CS7641 – Machine Learning

# Assignment 4

Jingyao Zhu

jzhu398@gatech.edu

## 1 Overview

In this report, three different algorithms such as Value Iteration, Policy Iteration, and Q-Learning are explored to solve two Markov Decision Processes (MDPs) including the Frozen Lake problem and Fire Management problem. Large size and small size will be applied. Markov Decision Processes is a mathematical framework to build a model for decision-making. It is useful for solving dynamic optimization problems because current rewards will be calculated in each state, future actions will be taken base on the current state, and future rewards will be maximum. In the following two problems, I picked different percentage of future rewards will be considered in current state. In other words, different discount factors will be applied to build model using three algorithms.

### I. Frozen Lake

In the grid world, the Frozen Lake problem is kind of famous because there may be only one optimized way to find the goal state. A Frozen Lake is quite similar to a maze problem that it is a path, typically from an entrance to a goal. An agent must navigate a maze to find the right paths. At the same time, the length of an entrance and a goal must be the shortest path in order to ensure the negative rewards of MDPs are small. In the real process, each tile will leave a small negative reward to make sure that the agent will be encouraged to find the right way to the goal state. An agent can move up, down, right, or left in the Frozen Lake problem. If there is totally random walk without any reward, each of them has a 25% chance to be acted. But in my problem, I set the 80% to transition functionto maximize reward, which means that an agent will act in an undesired way with a 20% chance. 4 * 4 is the small grid world with 16 states, and 8 * 8 is the large one with 64 states. I think the Frozen Lake problem is interesting because the problem is not a competitive game and is easy to understand without knowing the problem detail too much, which will make me learn MDPs easily. Reinforcement learning is proven to apply to that kind of game.

### II. Fire Management

The second problem that is going to be solved by MDPs is the fire management problem. It is a non-grid world problem and it is more close to a real-world problem. The problem is still going to be looking at two dataset sizes: small and big. The goal of this problem is to make the decision whether forest patches should be burned or not to allocate better resources. This problem is quite harder than Frozen Lake but it is interesting since it only takes a few inputs and to find the right decision itself. As I mentioned, this problem is quite close to the real-world problem. If the analysis process is proven here, many other real-world examples could be solved evenly.

## 2 Maze Problem

### 1. Overview

Figure 1 shows the 4*4 frozen lake grid world and figure 2 shows the 8*8 frozen lake grid world. The purple square with the letter G is the destination, which is the goal state. The light blue square with the letter S is the starting point. The dark blue with the letter H represents a wall, which cannot be walked through directly. The goal is to find the not only correct but also optimal action in each state so that the agent could find the best path to the destination. In this problem, the problem will be addressed by value iteration, policy iteration, and Q learning. The general rule is that the reward for all the steps before the destination is -1, while the immediate reward on the destination is 100. The discount factor (gamma), which is a total reward for each state including a percentage of future rewards, is going to be researched Learning rate and epsilon are the key parameters for Q learning.

