CS 4641 Project 4 Report

HU Heng

April 5, 2019

1 Overview

This report is for CS 4641 Machine Learning project 4 Markov Decision Process. In the following pages, you will see the solution to the chosen Markov decision process using value iteration, policy iteration and Q learning. I will show the convergence of value iteration and policy iteration as well as the influence of changing PJOG values. Finally, Q learning is discussed.

2 My MDP Problems

In this project, I use the RL_sim which is the java executable created by Carnegie Mellon University associates Rohit Kelkar and Vivek Mehta. The problems I would like to solve are both mazes given by the author. Fig.1 is a very simple 6x9 maze and Fig.2 is a more complicated 10x10 maze.

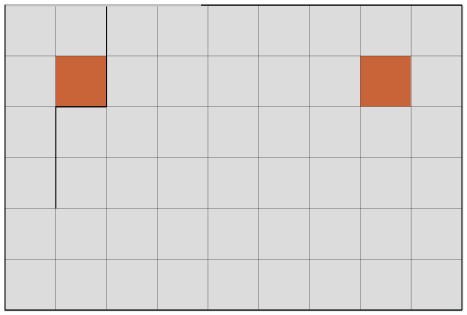

Figure 1: Simple Maze

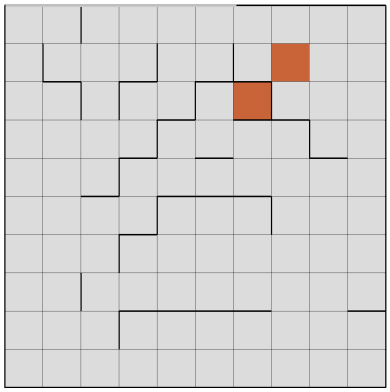

Figure 2: Complex Maze

In the figures, the highlighted grids are the destinations and the bold lines are walls and the agent cannot pass the wall. The possible actions for the agent are moving up, down, right or left and each action will have some constant cost. The parameter PJOG is the probability that the agent will act ineffectively. i.e. the agent will have 1-PJOG probability that its action will lead to the destination and PJOG/3 probability to choose other directions. Also, there would be some penalty if the agent tries to go into a wall.

I believe these two problems are interesting. The first reason is that they can be mapped to some real life problems. For example, a robot has to go back to its charging position when its battery is low. Another reason is that I believe the problems can show the performance of different algorithms as well as help the selection of algorithms in different situations.

3 Value Iteration

For value iteration, it works by finding the optimal value function which can represent how good is a state for the agent. It will apply the optimal Bellman operator to the value function in a recursive manner and converge to the optimal value. Then, we can get the corresponding optimal policy for every state. 1 shows the time and steps for value iteration to converge for different selection of PJOG value. Fig.3 to Fig.5 shows the result of small maze and Fig.6 to Fig.8 shows the result of large maze.

Table 1: Performance of Value Iteration

PJOG	**Small_Iterations**	**Small_Time(ms)**	**Large_Iterations**	**Large_Time(ms)**
0.1	24	4	42	40
0.3	48	19	84	65
0.5	123	23	181	143

From the experiments, we can see that the increase of PJOG will lead to longer run time and steps for value iteration to converge. This is within my expectation because large value of PJOG means the agent will have higher probability to do ineffective actions, that is, a higher chance to waste time. Also, the time and steps increase when we change from small maze to large maze, which is a very trivial result.

