CS 440 / ECE 448: Introduction to AI
MP 1

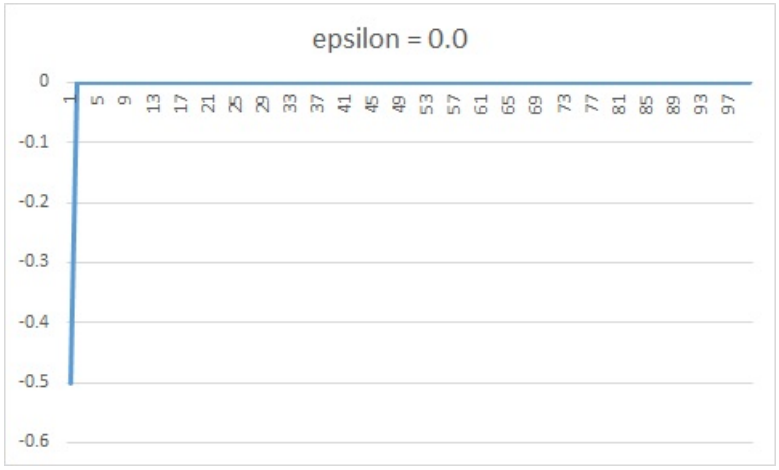
Ting-Jung, Chang

September 29, 2014

1.In **WorldWithoutThief**

(a) Set the discount factor to 0.9, the learning rate to 0.1, and the ϵ to 0.0.(which means your agent does not perform random actions except perhaps for ties). Simulate your agent for 1000 episodes of 10000 steps. Explain what you observe.

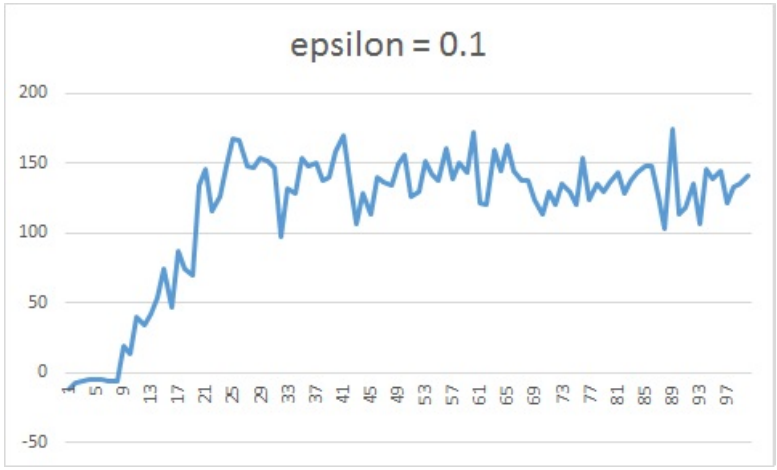
Figure 1: ϵ=0.0



From the Fig 1, we can soon observe that the reward would become 0 after the first episode, while the reward for the first episode would be more negative(such like -8 or -11). In the policy simulator, the robot would walk directly north, than stuck at the North-West corner.
In the first episode, since all Q-value are initialized to 0 first, the robot would randomly choose where to walk. It performs like a random agent in the very beginning. Since the ϵ is 0 and the Q-value is set to be negative once the robot drop due to the slipperiness, the robot has no chance to try different paths. It's nearly impossible for the robot to deliver both packages under this condition.
Looking at the map, we could find out that column three is filled with slipperiness. Since the robot seldom get the positive reward, the Q value for the east side might be negative. That is to say, the robot might get negative reward if it try to walk east after leaving the company. As a result, the robot always choose to go north and get stuck in the corner with reward equal to 0.

