Anand Ramachandran
aramach4@illlinois.edu

1. World Without Thief

(a) Epsilon = 0.

Except for a few episodes (< 10) at the beginning, which have negative rewards, the rewards for the rest of the episodes are all zero. Please refer to the figure below. The different levels of slipperiness are not marked.

		#		1
		#		
		#	#	#
R		#		
C		#		2

The agent starts from the bottom-left hand corner of the grid. If it keeps moving to the right, it will eventually meet column three that is slippery.

After a few trials, the agent will step into column three from column two only if the mean (until that point) negative reward from slipping is less than the mean discounted reward obtained after delivering both packages. Note that in the beginning, when all Q-values are 0, the agent behaves randomly in the process of breaking ties.

In the initial stages, after reaching column three, the probability that the agent visits '1' then '2' (or vice versa) within a certain small number of steps is very little, since the probability for taking any one of the four directions of movement is ¼ for a random agent. The positive reward is further reduced by the exponentially decaying discount factor. For example, from column three, the minimum number of steps to reach a customer is 2. This means we have a probability of ¼ * ¼ for the random agent to reach the customer. Mean discounted reward at column 3 on reaching one of the customers is then (¼ * ¼ * 0.9 * 0.9 * Reward) in the best case assuming no slip. This is smaller than the mean penalty due to slipping, the probability for which is 0.2 in the best case. Note that there is no reward for reaching just one customer, so this scenario is much less stringent than the scenario in the question, and even in this case, chances are very bleak. So, **the chances of slipping before delivering the packages are very high in the beginning when all Q values are not very well stabilized, and the agent is acting as a random agent. It is very highly probable that cells in column three will register a negative reward in these initial stages.**

Once a negative reward is registered on the cells in column three, that column will never be visited again with epsilon=0. This reward will also propagate to column 2 by the discounting factor. Thus, the agent will have no motivation to move into column 3 or column 2, and will remain trapped in column 1 and will keep receiving 0 rewards.

(b) Epsilon = 0.1. **There is some fluctuation in the rewards for different episodes, but on the whole, after a few initial episodes (<10) rewards become positive.** This means that with epsilon exploration, the robot eventually does get out of it's tendency to remain in the first column and delivers the packets successfully many times over the 10000 steps in an episode.

However, the variations in the rewards across different episodes need to be reasoned. These could be due to two reasons:

a. Epsilon randomness in choosing the best action
b. Q-values are not yet stabilized

The first one (a) cannot be avoided without a decaying epsilon. Even if the optimum policy has been decided, the agent would continue to choose bad steps at a rate of epsilon. The chosen sequence of steps will differ across episodes. The randomness in this case may not average out over the course of 10000 steps.

A side effect of the second point (b) can be observed with the policies printed out at the end of two different invocations of the simulation (for 1000 episodes of 10000 steps each). Multiple invocations will differ in the policy printed out. And this is actually observed. However this is not conclusive since multiple optimal policies are possible and the different invocations might be printing the different optimal policies (possible due to epsilon randomness). However the resulting simulation using the policy-simulator corroborates this. Since the policy-simulator doesn't have epsilon-exploration, a sub-optimal or incomplete policy can lead to the agent wandering aimlessly. Some policy results run on the policy simulator fail to complete the task of delivering the packets, while other policy results work satisfactorily in all cases (when the agent slips and when it doesn't slip).

A table is given below summarizing the results of four invocations of the policy simulator.

Trial	Reward Mean	Reward Variance	Policy Simulator Success?
0	133.051	3192.96	Yes
1	133.996	2798.84	No
2	133.908	3166.33	Yes
3	136.813	2948.82	No

When the agent is run with learning-rate set at 0.025, the mean reward is much higher at over 200. Thus, the agent is not doing its best with the settings given in this question, and Q is not yet optimal.

All these facts point to the Q-matrix not reaching the stable, optimal values. It indicates jitter in the Q-matrix values around the optimum, and also that different parameters of the agent, e.g., the learning rate, would possibly allow the Q-matrix to approach the optimum better.

(c) Epsilon = 0.5

In this case we observe negative rewards for almost all episodes. When epsilon=0.5, in half the cases, the agent chooses random actions. While Q-learning is off-policy and a high epsilon value should not prevent the agent from learning the optimal policy (provided we allow enough steps and episodes and set the learning rate correctly), **the agent will not use the optimal policy (or even a part-optimal policy that is incompletely formed), because it chooses random actions very frequently. This is the reason behind negative rewards.**

Since Q-learning is off-policy, the policy outputs of four invocations of the simulation were run through the policy simulator to understand the learning ability of the agent.