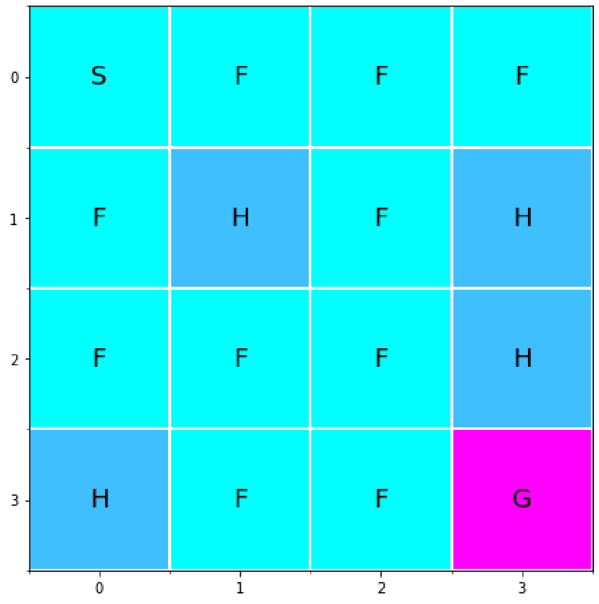


Figure 1: 4*4 grid world frozen lake

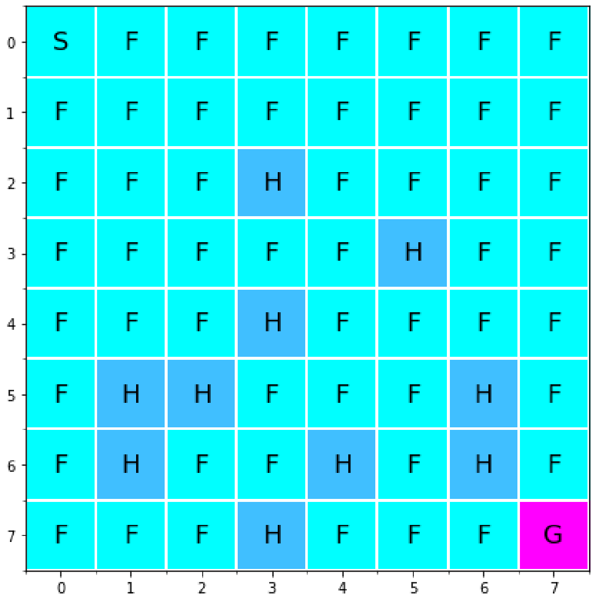


Figure 2: 8*8 grid world frozen lake

### 2. Tunning gamma and espilon

Overall, The parameter gamma will be tuned to find the optimal path for all three algorithms and epsilon will be tuned for Q learning in order to help get the best q learning model for large and small grid world. Q learning does not require a model of the environment. Thus, it is a model-free reinforcement learning algorithm. The algorithm is good because it can learn the value of an action in a particular state with stochastic transitions and rewards without requiring adaptations.

In Figure 3 to 5, graphs show how different gamma, which is from 0.5 to 1, affect optimal policy in different algorithms with small grid world. With lower gamma value, algorithms converge very quickly. At the same time, it loses too much value. With this tradeoff, the graphs show that gamma from 0.9 to 0.999 have the better movements (value closes to 0.5 after steady) because of better mean values even though max values and total rewards remain the same. With 4 or 5 iterations, mean values and errors of all gamma values are going to reach the steady state. Since iterations 4 (0.27s) and 5 (0.29s) take very close running time to finish, gamma 0.96 with 5 iterations are going to be the best option to all three algorithms. In figure 6 to 8, graphs show how different gamma, which is from 0.5 to 1, affect optimal policy in different algorithms with big grid world. In big grid world, gamma performs differently, which is from 0.94 to 0.999 have the better movements. And iteration 10 and 13 are going to have the best result for three algorithms based on values, errors, and running time. Since running time for 10 (0.41s) and 13 (0.46s) is very close, gamma 0.98 with 13 iterations are going to be the best option to all three algorithms. Thus, the small grid world needs a little bit lower gamma value while the big grid world needs a quite bit large gamma value. And the small grid world needs a few iterations while the big grid world needs a quite bit many interations. More iterations mean more time to run algorithms.

For Q learning, there is a parameter, which is esp, to set. With the 0.96 gamma value in small grid world, different esp values such as 0.3, 0.6, and 0.9 are going to applied in Q learning. In Figure 9 and 10, graphs show how many steps to converge and how much time is needed. It is very clear to show that lower esp value is easy to converge and becomes steady. Considered time to finish, lower esp is also show its advantage. Overall, 0.3 esp performs the best in small grid world. Based on the generated rule, esp value for Q learning should be setted as a low value. With the 0.98 gamma value in big grid world, different esp values such as 0.3, 0.6, and 0.9 are going to applied in Q learning. In Figure 11 and 12, graphs show how many steps to converge and how much time is needed. It is very clear to show that lower esp value is easy to converge and becomes steady. But the different is not as big as the one in small grid world. Considered time to finish, lower esp is also show its advantage. 0.3 and 0.6 are perform so close this time. Overall, 0.6 esp performs the best in big grid world since it is fast to converge in a few iteration. Thus, big grid world should pick a little bit higher esp value compared with eps in small world and it follow the rule from gamma value, which is that bigger grid world is larger value of parameter should be taken.

