Markov Decision Processes

CS7641: Machine Learning

Kim, Irene
skim3364@gatech.edu

I. Objective

This report aims to apply reinforcement learning techniques to an agent so that the agent can make decisions in a given environment. Two Markov decision processes (MDP) are selected to explore three different approaches to reinforcement learning. Three approaches include value-based reinforcement learning, policy-based reinforcement learning, and model-free reinforcement learning. First, value-based reinforcement learning tries to optimize the value function, which computes at each state the maximum expected reward in the future the agent will get. The policy-based approach targets to discover a policy that yields the highest expected utility without using a value function. Finally, model-free reinforcement learning interacts directly with the environment and the agent learns optimal actions in that environment. For each approach, value iteration, policy iteration, and Q-learning are chosen respectively. Each MDPs is tested with three different algorithms and thoroughly analysed. The code is released on GitHub [1].

II. Choice of MDPs

For this report, two Markov decision processes (MDP) are chosen, which are the frozen lake problem and the forest management problem. It is first interesting to see how each technique's performance varies by the number of states. The frozen lake is more simple compared to the forest MDP. The latter MDP has a relatively larger number of states, which is composed of 500 states whereas the former MDP only has 16 states (4x4 grid). The second interesting part is that each problem is based on different types of the world. The frozen lake is a grid world problem. The grid world follows a simple rule that the agent starts and a designated spot and ends at either +1 or -1 which are corresponding rewards. At every step, the agent has possible actions, which are going up, down, left, or right and tries to terminate at the highest reward state. The agent has approximately a 33.3% chance to move in the intended direction, a 33.3% chance to go to the opposite direction to the right and another 33.3% chance to go in the opposite direction to the left. Although simple, it is nice to see intuitively how three approaches to reinforcement learning behave in a stochastic environment. In contrast, the forest problem is a non-grid world problem. The forest management problem does not have a specific goal state but rather seeks to find the optimal sequence of actions that maximizes the reward. The actions are composed of waiting(W) to maintain the old forest or cutting(C) the trees to earn money. However, every action is followed by a probability p that the forest is burnt by wildfire, which initializes everything. This problem is intriguing as it is more directly related to real-world settings where people try to make the best decisions in the stochastic world.

[1]The code is available at GitHub: https://github.com/skirenekim/CS7641-Machine-Learning/tree/main/HW4

III. Frozen Lake

A. Value Iteration

The value iterations start with arbitrary utility values and update utilities based on neighbours until convergence. It is to make the values locally consistent with the Bellman equation. In every iteration, the values are updated explicitly and the policy is updated implicitly by taking the maximum over actions. It is guaranteed to converge. The latest truth becomes more important than the past.

How to Calculate Utility?
The utility of the state sequence is calculated by considering the additive discounted rewards. The discount factor(γ). A discount factor is a number between 0 and 1, which describes the preference of an agent for current rewards over future rewards. If the discount factor is close to 1, the agent is more likely to wait for the long-term rewards. When the discount factor is close to 0, the agent considers the long-term rewards insignificant. Different values are explored here to see how the discount factor affects the result.

How to Define Convergence?
The convergence is reached when the values of utilities are not too different from the next several iterations. In other words, we can define the convergence by setting some threshold(ϵ). If the delta is lower than the predefined threshold, then the algorithm can be said to be converged. Different thresholds are tested for this experiment to analyse the impact of the threshold on overall performance.

Frozen Lake Map for Value Iteration
The grey-shaded area is the initial state. The dark red spots are holes that need to be avoided. The yellow box is the goal state. Other uncoloured areas are frozen lakes, which the agent can freely move. Figure 1 below shows the initial

stage of the environment on the left side. The map on the right presents the map with the optimal policy discovered by the value iteration. The output of the value iteration is the optimal policy. The optimal policy shown on the right-side map has a discount factor of 0.9, the threshold set as 1e-2, and a success rate of 79.5% when tested over 1000 different times. The total time to try 6 different discount factors and 6 different threshold values took 0.18 seconds.

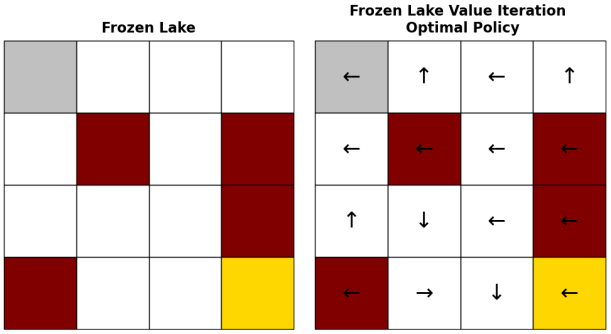

Fig. 1. Frozen Lake Policy Comparison Map

Value Iteration Result Summary

The results are summarized in Figure 2. A total of six different(γ) values were explored. To be specific, average steps, average time, success ratio, and average iterations to reach the goal state depending on different(γ) values are visualized in figure 2. The average steps, execution time, and iteration taken are the largest when gamma is also the largest(0.9). However, the success ratio is highest when gamma is the largest. The overall trend of the value iteration that is observed from the result is that as gamma increases the success ratio also increases but at the cost of taking more steps, time, and iterations. This is because as(γ) gets closer to 1, the agent will put more weight on long-term gains. Therefore, as the discount factor becomes closer to 1, the longer time to converge as the policy will try to optimize the gain over infinite time. The value iteration is already computationally expensive compared to a policy iteration because it computes the utility of the entire state to find the optimal value and finally return a policy. Table 1 summarizes the result of the best gamma value from figure 2.

