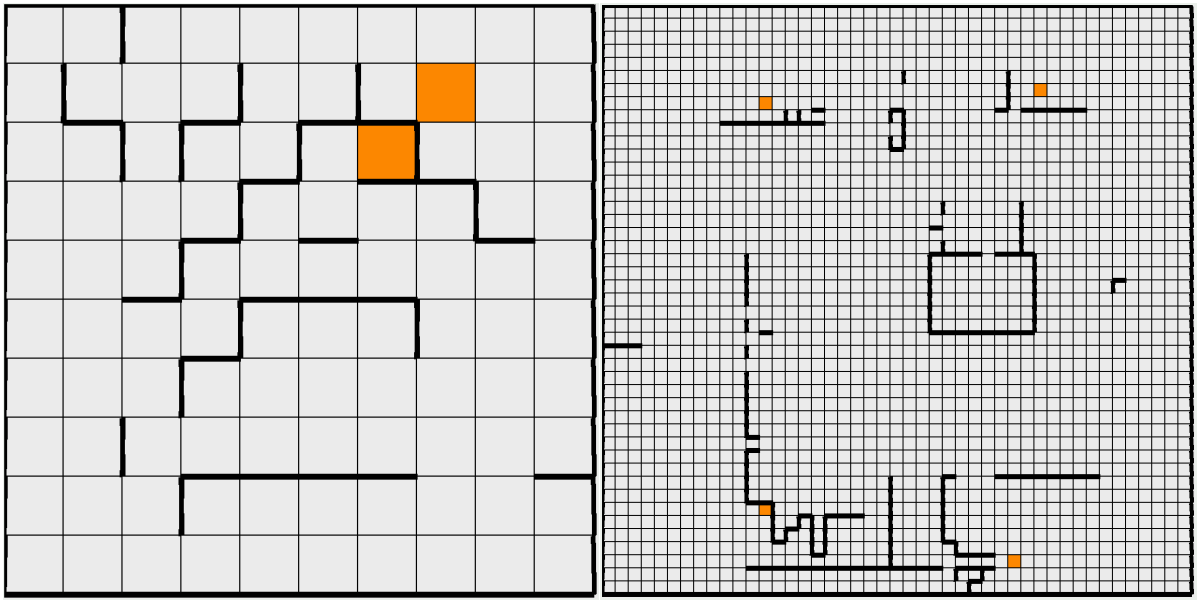
Min Ho Lee
#903122265
mlee432@gatech.edu

# MDPs and Reinforcement Learning Analysis

## Introduction

In this assignment, the main focus is to investigate Markov Decision Processes (MDPs) and Reinforcement Leaning. For all the experiments performed for this assignment's analysis, I used RL_Sim, Carnegie Mellon University's Reinforcement Learning simulator.

For the MDPs for this experiment, I used two mazes. The first maze (*maze 1*) is 10 x 10 maze which has 100 states total with 2 goals, and the second maze (*maze 2*) is 45 x 45 which has total 2,025 states total with 4 goals.



*Maze 1* (10 x 10) *Maze 2* (45 x 45)

I think these MDPs are interesting because it is very similar as the game *Pac-Man* and related how simple robotics works such as robotic vacuums. In this grid world, an agent will try to find the optimal path from a start location to the goal by moving in any four directions (up, down, right, left), which is the same concept as *Pac-Man*. If the agent attempts an action moving toward a wall, it will stay in the current state where it was and there will -50 penalty

on the reward. It will end when the agent reaches to one of the goals (grids marked with orange color).

**Value Iteration vs Policy Iteration**

In the simulator, there is PJOG parameter which is probability that the agent will not do the originally intended action, and the MDPs were tested with several PJOG values in order to compare results from experiments.