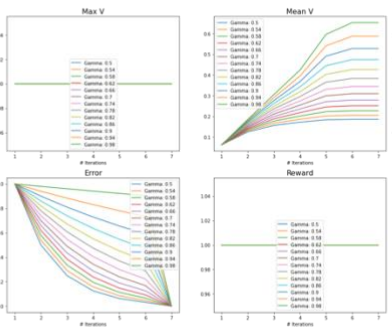


Figure 3: gamma in value iteration(small)

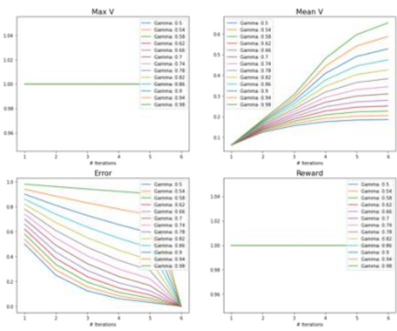


Figure 4: gamma in policy iteration(small)

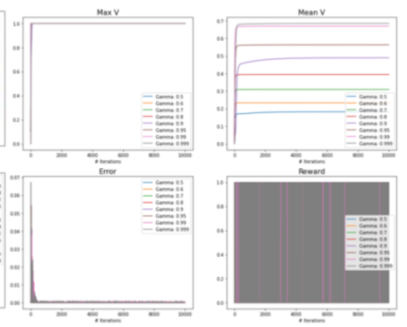


Figure 5: gamma in Q learning(small)

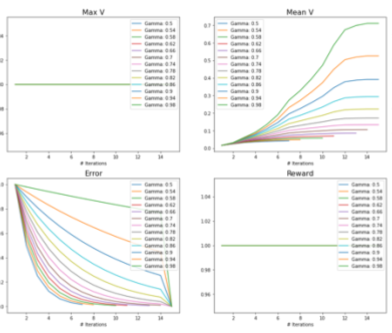


Figure 6: gamma in value iteration(big)

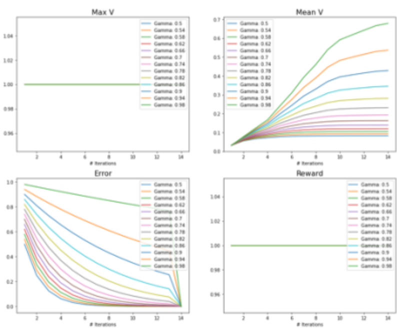


Figure 7: gamma in policy iteration(big)

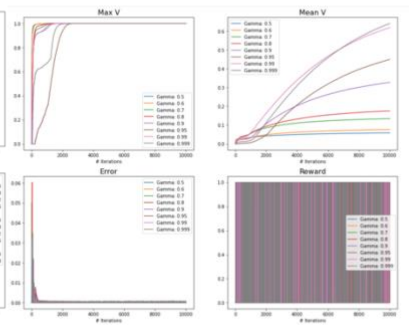


Figure 8: gamma in Q learning(big)

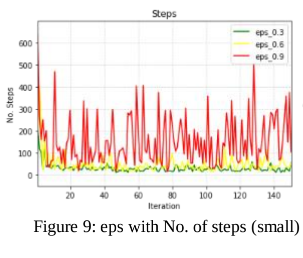


Figure 9: eps with No. of steps (small)

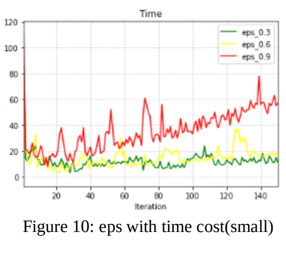


Figure 10: eps with time cost(small)