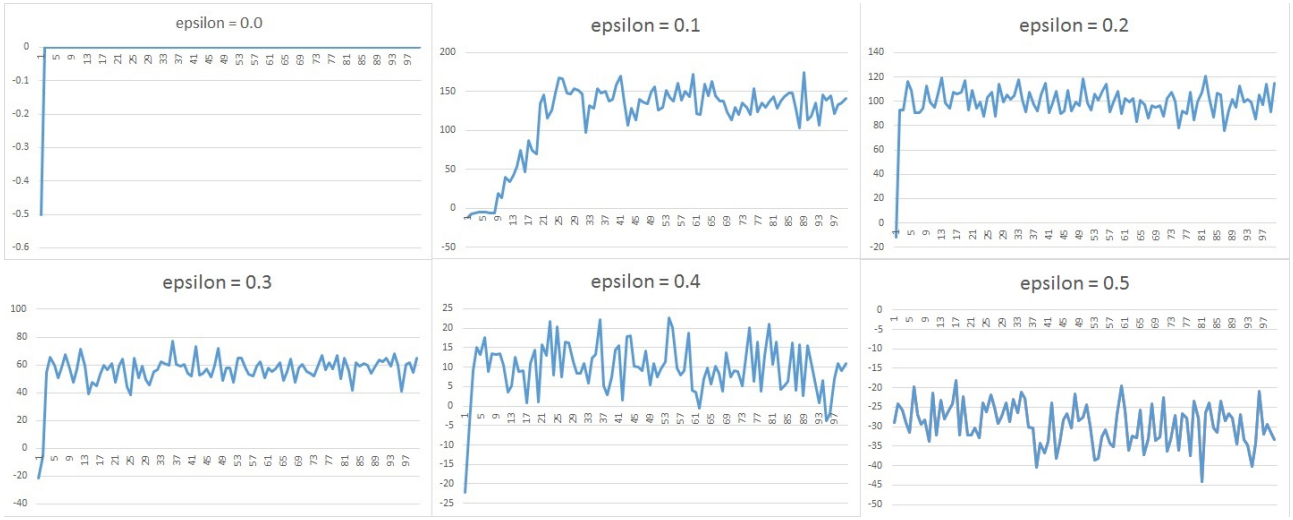
(b) Set ϵ as 0.1, and keep all the other settings. What did you observe? Explain it.
From the Fig 2, we can find the performance for the robot in this setting is much better than the previous one. The average reward for 1000 episodes is about 130. I use the average for every ten rewards to plot in order to have a more clear graph.
Compared with the setting in 1(a) ,since the ϵ here is set to be 0.1, the robot has more flexibility to try different path, instead of always following the greedy policy. The robot would have a higher chance to earn the reward and than follow the successful path afterwards.

Figure 2: ϵ=0.1



(c) Try larger ϵ(for example, ϵ= 0.5) and keep all the other settings. What did you observe? Explain it.

Figure 3: ϵ=0.0 to 0.5

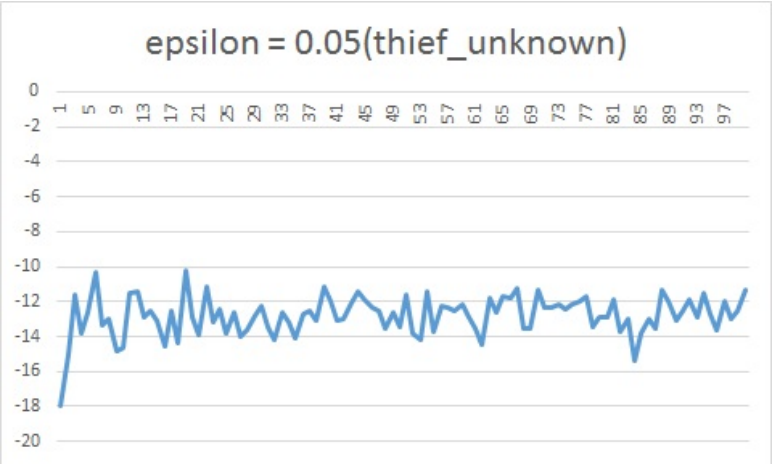


From the Fig 3, we could observe that the best performance occur at ϵ equal to 0.1. Then the performance decay with the larger ϵ. The average reward for ϵ 0.5 is around -29.
Though increasing ϵ from 0 to 1 improves the performance a lot, the agent would act more randomly as the ϵ increases. For ϵ equals to 0.5, the robot would act randomly in half of the time, but not follow the policy from Q-table. It would be harder for the robot to get positive reward. However, robot in this condition still has better performance than totally random one. The average reward for random robot is around -40.

