CS 7641 Markov Decision Processes

Hyesung Ji (hji61@gatech.edu)

Abstract— Generally, the Machine Learning algorithms are categorized into Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Reinforcement Learning is a training method to make the best decision by giving a reward. Also, the Markov Decision Processes (MDP) is used for the Reinforcement learning and the Bellman equations and value functions are used in the MD. For solving this MDP problems, the Value Iteration, Policy Iteration, and Q-learning algorithms are introduced. Also, two different problems were investigated to study the performance of three algorithms.

I. INTRODUCTION

Reinforcement learning is normally used for playing a game or training a vehicle to drive by awarding rewards, and it can be useful so that the game or vehicle can be moved into the right way [1]. In this project, all of these games/autonomous equipment are called 'game'. So, defining the environment of the game and reward system can the most important parts in Reinforcement learning. If the environment or reward system is incorrect or has a problem, the output of Reinforcement learning cannot be successful. And like a other Machine Learning methods (Supervised and Unsupervised Learning), the hyperparameter tuning is very important. In this project, two different Markov Decision Process problems are selected. The first one is grid problem ('Frozen Lake') and another problem is non-grid problem ('Forest Management'). In terms of algorithm, the value iteration, policy iteration, and Q-learning algorithm are used in this project.

A. Problems

In this project, grid problem and non-grid problem are researched. A small-size and a large-size random maps are used for the grid problem. The size of maps are 5×5 and 20×20. Also, the same numbers of states (25 states, 400 states) are used for the non-grid problem.

1) Grid problem: Frozen lake

The frozen lake problem is to begin from the start point of the up-left point of the grid and to arrive at the goal point of the down-right point of the grid. The number of states are $n \times n$ and start, frozen, hole, and goal states are exist. The action is to move up, down, left, and right, but the probability of movement to intended direction is just 1/3 due to the frozen surface. So, the probability of going in the intended direction is 1/3 and the probability of going in the perpendicular direction of the intended direction is 1/3 respectively [2]. In this problem, the agent can basically get a reward +1.0 when the agent arrives the goal. However, if there is no penalty to fall in the hall, or meaningless movement, the game is not easy to converge. Therefore, a penalty was introduced this problem to avoid this 'absorbing state' [3]. So, the environment code was modified so that the agent gets -1.0 points in the hall, and it gets -0.01 points in the frozen state.

2) Non-grid problem: Forest management

The Forest management problem is to decide wait and cut the tree in the state [4]. If cut the tree in a state, the agent can get a 1.0 reward. Also, if the agent is in the oldest state and decides wait, the reward is r_1. In this case, the agent decides cut, the reward is r_2 and the r_1 is greater than r_2. In this project, the r_1 is 4 and r_2 is 2.

B. Algorithms

In this project, the value iteration, policy iteration, and Q-learning are used. The value iteration and policy iteration are model-based algorithms and the Q-learning is 'model-free' algorithm.

1) Value iteration (VI)

For each iteration, the value iteration calculates the new 'value function'. It calculates all expected values by visiting all actions and makes an vector consists of the number of actions. And then it gets the maximum value from the vector. So, it can lead the value function to optimal policy without policy computation. Because, it computes all expectations for all actions and states according to the 'state-transition probability function' [3].

$$v_{k+1}(s) = \max_{a} \sum_{s',r} p(s', r|s, \boldsymbol{a})[r + \gamma v_k(s')] \quad (1)$$

$$err(\Delta) = \sum_s |v_{k+1}(s) - v_k(s)| \quad (2)$$

The value iteration is converged when the error (Δ) value ($\sum_s |v_{k+1}(s) - v_k(s)|$) is smaller than a small number (ϵ).

2) Policy iteration (PI)

For each iteration, the policy iteration performs 'policy evaluation' and 'policy improvement'. is similar to the value iteration. The policy evaluation computes the value function by visiting designated action according to the initialized policy rather than visiting all actions.

$$v_{k+1}(s) = \sum_{s',r} p(s', r|s, \boldsymbol{\pi(s)})[r + \gamma v_k(s')] \quad (3)$$

The difference between formula (3) and (1) is the action of the state-transition probability function. For formula (3), the action is determined by the policy π. If the error(Δ) is smaller than the ϵ, the value function(V) passed to the policy improvement. For the policy improvement, the new policy π^* is computed by using

the V. The new policy is compared with the previous policy π.

$$\pi^*(s) = argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] \quad (4)$$

If the new policy π^* is not equal to the previous policy π, the previous policy is replaced with the new policy. When the π^* is equal to the previous policy π, the policy improvement is over.

3) Q-learning

The Q-Learning calculates the 'Q-value' which is 'state-action' value. The Q-value is a value given action and state under a policy [3]. Especially, Q-learning computes and updates the Q-value in every iteration without using the state-transition probability and reward function. Therefore, it is called 'model-free' algorithm.

$$Q(s,a) = Q(s,a) + \alpha \left[R + \gamma \max_a Q(s',a) - Q(s,a)\right] \quad (5)$$

It determines the policy based on the Q-value.

$$\pi^*(s) = argmax_a q_\pi(s,a) \quad (6)$$

C. Parameters

1) Epsilon (ϵ) : For the VI and PI algorithm, the ϵ is compared to the error (Δ) value. If the Δ is smaller than ϵ, the value iteration and policy evaluation is stopped. So ϵ can be an significant parameter to determine the number of iterations and performance. For the Q-learning, ϵ is plays a role as a ϵ-greedy. If the random probability is smaller than ϵ, the action is chosen randomly. Otherwise, the action is chosen as the highest value of the action sets. The effect of ϵ is investigated for the VI and PI in the small-state cases. In the Q-learning, the ϵ is initially set to 1.0 and gradually decayed until 0.001.

