CS 4641/7641 Assignment 4: Reinforcement Learning

Sitong Wu

School of Electrical and Computer Engineering Department

Georgia Institute of Technology

Atlanta, GA, USA

swu321@gatech.edu

***Abstract* – This is a report about three difference reinforcement learning algorithms. They are model-based algorithms like Value Iteration and model-free algorithms like Policy Iteration, and Q-learning. Model-based means that the algorithm has domain knowledge while the model-free algorithm does not.**

I. INTRODUCTION

This paper talks about reinforcement learning applied to two problems. Both problems are 'grid world' problem which one is much simpler than the other. Value iteration, policy iteration, and Q-learning would be applied to these two problems and get the proper policies to help the agent reach the destination. Then further comparison and analysis will be discussed.

II. TWO MDP PROBLEMS

The two problems in this paper are both about 'grid world' problems, also can be considered as maze problem. The maze is in a rectangular shape formed by many small blocks. The agent starts from the beginning block and must reach the destination block. In each problem, the beginning block locates at the lower left corner of the maze and the ending block locates at the upper right corners. There is also a bunch of wall block which the agent cannot pass through. The agent can move through four different directions. They are north, south, east, and west, which are the action domain. In the maze problems of this paper, the agent has 80% probability that it will follow the guidance of the action. There is 20% probability that the agent might end up with going to the other directions. For each direction other than the action, the probability is 20%/3. If the agent takes an action of walking towards the wall, the agent will stay at the same spot. The reward value is correlated to the number of steps taken. It is about the step value times -1 plus 100. Every step during the trip will have cost. Generally, the fast the agent can get to the destination, the higher the reward.

In common sense, there are many distinct ways to reach the ending block from the beginning block. The idea of these two MDP problems is to find the shortest path for the agent and the proper action when the agent is currently reached a certain block.

The grid world problem can be considered as the abstract model for many other practical problems. It contains simple visual models which can be combined with rewards, penalties, obstacle, and action uncertainty. Researchers can modify the variables of the grid world problems to model problems like discounting rates and stochastic behavior. The variables include the walls, number of state and dimension of the maze to varying the problem difficulties. It is also possible to use more than one agent in the problem, for simplicity this paper only uses one agent for each maze, to model swarm robots or autonomous vehicles.

A. Easy Maze

The easy maze has a "small" number of states, 20, includes a beginning state and a destination state. It is a 5-time-5 maze with 5 obstacle blocks, as shown in figure 1 below.

B. Complex Maze

The complex maze has a "large" number of states, 513, beginning and ending inclusive. It is a 25-time-25 maze with 112 obstacle blocks, as shown in figure 2 below.

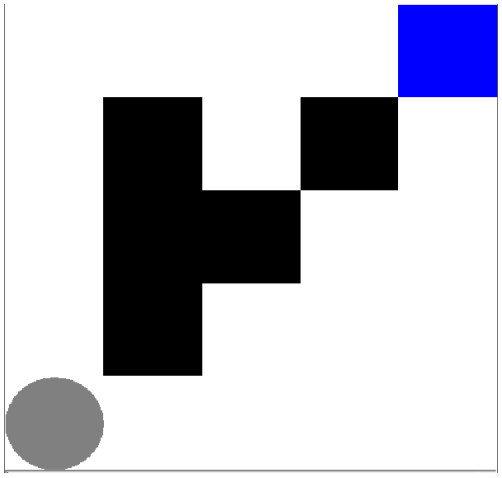

Figure. 1. Easy Maze

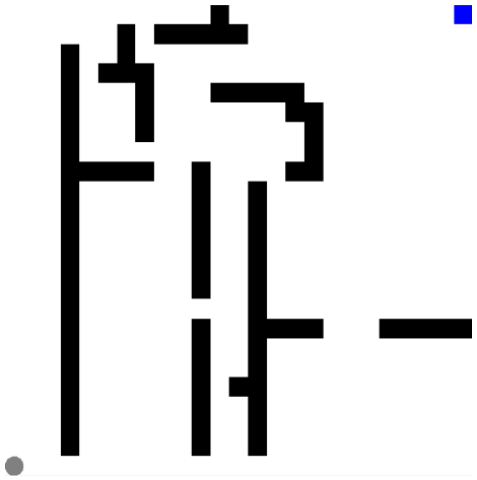

Figure. 2. Complex Maze

III. Solve Each MDP Problems Using Value Iteration and Policy Iteration

A. Value Iteration

The value iteration algorithm is an iteratively updated algorithm that calculates the value, also known as utility, of each state using the values of adjacent states with the update equation 1. When the update difference of value, V, is smaller than a certain threshold ε, the learning process is down. The value is guaranteed that it will converge.

$$V_{k+1} \leftarrow max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')] \qquad \text{(Equation 1)}$$

The result of value iteration of the simple maze is shown in figure 3 and the data is in figure 4 and figure 5 below. The value iteration algorithm is running for 100 iterations. The numbers in figure 3 is the resulted values and the arrows are the resulted policy actions. Though figure 4 and figure 5 show totally 100 iterations, the result from figure 3 is gotten around the iterator 4.

