CS 440: Artificial Intelligence
MP1: Part 2

Due on Monday, September 30, 2014

Prof. Gerald DeJong 3:30 - 4:45 pm

Giri Prashanth Subramanian

September 29, 2014

World Without Thief:

(a) In this case, the agent just goes straight up and gets stuck in the top left position. The reason for this outcome is that the Q-learning algorithm has found a local minima. It knows that if it goes to the right, there is a possibility of it dropping a package (and hence of a negative reward) and hence the algorithm will never take that route because it is a greedy algorithm. Since all the Q-values are initialized to 0 in the beginning, it has two options, to either go right or go up. But if it goes right, the next states are all negative rewards (and hence negative Q-values in subsequent iterations) and hence it goes up and keeps going till it hits the ceiling. Hence, the final outcome is that it gets 0 reward.

(b) In this setting, because it is not a purely greedy solution, it actually finds the global minima. From the company, the robot goes straight to location 2 and drops off the first package, then goes up and around the region with higher slipperiness and delivers the second package to location 1. It then goes back to the company and picks up 2 new packages and repeats the cycle. Sometimes, in the slippery region, it drops the one (two) package(s), it then immediately goes back to the company and picks up 2 new packages and repeats the cycle. The reason why this happens is very obvious. Since it has a randomness in its selection strategy, at times, it will randomly choose a state with a lower Q-value than its current state, which might end up with a bigger reward in the long run (thus finding the global minima). Hence, when it is on the left-most lane in the beginning, it will randomly choose moving to the right and so on, thus finding the reward at the right-most lane. This is also the global minima because the reward for delivering the package is actually higher than the penalty for dropping the package. If it would have been the other way around, the agent would stay on the left lane. Also, it goes to location 2 first because it is closer than location 1. Since we discount rewards to favor rewards that we obtain first, it goes to the closer of the two rewards.

(c) If we increase the value of ϵ, randomness will start dominating and hence wont let the solution converge to the optimum policy. Even after it finds good Q-values it will still take random steps, thus reaching random targets. Also, since we dont have a decaying ϵ, this will happen till the very end and hence no matter the number of steps or episodes we try it for, it will have very similar effects. Hence, in this case, we see that the robot starts moving, but gets stuck in some infinite loop and decides to stay there. Or in some other case, it will go through the more slippery region and end up dropping the packages more often than necessary.

World With Thief:

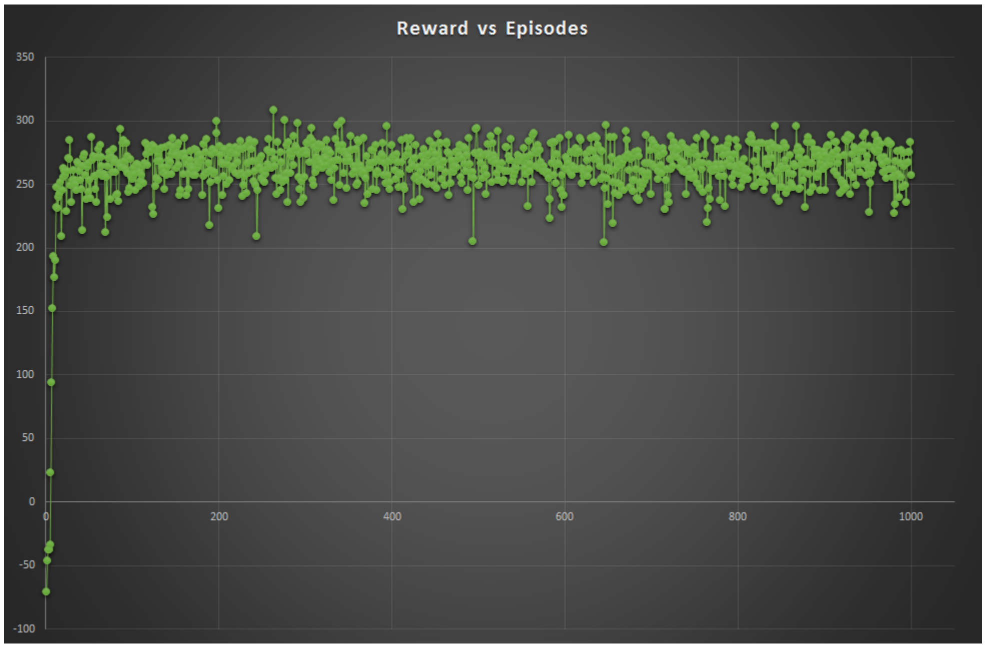
(a) Since the agent doesn't know about the presence of the thief, the thief row is not included in the state of the agent. Hence, according to the agent, when it gets to the thief row, it randomly gets a penalty. Hence, it will try and avoid that entire region because it may incur a penalty function if it goes there (a huge one). This penalty function outweighs the reward obtained by delivering the packages (-2 compared to 1). Hence, the agent always stays on the left lane. It moves to the top (for similar reasons as part (a) of q1) and stays there. This is the global optimum solution according to the agent (not local, because of the epsilon-greedy search).

