GTID: ywang3564

Assignment4: Markov Decision Processes

To present the Markov decision process, I choose two grid world problems. One is simple and the other one is a little complex.

The first problem starts from the orange starting points on the bottom right, the agent needs to find the shortest path to the top-right corner (green point). The agent can move up, down, left, and right unless there is a wall (defined as black blocks) or boundary. Moving will cost a reward lost and reaching the final destination will have a high reward. The first grid world (Fig1a) is a 10 X 10 panels. I specifically designed two alternative paths to reach the final destination. The path number are closed to each other and I will have some interesting observation from this problem.

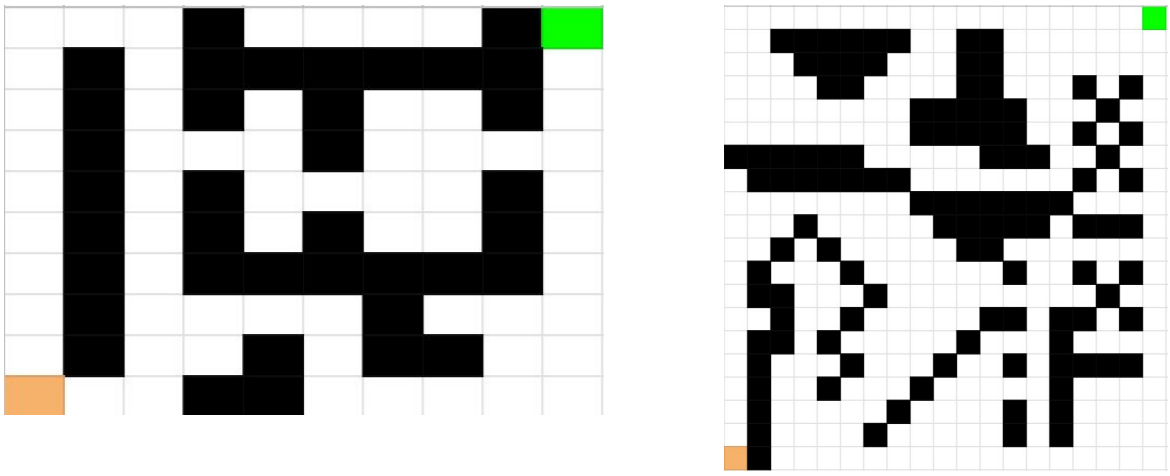


Figure 1. Two interesting problems

Reinforcement Learning

In the following analysis, I will apply three algorithms indicated in the assignment: value and policy iterations, and Q-learning. The package applied here is BURLAP and source code is downloaded from Jonathan Tay github.

Both value and policy iterations belong to dynamic programming (DP) and improve the performance from different aspects. For policy iterations, the policy π will be replaced by a better policy π' in the next round, then evaluate the policy improvement and improve it again. For a finite MDP problem, the policy number is limited. Therefore this method can converge to an policy within a finite number of iterations.

Value iterations has a different strategy with policy, it assigns different rewards to a different state, including positive and negative reward. Then the next iteration will make use of the state of last time and make their decision. During this process, the value function is defined and help the agent to make the decision.

The Q-learning algorithm mainly combined the value and policy interactions and defined a Q-function. The Q-learning algorithm will consider both the current state and next action, thus reconcile the advantages of both policy and value iterations.

Value and Policy Iteration on Problem I

For the simple grid world problem, I first tested method of value iteration and policy iteration. The discount factor was set to 0.99 and the max iteration time is set as 100.

Figure 2 showed the convergence time corresponding to the iterations. In general, both algorithms can converge within 50 iterations and reaching a convergence value as small as 7e-7. Specifically, the policy algorithm took 14 iterations to reach the smallest value while the value algorithm took 48 iterations.

Figure 3 showed the time for each method. Both methods spent a similar time to get the best policy: policy required 0.30s and value required 0.24s. But the iteration time for each method is very different: the policy spent 0.02s on each iteration and value spent about 0.005s. Policy tends to spend more time due to it is designed to find a better policy for the next step. This is very different with value iteration, which is linearly increased.

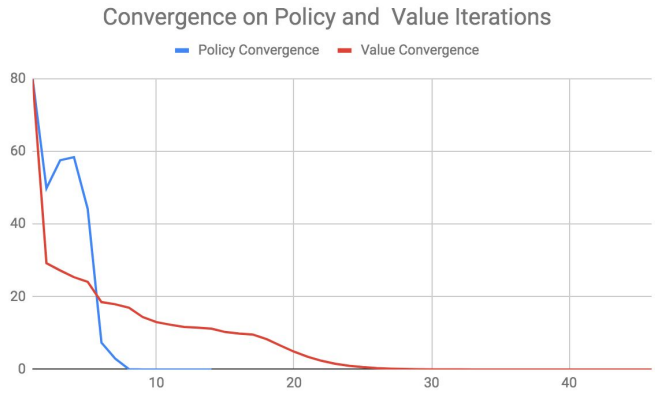


Figure 2. Convergence plot for value and policy iteration on problem I.