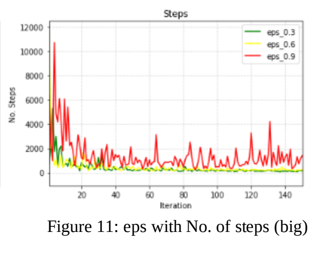


Figure 11: eps with No. of steps (big)

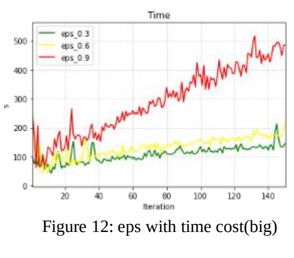


Figure 12: eps with time cost(big)

## 3. Algorithms comparison

There are three algorithms that are going to compare here. In Figure 13 to 15, value iteration, the optimal pathes in small grid world for policy iteration, and Q learning are showing below. In Figure 19 to 21, value iteration, the optimal pathes in big grid world for policy iteration, and Q learning are also showing below. Comparing three different algorithms in small grid world, value iteration and policy iteration are taking exact same action and Q learning algorithm gives different path and different steps in some cells. But overall, all three will converge. In Figure 16 and 17, Q learning has the less running time with 5 iteration to converage. The value iteration takes longest time with 5 iteration to converage. In Figure 18, Q learning is the least steady rewards algorithm compared with value iteration and policy iteration. But they all will converge within 5 iteration with very high value.

In large grid world, not only value iteration and policy iteration provide different optimal path to reach the goal state but also neither of them are right because they did not walk around the wall with a fewer iteration. But Q learning performs the best since it provide the right and optimal path to reach the goal. Figures 19 to 21 could show. In Figure 22, value iteration and policy iteration are very hard to converage since the graph shows that they are up and down (variance high). It is perfectly explain why those two algorithm could not provide the right path. Q learning is still the best here since it eventually took small steps even though some big steps are taken at the beginning. In figure 23, Q learning is still the best since it took less running time compared with other two algorithms. The value iteration performs the worst here. Comparing rewards, Q learning only took 13 iteration to reach a high reward and steady state (low variance). Considering all three comparison aspects, Q learning is the best algorithm with highest rewards and 13 iterations.

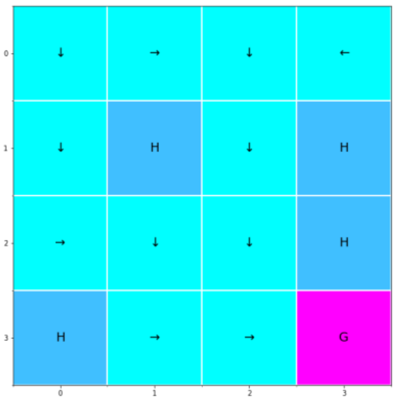


Figure 13: optimal value iteration (small)

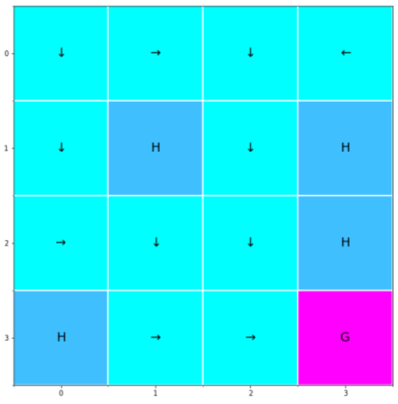


Figure 14: optimal policy iteration (small)

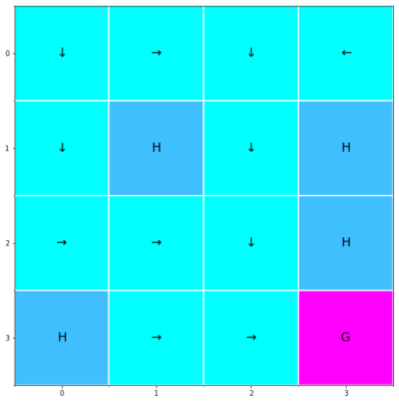


Figure 15: optimal Q learning (small)