2) The discount rate (γ) : the discount rate is the most important parameter in the reinforcement learning. By using the discount rate, the total reward (discounted rewards) is calculated. Normally, the discount rate should be closed to 1.0 to anticipate the future reward [3]. In this project, the effect of γ is investigated and the selection of γ is [0.7, 0.8, 0.9, 0.99, 0.999].

$$G_t(s) = \sum_{k=0}^{\infty} \gamma_k R_{t+k+1} \quad (7)$$

3) The learning rate(α): the learning rate is also significantly important in Q-learning. It determines the update rate and if it is too low, the Q-value is not update properly. In the Q-learning, the effect of α is investigated and the set of α is from Max(1.0), Min(0.1) to Max(1.0), Min(0.5).

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha \left[R + \gamma \max_a Q(s',a)\right] \quad (8)$$

II. Frozen Lake Problem (Small-size Map)

As a grid problem, the experiments for the Frozen Lake problem are performed. In this section, a small-states problem was investigated. As a result of the experiments, the rewards, the number of iterations, and time to train are analyzed.

A. Value iteration (VI)

1) Epsilon(ϵ): as introduced above, the effect of ϵ is studied and the set of ϵ is [10^{-5} ,10^{-7} ,10^{-10}, 10^{-12},10^{-15}].
As described above, the value iteration is converged when the error (Δ) is smaller than ϵ. It can affect the performance of the algorithm.

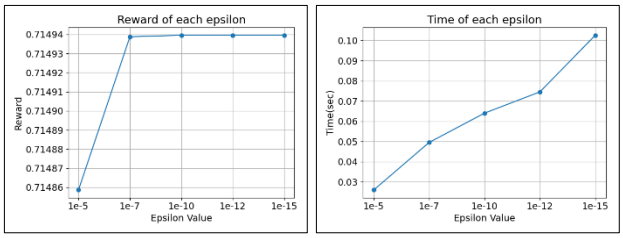

Figure 1. The reward and time of each ϵ

As ϵ value goes decreased, the reward is increased and time is also increased. When the ϵ is 10^{-10}, the reward does not change and it takes less time than the cases of 10^{-12} and 10^{-15}. So, 10^{-10} is selected as a ϵ.

2) The discount rate (γ)

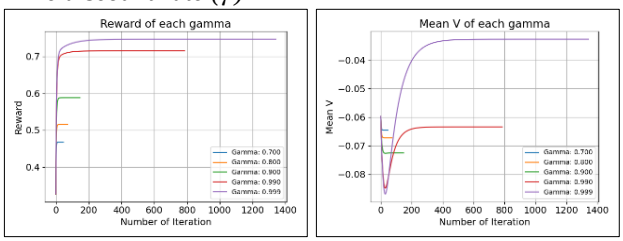

Figure 2. The reward and Mean V of each γ

In Fig 2, the reward and Mean V value converge as the number of iterations are increased. As we can see, when γ is 0.999, the reward and mean V value are the best. So, it is obvious to choose 0.999 as a discount rate.

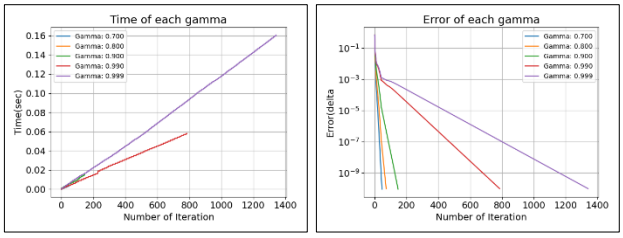

Figure 3. Time to train and error of each γ

When γ is 0.999, the number of iterations is the greatest, so the time to train is the longest. As a result, the error converges slower than the other cases. Because, if the γ is smaller, the computation is less. Also, the value function is less changed when the discount rate is lower.

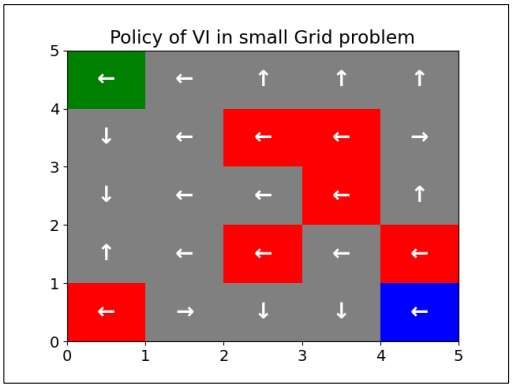

Figure 4. The finalized policy of value iteration (VI)

This figure shows that the final policy of the Value iteration algorithm when γ is 0.999.

B. *Policy iteration (PI)*

1) *Epsilon(ϵ)*

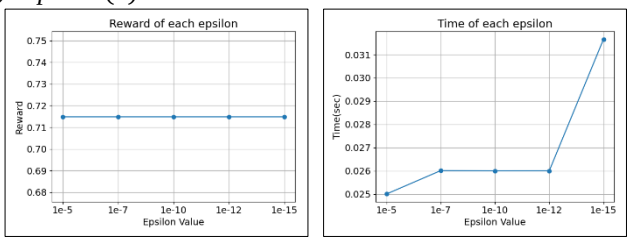

Figure 5. The reward and time of each ϵ

In Figure 5, the reward value is stable even though ϵ value is decreased. Because the policy evaluation is performed prior to the policy improvement, so the reward does not change. In terms of time, when the ϵ is 10^{-10}, the time to train is the shortest. So, the ϵ is selected as 10^{-10} like the previous case.

2) *The discount rate (γ)*

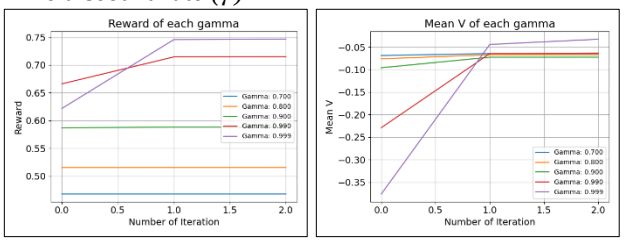

Figure 6. The reward and Mean V of each γ

In Fig6, the reward and mean V plots are stable and flat. The policy evaluation has calculated the value function already. So, there is not significant change in the policy improvement. Similarly, the discount rate is 0.999.