The learning ability of the agent was observed to be similar to that of case (b) with the agent successfully delivering packages in 2 out of 4 invocations of the policy simulator.

2. World With Thief

(a) Epsilon = 0.05, knows_thief = 'n'

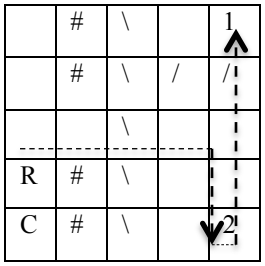
Rewards are predominantly negative in this scenario. To check the resultant policy, policy-simulator is run and the observation is that the agent is stuck in the first column.

Please refer to the figure below. '/' indicates high slipperiness and '#' indicates low slipperiness.

	#			1
	#		/	/
R	#	T		
C	#			2

The thief is moving randomly in column three. Without knowledge of the thief's location, the agent will see a uniformly distributed source of negative reward on column three of probability 0.20.

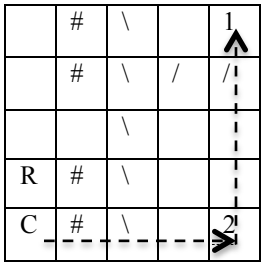
The following figure illustrates one of the paths for the agent that tries to minimize slipperiness and the number of possible encounters with the thief, at the same time traversing minimum number of cells. "\" indicates a probability of losing the package equal to 0.2 and replaces the thief as mentioned earlier.



Here, the agent passes through the thief's column once and a cell with high-slipperiness once. Expected discounted reward for the cell at the middle of column three, through which the agent passes is hence,
$0.2 * (-2) + (0.9)^7 * (-0.5) + (0.9)^8 * 1 = -0.20868$.

Moving into the cell in the middle of column three is hence undesirable for the agent, because moving in column one will result in only zero rewards (though the influence of columns 2 and 3 will tell on column 1 with reduced intensity due to discounting).

We will examine one more case (policy) where the agent moves directly to one of the customers. The path is shown below. In this case, the agent traverses one additional slippery cell but reduces the number of steps to the goal state hence making the goal more "visible" (less discounting).



Here we should examine the expected discounted reward at the lower-most cell in column 2 through which the agent passes.

The value is $0.2 * (-0.5) + 0.2 * 0.9 * (-2) + (0.9)^6 * 0.9 * (-0.5) + (0.9)^7 * 1 = -0.2208$

Thus moving into cell 2 is undesirable since column one gives higher reward. Paths through other cells in column 2 except the one in the middle will have even worse expected discounted reward since the distance to the goal would be larger, or the paths have to pass through multiple thief-cells or slippery-cells.

The examined paths are probably not the optimal paths. There maybe a better path that traverses more thief cells to avoid the high-slippery cells, or moves back into column one for the same reason. However these are longer paths, and the reward by delivery gets discounted exponentially and becomes comparable or less than the sources of negative reward on the way. For example, eight cells discount the reward of the goal state by $(0.9)^8 = 0.43$ which is comparable to the expected reward at any cell in column three $= 0.2 * (-2) = -0.4$.

Hence without knowing the thief, it is not possible for the agent to see a positive Q-value by moving into columns two or three. The agent will decide that there will be a net loss for the agent if it attempts to deliver the package wading into thief territory blindfolded.

Even in the absence of discounting, the expected rewards for both the paths given here are
Path 1: $0.2 * (-2) + 1 + 0.9 * (-0.5) = 0.15$
Path 2: $0.2 * (-0.5) + 0.2 * (-2) + 0.9 * (-0.5) + 1 = 0.05$

However, these paths will not be favoured by the optimal policy and hence can be the outcome of epsilon-randomness only, which will happen very rarely. Other, more complex paths (which will be more probable) will lead to more encounters with the thief or slippery cells and hence the rewards will be predominantly negative.

Hence in this scenario, rewards are too less for the goal-state and too much is lost by being attacked by the thief. The agent has to be able to avoid the thief to make the exercise profitable, which is not possible without having knowledge of the thief's location. Hence the agent will decide that it is better off in a non-risky column that is column 1. Negative rewards will be observed due to epsilon exploration in this case. Increasing the reward for the goal state, or adding a deadline and penalizing deadline misses with a large negative reward can solve this problem without changing other parameters.