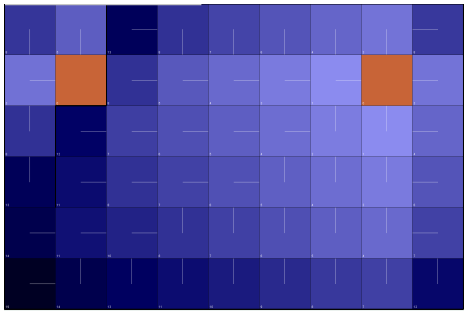

Figure 3: Small Maze, Value Iteration, PJOG=0.1

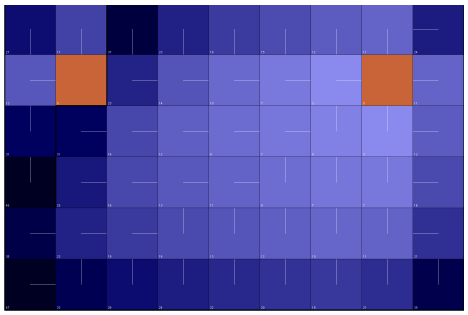

Figure 4: Small Maze, Value Iteration, PJOG=0.3

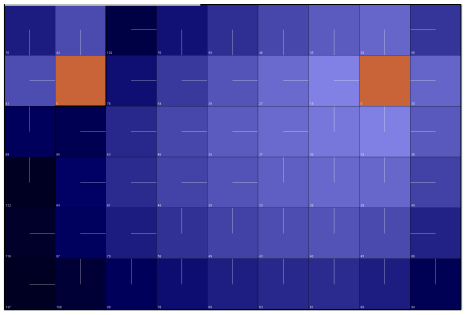

Figure 5: Small Maze, Value Iteration, PJOG=0.5

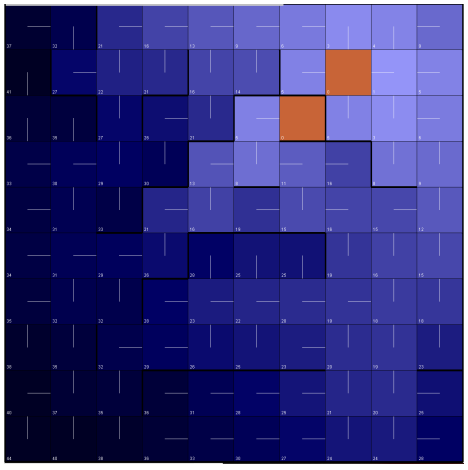

Figure 6: Large Maze, Value Iteration, PJOG=0.1

4 Policy Iteration

For policy iteration, it performs two steps in each iteration. First is policy evaluation, it is going to apply the Bellman operator for the current best policy in a recursive manner until convergence to the value function for the policy. Then, it will do policy improvement by taking the action that maximizes the value function for every state. 2 shows the time and steps for value iteration to converge for different selection of PJOG value. Fig.9 to Fig.11 shows the result of small maze and Fig.12 to Fig.14 shows the result of large maze.

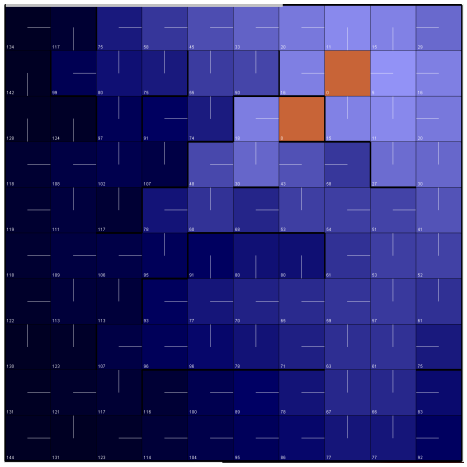

Figure 7: Large Maze, Value Iteration, PJOG=0.3

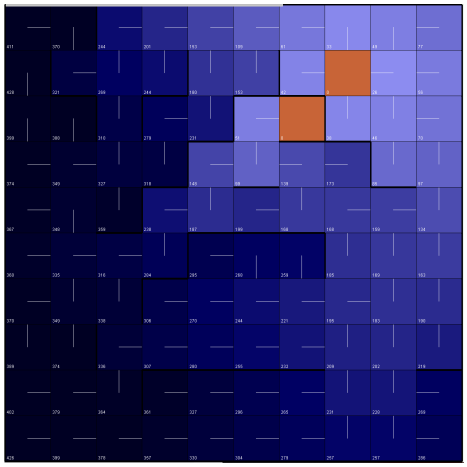

Figure 8: Large Maze, Value Iteration, PJOG=0.5

Table 2: Performance of Policy Iteration

PJOG	**Small_Iterations**	**Small_Time(ms)**	**Large_Iterations**	**Large_Time(ms)**
0.1	5	40	6	92
0.3	4	13	7	69
0.5	4	24	5	100