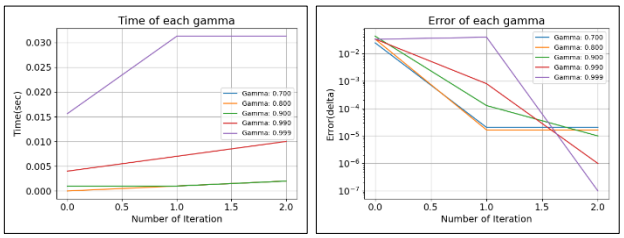

Figure 7. Time to train and error of each γ

When γ is 0.999, the time to train takes longer than other cases. And the error is greater than the VI case. The PI case computes the value function not with all actions, but with actions according to a policy. So, it makes sense that the error(Δ) is greater than the Value iteration case.

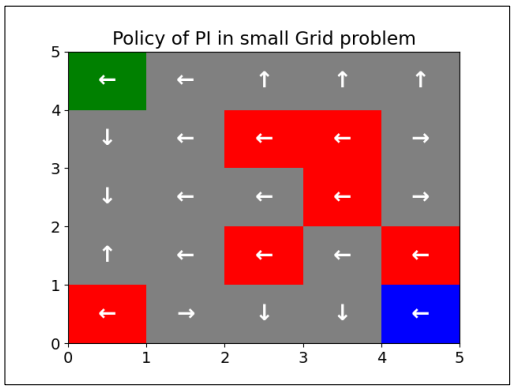

Figure 8. The finalized policy of policy iteration (PI)

Compared to the Fig 4, the policy of the PI is similar to the policy of the VI. The only difference is the (5, 3) value. A possible reason is that the PI case may not consider the situation when the agent goes to the state. It does not compute all actions.

C. *Q-Learning*

As introduced above, the Q-Learning is used for this problem. It calculate the policy with only Q-function updates. Also, the reward (Max V), mean V, time to train, and error were computed according to γ like the previous cases. In this problem, the number of iterations is set to 10,000.

1) *The Learning rate (α)*

Considering the initial steps of the Q-learning, the Q-table is not updated. So, it is important to select α. Additionally, considering the later steps, the learning rate α should be maintained for an update, but no need to be large number. So, in this problem, the initial learning rate is selected 1.0, 0.8, and 0.6. Also, the minimum learning rates are selected from 0.1 to 0.5. The learning rate is gradually decayed according to the iterations. The discount rate γ is 0.999. The following is the results of the learning rates experiments.

Max α	Value	Min α				
		0.1	**0.2**	0.3	0.4	0.5
1.0	Reward	0.8489	**0.9710**	0.9401	0.8893	0.8718
	Mean V	0.0225	**0.0297**	0.0713	0.0085	-0.0757
	Time	1.1611	**2.0706**	0.7348	1.3506	1.107
0.8	Reward	0.4888	0.4831	0.4824	0.3817	0.4599
	Mean V	-0.0307	-0.0383	-0.0734	-0.097	-0.1046
	Time	1.3486	1.4909	0.8443	1.9446	1.1097
0.6	Reward	0.4729	0.5816	0.6120	0.4818	0.6305
	Mean V	-0.0224	-0.0299	-0.0409	-0.1711	-0.1231
	Time	0.9567	2.1748	1.1148	1.7628	1.1001

Table 1. The result of Q-learning when $\gamma = 0.999$

In Table 1, the reward and mean V values are the best when α is 0.2. So, the α is set to 0.2 in this grid problem.

2) The discount rate (γ)

After the is set to 0.2, the effect of the discount rate is calculated like the VI and PI cases.

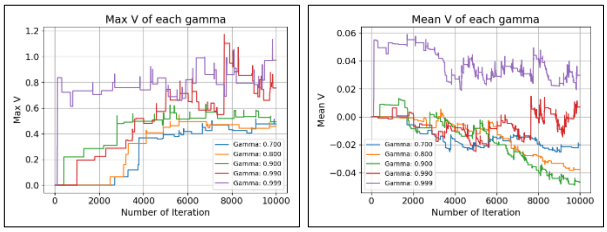

Figure 9. The reward (Max V) and Mean V of each γ

When γ is 0.999, the reward and mean V values are the best. So, γ is also set to 0.999 in this case.

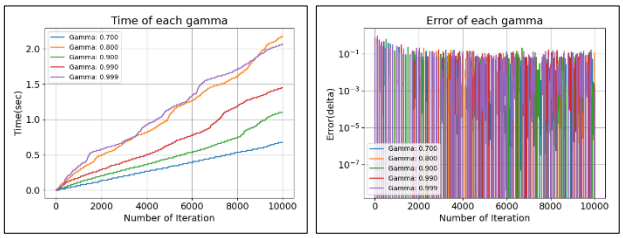

Figure 10. Time to train and error of each γ

In Fig 10, when γ is 0.999, the times to train takes a longer time compared to the other cases. For errors, the error values fluctuate significantly. The error is greater than the previous cases. A possible reason is that the Q-learning does not compute the value function directly. So, the error can be significantly great than the VI/PI algorithms.

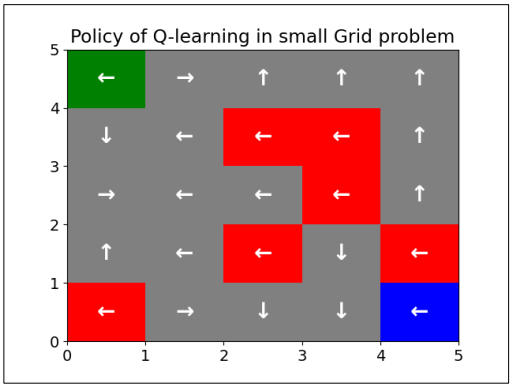

Figure 11. The finalized policy of Q-learning

Fig 11 shows the finalized policy and it looks different from the previous cases. A possible reason is that the Q-learning computes all of states and actions, so the large computation may affect the result.