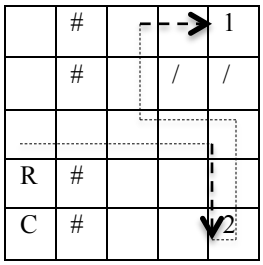
(b) Epsilon = 0.05, knows_thief = y

In this case, the agent performs very well giving positive rewards for all cases except the first few episodes. The mean reward is high at 270.057 (greater than when there is no thief), and policy simulator shows that the agent has developed admirable thief-dodging skills. The variance is also much lesser at 834.12 indicating a more stable policy.

Encoding the location of the thief into states, creating more states gives more information to the agent allowing it to make decisions regarding whether to stop or even retrace a step to avoid the thief when it has packages with it. In the absence of the knowledge of the location of the thief, the agent cannot make these specific decisions and will see column three as a uniformly distributed source of negative reward much like a slippery surface (as explained in part (a)).

Please refer to the following diagram. Here, the thief has been removed from the diagram. **Assuming that the agent can dodge the thief in all cases, there is a path indicated in the figure that poses no risks to the agent in reaching the two customers. This leads to a very large number of successful trips to the customers**, though the trip may take longer because the agent might spend some moves dodging the thief.

This explains why the agent is largely successful in this case.



(c) Search for best parameters

The search was first conducted with epsilon swept from 0.04 to 0.13 in steps of 0.01. Learning rate was held at 0.1 for this case. The results are as follows:

Epsilon	Average Reward for 1000 trials of 10000 steps
0.04	282.1675
0.05	271.8485
0.06	261.6405
0.07	248.7265
0.08	236.96
0.09	222.483
0.10	211.4685
0.11	199.4055
0.12	186.8475
0.13	174.615

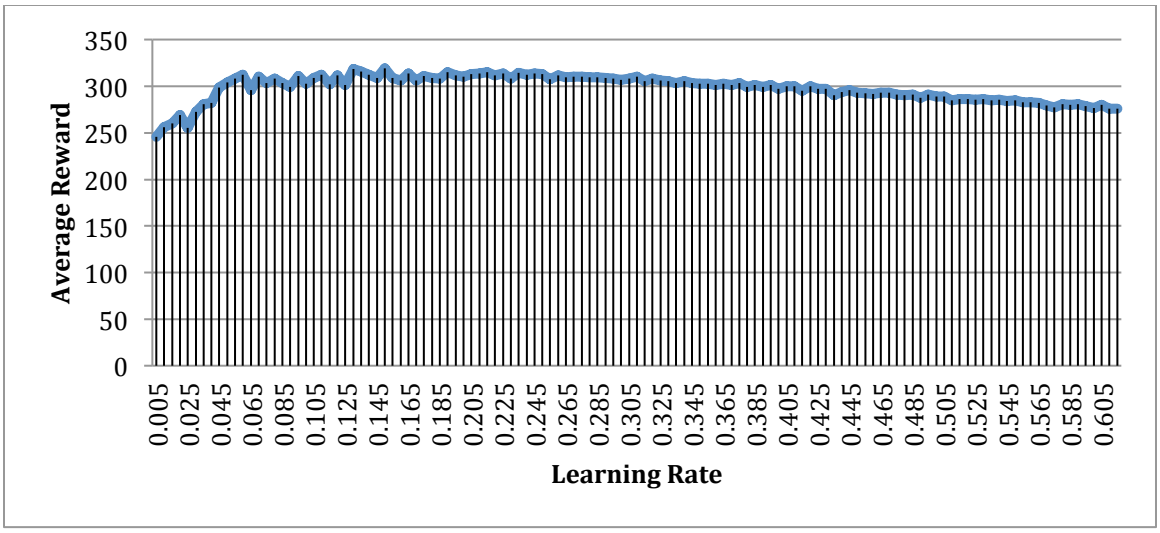
The average reward decreases monotonously for these values of epsilon. Hence the best value of epsilon is possibly less than 0.04. Further sweeps of epsilon were carried out from 0.005 to 0.035 in steps of 0.005. The results are as below:

