doesn't come up with an optimal solution.

References

[1]https://github.com/dennybritz/reinforcementlearning/blob/master/DP/Gamblers%20Problem%20Solution.ipynb
[2] https://gist.github.com/persiyanov/334f64ca14f7405f5b3c7372fecf2857
[3] https://www.kaggle.com/angps95/intro-to-reinforcement-learning-with-openai-gym