D. Comparison beween three cases.

When γ is 0.999, the results of three cases are below. The interesting result is that the reward and mean V of the VI and PI algorithm are the exactly same. And The PI takes significantly shorter time than the VI case and the reason is that the PI takes shorter iterations and computes the policy directly. So, it might not need a lot of computations compared to the VI. The Q-learning has the best values among three algorithms, however, it takes a significantly longer time than the VI/PI cases.

Algorithm	Reward	Mean V	Time (Iteration)
Value Iteration	0.7465	-0.0327	0.1606(1,324)
Policy Iteration	0.7465	-0.0327	0.0312(3)
Q-learning	0.9710	0.0297	2.0706(-)

Table 2. The results of three algorithms when $\gamma = 0.999$

III. Frozen Lake Problem (Large-size Map)

In this section, the large-states problem was introduced and it has 400 states. The results of VI, PI, and Q-learning according to γ are compared with the previous small-states case.

A. Value iteration (VI)

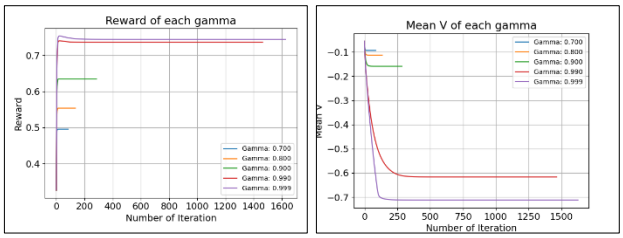

Figure 12. The reward and Mean V of each γ

The mean V values are lower than the previous case. The number of states are 16 times greater than the previous case. As a result, the initial steps might have a large negative value due to the numerous hole points, so the mean V values could be relatively lower value than in the previous case.

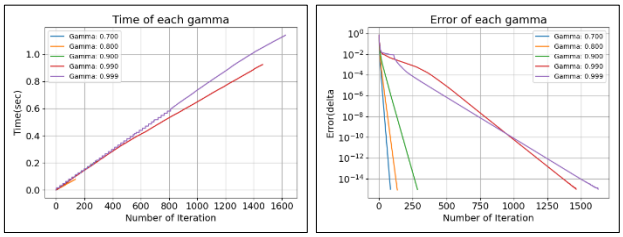

Figure 13. Time to train and error of each γ

When γ is 0.999, it takes 1.139sec and the number of iterations is 1,624. Compared to the previous case, it takes about 7 times more than the previous case. This map has 16 times larger than the previous case, so it can make sense.

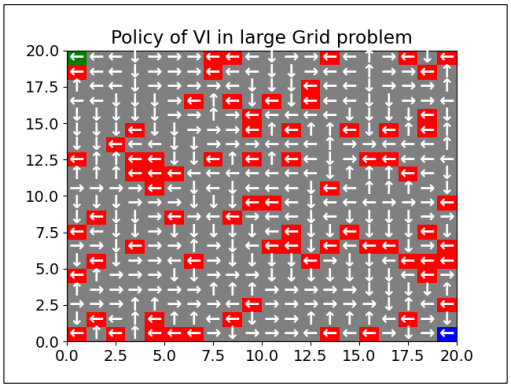

Figure 14. The finalized policy of value iteration (VI)

B. *Policy iteration (PI)*

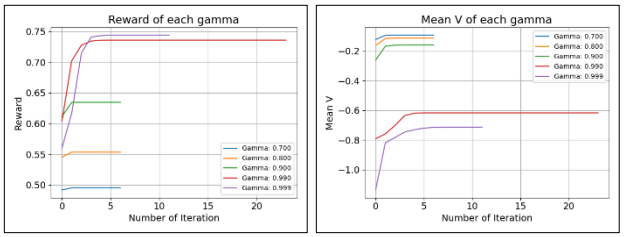

Figure 15. The reward and Mean V of each γ

In Fig 15, the rewards values are similar to the Fig 6. Similarly to the VI case, the mean V values are significantly lower than the small-states case.

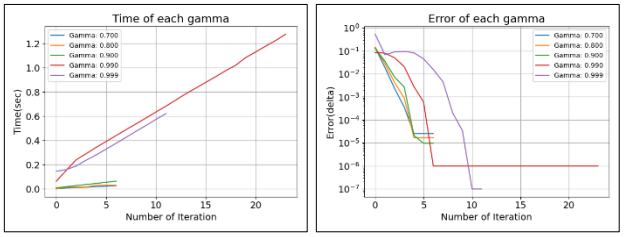

Figure 16. Time to train and error of each γ

The shape of Fig 16 is similar to Fig 7. It takes 0.622 sec and 12 times to converge. It takes about 20 times more than the previous case. A possible reason is that the policy iteration performs not only the policy evaluation, but also the policy improvement. So, it might take more time than expectation.

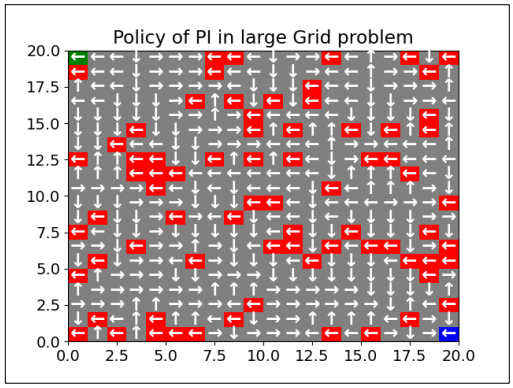

Figure 17. The finalized policy of policy iteration (PI)

Compared to the Fig 14, it show a little different results. However, the reward values are the exactly same as each other. So, two policies from VI and PI can be equivalent.