From the experiments, we can see that it always takes small iterations for policy iteration to converge. An interesting observation is that when PJOG is equal to 0.1, it will take much time to converge, compared with the case when PJOG is 0.3. For small maze, it even takes more time when PJOG is 0.5.

5 Comparison between Value Iteration and Policy Iteration

When we look at the results of value iteration and policy iteration, we can figure out that the results are the same. It means that both the two methods will converge to the same optimal value. Iteration steps vary a lot between value iteration and policy iteration. Policy iteration will take much smaller iterations steps to converge. This is because policy iteration is actually a

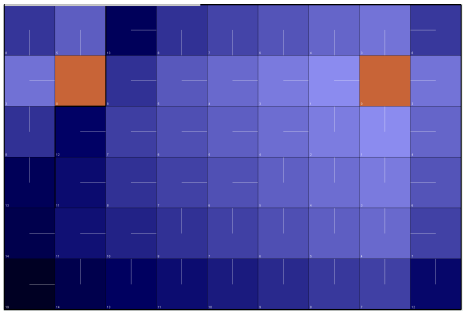

Figure 9: Small Maze, Policy Iteration, PJOG=0.1

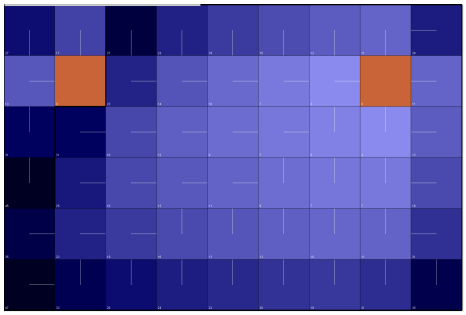

Figure 10: Small Maze, Policy Iteration, PJOG=0.3

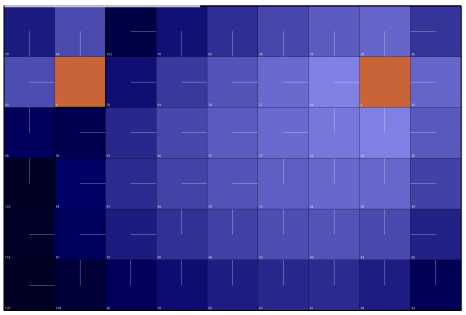

Figure 11: Small Maze, Policy Iteration, PJOG=0.5

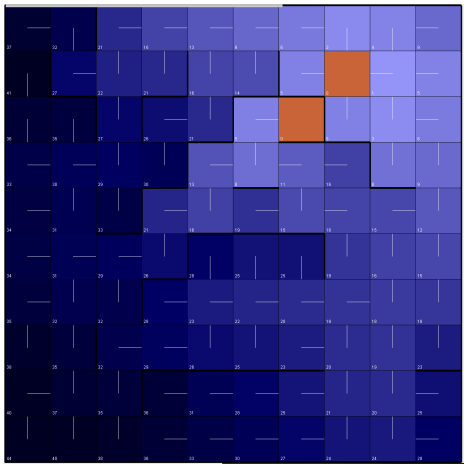

Figure 12: Large Maze, Policy Iteration, PJOG=0.1

kind of improvement of value iteration. Instead of waiting for convergence, we could now tolerate some error, evaluate the policy at some point and perform policy improvement over the not perfect value function. So one iteration in policy iteration is as powerful as multiple iterations in value iteration and the total iteration steps of policy iteration are consequently much smaller than that of value iteration.