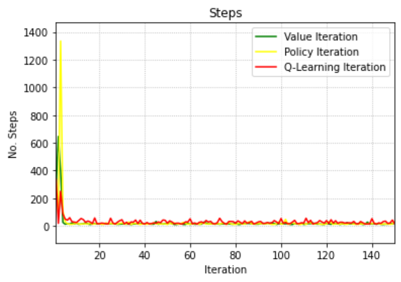


Figure 16: steps to goal (small)

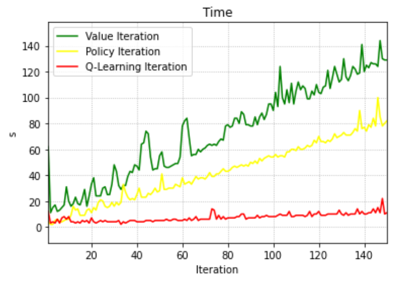


Figure 17: Running time (small)

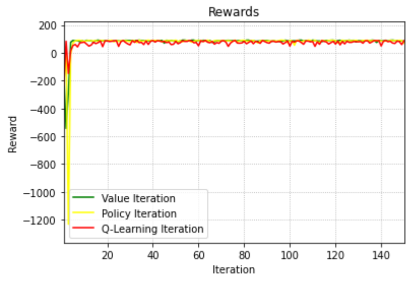


Figure 18: Rewards (small)

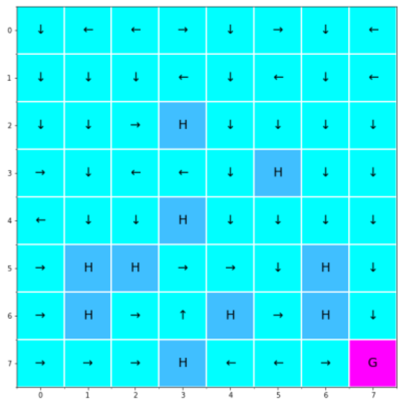


Figure 19: optimal value iteration (big)

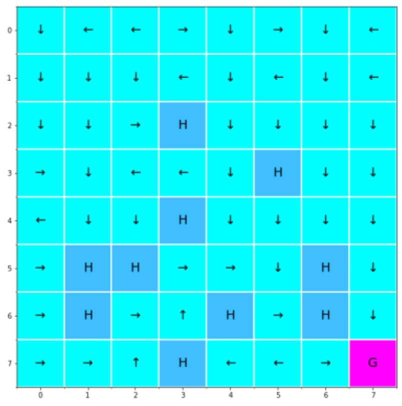


Figure 20: optimal policy iteration (big)

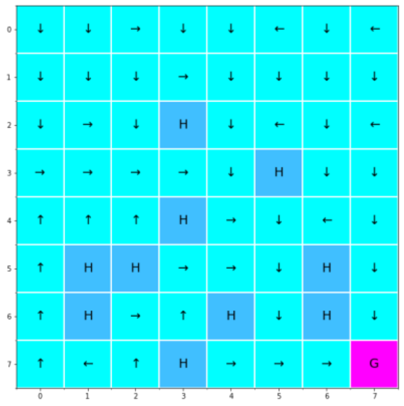


Figure 21: optimal Q learning (big)

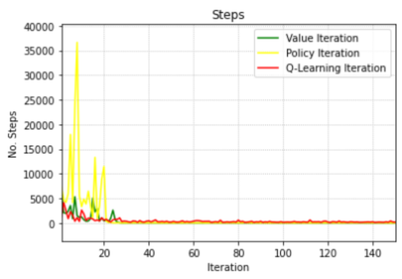


Figure 22: steps to goal (big)

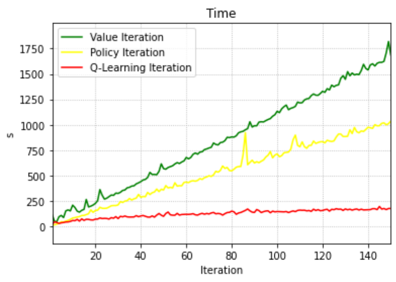


Figure 23: Running time (big)