TABLE I
FROZEN LAKE VALUE ITERATION BEST POLICY DETAILS

Best Gamma	Steps	Success Ratio	Time	Iterations
0.9	42.99	78.37	0.0082	127.7

Value Iteration Convergence

A total of six different thresholds (ϵ) values are compared with the gamma values to see the robustness of the convergence. Figure 3 shows the rewards achieved upon reaching the convergence for each gamma value. Every five ranges on the x-axis illustrate the 6 different(ϵ) tested in sequential order listed in the x-label. The rewards do not increase after passing

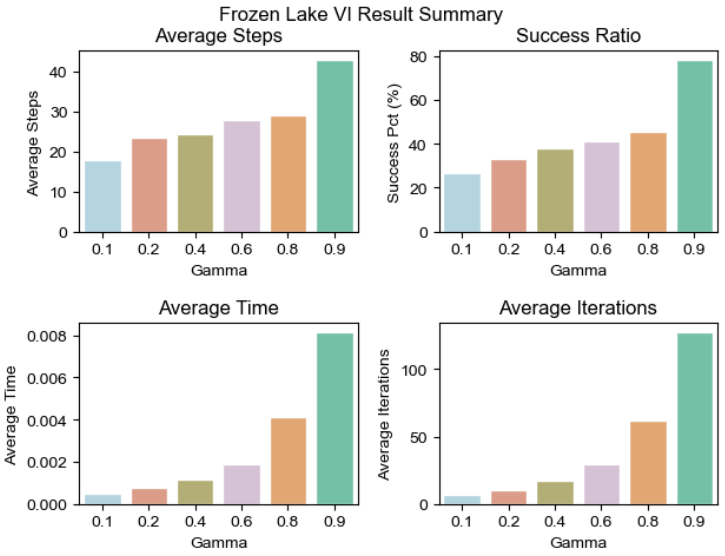

Fig. 2. Frozen Lake Value Iteration Results

1e-6 for the epsilon value even with the smaller values. Getting the same rewards with the smaller thresholds means that the algorithm is passing a more difficult test since the difference or delta is small enough to pass all the strict thresholds. Thus, 1e-6 for all gamma values output optimal results. The threshold lower than 1e-6 still outputs the same results. However, it needs to be higher than epsilon of 1e-2 for all gamma values.

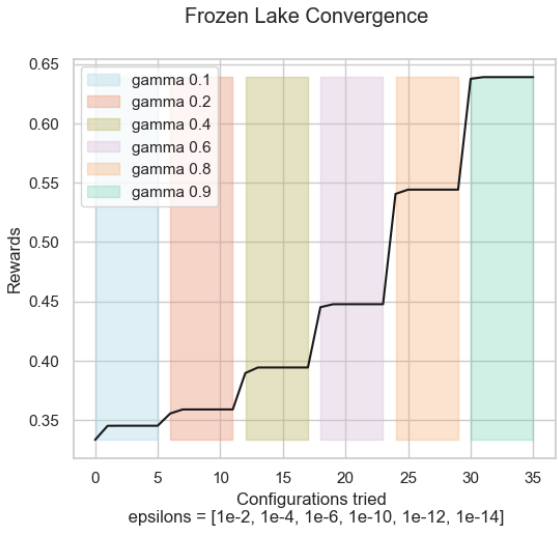

Fig. 3. Frozen Lake Value Iteration Convergence

B. Policy Iteration

Policy iteration is similar to the value iteration except for a max operation. To be more specific, the policy iteration includes iterative policy evaluation and policy improvement instead of finding the optimal value function and extracting

one optimal policy. The new policy will be better than the previous policy and do several passes that update utilities. Each pass will therefore be fast since the algorithm considers only one action per pass.

How to Calculate Utility?

The utility is evaluated on current policy instead of calculating the utility of each state like the value iteration. Besides this difference, the other parts of the utility evaluation are the same as with the value iteration. The discount factor(γ) is also taken into consideration to determine the preference over distant or long-term rewards as discussed in the value iteration section. The values experimented for this report range from 0.1 to 0.9, a total of 9 different discount factors.

How to Define Convergence?

The convergence for the policy iteration can be defined as when the difference between the prior and current iteration falls below a predefined threshold(ϵ). The iteration continues to improve the policy until the policy is no longer changing. In doing so, new policies will always be better than old policies. However, if the policy continually switch or has equally good policies, the iteration may never terminate.

Frozen Lake Map for Policy Iteration

Figure 4 presents the initial stage of the frozen lake problem where the grey area is the starting state and dark red areas are holes where it offsets everything. The gold box is a goal state we want to get to. The left side of the figure is the best policy found by the policy iteration. The policy found by the policy iteration has a gamma value of 0.90 and a success rate of 80.3% when tested 1000 times. The optimal policy extracted by the policy iteration returns the same result as with the value iteration. It took around 0.03 seconds to test 9 different discount factor values, which is roughly 6 times faster when compared to the value iteration. The detailed analysis of the result is continued below subsections.

