Greg Atherton
gatherto@illinois.edu
CS 440
MP 1 Part 2

1. a. The resulting policy simply travels north, never trying to cross the slippery path. The reason for this is, in the absence of any random experimentation, the robot will only choose the path it has experienced the best. In this case, it got 0 reward for going north, which is not worse than any other path, and due to the way it resolves ties it simply has no reason to change.

b. Around episode 16, the agent learned an optimal path. It travels up so that it crosses the low-slip portion of the midline to get to client 2, then crosses back over the same low-slip portion, north further to the other low-slip portion, and then to client 1. After episode 16, the reward was largely unchanged through episode 1000. However, it never learned to go back and get more packages.

c. With epsilon of 0.5, the robot did not learn at all. It showed no preference for path, and the final policy was simply to sit where it was. The reason for this is, with a 50% chance of taking a random action each time, and a significant number of sequential actions necessary to receive a reward at all, there is no real connection between policy and reward. When the robot can't link the policy to the reward, it simply can't learn.

2. a. The robot simply moves north until it hits a wall, and stays there. The likely reason for this is the presence of the thief. When the robot can't know where the thief is, there is a 20% chance whenever it crosses the midline of getting a negative reward. As it learns with a relatively low chance of acting randomly, it is unlikely to “stumble” onto the idea of crossing once and then braving the high-slip area and avoiding the thief (likely the best policy in an unknown thief scenario – depending on penalty, the other option is that the risk is too high and the best policy really is the 0 reward for doing nothing). This is especially true as early deviations from what ends up as the final policy are almost invariably penalized (the robot is more likely to simply sit in the path of the thief, as it doesn't know to keep moving) so the robot is disincentivized to keep trying.

b. The robot now performs optimally. It avoids the thief, delivers the packages, then returns and gets more. The reason for this is the separation of states. Whereas before, the robot would be seemingly arbitrarily punished for actions from states near the midline, now it sees states with the thief close by as entirely different than states with the thief far away. Now, it knows moving over the line with the thief close is risky, whereas doing so with the thief further is safe. It starts to get rewards around the 8th episode, and its reward steadily increases until around the 80th episode, where it seems to have learned its optimal policy.

c. My initial goal was to find the correlation between the inputs epsilon and rate and the outputs learning speed and final reward. I know initially that increasing the rate increases learning speed but decreases final reward, but I do not know exactly how epsilon will affect it. My numbers are tabulated as follows (E_i is the episode where the reward shows notable improvement, Ef is the episode where the reward reaches its steady state value, and the final reward is the average over the last 20 samples, rounded to the nearest 10):

Rate	**Epsilon**	**E_i**	**E_f**	**Final Reward**
0.05	0.03	4	237	300

0.05	0.05	5	67	280
0.05	0.1	7	125	220
0.05	0.3	1000	1000	-30
0.1	0.03	4	207	310
0.1	0.05	8	80	280
0.1	0.1	6	61	220
0.1	0.3	1000	1000	-10
0.3	0.03	2	51	300
0.3	0.05	5	49	270
0.3	0.1	5	44	200
0.3	0.3	1000	1000	0
0.5	0.03	3	32	260
0.5	0.05	4	11	210
0.5	0.1	6	18	180
0.5	0.3	1000	1000	-20

From the table, it's clear the optimal (from the perspective of the best final policy) epsilon is around 0.03 and the best rate is around 0.1. I will look at more values near these. As a note, reward is now rounded to the nearest 5.