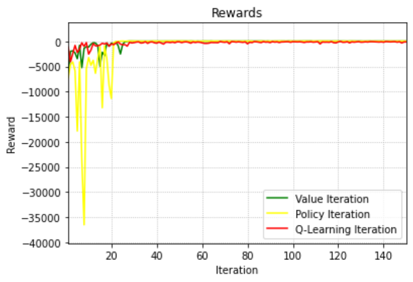


Figure 24: Rewards (big)

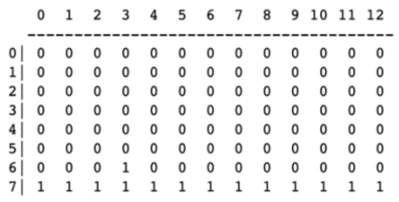
## 3 Fire Management

### i. Overview

I picked fire management problem is because it is a dynamic optimal resource allocation. The goal of this problem is to allcate resource by burning the forest. In small non-grid world, there is 104 states in total with 8 differet-level population and 13 different-level fires. In large non-grid world, there is 900 states in total with 30 differet-level population and 30 different-level fires. 0 in population means extinct and 1 at least in fire when there is a fire. In this problem, learning rate, eps, gamma will keep the same (learned from pervious problem) in order to compare.

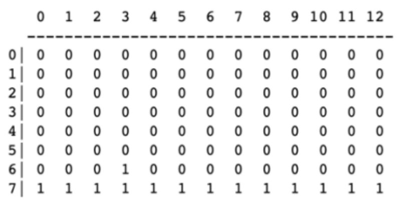
### ii. Small fire management

With fixed learning rate, eps, and gamma, three different algorithms performs differently. In figure 24 to 26, value iteration and policy iteration share the same optimal solution, which is to burn the forest when population hits 6 and it has been 3 years since last fire. But it is not the only condition. When population hits 7, the forest should be burned starting from the first year. In other word, when population hits 5 it is the margin for this small non-grid world. The Q learning shares a total different idea, which is to burn the forest when population hits 5 with 9 years later.



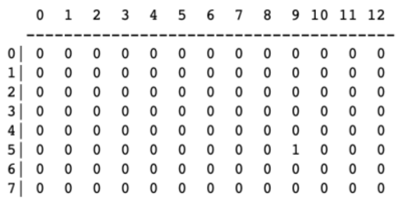
```
    0  1  2  3  4  5  6  7  8  9 10 11 12
  ---------------------------------------
0|  0  0  0  0  0  0  0  0  0  0  0  0  0
1|  0  0  0  0  0  0  0  0  0  0  0  0  0
2|  0  0  0  0  0  0  0  0  0  0  0  0  0
3|  0  0  0  0  0  0  0  0  0  0  0  0  0
4|  0  0  0  0  0  0  0  0  0  0  0  0  0
5|  0  0  0  0  0  0  0  0  0  0  0  0  0
6|  0  0  0  1  0  0  0  0  0  0  0  0  0
7|  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Figure 24: optimal value iteration (small)



```
    0  1  2  3  4  5  6  7  8  9 10 11 12
  ---------------------------------------
0|  0  0  0  0  0  0  0  0  0  0  0  0  0
1|  0  0  0  0  0  0  0  0  0  0  0  0  0
2|  0  0  0  0  0  0  0  0  0  0  0  0  0
3|  0  0  0  0  0  0  0  0  0  0  0  0  0
4|  0  0  0  0  0  0  0  0  0  0  0  0  0
5|  0  0  0  0  0  0  0  0  0  0  0  0  0
6|  0  0  0  1  0  0  0  0  0  0  0  0  0
7|  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Figure 25: optimal policy iteration (small)