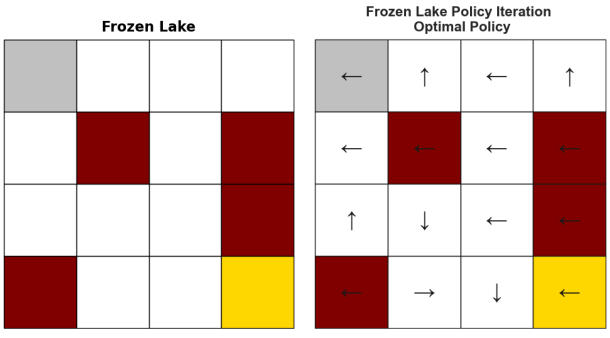

Fig. 4. Frozen Lake Policy Comparison Map

Policy Iteration Result Summary

Figure 5 displays the overall result of the policy iteration evaluated with different discount factors. Similar to the value iteration, the highest discount factor resulted in the highest success probability at the cost of taking more steps and iterations due to its preference for optimizing the policy over a long time. Thus, the agent weighs rewards in the future more heavily. However, the average time to run each test is considerably lower compared to the value iteration. Table 2 is the result of the best value for the discount factor. This table along with figure 5 starkly contrasts with the value iteration in terms of the time and iterations it took to converge. Although the best gamma value was 0.9 for both value and policy iteration, the total time needed to converge is roughly 35 times longer and 21 times more iterations to converge for the value iteration. As mentioned above, the policy iteration benefits from not evaluating all states in every loop, which significantly saves time.

TABLE II
FROZEN LAKE POLICY ITERATION BEST POLICY DETAILS

Best Gamma	Steps	Success Ratio	Time	Iterations
0.9	43	80.3	0.0023	6

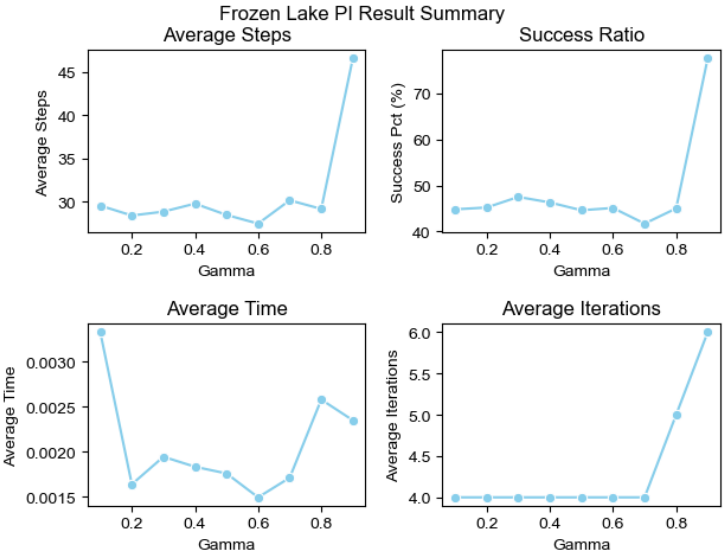

Fig. 5. Frozen Lake Policy Iteration Results

Policy Iteration Convergence

The rewards achieved by different gamma values can be seen in figure 6. As the discount factor gets bigger, the total reward also gets bigger because the agent prefers long-term rewards. Compared to the value iteration, the policy iteration found the same optimal policy that creates the same rewards but in a much shorter time. As described above, the pitfalls are that it might never terminate and keep alternating between policies if there are equally good policies in the worst-case scenario. Nevertheless, the speed of this algorithm makes the policy iteration very computationally efficient, especially when time and resource is constrained. In addition, the never-ending loop of evaluating policies is rare in real-life.

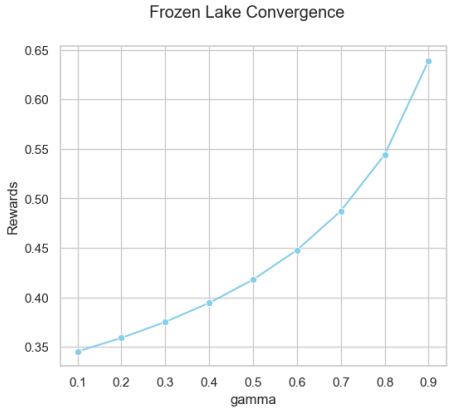

Fig. 6. Frozen Lake Policy Iteration Convergence

C. Q-Learning

In this section, the Q-learning algorithm with ϵ-greedy selection is chosen as the reinforcement learning algorithm. Unlike the other two previous algorithms, the Q-learning algorithm is model-free learning. In model-free learning, the agent interacts with the environment and derives an optimal policy directly instead of explicitly approximating the models of the environment state, transition, and reward. The Q-learning algorithm does not assume that the agent knows about the state, transition and rewards. With only trial and error, the agent is expected to learn an optimal sequence of action. However, there is an exploration-exploitation dilemma lurking in the learning process. The exploration-exploitation dilemma is a fundamental trade-off in reinforcement learning. The agent needs to exploit to use what it learnt but also should be able to explore or learn to be able to exploit profitably. Accomplishing the balance between these two is critical for reinforcement learning problems otherwise the agent is bound to either learn nothing or get caught in local minima.

How to Handle the Exploration-Exploitation Dilemma?
To achieve a nice balance between exploration-exploitation, we can use a method that is similar to simulated annealing optimization. Specifically, alpha and epsilon are utilized. Alpha is the learning rate that determines how much randomness we are going to allow during the search. Epsilon is how greedily we will allow the agent to exploit the current knowledge. The agent will either pick a random action based on alpha rate to explore or it will choose an action based on the current knowledge of Q-values with the probability of 1 ϵ. Each value has a decaying rate similar to simulated annealing that determines the degree of alpha and epsilon.