As for the convergence time, for the large maze, it is clear that when PJOG is a small value, value iteration is faster than policy iteration while policy iteration takes less time when PJOG is large. For the small maze, value

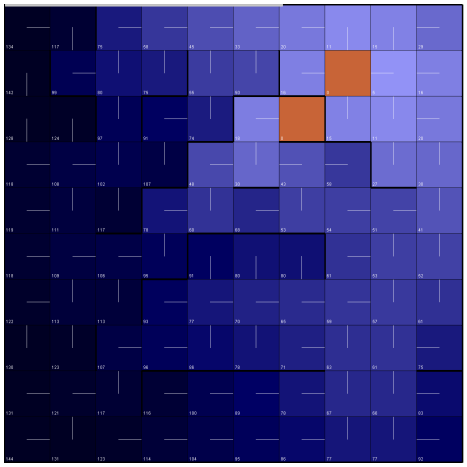

Figure 13: Large Maze, Policy Iteration, PJOG=0.3

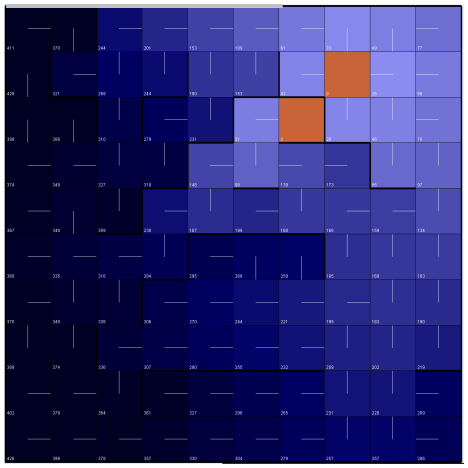

Figure 14: Large Maze, Policy Iteration, PJOG=0.5

iteration is smaller than policy iteration when PJOG is small and the convergence time become closer when PJOG increases. The observation indicates that policy iteration is a better choice when the problem is more complex and the agent cannot guarantee effect action. If the problem is simple or the action has a higher probability to be effective, value iteration may be quicker to converge.

6 Q Learning

Value iteration and policy iteration are both dynamic programming strategies, they can be used for offline planning where we make the assumption that the agent has prior knowledge about the effects of its actions on the environment. Now, assume the agent does not have prior knowledge of the system. i.e. the state transition and reward models are not known and the agent only know the possible states as well as actions. Q learning is an example of model-free learning algorithm, the agent will discover the environment and learn the transition and reward model. It works by estimating the vale of state-action pairs. The agent initialize the pairs using arbitrary values and by observing the reward and state transition of its action, it can update the estimation of state-action pairs. A problem here is how the agent select actions during learning. The approach used is ϵ-greedy approach where the agent will explore by picking random actions with probability ϵ or select an

action based on the current estimate of Q values with probability $1 - \epsilon$.

Fig.15 to Fig.18 are the results of running Q learning algorithm on small maze with PJOG=0.1, ϵ=0.1 after 10, 25, 50, 100 episodes correspondingly. Fig.19 to Fig.22 are the results of running Q learning algorithm on small maze with PJOG=0.1, ϵ=0.3 after 10, 25, 50, 100 episodes correspondingly.