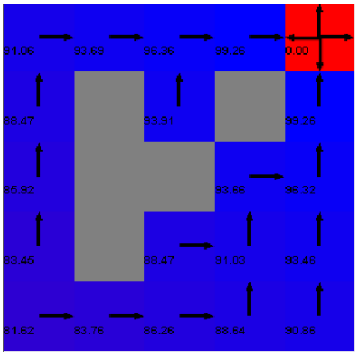

Figure. 3. Simple maze VI resulted policy

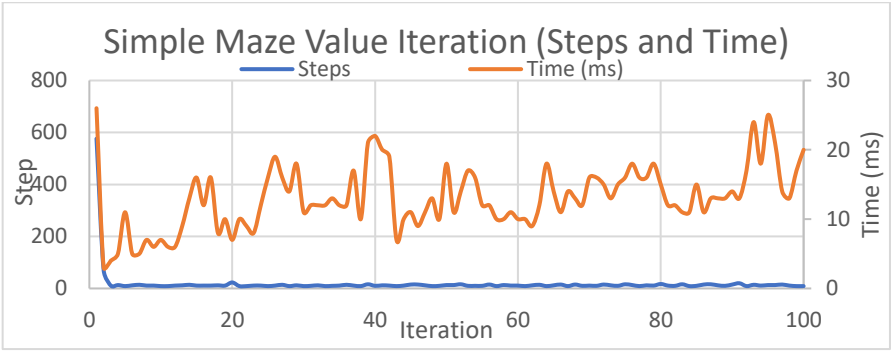

Figure. 4. Simple maze VI iteration vs. step & time

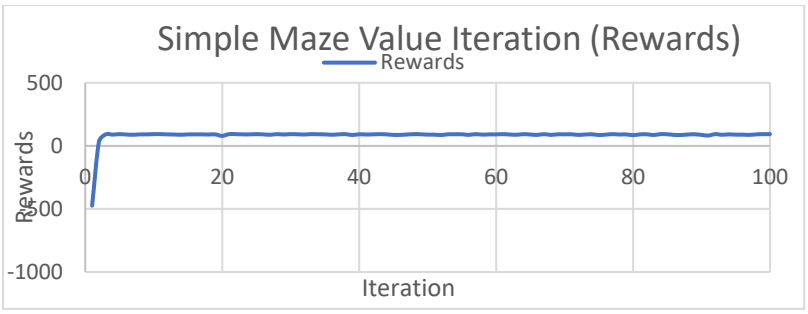

Figure. 5. Simple maze VI iteration vs. reward

The time taken for each iteration round is also shown in figure 4. The time plot in figure 4 shows that the time taken is kind of random. The duration of each iteration seems do not depend on the previous iteration round.

The most important information that figure 4 and figure 5 give is that the value converges when the iteration is around 4. It is hard to plot the variation of the values of states, but the reward values can show the situation when the values of states converge. As the actions are taken from iteration to iteration, the values of states converge, and the total reward will also reach and maintain its maximum. Notice that the reward function is correlated to the step number which is described in the previous section. Generally, fewer steps are taken from the starting block to the final block, the higher reward the agent gains.

The figure below shows the result of value iteration from the complex maze. Figure 6 shows the values of each state and the action in the resulted policy of each state. The color code, except the destination block, in each block, denotes whether the agent wants to enter. BLUE means the agent prefers to enter the state and RED stands for the contradict case. It is clearer in the complex maze than the simple maze that the blocks closer to the destination is bluer than then blocks near the beginning block which are more red, which reflects the desire of getting to the destination.