How to Calculate Utility?
Q-learning is the evaluation of the Bellman equations from data. The below equation shows how optimal policy is evaluated.

$$Q(s,a) = R(s) + \gamma \sum_{s}^{'} T(s,a,s') max_{a}' Q(s',a') \quad (1)$$

Once arriving in s, the agent leaves by taking an action of a, and then proceeding optimally thereafter. Once the agent arrives to s', the agent takes whichever action has the highest Q value from there.

How to Define Convergence?
To reach the convergence in the limit, the Q-learning algorithm must visit all possible state-action pairs. This means that it needs to run for a very long time. Also, the learning rate must gradually approach zero. As this process can take an infinitely long time, the policy is returned once it reaches the number of fixed iterations.

Frozen Lake Map for Q-Learning
Figure 7 displays the optimal policy retrieved by the Q-Learning algorithm. The initial map of the frozen lake is omitted since it can be viewed in the above sections. Compared to the value iteration and the policy iteration, the optimal policy resulting from the Q-learning shows a different pattern in the first and the last row. The best policy from the Q-learning has a discount factor of 0.99, success percentage of 42.2, alpha of 0.01, alpha decay of 0.999, epsilon decay of 0.9, and iteration of 1e+6. These values are specific to the Q-learning algorithm, which is further discussed below.

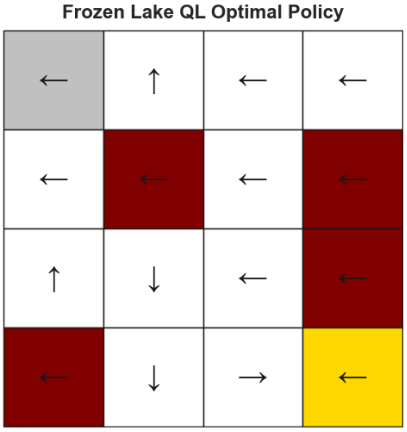

Fig. 7. Frozen Lake Q-Learning Optimal Policy Map

Q-Learning Result Summary
Table 3 records the values used for the policy derived from the Q-learning. The best discount factor is 0.99 with a success percentage of 11 and 1e+6 iterations that took 33.22 seconds to complete. By far the Q-learning returns the worst success ratio with the largest number of iterations that took the most time. This is due to the property of Q-learning that does not assume any prior knowledge of the environment nor reward function. Therefore, it only relies on learning its surroundings and

making the best move as it interacts with the environment. This makes the algorithm intrinsically more difficult to perform better than value and policy iteration in a short time. However, given long enough time and iterations, it is guaranteed to converge to the optimal policy. If the lager number of iterations were allowed, the success ratio would be higher than the current value.

TABLE III
FROZEN LAKE Q-LEARNING BEST POLICY DETAILS

Best Gamma	Steps	Success Ratio	Time	Iterations
0.99	25.6	11	33.22	1000000

Figure 8 depicts how different parameters affect the result. The discount factor used for this experiment were 0.8, 0.9, and 0.99. The learning rates tried were 0.01, 0.1, and 0.2. There decay rates and limit of iteration are provided in table 4. Figure 8 results show that the gamma value itself does not increase the number of iterations since Q-learning iterates until it reaches the predefined number of iterations. The time it needs to iterate more increases drastically when the iteration is set to 1e+6. The plot on the left side of the second row presents that as alpha decay gets closer to 1, the success rate increases. Additionally, the success rate also increases when the algorithm is allowed to iterate more, which can be seen in the right-side plot of the second row. How other factors contribute to each other can be seen compactly in figure 9. The reward and success ratio is heavily affected by time and iteration, which is the obvious result of Q-learning. We can also see that alpha decay also contributes to success percentage. The values used for the Q-learning algorithm(alpha, gamma, and decay values) also show a positive correlation with the reward. Taking advantage of what it learnt(exploit) and learning new things(exploring) may have led to choosing more optimal actions. The takeaway is that securing long iterations affects the most to reach the convergence. At the same time, adjusting the learning rate and epsilon values can prevent the agent from learning things that reinforce sub-optimal or bad actions.