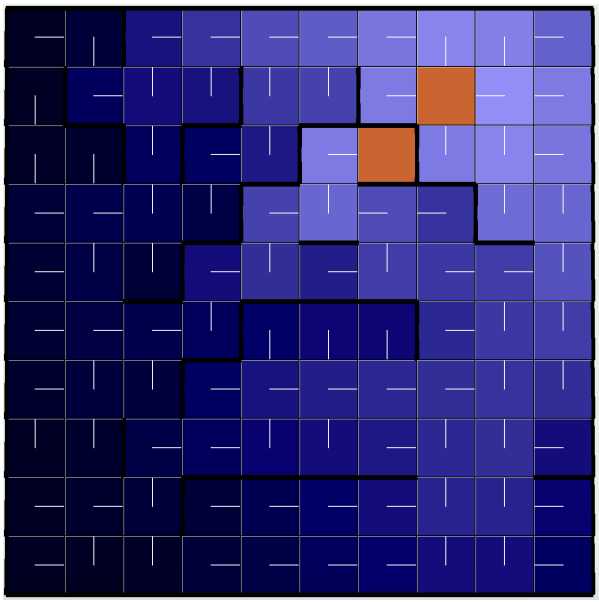
| Maze 1 | | | | |
|---|---|---|---|---|
| | Value Iteration | | Policy Iteration | |
| PJOG | Performed Iteration | Time Taken (ms) | Performed Iteration | Time Take (ms) |
| 0.1 | 84 | 39 | 6 | 67 |
| 0.2 | 62 | 47 | 6 | 62 |
| 0.3 | 84 | 37 | 7 | 40 |
| 0.4 | 115 | 58 | 7 | 50 |
| **Maze 2** | | | | |
| | Value Iteration | | Policy Iteration | |
| PJOG | Performed Iteration | Time Taken (ms) | Performed Iteration | Time Take (ms) |
| 0.1 | 50 | 903 | 19 | 5829 |
| 0.2 | 72 | 1362 | 17 | 6087 |
| 0.3 | 104 | 1850 | 16 | 6233 |
| 0.4 | 162 | 2888 | 15 | 6532 |

Result table for Value Iteration vs Policy Iteration

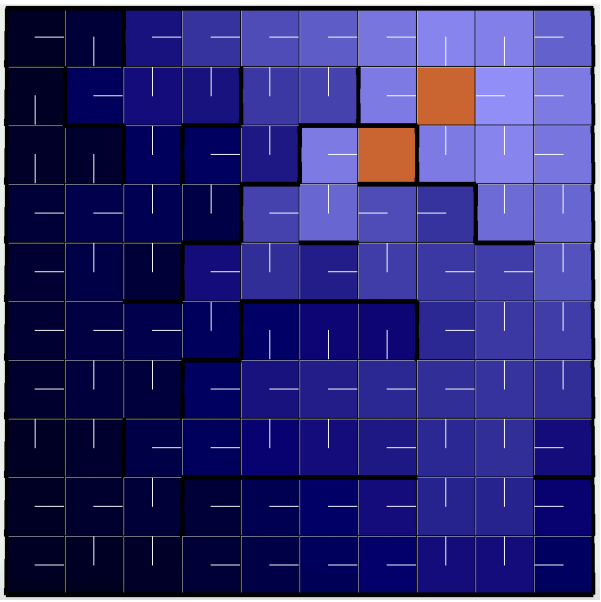
As we can see from the table above, we can observe that value iteration takes distinguishably more iteration to converge than policy iteration. This is expected result since there two algorithms have different termination condition. Value iteration will keep iterate through if there is any change on utilities $U$. It will continue until there is no change on $U$ within a given tolerance. Thus, it would make value iteration to keep iterate through even though there would be no more change on the actual action taken by the agent, and it makes value iteration takes a lot more iteration to converge. On the other hand, policy iteration continues only until there is no change on the policy, meaning it terminates if there is no change on the actual action taken by the agent. This makes policy iteration need a lot fewer iteration needed to converge.

However, in the time respect, policy iteration takes a lot longer than value iteration. This is because value iteration applies the optimal Bellman operator recursively to the value function and so it converges to the optimal value, but policy iteration applies the Bellman operator for the current best policy recursively until it converges to the value function and then improve the current policy by taking the action that maximizes the value function for every state. Since policy iteration does more computation, it requires more time to get the result.