Figure 3. Time for value and policy iteration on problem I

Figure 4 illustrated the reward result from both methods. The starting reward for two methods are both negative and they quickly reach their maximum reward. The iteration for policy and reward are 5 and 12. Therefore, the time for getting the best reward is earlier than the convergence time. The reward will a slight fluctuation and keep stable after that. This is caused by the uncertainty of exploration probability.

In general, policy iteration had a better performance than value iteration. This observation matched the design of the grid. In our grid, we only have binary value for each step and the total state is simple. Therefore, policy function is better than the value function.

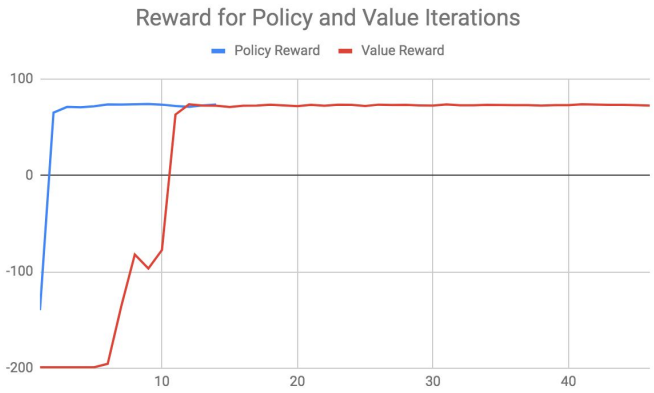
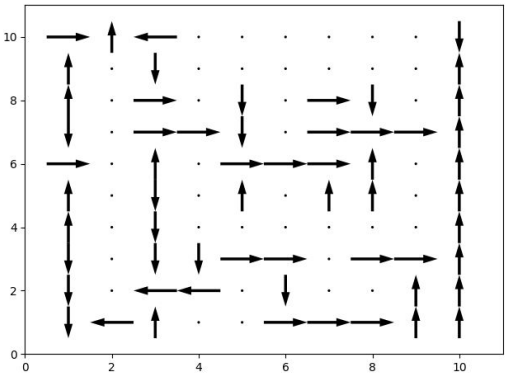
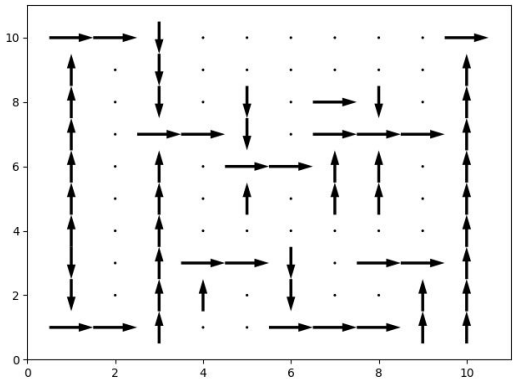


Figure 4. Achieved reward by value and policy iteration on problem I

Figure 5 showed the solution for two methods. Fig5a and Fig5b shown the policy for value 5th iteration and 46th iteration. When the iteration is 5, the policy is still unclear. We can see a lot of down or right arrows in the first column, while most of them are up in the iteration 46. In another way, the policy is still not so good when iteration is 5. Similarly, Fig5c and Fig5d shown the result of policy iterations. We also see some difference in the first column. Finally, both value and policy iterations generated the same result, except the direction of the destination, which actually does not affect the final result.

It's also very interesting to review the final result of both methods. From the first column, you can find two arrows have down arrays. And this exactly matched my design for the grid. In this grid, I have two paths: the upper one has 25 steps while the lower one has 21 steps. If you wrongly choose to go up first, you still have the chance to go back. The total step is still under 25 steps. That's the reason why these two steps have a down arrow.