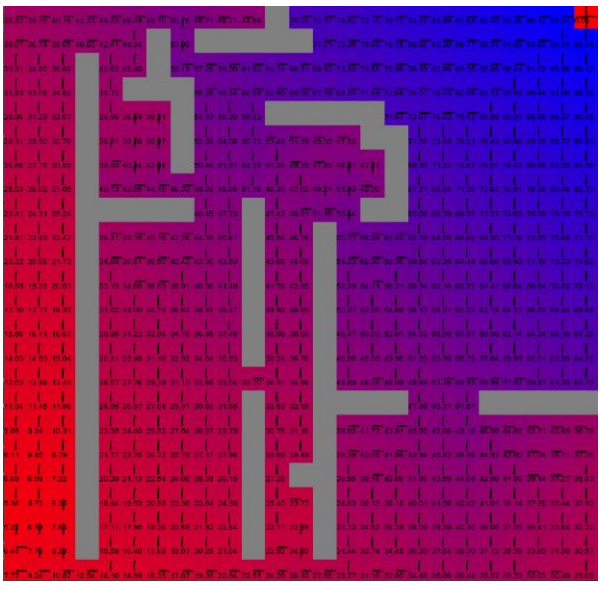

Figure. 6. Complex maze VI resulted policy

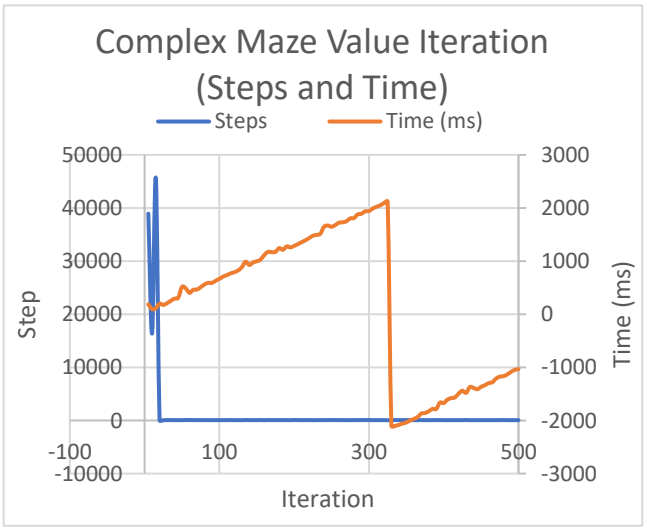

Figure. 7. Complex maze VI iteration vs. step & time

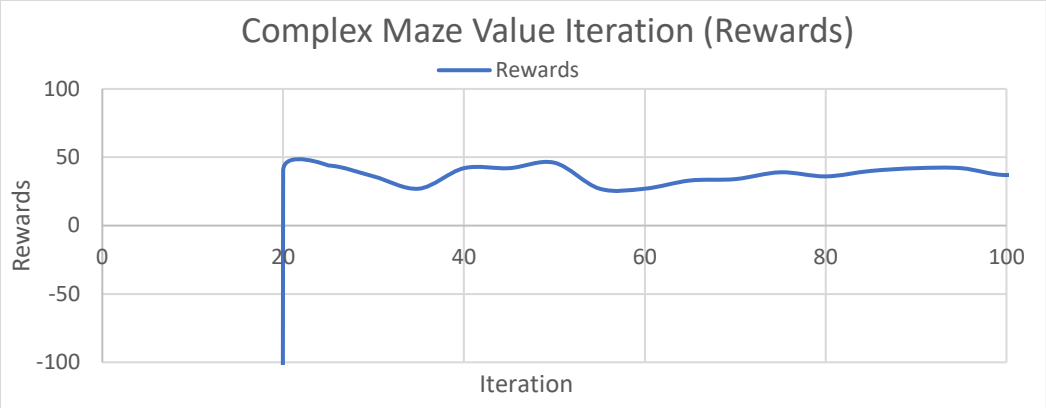

Figure. 8. Complex maze VI iteration vs. reward

From figure 7, it is shown that the time taken for each iteration is also much longer than that in the simple maze case. The time plot in figure 7 sometimes goes to negative is because of the time function implemented. There is a threshold for the maximum time that can be recorded. When the time is way too large, the time number would result to negative. From figure 7 and figure 8, a conclustion of converge iteration is around 20 which is also much larger than the converge iteration number in the simple maze case.

Figure 9 below does a comparison of two value iterations. It shows that the simple maze converges faster than the complex maze due to the fewer state values to calculate so that less variation between iteration rounds. Another thing can be noticed is that the simple maze can come up with a higher reward which is more close to 100 comparing to that of the complex maze. This is because of the longer path in the complex maze.