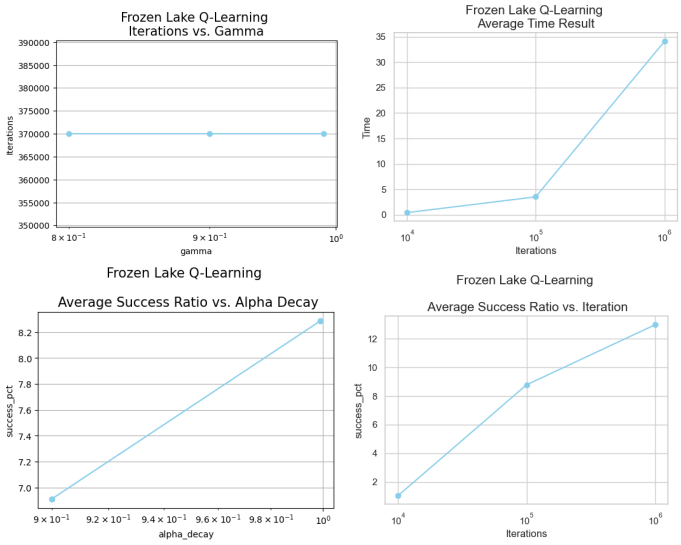

Fig. 8. Frozen Lake Q-Learning Results

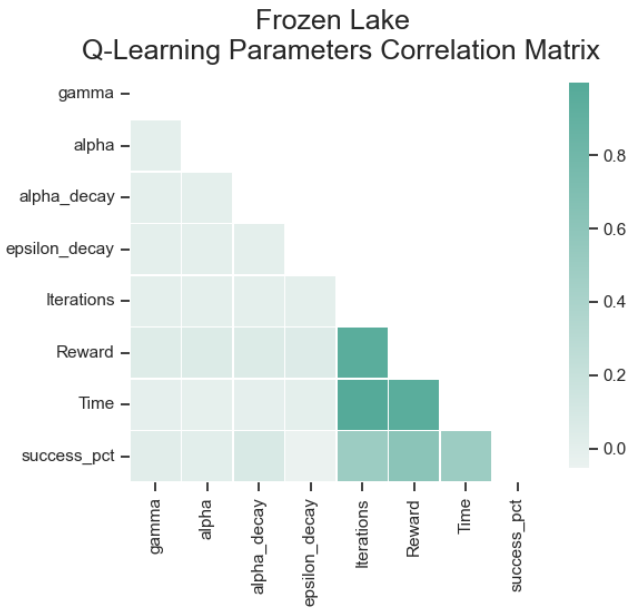

Fig. 9. Frozen Lake Q-Learning Correlation Result

TABLE IV
FROZEN LAKE Q-LEARNING PARAMETERS

Gamma	Epsilon Decay	Alpha	Alpha Decay	Iterations
[0.8, 0.9, 0.99]	[0.9, 0.999]	[0.01, 0.1, 0.2]	[0.9, 0.999]	[1e4, 1e5, 1e6]

Q-Learning Convergence

Running the Q-learning algorithm on the frozen lake problem the convergence was not reached given the limited iteration. The reward received was 0.497, which is much less compared to other algorithms attempted above (see figure 10). Given enough time, the algorithm would learn more optimal decisions in the first row and the last row of the map in figure 7. Increasing the number of iterations to 1e+7 or 1e+8 may produce a policy that is more similar to the ones found by the value iteration and policy iteration.

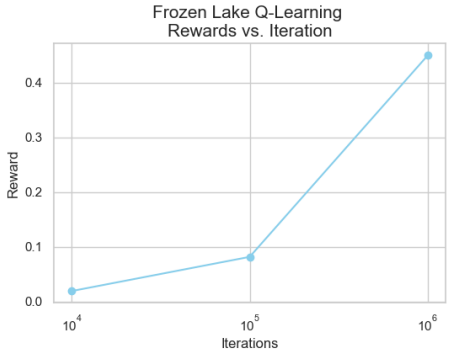

Fig. 10. Frozen Lake Q-Learning Convergence

IV. Forest Management Problem

The forest management problem is a more complicated non-grid world problem consisting of 500 states. The algorithms used for the frozen lake will be tested again to solve the forest problem. The explanation of the forest problem is written in section 2.

A. Value Iteration

The descriptions of how the value iteration works, calculating the utility, and defining convergence are illustrated in the above-frozen lake value iteration section. The same logic applies to solving the forest problem.

Forest Map for Value Iteration

The optimal policy resulting from the value iteration is visualized in figure 11. The green state is when the agent waits and decide not to cut the trees. The rest of states are cutting trees actions. The detail analysis of how the value iteration works, calculating the utility, and defining convergence are illustrated in the above-frozen lake value iteration section. The same logic applies to solving the forest problem. t of boxes coloured in brown are cutting trees. The optimal policy has a reward of 77.81, a gamma discount factor of 0.99, and an epsilon threshold of 1e-14. The total time to finish the value iteration took 1.28 seconds, which took roughly seven more times to complete compared to the frozen lake as we now have approximately 31 times more states than the frozen lake states.

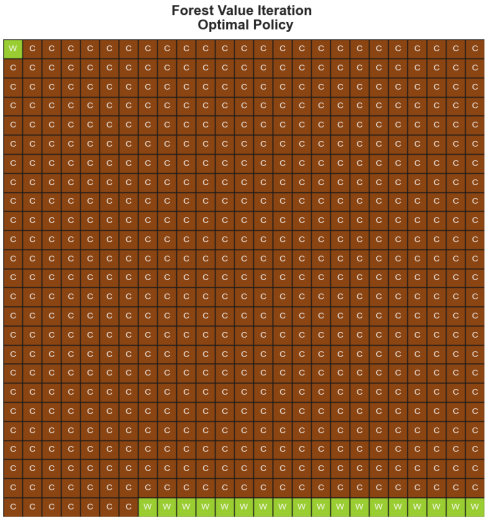

Fig. 11. Forest Value Iteration Optimal Policy Map

Value Iteration Result Summary

Below table 5 presents the average result of the best gamma chosen by the value iteration. The detailed information is also depicted in figure 12, which shows the average results of all the discount factor values. Similar to the frozen lake analysis, the further we push the Gamma close to 1 the better rewards we get. However, it takes more time and iteration to converge. Again, the reason why we get more rewards by increasing the discount factor is because it allows the agent to weigh future rewards more heavily than immediate rewards. Thus, the agent will continue to take more steps to figure out a better but longer solution than taking sub-optimal policy.