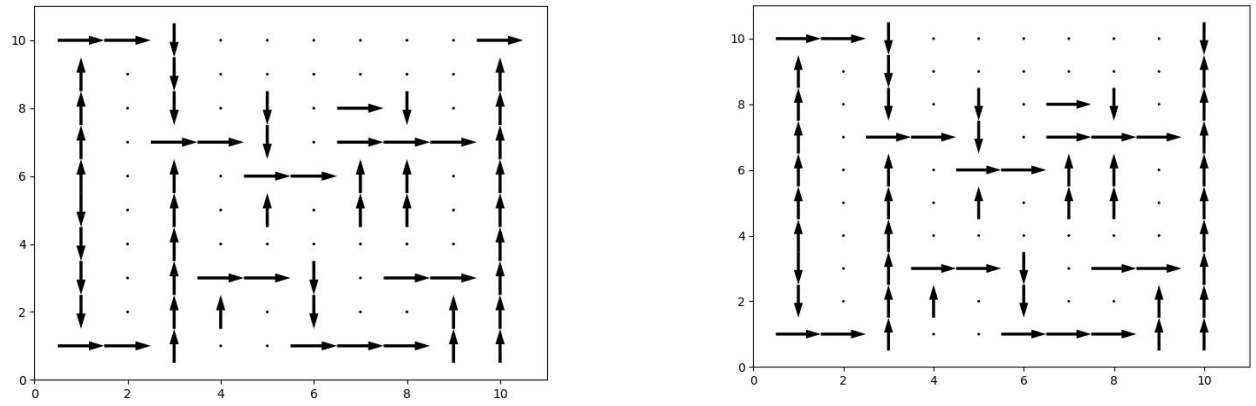


Figure 5. Solutions generated from value iteration (top 5a, 5b) and policy iteration (bottom 5c, 5d) on problem I.

Q-learning for Problem I

For the problem I, I applied Q-learning algorithms with a combination of parameters, including learning rate (donated as L) and exploration rate (donated as E: epsilon). The learning rate was set as 0.1, 0.5 and 0.9, where the epsilon is set as 0.1, 0.3, and 0.5. The initial score was set as zero and max iteration time was set as 1000.

Figure 6 shown the convergence for each parameters combinations. To make it clear, convergence is shown as both time-series and box-plot manner. From the left figure, we can find the convergence quickly decreases when the learning rate is 0.1, which means the agent learned a small proportion from the most recent result. From the right figure, the general convergence is high that others when learning rate is 0.9. The general trend here is: when the learning rate increase, the agent takes more time to converge and have a higher value.

The effect of epsilon can be obtained from the median value of boxplot. If the learning rate is the same (for example L=0.9), epsilon = 0.3 had the lowest value(8th is the lowest among 7th, 8th, and 9th boxplot). This trend is very similar when L = 0.5. This may suggest a balanced exploration rate is better in our case.

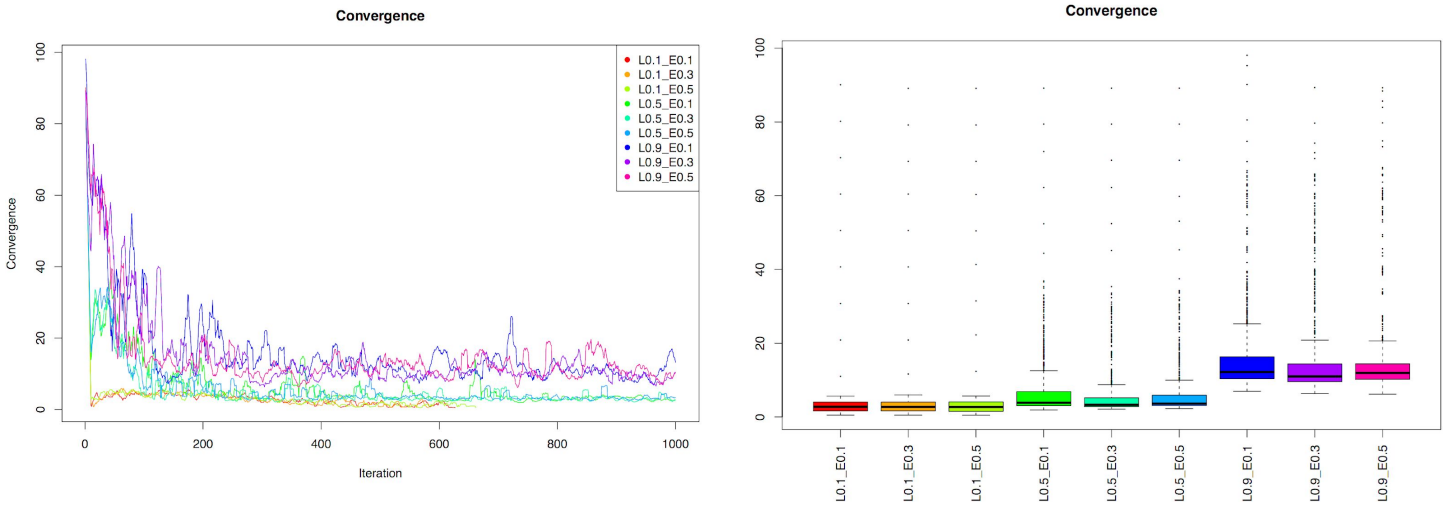


Figure 6. Convergence score for Q-learning on problem I

Figure 7 listed all running time for Q-learning. In general, all nine combinations have a very similar time: ~ 0.15s. Specifically for analysis with learning rate 0.5, the time is lower than other analysis. If we compared all time with previous value and policy iterations, Q-learning used less time and increased linearly with iteration.

