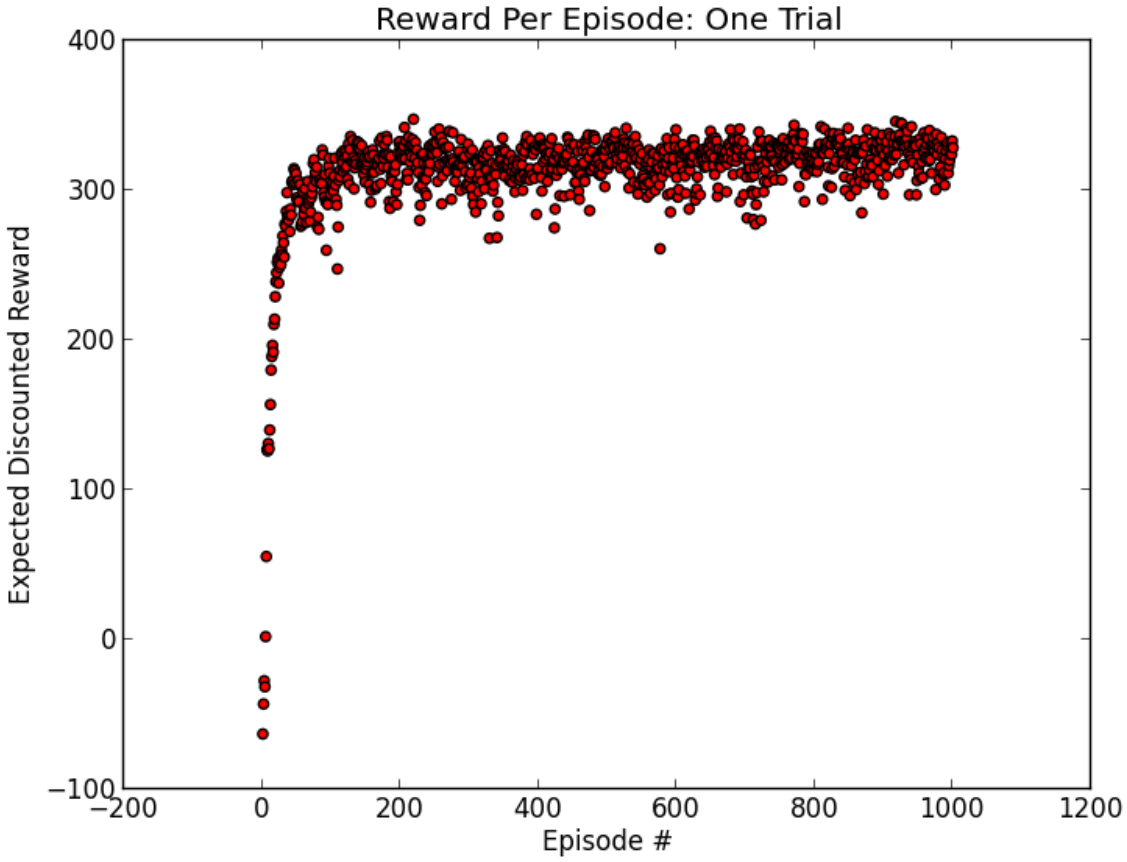
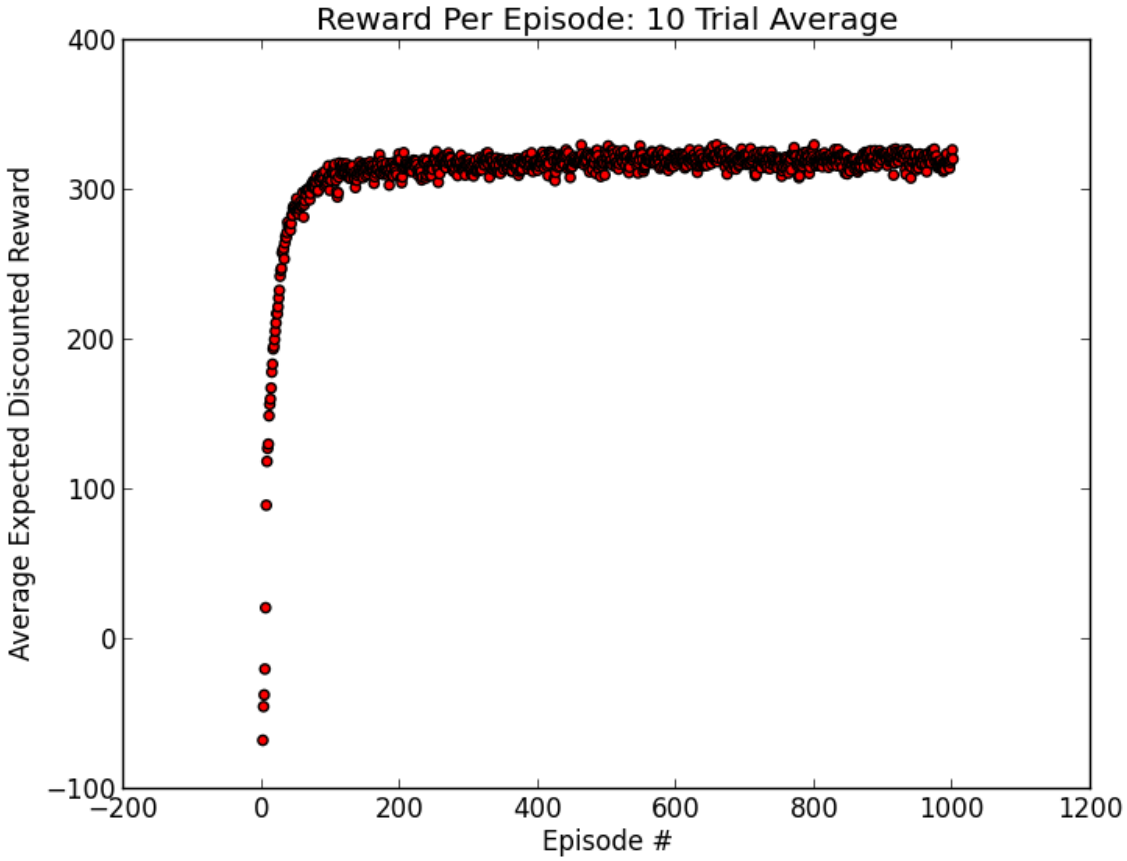
1. WorldWithoutThief
    a. Discount factor = 0.9, learning rate = 0.1, epsilon = 0.0 (1000 episodes of 10000 steps): When $\varepsilon = 0$, the robot's policy is just to keep moving up from the company. The robot's policy never tells it to step on a slippery square because that results in a negative reward. Thus, the robot just gets stuck in the northwest corner of the board. Setting $\varepsilon = 0$ does not allow the robot to explore spaces that could potentially result in a positive reward. In the first episode, the robot essentially is making random moves until he slips and breaks the packages. Once he slips, he will never visit that spot again because it results in a negative reward. Since the robot never explores and makes random moves through spots with potentially negative rewards, the robot can never deliver deliver both packages and return to the company.
    b. Same as (a) but now $\varepsilon = 0.1$: With $\varepsilon = 0.1$ the robot is able to explore random spots 10% of the time. This allows the robot to eventually find a positive reward. Unlike before, where the robot explored random spots only the first episode, this agent now goes to a random spot 10% of the time. In future episodes, the agent sometimes goes over slippery spots allowing him to potentially deliver the packages. This randomness allows the agent to eventually find that the best path is to deliver both packages and return to the store. Without this randomness, the robot would never step on the slippery spots.
    c. Same as (a) but now $\varepsilon = 0.5$: With $\varepsilon = 0.5$, the exploration probability is too high. The agent converges into a large region around π^*. The agent chooses a random move 50% of the time. The agent will be able to explore all regions of the board, but it will usually reach the goal state with a negative total reward. For example, say the robot is 1 spot away from a positive reward. The robot will only go to the positive reward 50% of the time. It will choose another spot the other 50% of the time. If it chooses another spot, then it will have to make two optimal moves to get to the reward. There is only a 25% chance of this. The high number of random moves makes the agent choose poor spots too often. This usually results in a bad policy at the end.