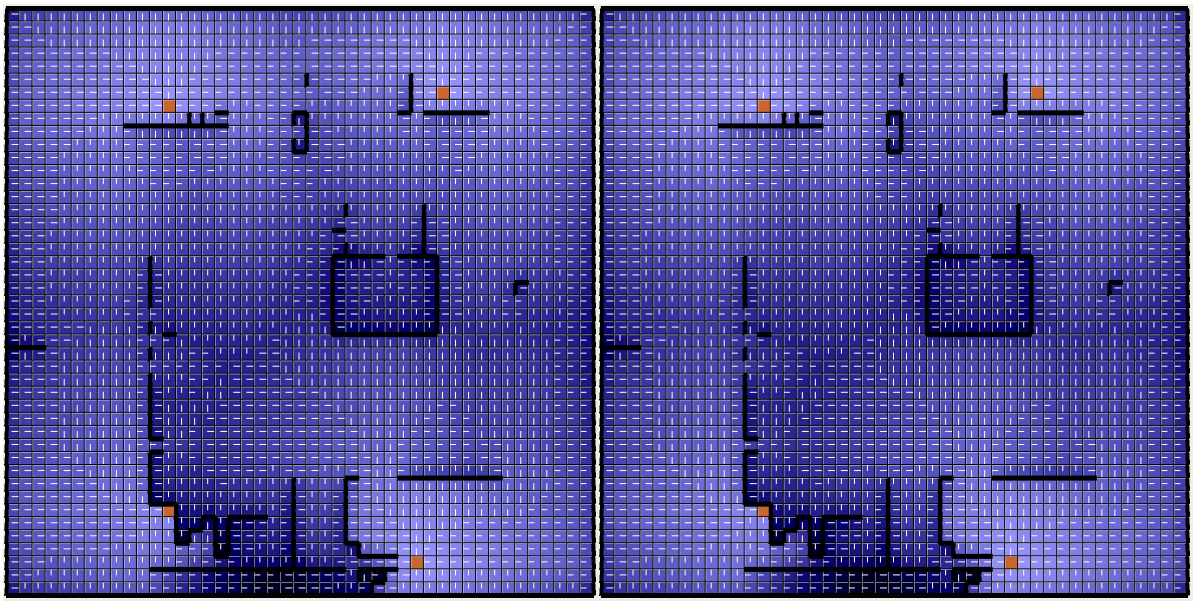
In PJOG respect, the number of iteration to converge increased as the value of PJOG increased. This is obvious because PJOG is probability that the agent will not do the intended action. When the value of PJOG is close to 0, the agent does not have to be afraid of the penalty coming from hitting a wall. Thus, it does not need to make unnecessary moves such as take detours in order to go farther from walls.



Value Iteration for *maze 1* (PJOG = 0.2)



Value Iteration for *maze 1* (PJOG = 0.2)

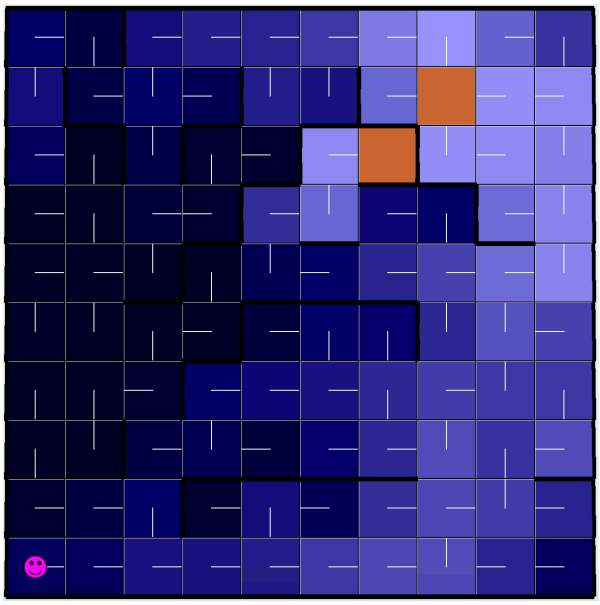


Policy Iteration for *maze 2* (PJOG = 0.2) Policy Iteration for *maze 2* (PJOG = 0.2)