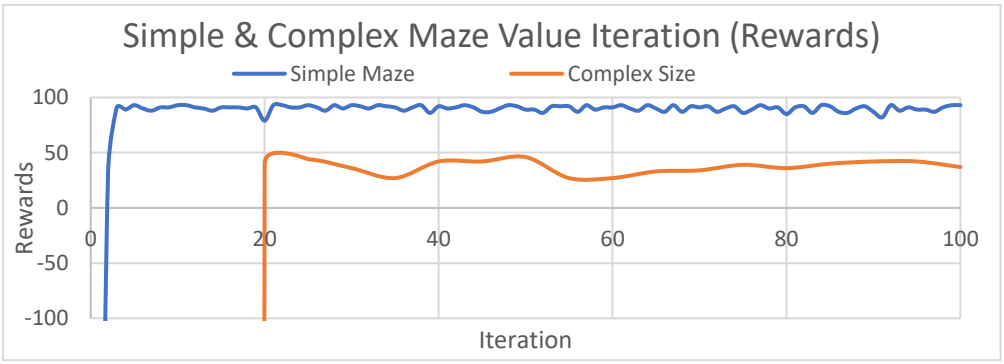

Figure. 9. Comparison of VI results between mazes

B. Policy Iteration

Policy iteration starts from evaluating a fixed policy which is not necessarily the optimal one and do evaluation and improvement until the values of states converge. The evaluation function is equation 2 below which calculate all state values based on a policy π_i, and put the state values into equation 3 to do improvement. Then get a new policy π_{i+1}. This process will iteratively repeat until state values converge.

$$V_{k+1}^{\pi_i} \leftarrow \sum_{s'} T(s, \pi_i(s), s')\left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')\right] \qquad \text{(Equation 2)}$$

$$\pi_{i+1}(s) \leftarrow argmax_a \sum_{s'} T(s, a, s')\left[R(s, a, s') + \gamma V_k^{\pi_i}(s')\right] \qquad \text{(Equation 3)}$$

This algorithm would also converge but it is more computationally heavier than value iteration, which can be observed by the two steps above while the value iteration only has one step.

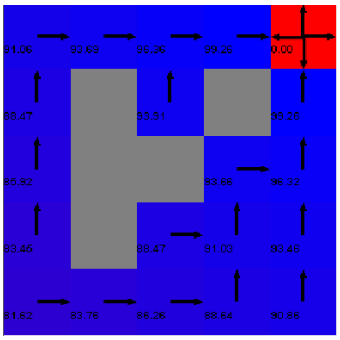

Figure. 10. Simple maze PI resulted policy

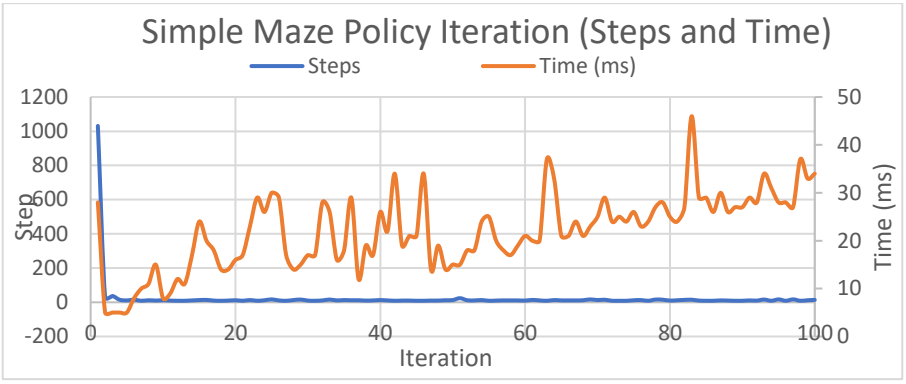

Figure. 11. Simple maze PI iteration vs. step & time

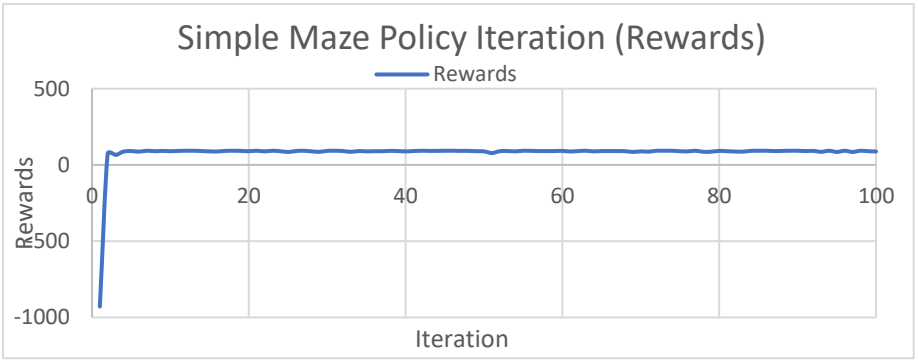

Figure. 12. Simple maze PI iteration vs. reward

Figure 10, figure 11, and figure 12 shows the result of policy iteration on the simple maze. The convergence happens at around iteration 3. This iteration number is very close to the value from value iteration. The pre-experiment assumption is that the value iteration should take more iteration round to converge; however, the result gotten here is not very surprising. First, the MDP problem of simple maze chosen here is very easy and secondly, two converged iteration values are also very close. Any tinny measurement tolerance might affect the result a lot.