2. World With Thief
    a. Same settings as (1a) except $\varepsilon = 0.05$ and $knowsThief = n$: Since the agent is not sensitive to the thief's position it learns that it gets a negative reward a lot of the times when it goes into the third column. Thus, the policy tells the agent to never go to the third column, and the robot gets stuck in the northwest corner again.
    b. Same settings as (2a) except $knowsThief = y$: Now there are many more states, and the agent is aware of the thief's position. Thus, the agent can learn when it is safe to move into the third column based on the thief's position. We end up with a positive reward after 10000 steps because the agent is able to deliver both packages and avoid the thief.

c. **Observed Optimal** $Learning\ Rate = 0.3$**. Observed Optimal** $\varepsilon = 0.01$. I tested various epsilon ranging from 0.0 to 0.5. I noticed that lower epsilon values tended to produce higher average rewards than higher epsilon values. So, I tested very small epsilon values and found optimal to be $\varepsilon = 0.01$**.** I tested learning rate in the same way. I iterated through values of learning rate along with values of epsilon and got that 0.3 was the optimal learning rate for this agent. These values make sense because you do not want epsilon to be too high or else the agent is just going to be making dumb random guesses too much of the time. A learning rate of 0.3 makes sense because you want to take previous values into consideration, but the most current values matter more.

3.



As you can see, after about 200 episodes the expected reward converges to around 320. There are some episodes that have a lot less final reward, while there are also some episodes that have a higher total reward. This data has more variance than the next graph of the average of 10 runs.



Again, we see that the average reward per episode converges to around 320. However, this average plot has a lot less variance. We can more clearly see the value at which the reward converges to.