AI MP 1 Part II

Goran Tomic

1a) After simulating 1000 episodes and 10000 runs the agents reward was exactly 0 for every episode except for the first one. What happened in the first episode was that the agent received a negative reward, then the policy was updated to avoid those locations. By avoiding those locations the agent just traveled to spaces with no net reward. Since there is no randomness the agent will not encounter a state in which he receives a positive reward, and thus will not reach a goal state. Randomness is essential for this very reason. It is beneficial for our agent to move against the policy, that is, move randomly some of the time. This can lead the agent to larger rewards that the policy hasn't taken into account since the agent has not yet moved there.
1b) By changing epsilon to .1 randomness was introduced. epsilon > 0 is very beneficial as randomness can lead the agent to a more optimal path for reaching the goal state. As was seen in a) since there was no randomness the agent never reached the goal state. Generally, it could be the case that the policy indicates a move just because *some* positive reward is there, but there may be an unexplored move that contains a higher reward, and we need randomness to find that state if a policy is directing us toward a state with a lower positive reward.

1c) Changing epsilon to 0.5 greatly reduced the reward. In fact, as epsilon is increased from 0.5 to 1 the expected reward per episode gets more and more negative. This is due to the fact that the larger epsilon is the less likely the agent is to follow the policy. The policy is what tells the agent his best action, and not obeying it results in random moves. It is not surprising that as epsilon grows from 0.5 on the reward gets more and more negative since the agent is doing random moves, and there are a lot more negative rewards than positive ones.

2a) In WorldWithThief the performance of the agent is poor given that he does not know where the thief is. The expected reward at the end of each episode is about -5. If knows thief is set to "no", then there does not exist a way for the agent to figure out where the thief will be, thus the policy may in fact lead him right into the thief which will result in negative rewards. The thief is moving according to his own policy, and when knows thief is set to no there is nothing that will let the agent know the location of the thief, or even predict where the thief might be.

2b) If the agent knows where the thief is the performance significantly improves. The agents expected reward was around 300, and this is a huge improvement from -5. Since the agent knows where the thief is he can avoiding moving to those spots. In 2a the agent couldn't learn because he never knew where the thief was, but when we set knowsthief to yes then the learning is significantly improved because he knows the theifs policy.

2c) The best value for epsilon is around 0.02, and the learning rate is around 0.1 . As you increase the learning rate from 0.1 the expected reward per episode begins to slowly drop. This is due to the fact that as the learning rate increases the agent places less weight on the old information (old q- value). This can be seen in the equation for the q-value:

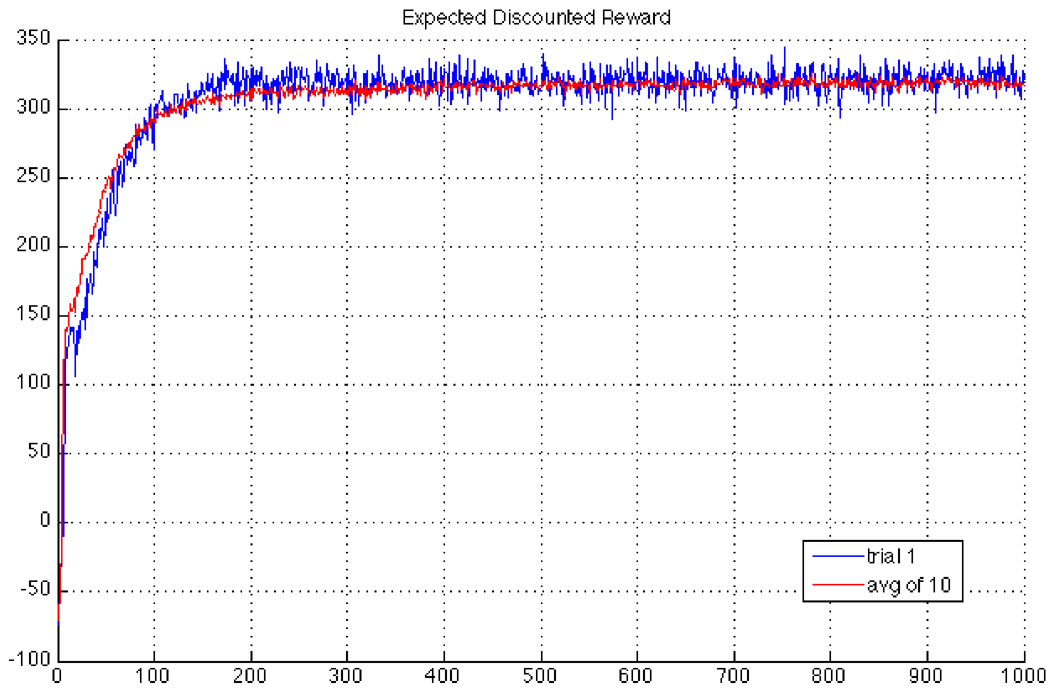
```
qValue[oldState][action] += rate*(reward + discount*bestUtility(newState) -
qValue[oldState][action])
```

When the learning rate is 1, then the old q-values cancel out, and the new q-value is determined solely based on the new information, that is :

```
        qValue[oldState][action] = reward + discount*bestUtility(newState)
```

. In order to model knowledge in our agent we need it to be able to consider old information, and slowly override old information with new information, and 0.1 is a good candidate based on experimenting with different rate values. The best value for epsilon seems to be about 0.02. As epsilon increases from about 0.05 to 1.0 you can notice easily a decrease in the expected reward. As epsilon increases the randomness of our agent increases as is described in problem 1.

3) After simulating the agent for 1000 episodes and 10000 steps it appears that the agent learns very rapidly at first. After about 150 moves his expected reward stabilizes. After repeating the simulation 10 times and taking the average of those I compared it against one of the simulations (plot on next page) and it fits very nicely. The plot of averages fluctuates a lot less than the plot of the single simulation; however, they follow approximately the same curve.


Expected Discounted Reward
350
300
250
200
150
100
50
0
-50
-100
0
100
200
300
400
500
600
700
800
900
1000
trial 1
avg of 10

r