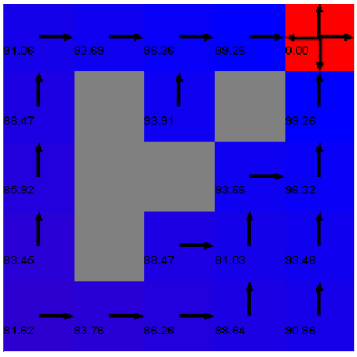

Figure. 13. Simple maze VI resulted policy

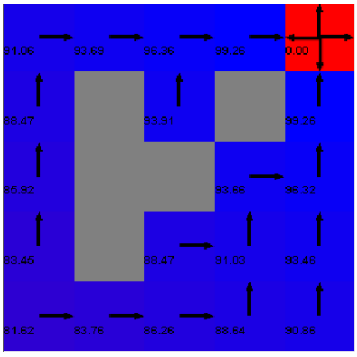

Figure. 14. Simple maze PI resulted policy

Comparing the result from both algorithms for the simple maze case, it is no surprise that they end up with the same policy. Since both follow their own update rules and get state values to converge with such a simple maze case, the same and optimal policy will be generated.

The results of policy iteration on the complex maze is shown below in figure 15, figure 16, and figure 17. The converge iteration is around 30 which is faster than that of the value iteration case, 20. The difference is not very large. The ideal case is that the policy usually converges before the values do.

The result polices from two algorithms for the complex maze is not compared state by state here due to the complexity of the graph. Through the color code, it looks like they are similar. The reward plot shows that both algorithms converge to a very close value.