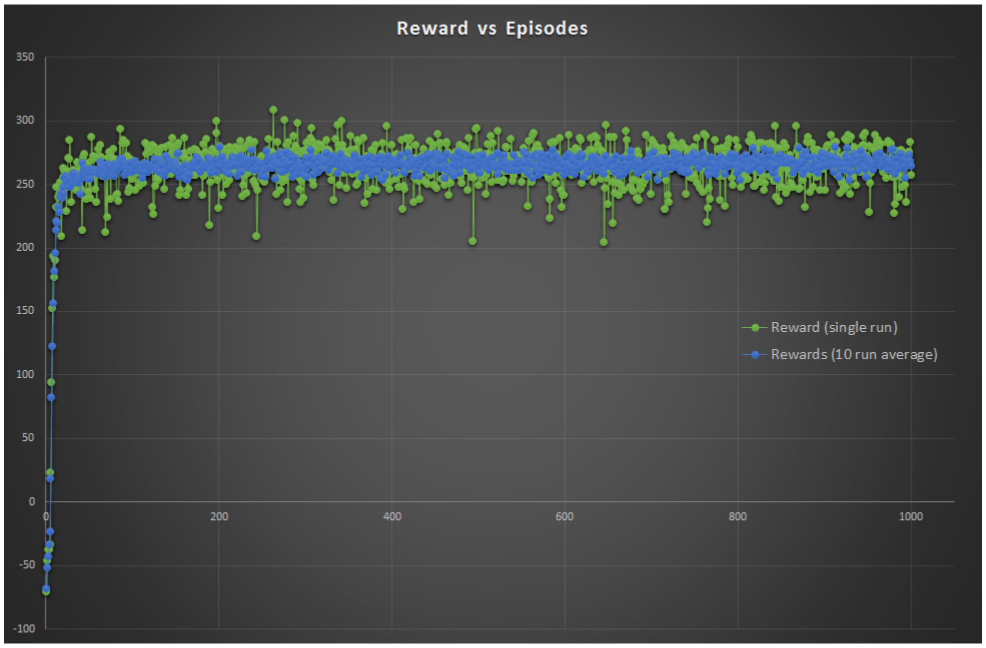
(b) In this case, since the agent is aware of the thief, it is included in the state of the agent. Hence, it knows to avoids those states only. Hence, it always polls for the location of the thief and avoids only its row and column instead of avoiding the entire column. Hence, the agent actually moves optimally and delivers the two packages while avoiding the thief and the more-slippery regions. Also, since it is epsilon greedy search, it actually finds the global optimum solution in this case as well. It starts from the company, goes to location 2 while avoiding the slippery regions completely, avoids the robot, avoids the more-slippery regions and goes to location 1 and then goes back to the company. Thus the reward obtained in this case is 2 per iteration. It also goes to location 2 first because of the same reasons mentioned in part 1b.

(c) Since we want to find the effect of both those factors independently, I varied one of them while keeping the other one constant. With the learning rate kept constant, increasing ϵ beyond 0.15 started giving negative rewards in lot of the trial runs. This is possible because it going random beyond a point of necessity. Hence, any value greater than 0.15 was discarded. The rewards seemed to peak at ϵ approximately equal to 0.05. Hence, this value of ϵ was chosen. Next, the learning rate was varied keeping ϵ constant at 0.05. At 0, the agent didn't learn anything and was useless. The reward function seemed to peak around 0.3 and then started to fall off again. Anything beyond 0.5 and the agent started behaving non-optimally by going back and forth in some regions when not necessary. This starts happening because the agent is not taking advantage of the previous found "good" solutions and is depending more and more on the current iteration. Hence a value of 0.3 was chosen for the learning rate. Hence, in conclusion, a learning rate of 1 would be optimal if the environment was deterministic. But since it is not deterministic in this case, we need a lower value of the learning rate. Also, too much randomness in finding the solution doesn't let the solution converge and again gives a bad result. Hence, we want a small value of ϵ. These parameters will vary from problem to problem. For the given problem, the best value seemed to be 0.3 for the learning rate and 0.05 for ϵ.

Plotting

The plot of the rewards after each iteration has been shown below. We can see that the reward starts off at a negative value and then increases and settles to some value. However, there is a lot of variation in a small band as every run is stochastic. Hence the rewards obtained between any two runs is different. However, we can see from the next graph that this value starts converging to the true value when we take an average value between multiple runs. The blue band in center represents the average over 10 runs. We can clearly see that the band is much smaller than the green band.





Thus, if we keep increasing the number of runs and start taking an average, the value will converge further according to the law of large numbers.