C. *Q-Learning*

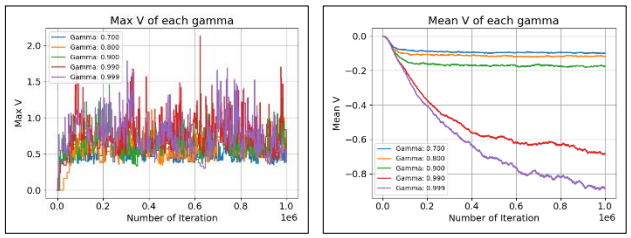

Figure 18. The reward and Mean V of each γ

When γ is 0.999, it has the best result in the small-state problem, however, it does not have the best result in the large-state problem. In this case, the γ has the best value when it is 0.99. A possible reason is that the number of iterations is 10^6 and it cannot be a very big number due to the memory issue. When the number of states is very high, it might need more iterations. And the learning rate is simply compared. So, when the number of states is large, the parameters might be tuned more accurately. Due to these reasons, the result might be not very excellent.

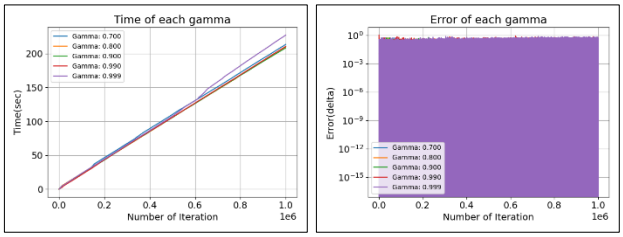

Figure 19. Time to train and error of each γ

When γ is 0.999, the times to train takes a little bit longer than other cases. Also, the error is significantly higher than the small-state case. The more iterations might be needed to obtain better result.

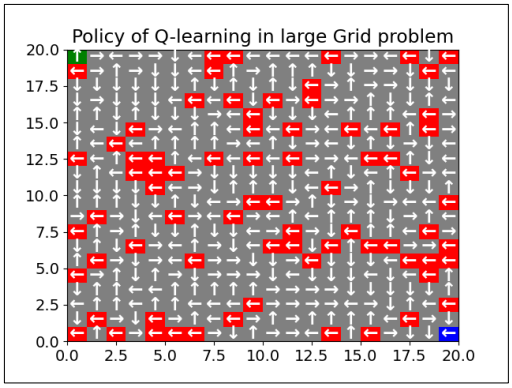

Figure 20. The finalized policy of Q-learning

Compared to the Fig 14 and 17, it show a significantly different results. Also, the reward is not good as the VI/PI case.

D. *Comparison beween three cases.*

When γ is 0.999, the results of three cases are below. In this case, the results from VI/PI are also the exactly same like the previous case. The computation of VI/PI might be very similar by using the value function. So, this similar computation might lead to the same results. Similarly, the PI takes shorter than the VI and Q-learning takes significantly long time.

Algorithm	Reward	Mean V	Time (Iteration)
Value Iteration	0.7438	-0.7125	1.1397(1,624)
Policy Iteration	0.7438	-0.7125	0.6225 (12)
Q-learning	0.6086	-0.6432	215.2657(-)

Table 3. The results of three algorithms when $\gamma = 0.999$

IV. Non-Grid Problem (Small-state)

In this section, the experiments for non-grid problem are performed and the outputs are investigated. The small-state has 25 states and the effects of ϵ and α are also investigated.

A. Value iteration (VI)

1) Epsilon(ϵ)

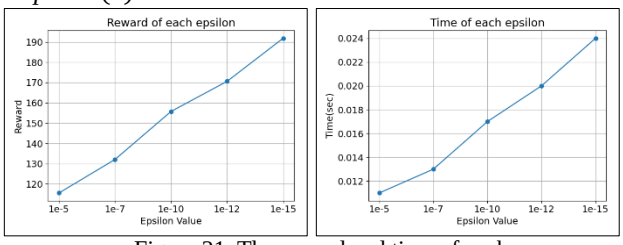

Figure 21. The reward and time of each ϵ

In Fig 21, when ϵ is smaller, the reward value is better. Also, the time is inversely proportional to ϵ. Considering the reward value, ϵ is determined as 10^{-15}.

2) The discount rate (γ)

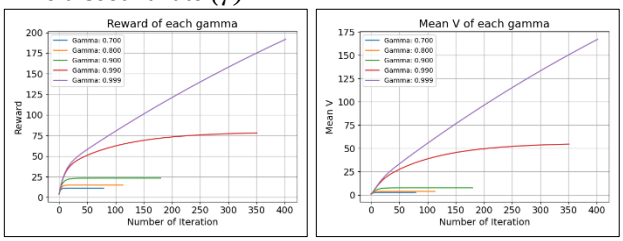

Figure 22. The reward and Mean V of each γ

When γ is 0.999, the reward value is the best and the mean V is also the best.

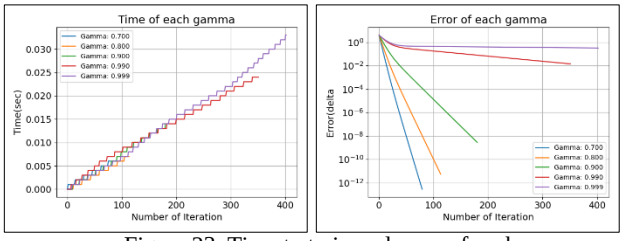

Figure 23. Time to train and error of each γ

When γ is 0.999, the time to train takes a longer than other cases. Even though the error values converge, the values are different.