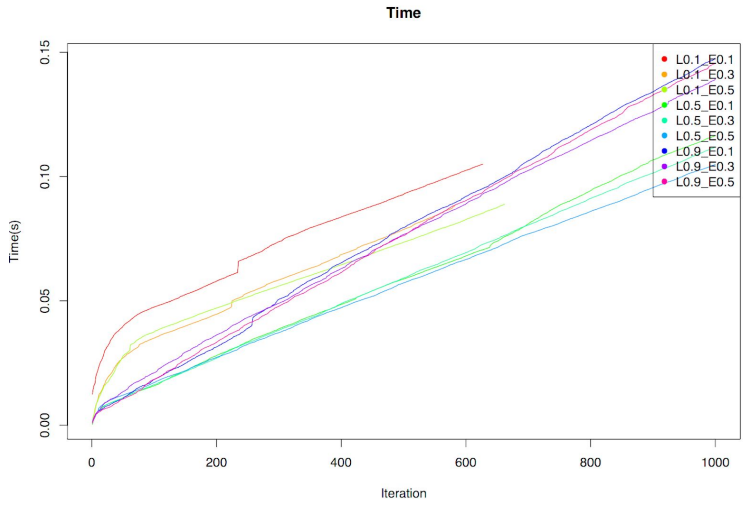


Figure 7. Time for Q-learning on problem I

In Figure 8, you will find the rewards for all nine combinations. Among all combinations, the agent with a learning rate equals to 0.1 can quickly reach the best reward, and the iteration is less than 50. From the left figure, we can see the best reward can only be acquired when the learning rate is not so high. We also observed a large variance when L=0.9, indicating the effect of the most recent policy has affect the next step policy.

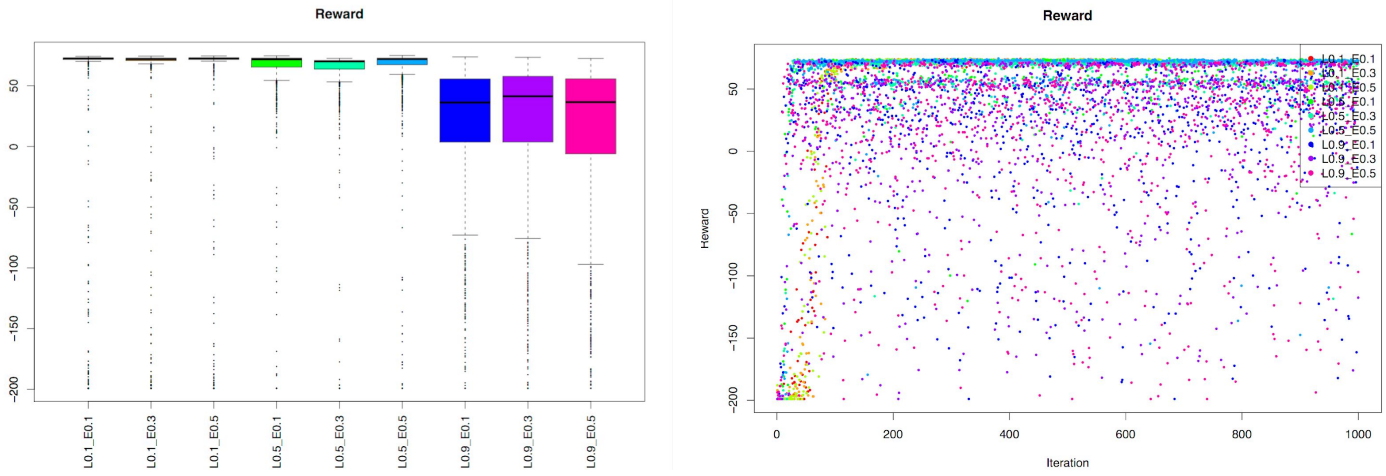


Figure 8. Reward by Q-learning on probme I

Figure 9 shown the final solution when L=0.1 and E = 0.1. Very different from the previous value iteration, it only chooses the upper path.