TABLE V
Forest Value Iteration Best Policy Details

Best Gamma	Rewards	Time	Iterations
0.99	71.53	0.047	212

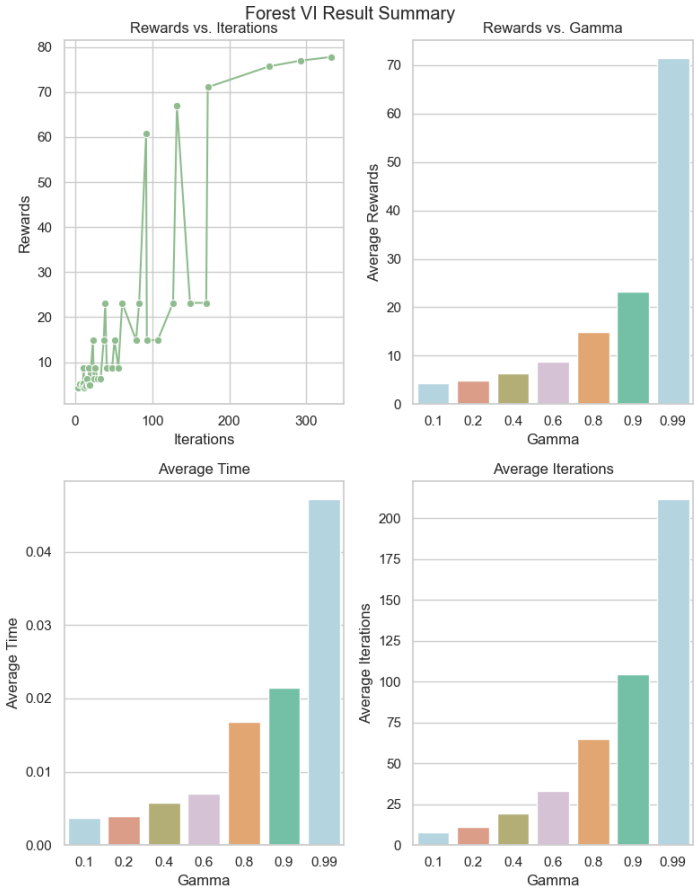

Fig. 12. Forest Value Iteration Results

Value Iteration Convergence

The same threshold(ϵ) values used for the frozen lake problem are used again to see how the algorithm reaches convergence with different gamma values. Figure 13 shows the rewards achieved upon reaching the convergence for each gamma value. Every five ranges on the x-axis show the reward result of the six different(ϵ) tested in sequential order listed in the x-label. The rewards do not increase from the epsilon value of 1e-2. Getting the same rewards with the smaller thresholds means that the algorithm is passing the strict thresholds well. When gamma is 0.99, however, it shows a different trend compared to the other gamma values. As we vary the epsilon values, it achieves higher rewards. Setting the threshold value low and allowing the agent to take actions that would eventually be optimal in the long term helps the agent to find the optimal result.

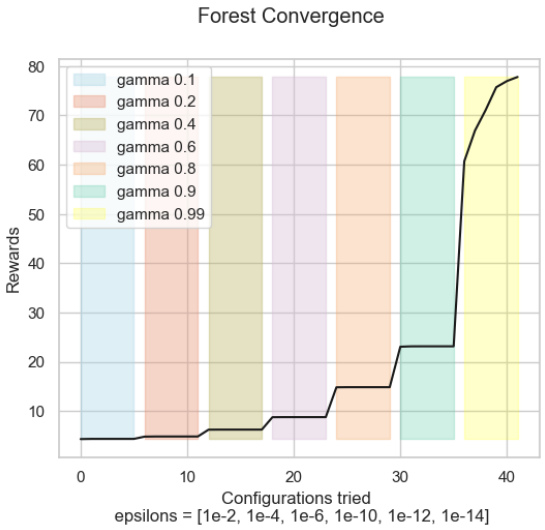

Fig. 13. Forest Value Iteration Convergence

B. *Policy Iteration*

The descriptions of how policy iteration works, calculating the utility, and defining convergence are discussed in detail in the frozen lake policy iteration section above. The same logic applies to solving the forest problem.

Forest Map for Policy Iteration

Figure 14 presents the optimal policy found by the policy iteration. The best policy reached 79.5 for rewards with the gamma of 0.99. It took 0.64 seconds to converge, which is twice the faster than the value iteration. While having the same discount factor as with the policy found by the value iteration, the policy iteration resulted in a higher reward value. The explanation of why the policy iteration is faster to converge is written in detail in the frozen lake policy iteration section.

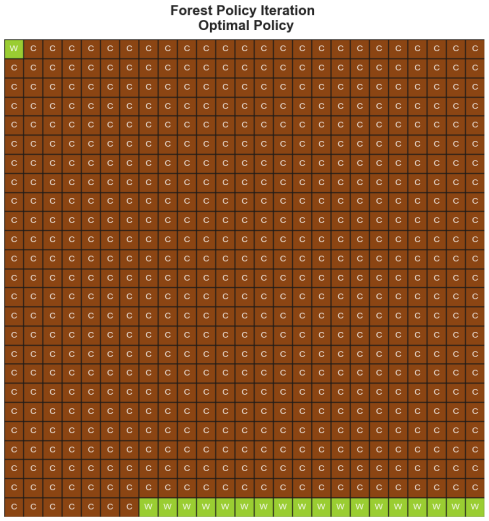

Fig. 14. Forest Policy Optimal Policy Map

Policy Iteration Result Summary

Table 6 is the average result of the best gamma chosen by the policy iteration. Each gamma value explored and its results are shown in figure 15. For the policy iteration, the same rule applies with preceding results. It takes significantly more iteration and time as the discount factor gets close to 1, favouriting future rewards. In doing so, the agent may find optimal actions in the long run. Compared to the value iteration, the amount of iteration needed to converge is drastically lower, but each iteration is more expensive than the value iteration. In addition, the time required to converge is almost half the time it took to converge for the value iteration. As described multiple times in the above sections, the policy iteration benefits from not evaluating all states in every iteration.