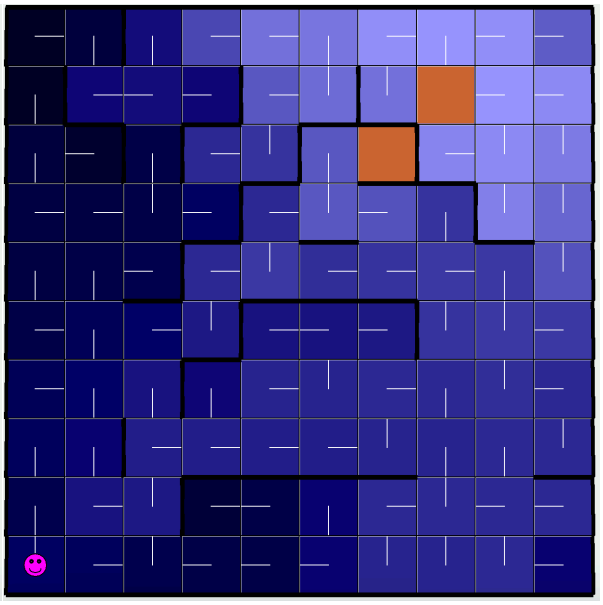
**Q – Learning**

I chose Q-Learning for my favorite Reinforcement Learning because the agent does not need prior unlike value iteration and policy iteration which we assume the agent knows the transition function and reward for every single state. Q-Learning only needs a set of states and the possible actions from each state to another. Since in real world it is hard to give the transition function and reward for every state, it might be more practical algorithm. Also, I think it will explain the best about the exploration – exploitation dilemma in Reinforcement Learning.

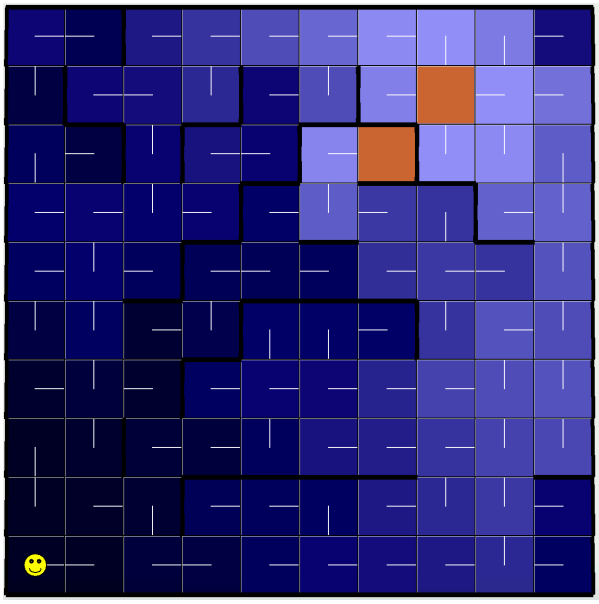
For this experiment, decaying learning rate of 0.7 and PJOG = 0.2 was used to make it easier to compare with value iteration and policy iteration since both of the result figures are with PJOG = 0.2. The number of iteration for *maze 1* was 1,000 and for *maze* 2 was 100,000. Different *Epsilon* values have been tried for this experiment so that exploration – exploitation trade-off can visible.



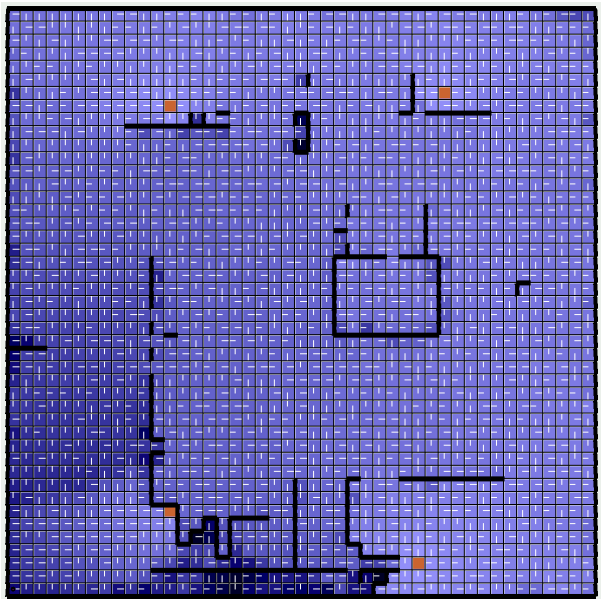
Q-Learning for *maze 1* (*Epsilon = 0.1*)