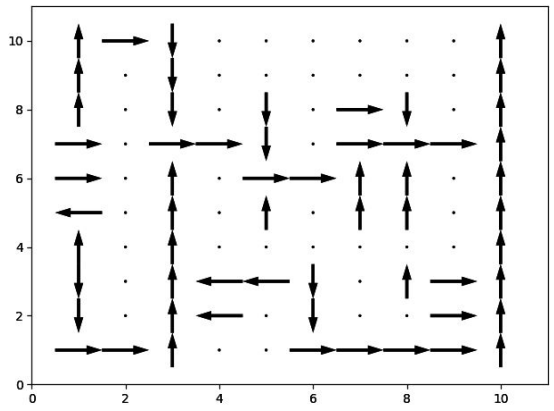
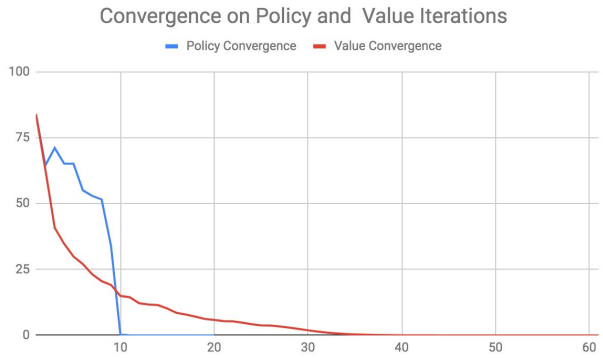


Figure 9. The final solution obtained by Q-learning on probem I

Combining all together, a better solution for problem needs a lower learning rate (0.1) and a balanced epsilon (E= 0.5).

Value and Policy Iteration on Problem II

Compared with the first problem, the second problem is a little complex. And we are going to test three algorithms on this problem: value iteration, policy iteration, and Q-learning. Similar to the first problem, the second will start from the orange starting points. the agent needs to find the shortest path to the top-right corner (green). The grid world size is 20X20 panel (Fig1b). The second problem has only paths but has more spaces to search. All these complex settings will help us better understand three algorithms.



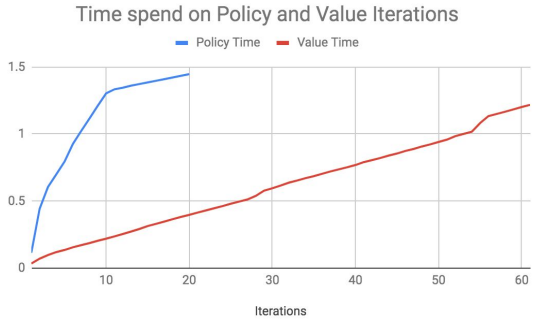


Figure 10. Convergence plot for value and policy iteration on problem II

Figure 11. Time for value and policy iteration on problem II

Figure 10 showed the convergence time corresponding to the iterations. In general, both algorithms can converge within 30 iterations and reaching a convergence value as small as 8e-7. Specifically, the policy method took 10 iterations to reach the smallest value while the value method took 38 iterations. The result is very similar to what we observed from problem I.

Figure 11 showed the time for policy and value iteration. Both methods spent a similar time to get the best result: policy required 1.5s and value required 1.3s. But the iteration time for each method is very different: the policy spent 0.075s

each iteration and value spent about 0.02. Similar to the performance of the problem I, policy tends to spend more time due to the design of seeking better policy to replace the current one. In general, policy iteration had a better performance than value iteration.

Figure 12 included the reward from both methods. The starting reward for two methods are both negative and they quickly reach their maximum reward when iteration time is 5 and 20, respectively. Therefore, the time for getting a best reward is earlier than the convergence time. The value was achieved and will fluctuate due to the probability of each direction.

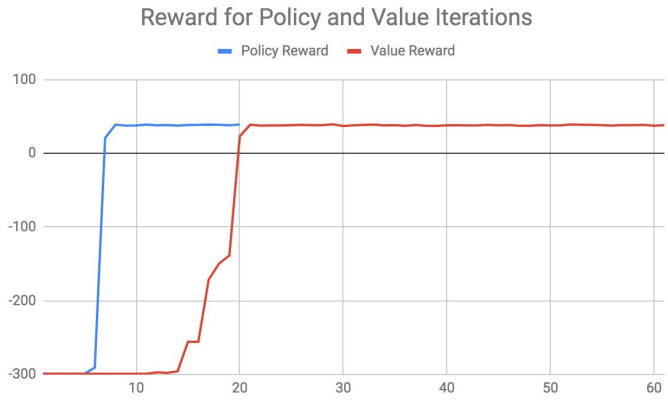


Figure 12. Achieved reward by value and policy iteration on problem I

Figure 13 showed the solution for two methods. Fig5a and Fig14b shown the policy for value and policy iteration respectively. Two method showed the same solution. Actually, there is only one best solution for this problem, even the grid is a little complex. All points shown the same arrows between value and policy iterations.

