Colton Cannon - cjcanno2

MP1 Part 2

1. a.) The final reward of every episode is 0, except for the first episode, which is a negative reward. Running the policy simulator shows the robot traveling straight upwards until it reaches the corner and gets stuck. This is because the lack of randomness means the robot will always take the most optimal path, meaning as soon as any negative path is found, the robot will never try that route again, ultimately getting stuck as all other possible routes are deemed unsuitable.

b.) This time the total rewards of the first couple episodes are still negative as the robot is mostly just taking random paths, but they eventually become positive and reach as high as 250 in the later episodes. This is because with the epsilon set larger than 0, the robot has a chance to try any path and determine if it is optimal. Therefore, paths that were initially passed up or deemed sub-optimal that were actually successful paths can now be tried again and updated accordingly. Over a lot of iterations, this means that a fairly optimal path will be found, leading to a high total reward. However, the introduction of more randomness means that there is still a chance the path taken will not accrue a high total reward, as the total rewards for the episodes get as low as 30 even in the later episodes.

c.) With epsilon set to .5, the total rewards for each episode are almost always negative. This is because the robot now diverges wildly from the optimal path it calculated. In the previous part, the robot would stick to the path with the highest Q value 90%, but now this chance is only 50%. This means that over the many steps in each episode, the robot eventually gains a very high chance to take a random path that either got the robot stuck, minimizing point gain, or having it going over slippery patches, increasing the number of packages dropped and thus increasing the points lost. While some amount of randomness is needed to ensure success, too much will cause the robot to take paths with very low Q values too often to be considered successful.

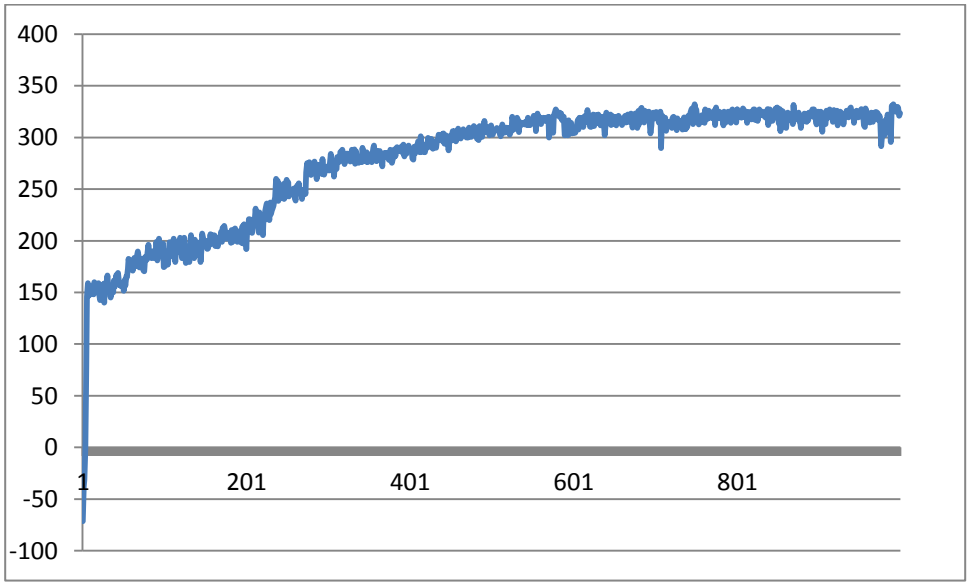
2. a.) The rewards almost exclusively negative. Running the simulator shows the robot will eventually just avoid delivering packages altogether, instead sitting in a corner as it loses too many to the thief to consider any attempts at actually delivering, since the penalty from the thief is too high. This is because the column that the thief travels in has no slippery patches, therefore in a world without a thief, the robot would take that column as it minimizes risk of dropping the packages. Because it would spend so much time in that column, the robot would run into the thief often, incurring a very large penalty, so eventually the robot decides to stay out of the thief's way entirely.

b.) If the robot knows where the thief is, the rewards become much higher, and eventually converge to a number in the high 200s for the given settings. This is because the robot can now deliver packages effectively as it can avoid the thief. The high convergence arise from an observation in the last part that the map has a path from the company to the delivery sites that does not pass over any slippery patches, therefore the amount of randomness is severely diminished, so each episode will end with a similar reward, varying only based on when the thief makes the robot wait.

c.) Through experimentation, I found that lowering the epsilon value will have it converge better and to a higher total reward, though it takes longer to reach this higher value than higher epsilons. This is

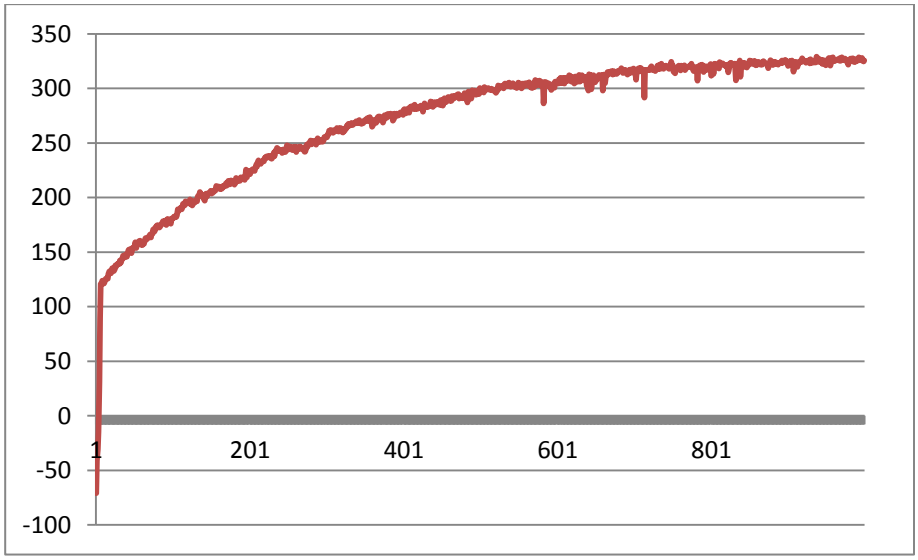
because with a lower epsilon value the robot will explore for better paths less, but also once it founds the best path, will diverge from it much less often. For the learning rate, I found that increasing it makes the robot learn the best route faster, but increasing it too high causes a lot of fluctuations in the total reward, as little changes in reward cause too large of changes in the Q value causing the robot to try new paths too often and take longer to settle on the best path. Overall, I found an epsilon of around .001 and learning rate of around .2 to give the best results.

3.) For the single simulation plot, the results were as follows:



The x-axis represents the episode, while the y-axis represents the total reward for that episode. The graph starts very negative as the robot is essentially taken random paths the first few episodes but very quickly the robot finds a good solution, and then overtime due to the epsilon factor learns about other better paths. The epsilon factor also injects some randomness into the solution, leading to a thicker line and a less regular shape.

For the average of 10 simulations, I got:



Like the first graph, it starts poor, becomes good very fast, and then slowly becomes better. However, unlike the first graph, the line is a lot smoother, and the general shape of the graph is more regular. This is because of the law of large numbers, where the random outliers of each individual simulation slowly cancel and the total reward converges on the actual expected.