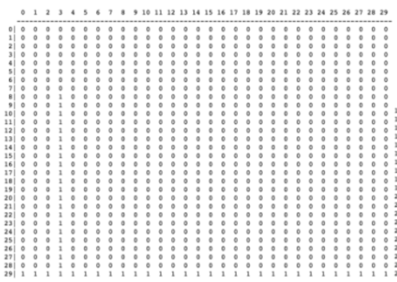


```
    0  1  2  3  4  5  6  7  8  9 10 11 12
  ---------------------------------------
0|  0  0  0  0  0  0  0  0  0  0  0  0  0
1|  0  0  0  0  0  0  0  0  0  0  0  0  0
2|  0  0  0  0  0  0  0  0  0  0  0  0  0
3|  0  0  0  0  0  0  0  0  0  0  0  0  0
4|  0  0  0  0  0  0  0  0  0  0  0  0  0
5|  0  0  0  0  0  0  0  0  0  1  0  0  0
6|  0  0  0  0  0  0  0  0  0  0  0  0  0
7|  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 26: optimal Q learning (small)

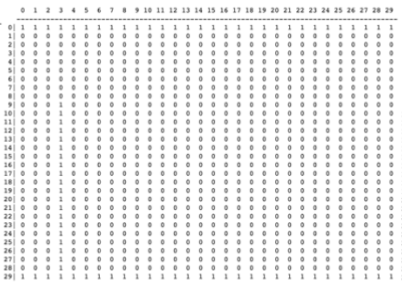
### iii. Large fire management

In figure 27 to 29, value iteration suggests that burn down forest when population hits 9 and it has been 3 years later than pervious fire. the burden line is population 8. The policy iteration share another solution, which is to burn the forest with 0 population from the beginning. But it is not a reasonable solution. The Q learning shares a total different idea, which shows that there are many ways to do it. For example, burn the forest when population hits 8 with 12 years later.



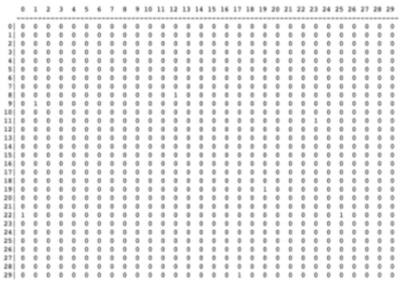
```
     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
   ------------------------------------------------------------------------------------------
 0|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 4|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 5|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 6|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 7|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 8|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 9|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
17|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
18|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
19|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
20|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
21|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
22|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
23|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
24|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
26|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
27|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
28|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
29|  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Figure 27: optimal value iteration (big)



```
     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
   ------------------------------------------------------------------------------------------
 0|  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 1|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 4|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 5|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 6|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 7|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 8|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 9|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
17|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
18|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
19|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
20|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
21|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
22|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
23|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
24|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
26|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
27|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
28|  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
29|  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Figure 28: optimal policy iteration (big)



```
     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
   ------------------------------------------------------------------------------------------
 0|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 4|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 5|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 6|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 7|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 8|  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 9|  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
12|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
17|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
18|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
19|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
20|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
21|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
22|  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
23|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
24|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
26|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
27|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
28|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
29|  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 29: optimal Q learning (big)

### iv. Comparison of algorithm

In this part, the comparison of three algorithm will focus on rewards and iterations in small and large fire management problems. Figures 30 to 32 show the relationship between reward and iteration for small states and Figures 33 to 35 show the relationship in the large one. Value iteration algorithm converges in small and large problem. In small problem, value iteration takes around 43 iterations to be steady and converge and it takes around 50 iterations in the large problem. Policy iteration algorithm also converges in small and large problem. In small problem, value iteration takes around 3 iterations to be steady and converge and it takes more around 2 iterations in the large problem. But the policy iteration provides some unreasonable solution in the large problem, which means that it is not the best algorithm with few iteration to converge. Q learning does not converge in either small problem or large problem with around 10000 iteration. The value iteration hits the rewards smoothly while policy iteration does not.

Based on running time, Q learning took 100 times to finish compared with value iteration and policy iteration. The big reason is that Q learning stucks in local optima. Even though any iteration in Q learning is fast, it has to finish max iteration, which is 10000 to finish. Overall running time is the slowest.