2.In **WorldWithThief**

(a) Set ϵ = 0.05, and keep all the other settings. In the simulation command, set knows_thief as n. How is the performance of your agent? Explain it.
From the Fig 4, we could observe that the reward is negative in this condition, which is quite awful. The average reward is about -13. When we set knows_thief as n, the robot has no idea where the thief is. That is to say, the robot does not have different states corresponding to the position of the thief. Without corresponding state, it is hard for the robot to learn how to avoid

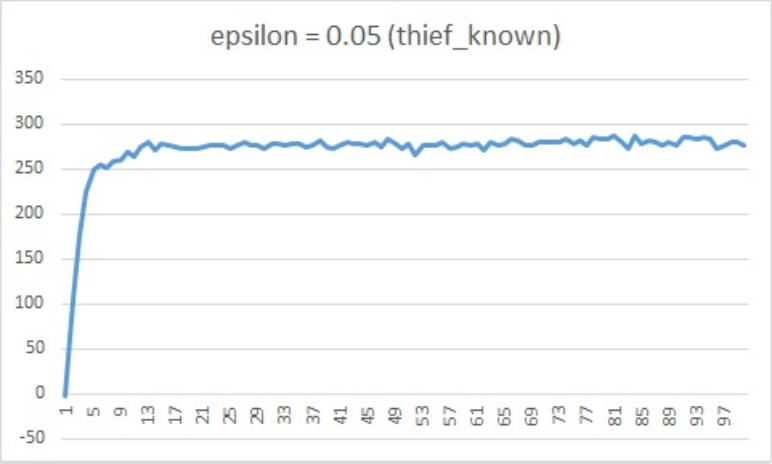
Figure 4: ϵ=0.05 thief_unknown



the thief, since the thief would move randomly along the column. Without knowing the position of the thief, robot could not find a optimal path to get more reward.

(b) Set knows_thief as y. Explain any performance difference.

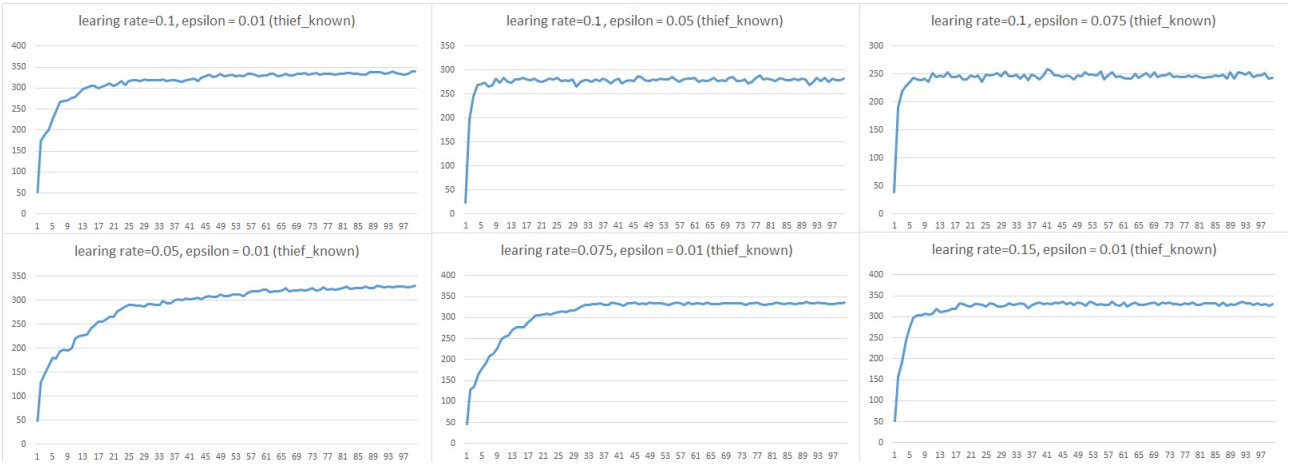
Figure 5: searching the best learning rate and ϵ



From the Fig 5, we could observe that performance is much better than 2(a). The average reward is about 275. When we set knows_thief as y, the program would store more states for the thief in different rows. That is to say, in addition to robot position and the amount of the packages, the sates also record the row of the thief. In this way, we could compute different Q-value for every different state and the robot would have more accurate action due to a more complexed Q-table with more states. With more states, it would be much more easier for the robot to avoid the thief.

(c) Search for the best learning rate and ϵ. Describe your investigation and conclusions.
By testing many different learning rate-ϵ pair, I found that the robot get highest average reward about 320 at learning rate=0.1 and ϵ=0.01.
The ϵ determines the possibility for the robot to act randomly. For the ϵ that is too small, the curve in the graph would rise slower in the beginning because the robot need more time take random action and learn the optimal path. The robot needs to take random action sometimes in order to learn faster and better since purely greedy behavior would prevent us from exploring potentially better alternatives. On the other hand, if the ϵ is too large, than the robot has a higher