TABLE VI
FOREST POLICY ITERATION BEST POLICY DETAILS

Best Gamma	Rewards	Time	Iterations
0.99	79.49	0.0248	18

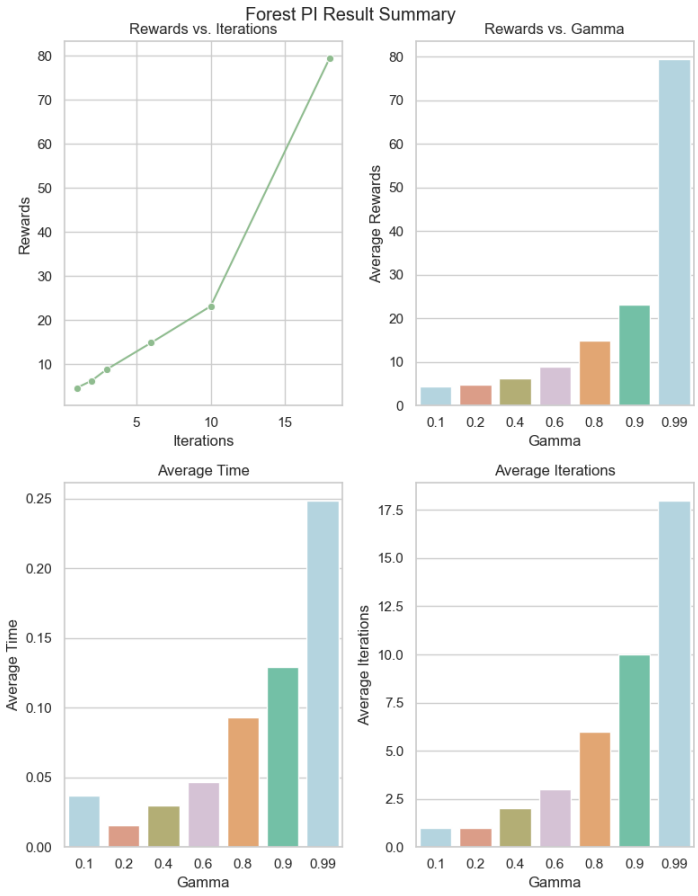

Fig. 15. Forest Policy Iteration Results

Policy Iteration Convergence

Figure 16 illustrates the rewards achieved by different gamma values. For the same reason explained above, rewards get bigger as the gamma value gets bigger. Similar to the frozen lake problem, the policy iteration found the same policy that was found by the value iteration. The value iteration, on the other hand, took much shorter execution time. The policy iteration could end up never terminating when there are equally good policies. Regardless of this rare case, the policy iteration has the benefit of increasing computational efficiency and fast speed.

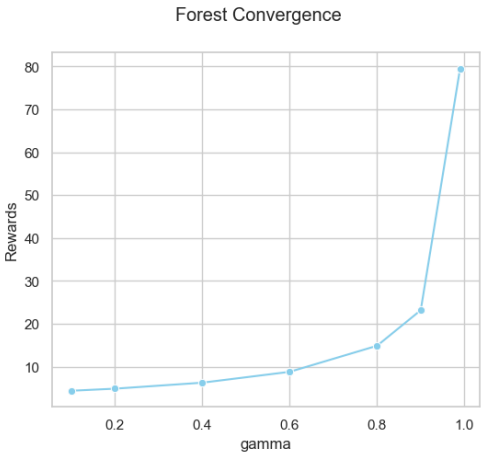

Fig. 16. Forest Policy Iteration Convergence

C. Q-Learning

The explanation of the Q-learning algorithm, its difference compared to value and policy iteration, the importance of exploration-exploitation to derive optimal policy, calculating utility, and defining convergence are well discussed in the frozen lake Q-learning section. The same details and properties apply to solving the forest problem.

Forest Map for Q-Learning

Figure 17 is the optimal policy retrieved by the Q-Learning algorithm. The best policy from the Q-learning has a discount factor of 0.99, alpha of 0.2, alpha decay of 0.999, epsilon decay of 0.9, and iteration of 1e+6. These values are specific to the Q-learning algorithm, which was explained well in the frozen lake Q-learning section above. The forest map looks very different compared to value iteration and policy iteration. This is because the Q-learning algorithm never assumes any prior knowledge of the environment or reward function. The agent can only learn how to behave optimally via trier and error. Therefore, it takes much more time and iteration to find optimal sequence of actions compared to the value iteration and the policy iteration.