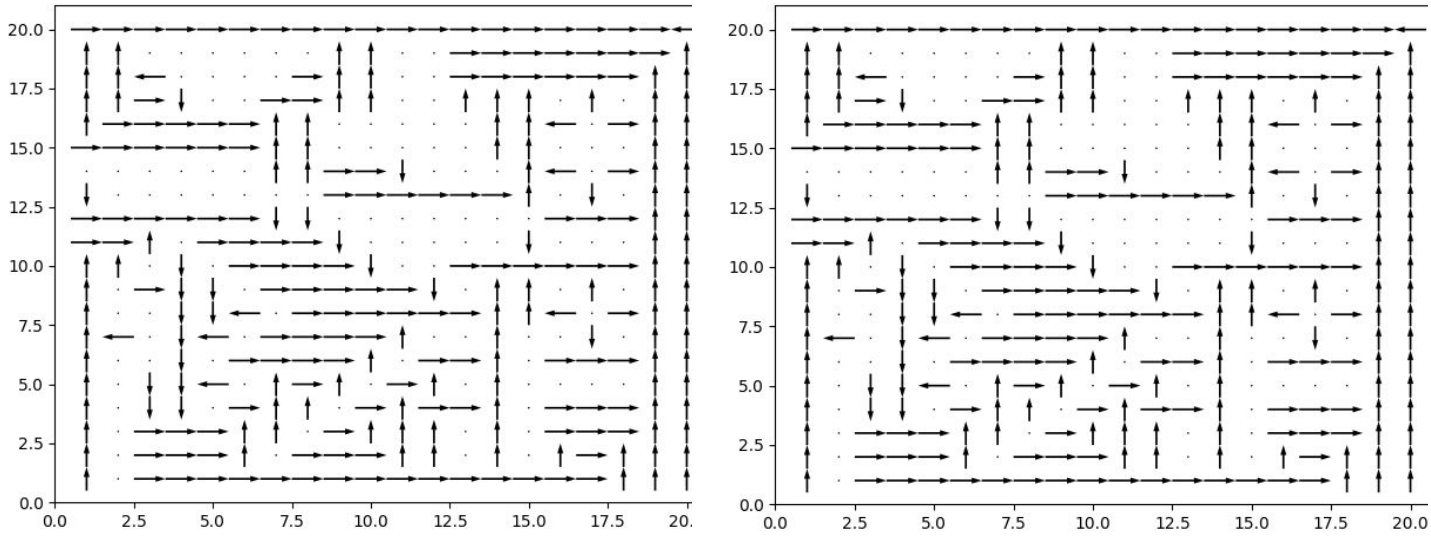


Figure 13. Solutions generated from value iteration (left) and policy iteration (right) on problem II

Q-learning for Problem II

For problem II, I also applied Q-learning algorithms with a combination of parameters, including learning rate (donated as L) and exploration rate (donated as E: epsilon). The learning rate was set as 0.1, 0.5 and 0.9, where the epsilon is set as 0.1, 0.3, and 0.5. The initial score was set as zero and max iteration time was set as 1000.

Figure 14 showed the convergence for each parameters combinations, with both time-series and box-plot manner. Similar to the problem I, the agent can quickly convergent when the learning rate is 0.1, which means the agent learned a small proportion from the most recent result. The left figure also indicated that the convergence value is always higher than the others when the learning rate is 0.9. The right figure indicates the difference between different epsilon. From the general distribution of convergence value, epsilon does not have any effect on this problem. This is a little different with problem I. Combining all together, a lower learning rate is preferred to generate a good solution.