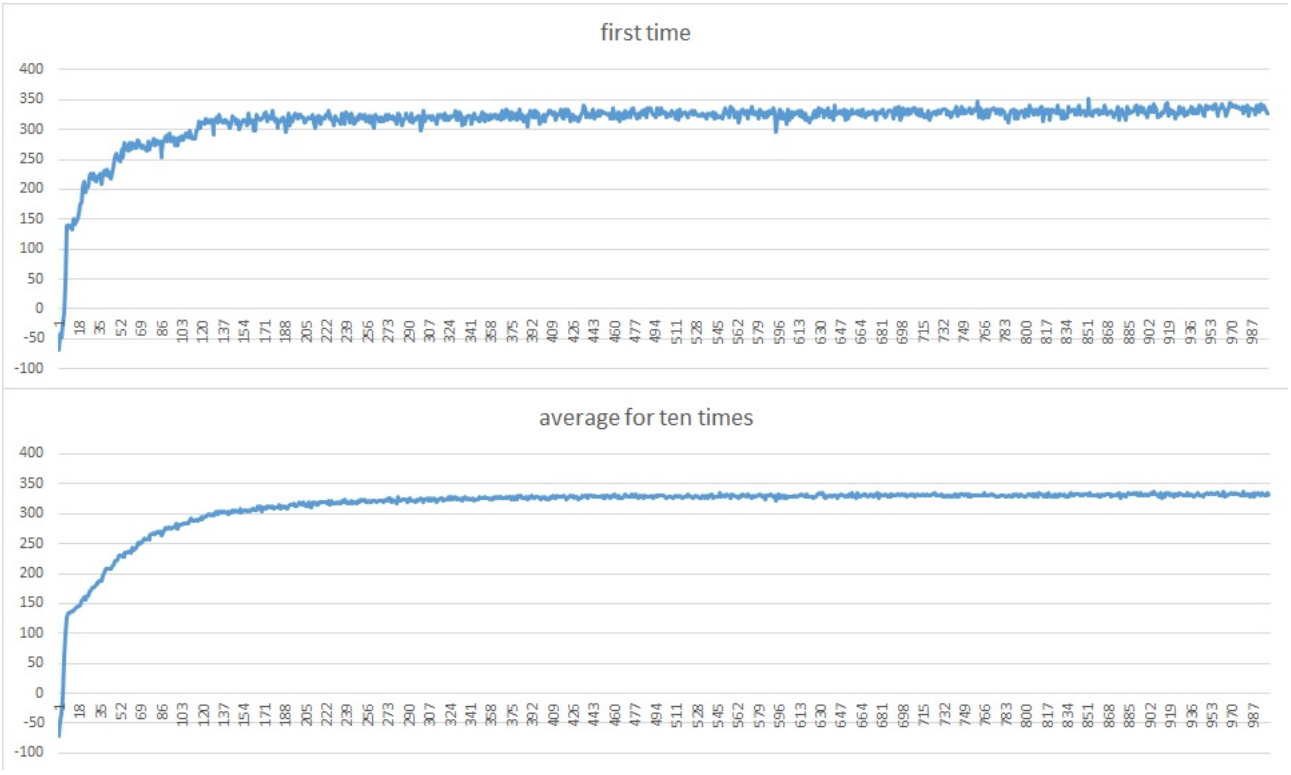
Figure 6: find the best



chance not to act due to the policy, and that could cause the drop in the reward.
The learning rate determines how fast does the robot learn. For the learning rate that is smaller, the curve in the graph would rise slower in the beginning because the robot need more time to learn. If the curve rises too slow, then the average reward would be small. For the learning rate that is larger, the curve rises faster. However, if the learning rate is too large, the robot would learn too fast that it could not actually find the optimal path.

3.Using your best parameters from 2c, simulate your agent for 1000 episodes of 10000 steps. Plot the expected discounted reward achieved at the end of each episode showing how the agent improves. What do you notice? Repeat the procedure nine more times and again plot the expected discounted reward at the end of each episode but now averaged over all ten simulations. Explain what you observe.

Figure 7: first time and average



The reward gained by the robot improved from about negative 70 to about positive 330 at the end. The reward enters a steady state after 120 episodes, then it increases in a quite slow manner. Though the agent may figure out the optimal policy after a few hundreds of episodes, it is still possible for the agent to move randomly and get caught by the thief which also moves randomly. I think this is one possible reason that causes the variation in the curve instead of getting the same reward in all episodes.
After averaging over all ten simulations, the curve becomes smoother. We can now observe the curve clearly. The average action make the variation in the curve more implicit.