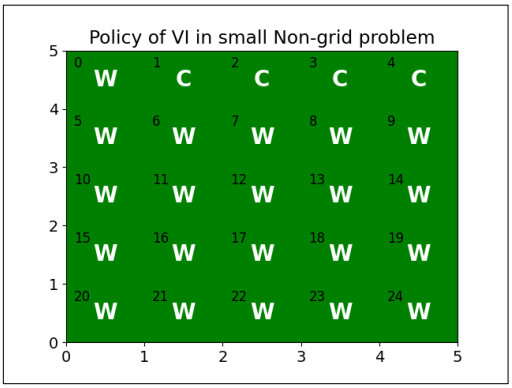

Figure 24. The finalized policy of value iteration (VI)

B. Policy iteration

1) Epsilon(ϵ)

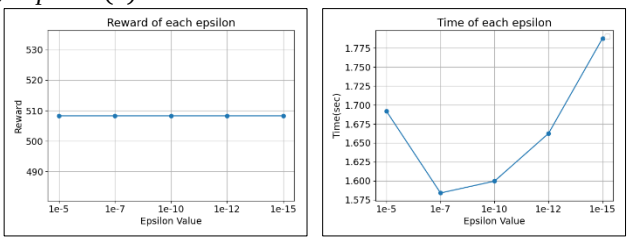

Figure 25. The reward and time of each ϵ

In Fig 25, the reward does not change according to ϵ. And the time is also very similar regardless ϵ. Therefore, ϵ does not affect significantly the performance of PI algorithm.

2) The discount rate (γ)

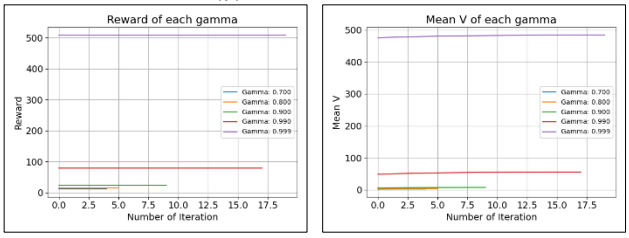

Figure 26. The reward and Mean V of each γ

In terms of the reward, there are significantly large gap between the discount rates. Compared to Fig 22, the reward value is very improved.

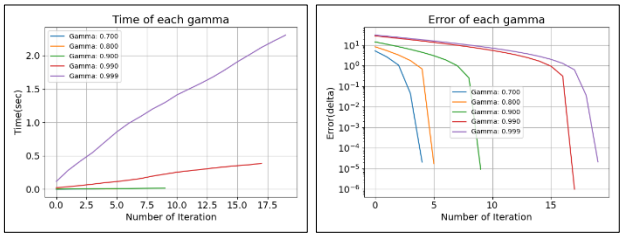

Figure 27. Time to train and error of each γ

As expected, the time to train of γ=0.999 takes the longest time. The error is significantly lower than the VI case and it shows the PI can be more effective algorithm than VI.

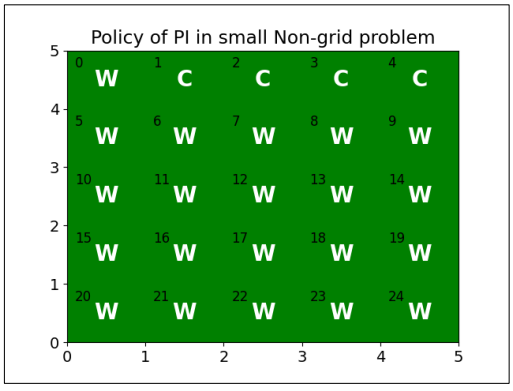

Figure 28. The finalized policy of policy iteration (PI)

The policy is the exactly same as the policy from the VI algorithm.

C. Q-learning

1) The Learning rate (α)

In previous grid problem, the minimum learning rate was 0.2. In this non-grid problem, the results of different minimum learning rates is the following. The number of iterations is also 10,000.

Max α	Value	Min α				
		0.1	0.2	0.3	0.4	**0.5**
1.0	Reward	246.961	332.715	395.754	438.179	**464.728**
	Mean V	88.1737	152.1796	221.7572	287.7	**344.440**
	Time	1.7935	1.1762	1.1768	1.2642	**1.146**
0.8	Reward	231.272	323.726	390.641	435.087	462.220
	Mean V	79.875	146.451	219.477	288.180	345.345
	Time	1.5124	1.506	1.1919	1.2082	1.003
0.6	Reward	213.055	313.823	385.212	431.808	460.345
	Mean V	68.7161	140.839	221.551	298.423	364.043
	Time	1.8921	0.6898	1.6285	1.2967	1.1324

Table 4. The result of Q-learning when $\gamma = 0.999$

Table 4 shows when the maximum learning rate is 1.0 and the minimum learning rate is 0.5, the Q-learning has the best results. In this non-grid problem, the minimum learning rate is relatively higher than the grid problem case.

2) The discount rate (γ)

After the learning rate is set to 0.5, the effect of the discount rates is calculated.

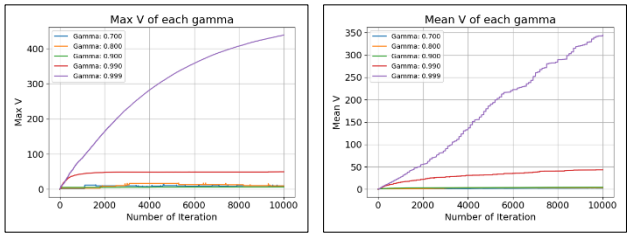

Figure 29. The reward and Mean V of each γ

In Fig 29, when γ is 0.999, the max and mean V has the best value.

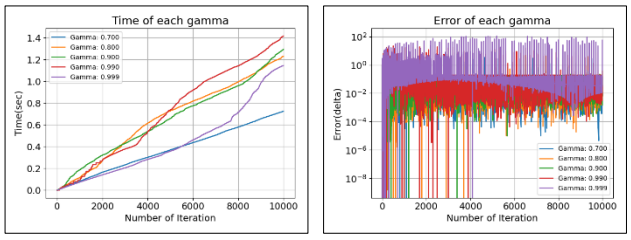

Figure 30. Time to train and error of each γ

Fig 30 shows that there is no significant difference between γ values in terms of time to train. On the other hand, if γ is larger, the error is higher accordingly. And the error is much higher than the VI and PI cases.