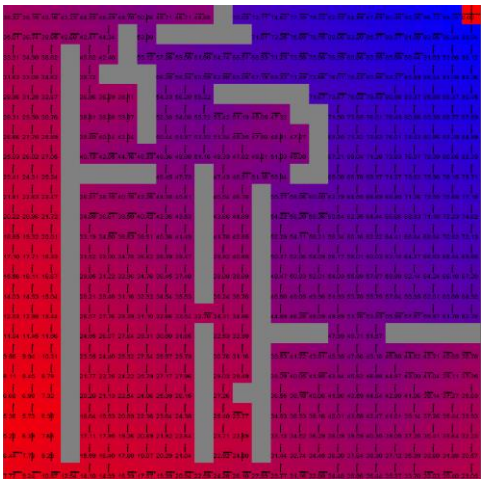

Figure. 15. Complex maze PI resulted policy

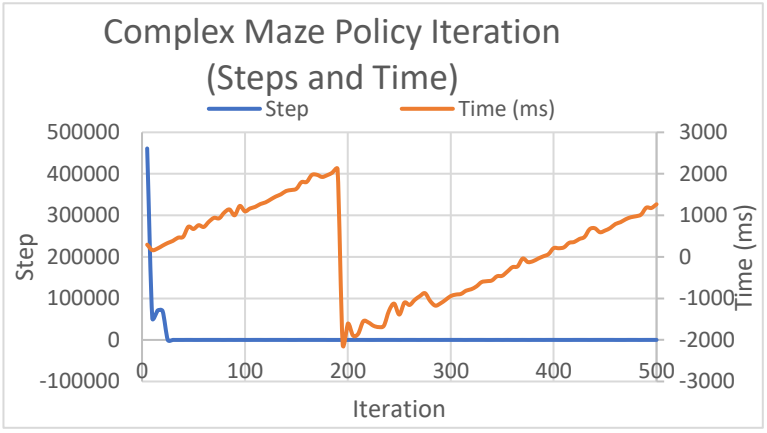

Figure. 16. Simple maze PI iteration vs. step & time

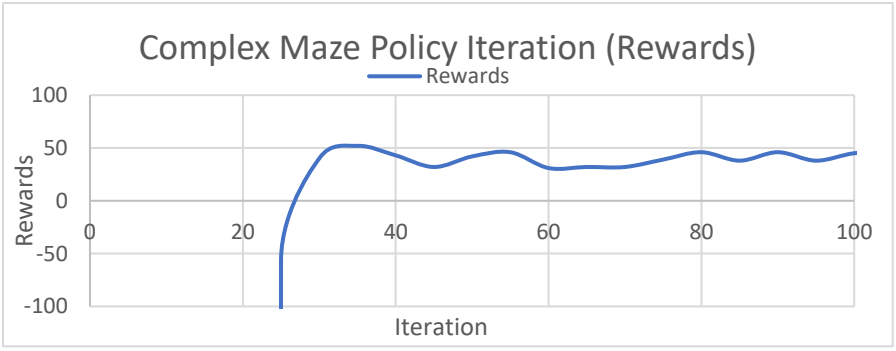

Figure. 17. Complex maze PI iteration vs. reward

Figure 18 below compare the policy iteration results from the simple maze and the complex maze. Not, surprisingly, the complex maze with more states requires more iterations to converge and get lower total rewards when evaluating both maze policies with the same reward function.

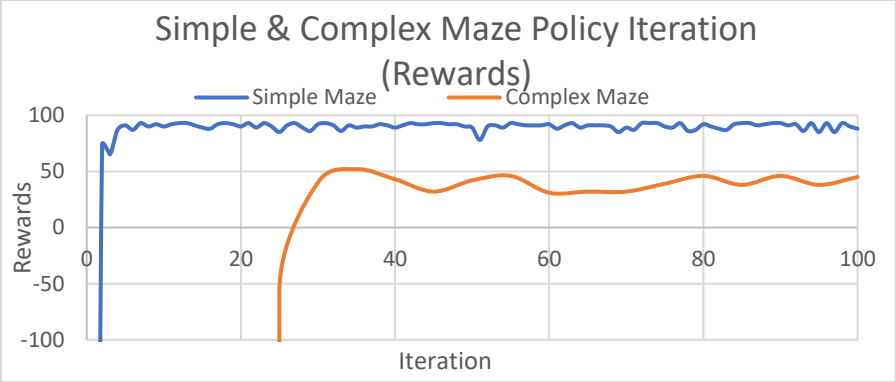

Figure. 18. Comparison of PI results between mazes

IV. Q-LEARNING SOLVING TWO MDPS

Q-learning algorithm is chosen as the model-free reinforcement learning algorithm to solve the two MDP grid world problems. The reason for choosing Q-learning is that this algorithm will guarantee to generate the optimal solutions with proper parameter choices. Different from the previous two algorithms, Q-learning does not require the domain knowledge. The domain knowledge, in this case, include the model function, T, which generates the probability of ending at next state, s', with given current state, s, and action, a, and the reward function R(s). Initially, in Q-learning algorithm, Q[s,a] of each state is randomly signed with a small value and get updated during exploration and exploitation. Q[s,a] values are updated by considering the old value and the improved estimate by the following function.

$$Q'[s,a] = (1-\alpha)Q[s,a] + \alpha(r + cQ[s', argmax_{a'}(Q[s',a'])] \qquad \text{(Equation 4)}$$

As the learning rate, the higher the α, the faster the learning process can reach its maximum. γ is the discount factor that determines the weight of the later reward consideration.

Figure 19 below shows the result for Q-learning on the simple maze with a constant discount factor 0.99 with different learning rate. Due to the algorithm idea of random searching, the outcome reward curve is not very smooth, but they are jumping in a certain range. From figure 19, the jump amplitude of α = 0.2 is the smallest and that of the curve α = 0.8 is the largest. If only compare the Q-learning with the learning rate, the converge iteration is not very clear, but for sure they are approximately smaller than 10. From the general idea of machine learning, the bigger learning rate will make the algorithm result converge faster; however, a bigger learning rate might cause an overshooting issue.

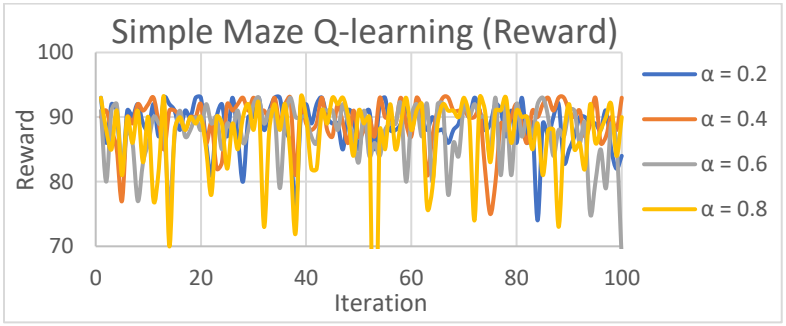

Figure. 19. Simple Maze Q-learning (γ = 0.9)

Due to the running for Q-learning is very long which is much longer than the previous two algorithms, α value changes only starting from 0.2 with interval 0.2 and only discuss α in this paper. Figure 20 shows the first resulted policy, the last second resulted policy and the last resulted policy. The simple maze is very easy to visualize and to compare the policies gotten from different α. Basically in this simple maze, if the agent starts from the beginning block and goes in the direction either north or east should be fine. Ironically, in the α = 0.8 case, the policy at the 1st-row 2nd-colon block points to west. Intuitively, it is not the best decision, which means overshoot happens here. So the fastest and more optimal choice for the simple maze is to run the Q-learning with a learning rate of 0.6.