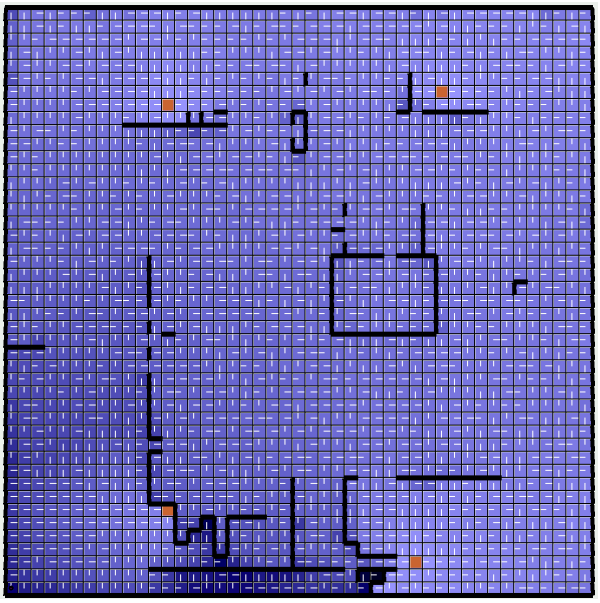
Q-Learning for *maze 1* (*Epsilon = 0.5*)



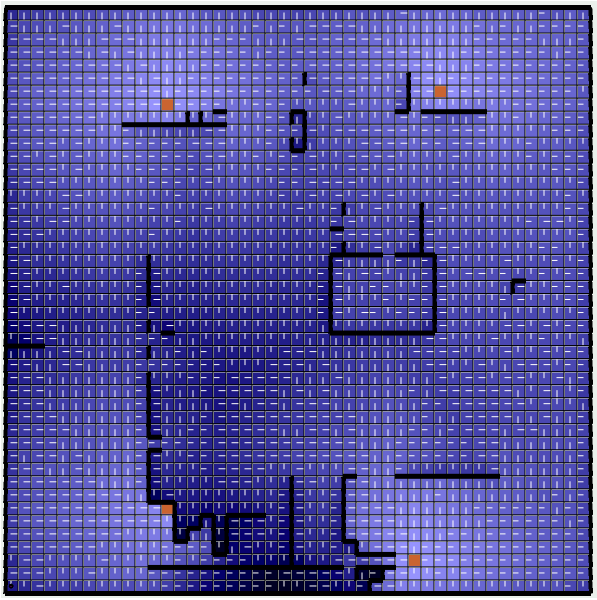
Q-Learning for *maze 1* (*Epsilon = 0.9*)



Q-Learning for *maze 2* (*Epsilon = 0.1*)



Q-Learning for *maze 2* (*Epsilon = 0.5*)



Q-Learning for *maze 1* (*Epsilon = 0.9*)

The experiment for Q-Learning does not seem to perform as good as value iteration or policy iteration even though it needed a lot more iteration and time taken than those. As we can see from the figures above, the action the agent had taken does not seem to organized. However, this result was expected since Q-Learning does not need prior knowledge of the state space nor transition model. Thus, we can say it showed good performance since it returned good policy.

For *maze 1* the time taken to converge was reasonably short since it does not have too much states to explore, but for *maze 2* the converging time was incredibly high because there are 20 more times of states which are in the maze. Thus, it takes much longer time because it needs to explore more. When the *epsilon* value was 0.1, it looks for the closest goal and try to reach there. However, when the goal is not close to where the agent is, as we can see from the figures above, it seems like the agent had wandered around not heading toward any goals. It is because it greedily looked for goals near the agent. When 0.9 was given as *epsilon* value, we can see the optimal policy on the figure. It is because instead of satisfying when it reached one of the goal, it explores more and more to look for potential better policy. It took significantly longer time to converge because of the reason, but it returned a lot better result. Thus, the figure looks very similar as value iteration and policy iteration which are with the given prior knowledges. We can see exploration – exploitation dilemma in Reinforcement Learning here.

**Conclusion**

We have so far seen the three algorithms dealing with MDPs, *maze 1* and *maze 2.* Value iteration had shorter converging time over policy iteration while policy iteration had a benefit on the number of iteration to converge. Q-learning required a lot longer time to converge, but it is because it does not need prior knowledge. We could observe the increase of optimality as the value of *epsilon* increases, but it increased the time to converge significantly. It is because the exploration vs exploitation dilemma in Reinforcement Learning. Exploration may lead to the agent taking less optimal actions because it wanders around even though it reaches to a goal once, but it could find more optimal solutions if the number of iteration is big enough. Exploitation causes the agent to utilize actions which can produce good results, but also it could cause to converge to only local optima because it just satisfies with the reward it found and try to stay there instead of looking for better rewards.