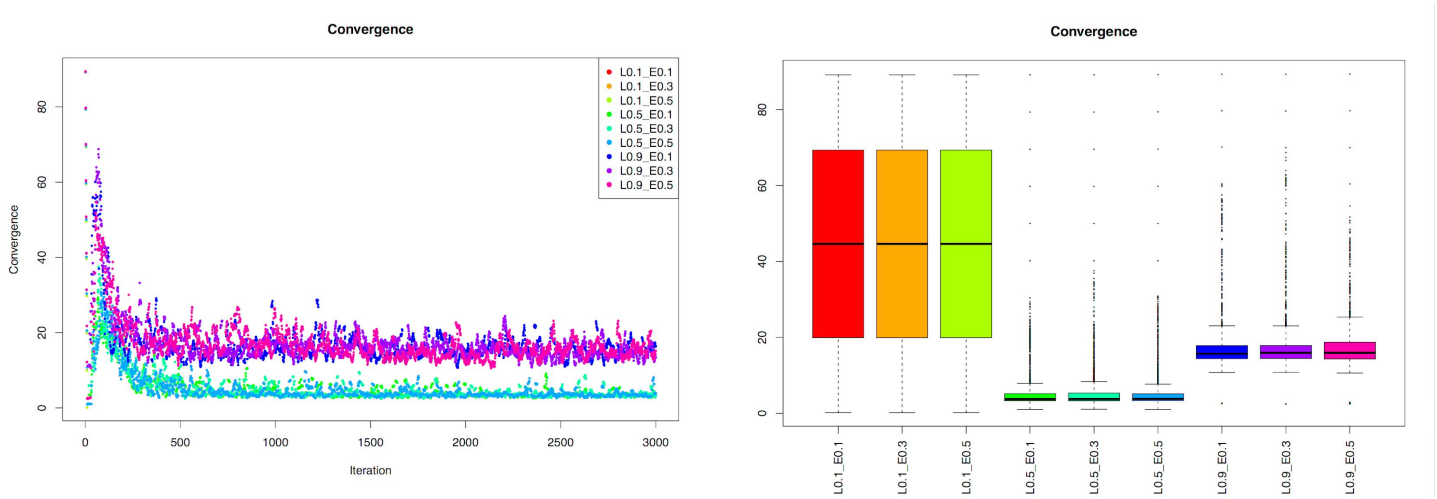


Figure 14. Convergence score for Q-learning on problem II

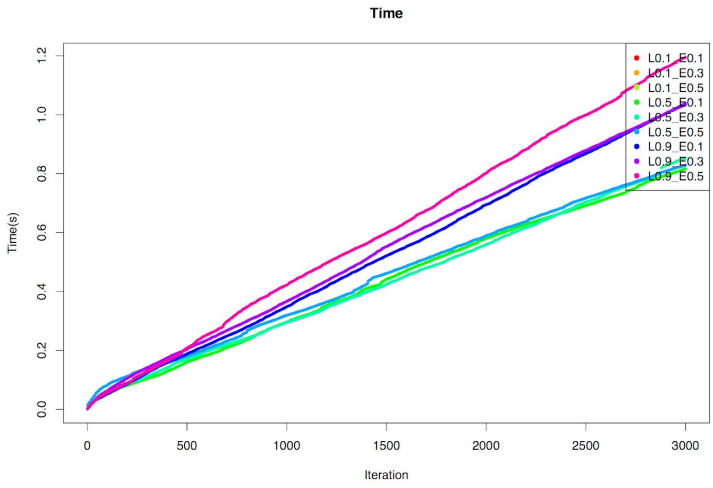


Figure 15. Time for Q-learning on problem II

Figure 15 listed all running time for Q-learning. All nine combinations have a very similar time: around 1.2s when iteration time 1000. Similar with the problem I, when learning rate is 0.5, the time is lower than other analysis. If we compared all time with previous value and policy iterations, Q-learning used less time and increased linearly with iteration time.

In Figure 16, you will find the rewards for all nine combinations. Surprisingly, when learning rate = 0.1, it can not get the best reward even it can quick convergent: the reward is -300. Among all three parameters, the best solution is: learning rate equals to 0.5. The right figure is a little messy, but the left figure indicated that learning rate 0.9 is not suitable for this problem and always have a lower reward score.

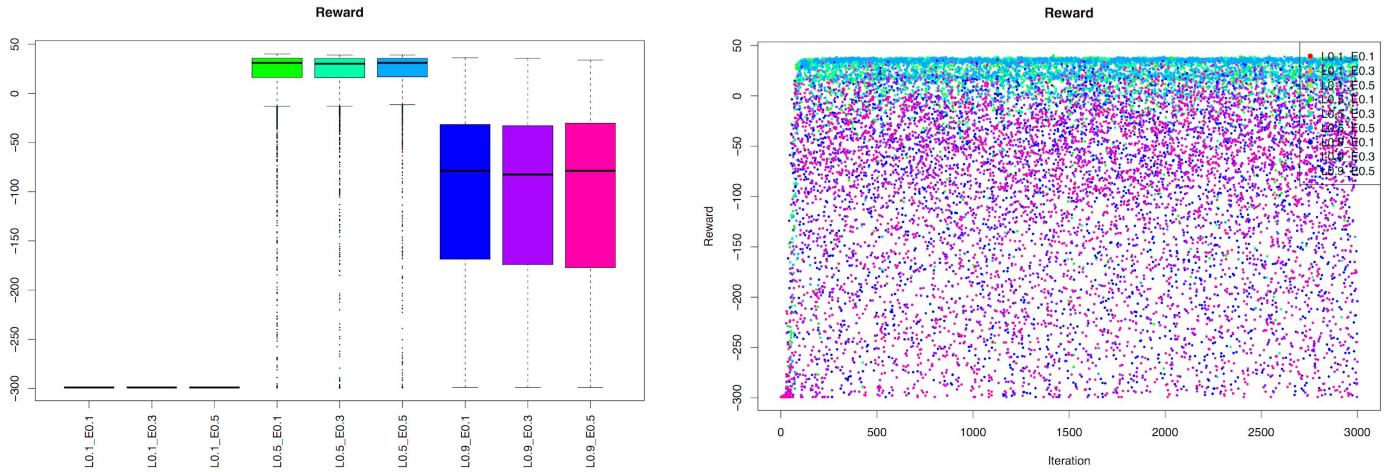


Figure 16. Reward by Q-learning on probme II

Figure 17 showed the final solution when L= 0.1 and 0.5 respectively. The exploration rate is 0.1 for both figures. After comparing the left and right figure, we concluded that L = 0.5 provided a better solution. For example, we found a lot of arrows pointing to left or right on the first column of the left figure, which was wrong directions. Therefore, a higher learning rate is suitable for the second problem.