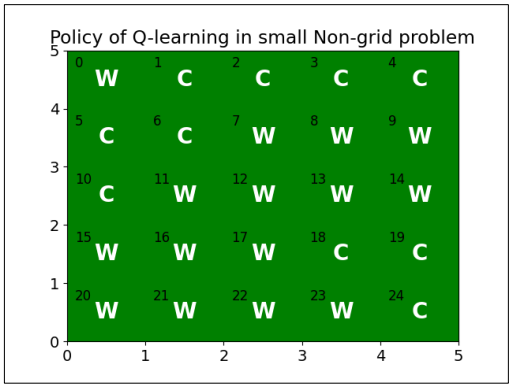

Figure 31. The finalized policy of Q-learning

Fig 31 show that the finalized policy is quite different from the VI and PI cases.

D. Comparison beween three cases.

When γ is 0.999, the results of three cases are below. The interesting result is that the reward and mean V of VI is significantly lower than the other cases. The VI algorithm does not update the policy, so the VI algorithm just calculates the V value of the current state. This might be a possible reason that the VI could have an inferior performance than other cases. In terms of time to train, the PI takes longer than the VI algorithm. In grid problem, the PI algorithm takes less than the VI. A possible reason is that the agent should visit all states in the non-grid problem. So, the agent should compute the policy evaluation and policy improvement. On the other hand, the agent moves to the designated states according to the maximum action in the grid problem. This difference might result in a long time to train than the VI algorithm.

Algorithm	Reward	Mean V	Time (Iteration)
Value Iteration	191.880	167.103	0.0785 (403)
Policy Iteration	508.365	483.587	2.9355(20)
Q-learning	464.728	344.440	1.146

Table 5. The results of three algorithms when $\gamma = 0$

V. Non-Grid Problem (Large-state)

In this section, the experiments for large non-grid problem are performed and the outputs are investigated. Equivalently, the large-state is 400.

A. Value iteration (VI)

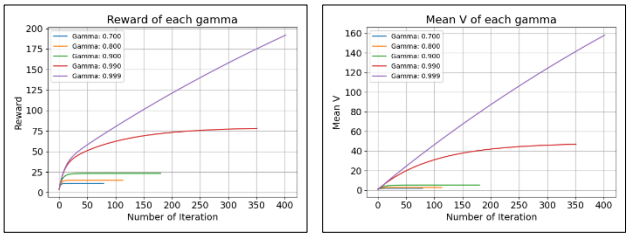

Figure 32. The reward and Mean V of each γ

The reward and mean V values are similar to the small-state case.

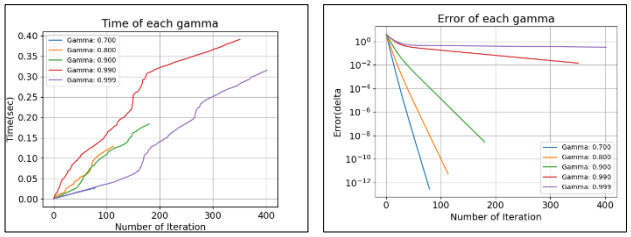

Figure 33. Time to train and error of each γ

It takes 0.316 sec and it is just 4 times more than the small-state case.

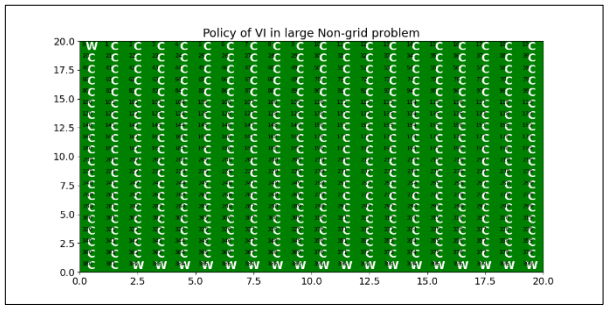

Figure 34. The finalized policy of value iteration (VI)

Fig 34 shows the finalized policy of the VI algorhitm. The interesting thing is there is a consecutive "w" action like the previous small-state case.

B. *Policy iteration (PI)*

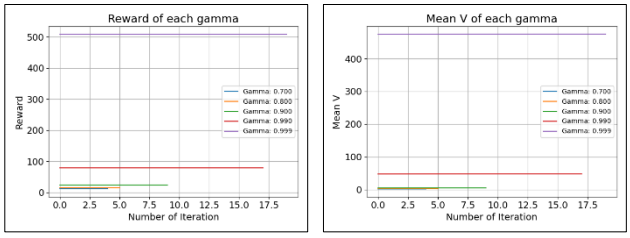

Figure 35. The reward and Mean V of each γ

In terms of the reward and mean V values, the results are pretty similar to the small-state case.

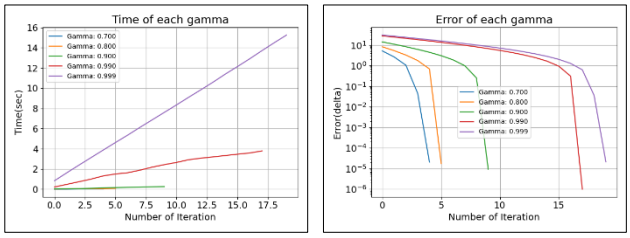

Figure 36. Time to train and error of each γ

In Fig 36, it takes 15.256 sec and it is similar to the VI case. The difference between small-state and large-state is much smaller than the grid problem case. On the other hands, it takes much longer time than the grid problem case. The large-state case of the grid problem takes less than one second, however, this case takes 15 to 16 seconds. As stated above, the agent of PI algorithm should visit all states in the non-grid problem and it might be a primary reason to take much longer time.