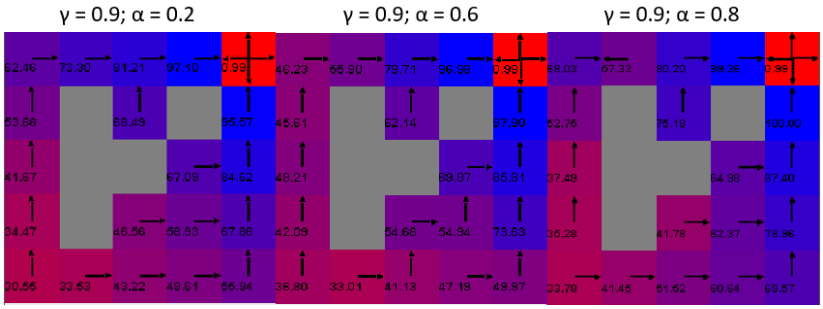

Figure. 20. Simple Maze Q-learning policy with 3 different α (γ = 0.9)

Compare the α = 0.6 Q-learning result with the other two algorithm results on the simple maze, figure 21 is gotten. It is clear that Q-learning converges much faster at the beginning while the value from value iteration and policy iteration have some crazily low values. After convergence, the value iteration and policy iteration tends to stay at the current value while the Q-learning is still searching for new possibilities.

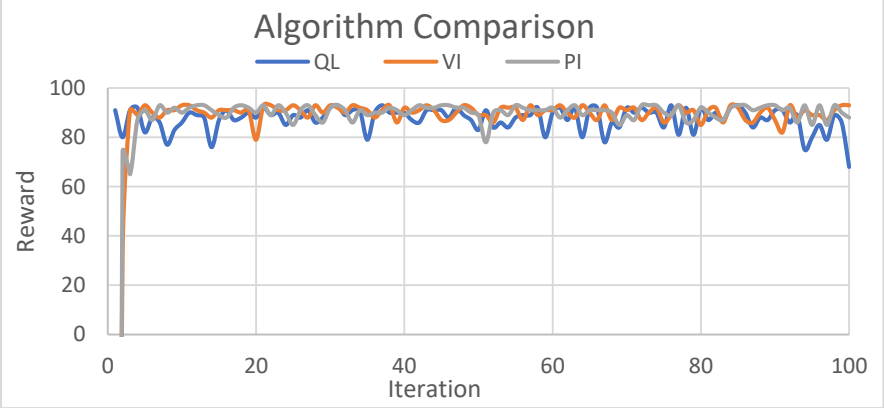

Figure. 21. Comparison of three algorithms on simple maze

Figure 22 is the result for Q-learning on the complex maze. A similar step has been done as above with the simple maze case. Except for this time, a more exaggerated α value is chosen while α = 0.01. This is a very slow learning rate and the curve of α = 0.01 in figure 22 shows the slowest convergence speed comparing to the others. α = 0.06 curve converges around iteration 150; α = 0.04 curve converges around iteration 205; α = 0.02 curve converges around iteration 335; α = 0.01 curve cannot converge within iteration 500.

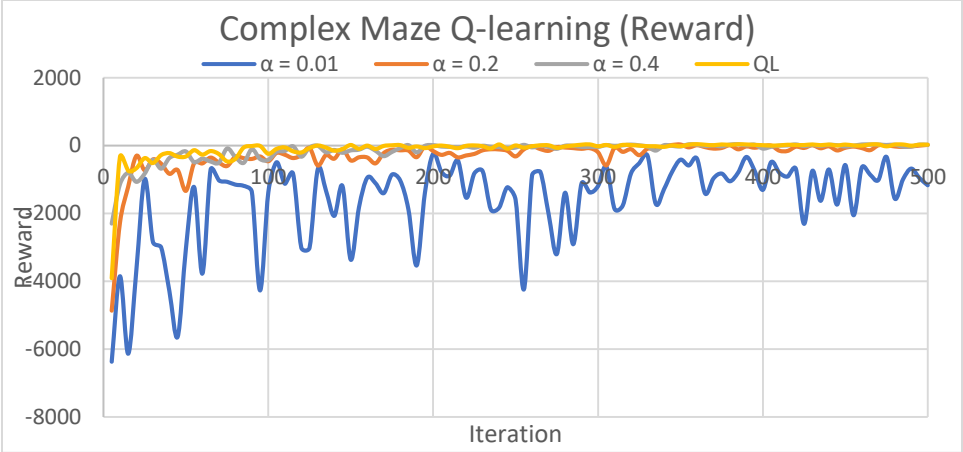

Figure. 22. Complex Maze Q-learning (γ = 0.9)