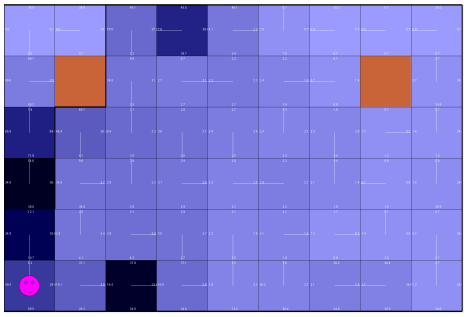

Figure 15: PJOG=0.1, ϵ=0.1, episode=10

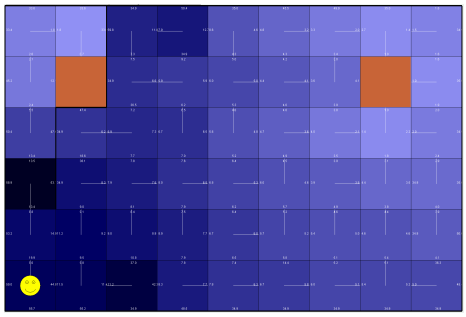

Figure 16: PJOG=0.1, ϵ=0.1, episode=25

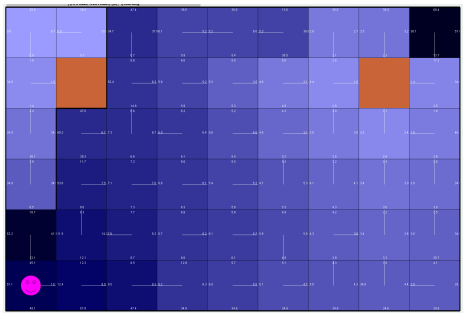

Figure 17: PJOG=0.1, ϵ=0.1, episode=50

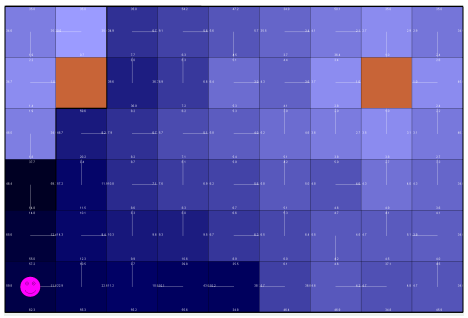

Figure 18: PJOG=0.1, ϵ=0.1, episode=100

Fig.23 to Fig.26 are the results of running Q learning algorithm on large maze with PJOG=0.1, ϵ=0.1 after 10, 25, 50, 100 episodes correspondingly. Fig.27 to Fig.30 are the results of running Q learning algorithm on large maze with PJOG=0.1, ϵ=0.3 after 10, 25, 50, 100 episodes correspondingly.

We can see that Q learning algorithm can produce some useful information quickly even when episode is small. When episode=10, all agents in four experiments have learned some effective actions. As mentioned above,