Epsilon	Average Reward for 1000 trials of 10000 steps
0.005	310.989
0.010	312.8785
0.015	314.704
0.020	308.814
0.025	299.4595
0.030	295.6765
0.035	291.606

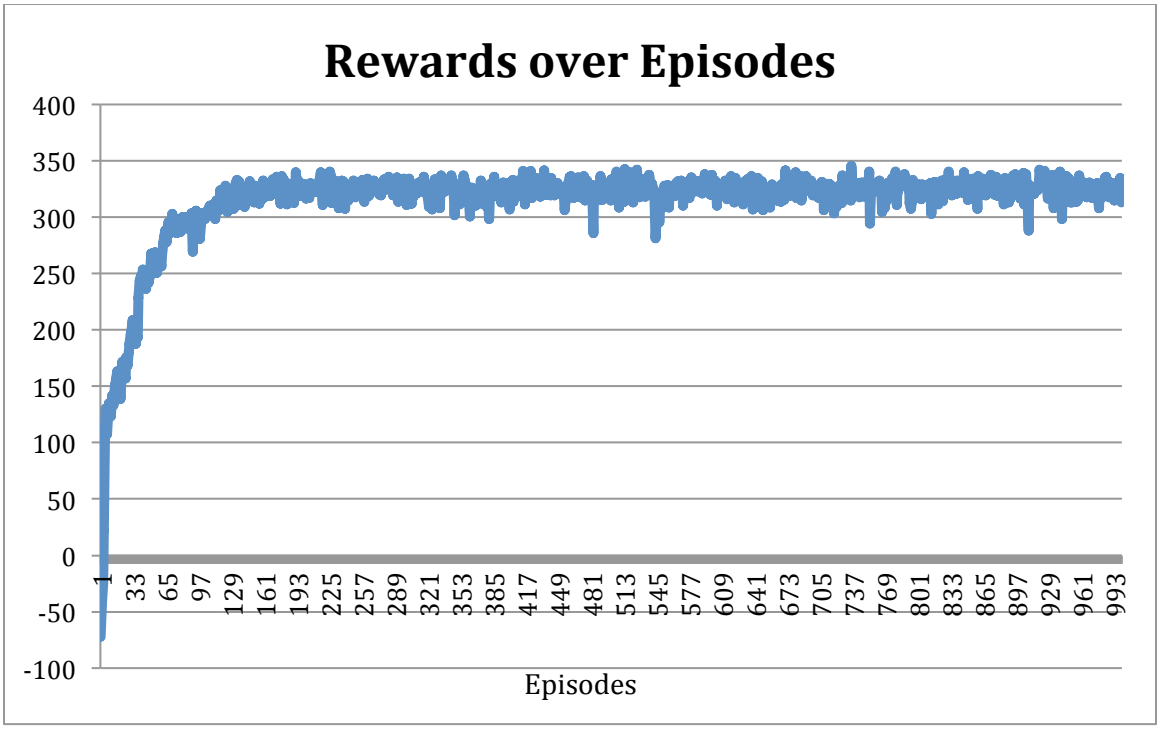
There is a peak in this case at epsilon = 0.015. We fix the best-case epsilon as 0.015.

In exploring alpha, epsilon was set at 0.015, and alpha was swept from 0.005 to 0.615 in steps of 0.005. The average reward for 1000 episodes of 10000 steps is measured. The result is plotted below. The plot indicates that rewards increase first for small values of the learning-rate, and slowly start decreasing again after a point. The best average reward occurs for learning rate equal to 0.15 equal to 319.0445. We choose this as the representative best value for alpha, since the peak is flanked on both sides by regions of lesser value indicating a local maximum. Hence we have,

Best alpha is 0.150
Best epsilon is 0.015


Average Reward
350
300
250
200
150
100
50
0
0.005
0.025
0.045
0.065
0.085
0.105
0.125
0.145
0.165
0.185
0.205
0.225
0.245
0.265
0.285
0.305
0.325
0.345
0.365
0.385
0.405
0.425
0.445
0.465
0.485
0.505
0.525
0.545
0.565
0.585
0.605
Learning Rate