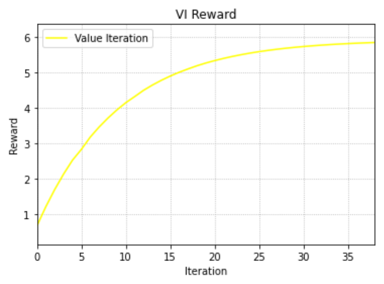


Figure 30: VI Reward (small)

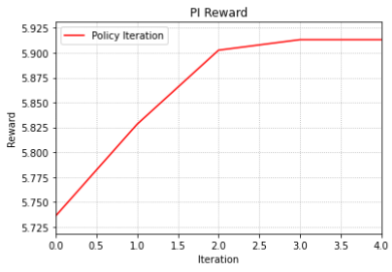


Figure 31: PI Reward (small)

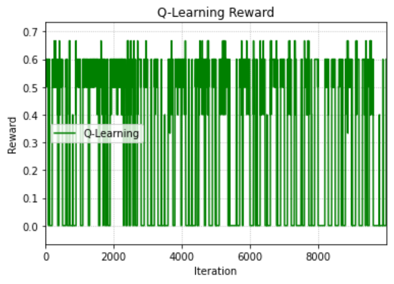


Figure 32: Q learning Rewards (small)

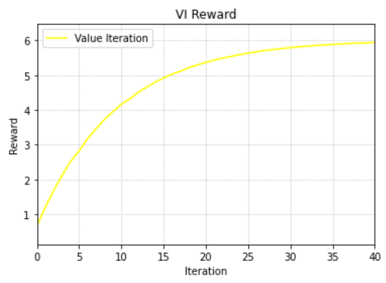


Figure 33: VI Reward (big)

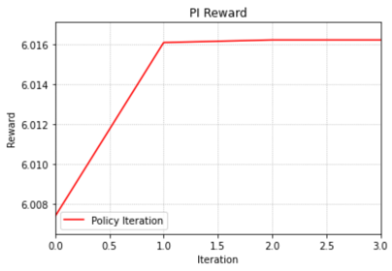


Figure 34: PI Reward (big)

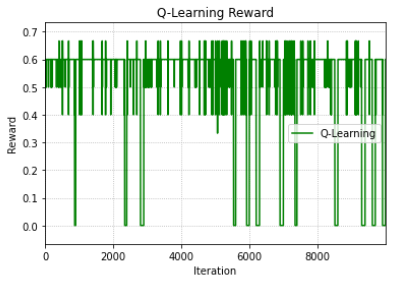


Figure 35: Q learning Rewards (big)

## 4 Summary

### i. How many iterations does it take to converge?

For Frozen Lake problem, three algorithms converge in 5 iterations in small grid world. Q-Learning converges in 13 iterations, value iteration and policy iteration do not converage in large grid world. For small and large fire management problems, the situation is going to inverse. Q-Learning does not converge. But policy iteration and value iteration converge within 5 iteration and 60 iteration.

### ii. Which one converges faster? Why?

Q-Learning is always the fastest one in either small or large two problems. The key reason is that Q-Learning can learn the value of an action in a particular state with stochastic transitions and rewards without requiring adaptations. But the disadvange is that it will stuck in local optima sometimes.

### iii. How did you choose to define convergence?

The good defination of Convergence is that with compatiable running time it should hit the highest rewards and remain in the steady state.

### iv. Do they converge to the same answer?

The answer is No. value iteration and policy iteration share the same optimal path in small grid world while Q-Learning provides different solution in Frozen Lake problem. In the large grid world, only Q-Learning converges. The policy iteration converges and value iteration do not even converge. Even solutions without converge are different in policy iteration converges and value iteration. In either small or large the fire management problem, different algorithms provide different solution to burn forest but only Q-Learning does not converge.

### v. How did the number of states affect things, if at all?

In either Frozen Lake or Fire Management, when the number of states increases the steps from starting to goal will increase. In other words, many possible solution will be remained and time to converge will be increasing. For the Frozen Lake problem, it increases from running time when states increase from 16 to 64 as an example. The another aspect is that the variance will increase when states increase. At the same time, steady state reward drops since it will take more steps to reach the goal and any extra step will lead to drop the rewards.