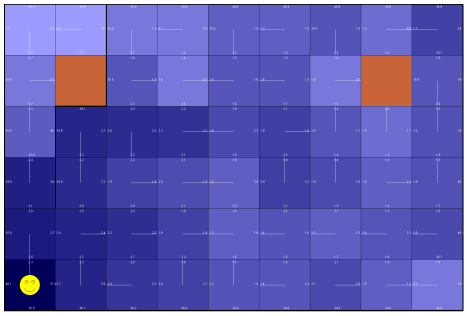

Figure 19: PJOG=0.1, ϵ=0.3, episode=10

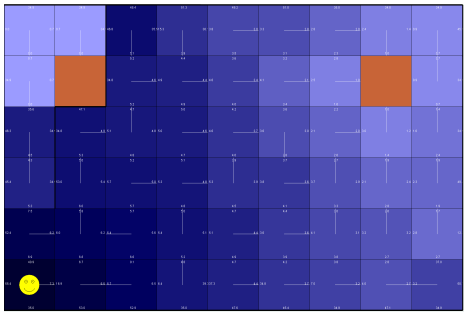

Figure 20: PJOG=0.1, ϵ=0.3, episode=25

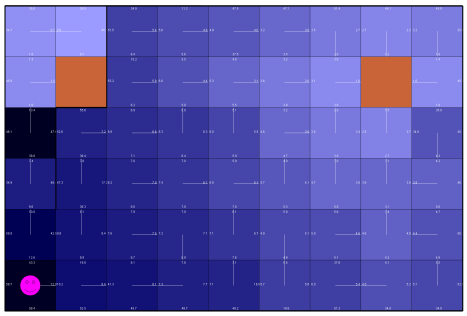

Figure 21: PJOG=0.1, ϵ=0.3, episode=50

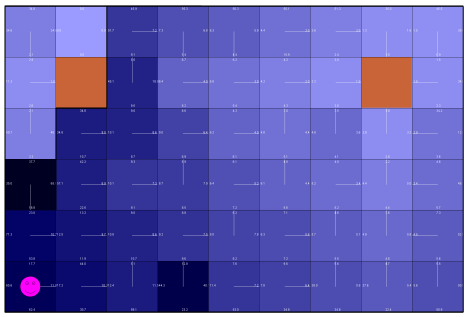

Figure 22: PJOG=0.1, ϵ=0.3, episode=100

Q learning assume that the agent knows nothing about the environment and will explore the world by itself, there are some non-optimal actions and the agent may even get stuck at a corner at first. But as episode increase, the actions are more close to the optimal solution, which indicates that the agent is learning more about actions and policies. In my experiment of using different ϵ value, I will pause every ten episode and see what happens for the agent. I find out that small ϵ(=0.1) leads to slow learning at very beginning but the speed increases as the environment is better explored. On the contrary, the exploring process of large ϵ(=0.3) at beginning is faster than small ϵ while it becomes slower later. This is because ϵ measures the "random walk" probability, when the agent does not know the world very much, it will fasten the exploration. But when the agent has enough information about the environment, it is regarded as negative influence.