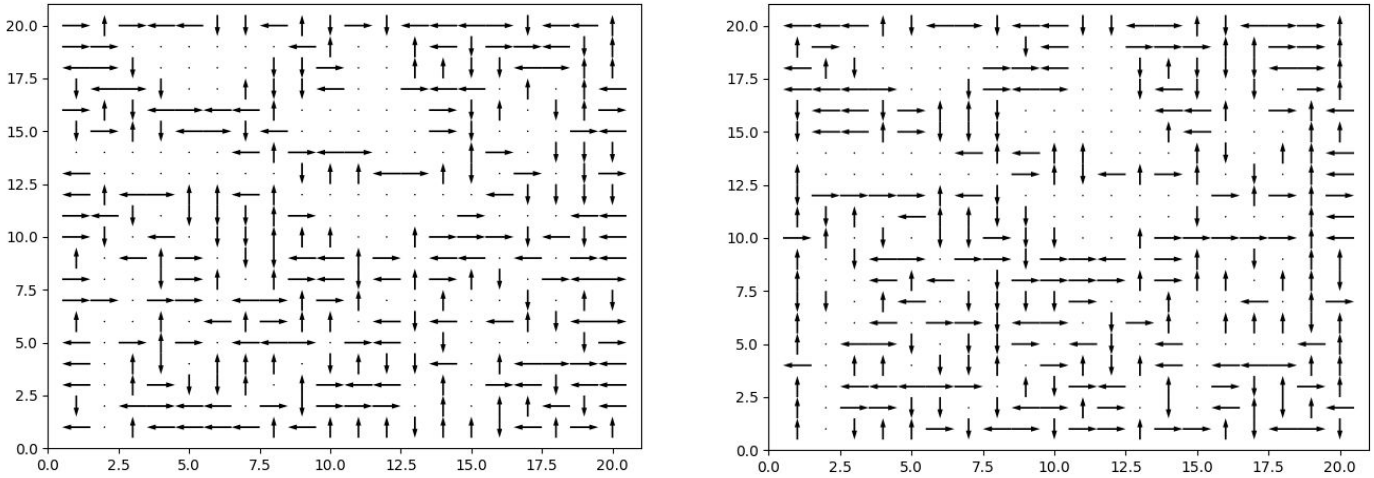


Figure 17. The final solution obtained by Q-learning on probem II.

Conclusion and Discussion

Based on the assignment requirement, here is the checklist for all tasks:

- 1. Present two MDP problems:
    - I presented one simple task and one complex task (Fig1).
- 2. Why they are interesting:
    - I designed the grid with two different strategies: the simple task has two solutions and the searching space is small, the complex task has only one best solution but the searching space is large. By the way, I designed only binary grid to test the performance of value and policy iteration.
- 3. Solve each MDP with **value iteration and policy iteration,** How many iterations to converge:
    - 5 for problem I, 10 for problem II. (Fig 2 and Fig10)
- 4. Solve each MDP with **policy iteration**? How many iterations to converge:
    - 12 for problem I, and 38 for problem II. (Fig 2 and Fig10)
- 5. Which one is faster;
    - For both problems, policy iteration convergent faster, but takes more computation time. (Fig3 and Fig 11)
- 6. Do they give the same answer;
    - Yes for both problems. Specifically for the two paths in problem I, I found an interesting pattern from the final solution. (Fig 5 and Fig13)
- 7. Does the number of states affects results;
    - Yes, I designed a simple case here and policy function always have a better performance than value
- 8. Use another favorite algorithm to solve two MDP:
    - Q-learning with time, convergence, reward and final solutions (Fig6-9, Fig14-17)
- 9. How does it perform, especially in comparison to the cases above where you knew the model, rewards, and so on?
    - In general, Q-learning has a worse performance than both value and policy methods. Specifically for problem I, Q-learing can acquire a similar result when L=0.1 and E=0.5. For problem II, the best L is 0.5 and E has no difference among three (0.1, 0.3,0.5)
- 10 What exploration strategies did you choose?
    - E = 0.5 seems a better strategy for both problems. (Fig 6, Fig 14).
- 11 Did some work better than others?
    - Policy iteration seems a better strategy for my designed grid since it's a simple state grid. The problem for policy is it may take a longer time to find a better strategy. I am wondering is it possible to apply value iteration to convergence at a certain point, then apply the policy iteration, which can combine both advantages together.