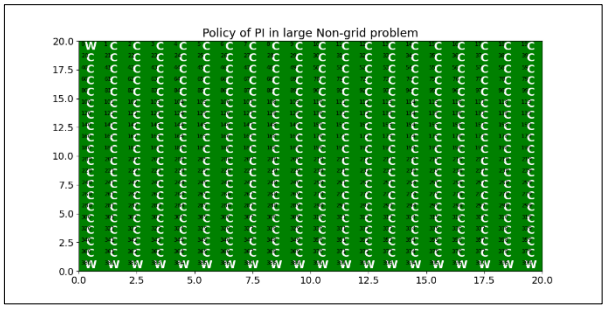

Figure 37. The finalized policy of policy iteration (PI)

Fig 37 is a little bit different from Fig 34. The only two different thing are (1,1) and (1, 2) actions. Also, the number of consecutive 'w' actions is the exactly same as the small-state case. (See Fig 28.)

C. *Q-learning*

Equivalently, the number of iterations is set to 10^6 due to the limitation of the memory issue.

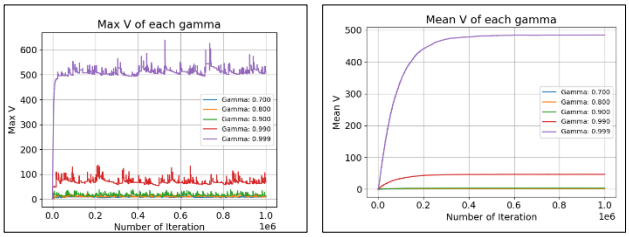

Figure 38. The reward and Mean V of each γ

In this case, when γ is 0.999, the reward and mean V have the best value.

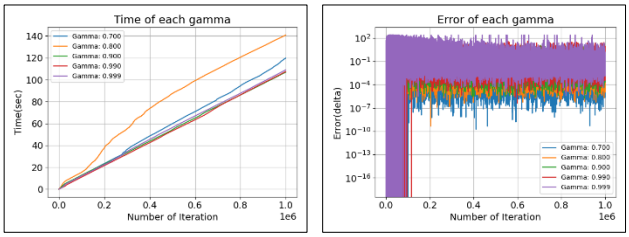

Figure 39. Time to train and error of each γ

Fig 39 shows that the time to train for each γ has not significant difference and the error is much higher than the VI and PI cases.

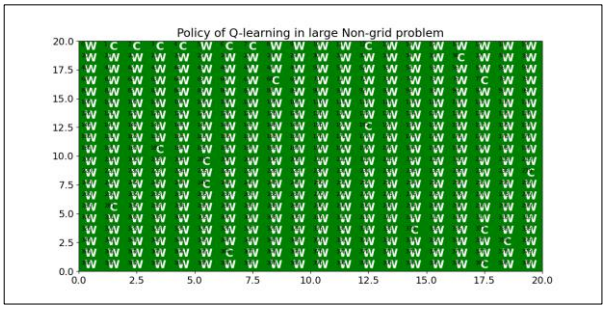

Figure 40. Time to train and error of each γ

The policy of Q-learning is significantly different from the VI and PI cases.

D. *Comparison beween three cases.*

When γ is 0.999, the results of three cases are below. significantly long time. For the VI and PI algorithm, there is no significant changes in reward and mean V values. On the other hand, the performance of Q-learning is improved compared to the small-state case.

Algorithm	Reward	Mean V	Time (Iteration)
Value Iteration	191.881	158.059	0.3156(403)
Policy Iteration	508.365	474.543	15.2556 (20)
Q-learning	511.897	483.707	108.984

Table 6. The results of three algorithms when $\gamma = 0.999$

VI. Conclusion

In this project, two dynamic algorithms and one Q-learning algorithm to solve two MDP problems. The two MDP problems have 25 states and 400 states respectively.

A. *Value iteration (VI)*

The value iteration algorithm can be the most fast algorithm. Even it is slower than the PI algorithm in grid problem, it is much faster than the PI in non-grid problem. Also, it is simple to implement. Because it does not evaluate the policy. However, the value iteration algorithm is not accurate algorithm in the non-grid problem. The performance of the VI is significantly lower than the PI and Q-learning. Therefore, if the MDP problem is grid problem, the VI algorithm could be a good algorithm to calculate policy.

B. *Policy iteration (PI)*

In most cases, the policy iteration has the best reward. Even though the reward of large-state of non-grid problem is slightly lower than that of Q-learning, the reward values are always high. Additionally, in terms of time to train, it is shorter than the VI algorithm in the grid problem. Normally the PI has significantly less steps than the VI algorithm, it results in the shorter time to train.

C. *Q-learning*

Q-learning is the hardest algorithm to train. In grid problem, the Q-learning has the best performance in the small-state. Also, the Q-learning has the best performance in large-states of non-grid problem. However, the performance of large-state of grid problem is not better than the VI/PI algorithms. This might cause that the tuning of parameters (α and ϵ) was not sufficient and the iteration might not be sufficient to obtain the best result. Due to the memory issue of computer, the iterations was difficult greater than a million. Additionally, it takes the most longest time compared to the VI and PI algorithms. When the number of states is higher, it takes a lot of time proportional to the number. Despite this drawback, the Q-learning is the most potential algorithm to improve the performance of given problems.

In summary, this project was very interesting to have an opportunity to analyze three reinforcement algorithms. Even though it was difficult to get improved results, the pros and cons could be investigated. For better performance, the tuning of parameters should be more accurate and more iterations should be performed. For the next time, more developed ways should be used and it will be able to result in better performance.

References

[1] https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[2] https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

[3] R.S.Sutton, A.G.Barto, Reinforcement Learning, An introduction, 2nd Edition, MIT Press, (2020)

[4] https://pymdptoolbox.readthedocs.io/en/latest/api/example.html