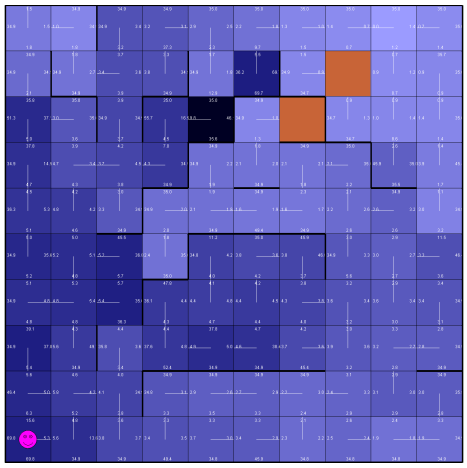

Figure 23: PJOG=0.1, ϵ=0.1, episode=10

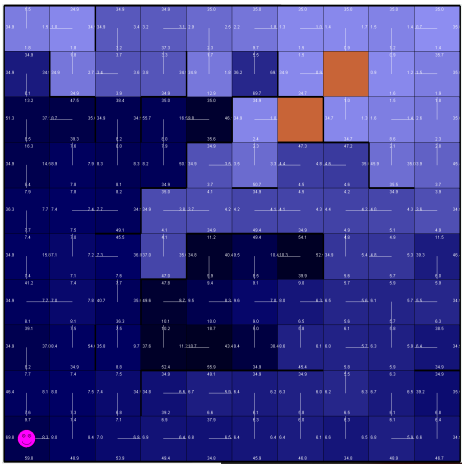

Figure 24: PJOG=0.1, ϵ=0.1, episode=25

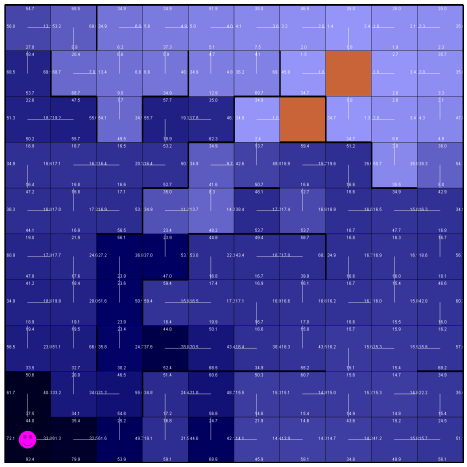

Figure 25: PJOG=0.1, ϵ=0.1, episode=50

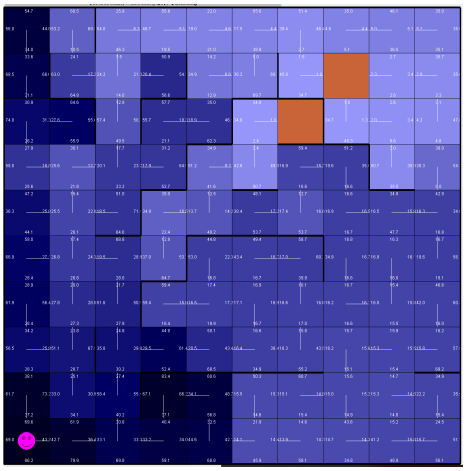

Figure 26: PJOG=0.1, ϵ=0.1, episode=100

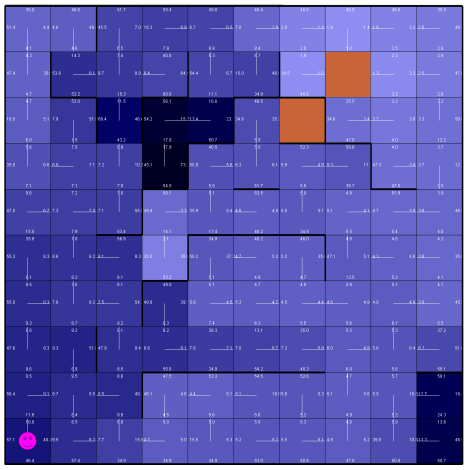

Figure 27: PJOG=0.1, ϵ=0.3, episode=10

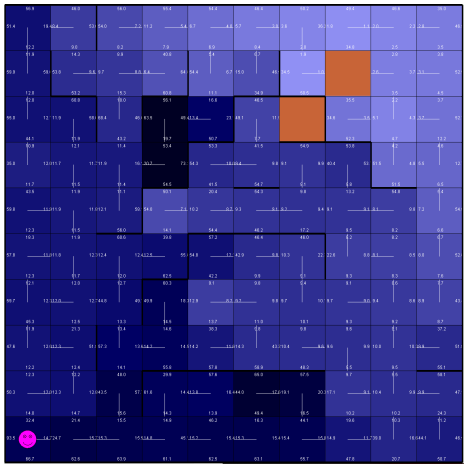

Figure 28: PJOG=0.1, ϵ=0.3, episode=25

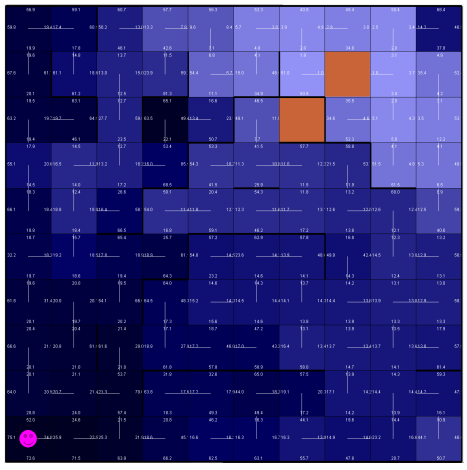

Figure 29: PJOG=0.1, ϵ=0.3, episode=50

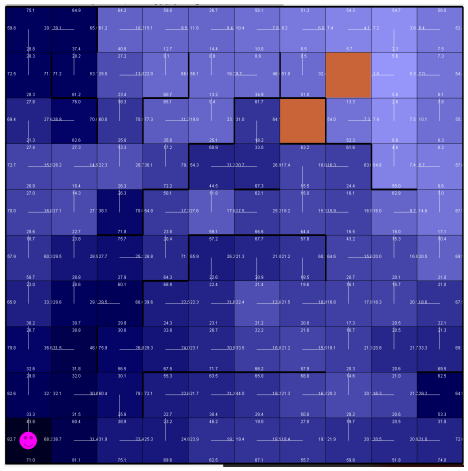

Figure 30: PJOG=0.1, ϵ=0.3, episode=100