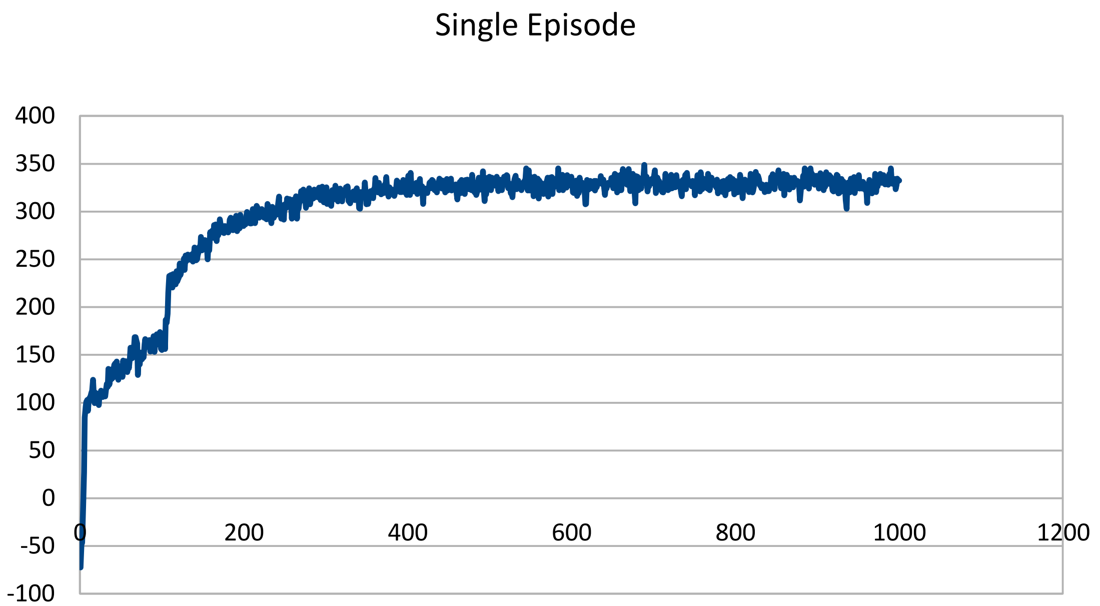
Rate	**Epsilon**	**E_i**	**E_f**	**Final Reward**
0.075	0.02	5	225	315
0.1	0.02	3	409	310
0.125	0.02	5	237	320
0.075	0.03	6	564	310
0.1	0.03	5	109	310
0.125	0.03	5	93	305
0.075	0.04	8	102	295
0.1	0.04	5	106	295
0.125	0.04	6	149	300

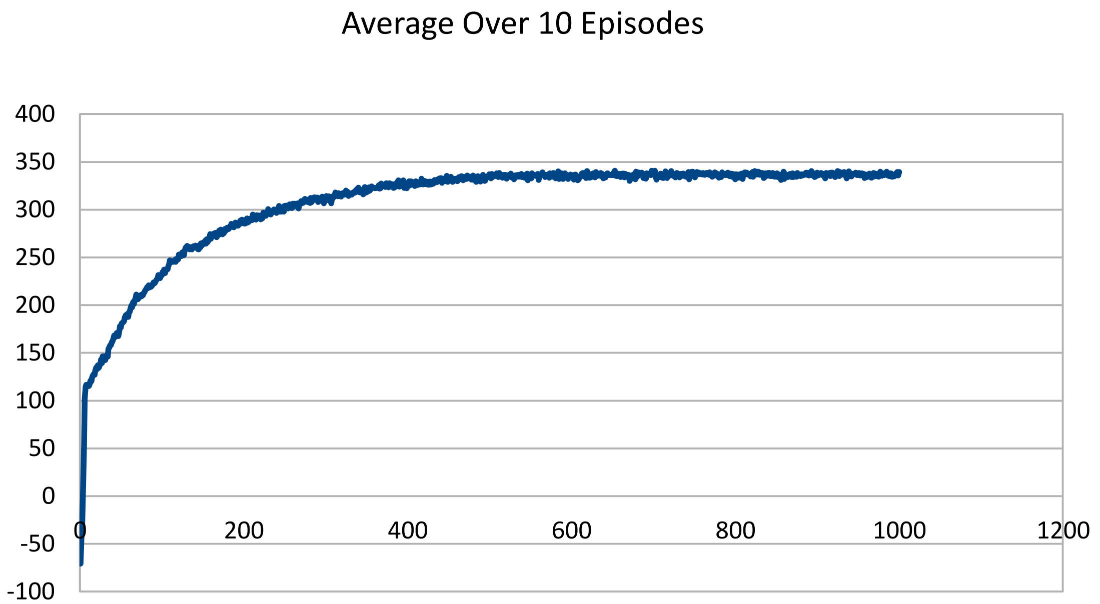
I am therefore choosing my rate as 0.125, and investigating epsilon lower than 0.02 for feasibility.

Rate	**Epsilon**	**E_i**	**E_f**	**Final Reward**
0.125	0.005	5	847	345
0.125	0.01	4	189	315
0.125	0.015	6	217	330

As 0.005 reaches the highest eventual reward, yet takes almost the entire allotted learning time to do it, I will choose it without searching for yet lower epsilon. 0.015 is a close second, in a fifth the time, so a case may be made for it, but I will stick with my original selection criteria.

3.





The averaged plot looks far more like what I would expect a learning algorithm to look like (exponential approach to a final value). The amount of noise in the single-episode case makes me reconsider the validity of my initial method of choosing epsilon and rate, though I still notice that the final result is far higher than any other epsilon or rate got close to. Given the very low epsilon and rate, it makes sense that the reward does not increase quickly, but it reaches an impressive final value.