The visualization of the outcome policies is very hard and not practical at all for the complex maze case due to a large number of states. Therefore, the learning rate α = 0.6 is temporarily used for the next part. Due to the similarity of the problem case, the parameter used for the simple maze might be applicable to the complex maze. The comparison of three algorithms on the complex maze is shown in figure 23. It shows the advantage that the value can converge by Q-learning with a much smaller iteration number. In the plot below, Q-learning can converge around 15 while the value iteration curve converges around iteration 20 and the policy iteration curve converges around iteration 30.

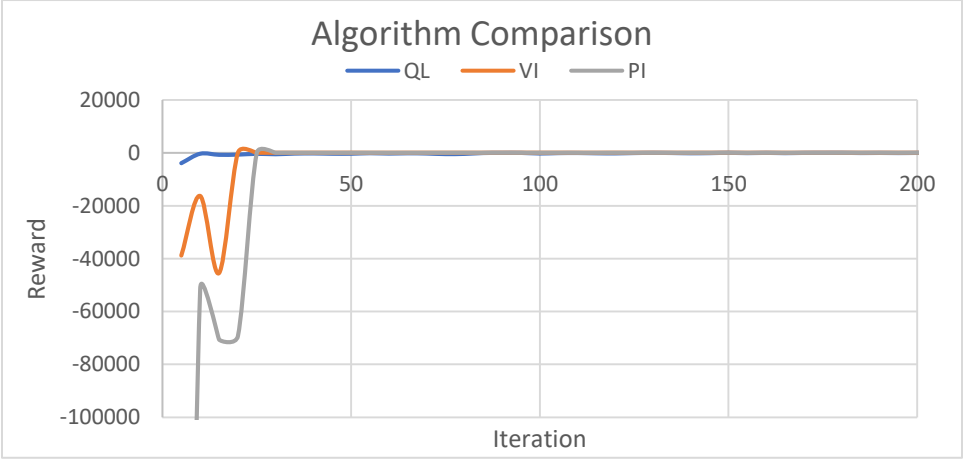

Figure. 23. Comparison of three algorithms on complex maze

V. Conclusion

This project uses grid world model to form two maze problems. The different algorithms have been applied and analyzed. The maze problems are relatively easy for value iteration, policy iteration, and Q-learning. Especially the simple maze which has small state space, can be eyeball solved. So researcher could easily compare and evaluate the resulted policies from those algorithms.

When comparing value iteration and policy iteration, while both perform well, value iteration could require a little bit smaller iteration number to converge. While comparing all three algorithms together, the Q-learning can always converge faster than the other two even it does not have the domain knowledge. Another phenomenon from Q-learning is that it is way slower than the other two during the

iteration cycle. It makes sense because ,during each iteration, Q-learning has more things to take care. So one conclusion can be that Q-learning can converge with a smaller iteration number but a slow running speed during each iteration.

When the problem comes to Q-learning, overshooting should be prevented. This means that the learning rate cannot be too large. There are some good ways to prevent overshoot with an even faster converge speed. One way is to use the learning rate descendant. If the beginning learning rate is not exaggeratedly large, a decreasing learning rate can help the value converge faster at the beginning. As the value gets closer to the local optimal, the smaller learning rate will help the progress aggressively approach the local optimal. Another more complex way is applying the idea of annealing. Annealing adopts the idea that explores more at the beginning iteration and exploits more at the latter iteration. With annealing, the learning process will try to reach more cases as possible to search for the global optimal solution at the beginning and later focus more on the more optimal one.

As a follow-up experiment, several more complex problems can be considered, such as block dude problem. For the model-based algorithm, it might be fun to run multiple agents and add more rules, such as no overlapping, to increase the complexity of the problems. About Q-learning, a fixed learning rate with varying discount factor experiment can be done to investigate the effect of the discount factor on the solution. There are also several other reinforcement algorithms that can be applied and got compared with the algorithms in this paper.

VI. References

[1] http://burlap.cs.brown.edu

[2] https://classroom.udacity.com/courses/ud501/lessons/5247432317/concepts/53538285920923

[3] https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4