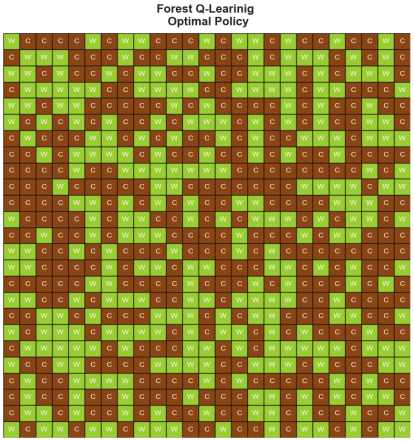

Fig. 17. Forest Q-Learning Optimal Policy Map

Q-Learning Result Summary

Table 7 records the reward, time, and iteration results of using the best policy extracted from the Q-learning algorithm. The average reward result is 47.38, 53.25 seconds to converge and 1,000,000 iterations to complete. By far Q-learning returns the worst reward result and the largest number of iterations that took the most time. The result is similar to the previous simple grid-world problem. The Q-learning reinforcement algorithm only learns its surroundings as it interacts with the environment. Thus, it needs much more time and iteration to converge compared to the model-based learning algorithms. However, the Q-learning algorithm could be a more practical tool in real-life. In reality, it is difficult to model the reward and environment accurately. Furthermore, given sufficient time and iteration, the Q-learning is guaranteed to converge to the optimal policy. Hence, provided a larger number of iterations the rewards would have been higher than what we have now. Figure 18 is the detailed result of different parameters selected for the algorithm. The explanation of each parameter is stated in the frozen lake Q-learning section, thus omitted here. We can observe that the more iteration and the bigger the gamma value is, the better the results get. However, it takes significantly more time. As the alpha decay grows, the reward also gets higher. Figure 19 visualizes the correlation matrix of Q-learning parameters to show how different parameters influence each other. Besides the high correlation result between the gamma, iteration, time, and reward, we can also see that alpha decay has an impact on the reward. In the frozen lake problem, alpha decay had little influence on the success percentage. The results highlight that exploration is an important part of finding optimal actions for Q-learning. The last table below figure 19 is the table of all the parameters tried in the Q-learning algorithm.

TABLE VII
FROZEN LAKE Q-LEARNING BEST POLICY DETAILS

Best Gamma	Reward	Time	Iterations
0.99	47.38	53.25	1000000

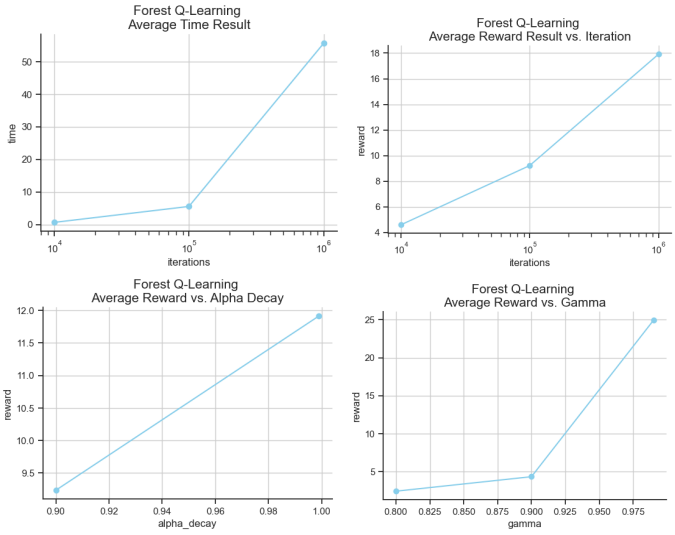

Fig. 18. Forest Q-Learning Results

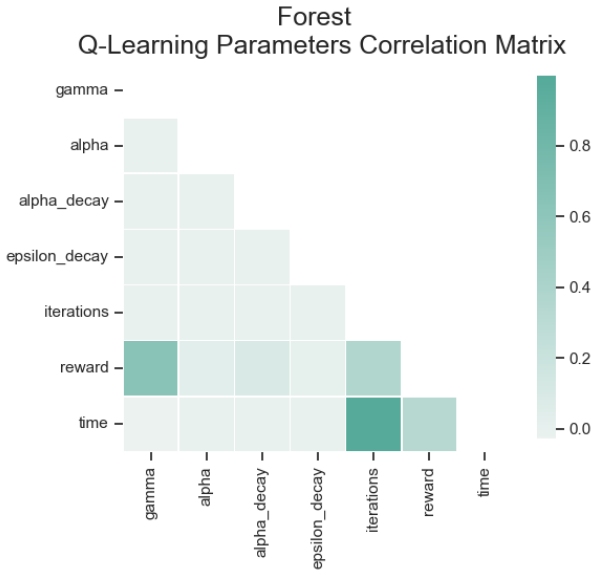

Fig. 19. Forest Q-Learning Correlation Result

Q-Learning Convergence

The Q-learning algorithm for the forest management problem could not find the optimal policy given the restricted number of iterations. Due to time and resource limitations, a larger number of iterations were not attempted in this experiment. However, increasing the number of iterations to a significantly larger value can further improve the Q-learning algorithm.

TABLE VIII
FOREST MANAGEMENT Q-LEARNING PARAMETERS

Gamma	**Epsilon Decay**	**Alpha**	**Alpha Decay**	**Iterations**
[0.8, 0.9, 0.99]	[0.9, 0.999]	[0.01, 0.1, 0.2]	[0.9, 0.999]	[1e4, 1e5, 1e6]

V. CONCLUSION

Reinforcement learning allows tasks that were not previously possible with machine learning algorithms. It enabled the agent to learn and behave optimally without any information or with only a little prior knowledge. This opens a new opportunity that can solve many real-world issues ranging from game-playing to maintaining the wild forest. In this report, model-based algorithms, which are value iteration and policy iteration experimented on two different problems. Model-free Q-learning algorithm was tested on those problems as well. Although the Q-learning algorithm continuously showed the worst result compared to the other two algorithms, I believe it has more potential in real-life as it does not require any information about the world. It only reinforces its learning by experience, which is similar to how humans behave.