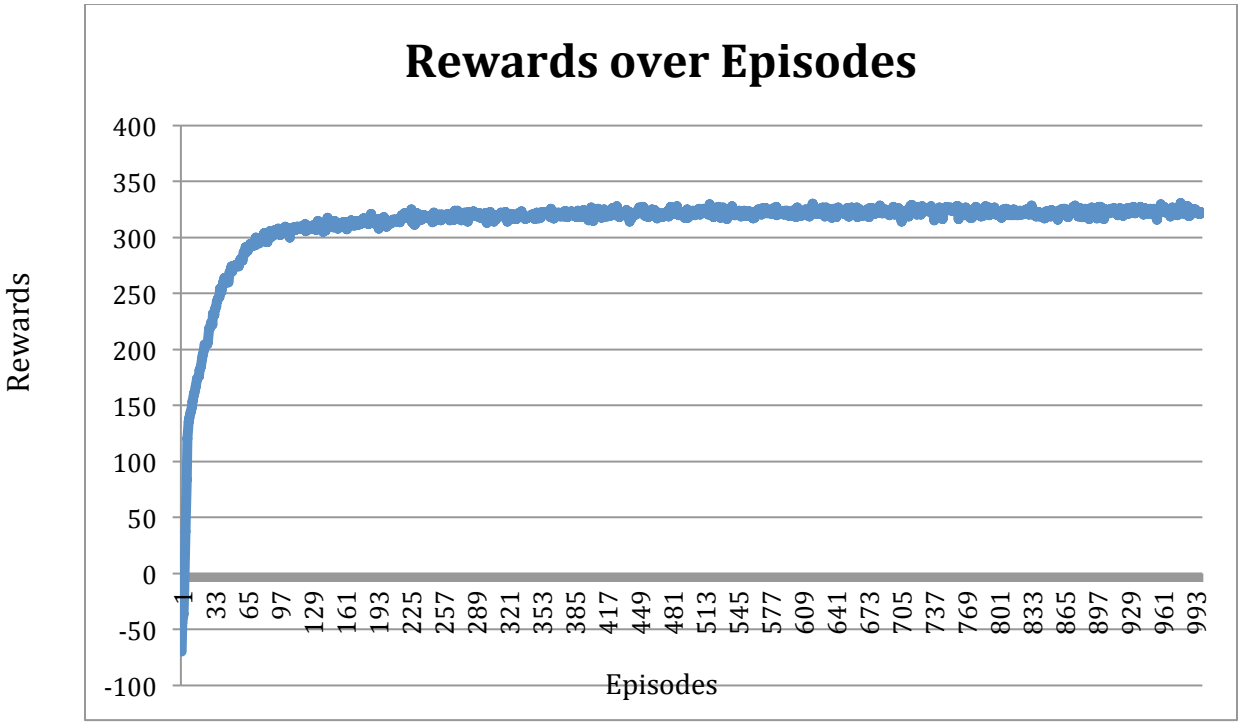
3. The plot showing the agent's evolution over a period of 1000 episodes of 10000 steps each is given below:



The rewards are initially negative and steadily increase. After a sufficient number of episodes rewards settle on values above 300. Initially, the agent follows a random policy breaking ties during which time, it largely faces losses due to lack of deliberate movement towards the customers from the company. After a while, the policy is improved due to Q-learning and approaches the optimal policy. However, the behavior of the agent still shows variations after this point. The variations can be due to many things:

1. **Learning-rate is high, non-decaying.** For absolute convergence towards the optimal policy, the learning-rate should decay towards zero as confidence improves on Q-values. If alpha-decay is not incorporated and when alpha is high, Q-scores can jitter visibly around the optimal values without settling down.
2. **Randomness due to epsilon**. The agent doesn't have a decaying epsilon because of which it is randomly exploring bad cells in the grid. The sequence of these random cells will change from episode to episode. To weed out the effects of these random explorations, the results will have to be averaged across a larger number of runs.
3. **Randomness of the World**. The new world released for part 2 of the MP has randomness incorporated in it. The agent might execute an action and find itself in a cell that it didn't want to end up in. The sequence of these random events will vary from episode to episode. Averaging over a larger number of episodes (which are at an equivalent stage of evolution of the agent) will weed out the effects of the World's randomness.

As specified in the question, ten invocations of the simulation were run and averaged. The results are plotted in the figure below.



Averaging over a larger number of trials confirms the points raised in the previous section of the answer. This graph is much better behaved than the previous one. Random effects such as the effects of epsilon exploration and the randomness of the world are averaged out in this case. This is the reason for the lesser fluctuations in the graph.