**Project 3 –Blackjack**

Weipo Lai 19340980040

**Problem 1: Value Iteration**

**a. Give the value**

1. Iteration 0:

V(-2) = V(-1) = V(0) = V(1) = V(2) = 0

1. Iteration 1:

V1(-2) = V1(2) = 0

V1(-1) = max{0.8(20+0)+0.2(-5+0), 0.3(-5+0)+0.7(20+0)} = 15

V1(0) = max{0.8(-5+0)+0.2(-5+0), 0.3(-5+0)+0.7(-5+0)} = -5

V1(1) = max{0.8(-5+0)+0.2(100+0), 0.3(100+0)+0.7(-5+0)} = 26.5

1. Iteration 2:

V2(-2) =V2(2)=0

V(-1) =max{0.8 (20+0) +0.2(-5-5), 0.3(-5-5) + 0.7(20 +0)}=14

V2(0) =max{0.8(-5+15)+0.2(-5+26.5), 0.3(-5+26.5)+0.7(-5+15)}= 13.45

V2(1) =max{0.8(-5-5) + 0.2(100 + 0), 0.3(100 + 0) + 0.7(-5-5)}=23

**b. What is the corresponding optimal policy πopt for all non-terminal states?**

πopt(-2) = πopt(2) = endstate

πopt(-1) = s-1 = -1

πopt(0) = s+1 = +1

πopt(1) = s+1 = +1

**Problem 2: Transforming MDPs**

**a. Not always V1(S start) ≥ V2(S start) (can see in submission.py)**

**b. MDPs, find the optimal value at each node**

Markov chain just goes forward and doesn’t go back.Therefore, if we want to get the optimal global reward, we only need to get the optimal reward for each recursion. Therefore, we can use a dynamic programming algorithm to compute V(s), which then only needs to be computed once for each (s, a, s') triple.

**c. Set the state o as a new state**

Set the state o as a new state that each state can lead to. In other words, we add an option for each iteration, as shown in below.

R\*T(s, a, s’) s’ ∈ States

1-r s’ = o

T(s, a, s’) =

1/r\*R(s, a, s’) s’ ∈ States

0 s’ = o

R’(s, a, s’)

V’opt(s) = Vopt(s)

That is, the new MDP has the same optimal values as the original one.

**Problem 3: Peeking Blackjack**

**a. Implement the game**

We implement the Blackjack game in submission.py.

**b. Build a peeking MDP**

We just put a card with large value, so that can lead players to peek more.

**Problem 4: Learning to play Blackjack**

**b. Comparing Q-learning policy and value iteration policy, the different ratio as follows:**

Small MDP: 0.2593

Large MDP: 0.3315

According to the values, we can find that Q-Learning performs better in small MDP. We think, maybe the state space is relatively small, so that Q-Learning can learn in a better environment. However in large MDP, it requires more steps and be much complicated to converge.

**d. After 30000 trials, we can get each reward as follows:**

Value Iteration: 6.83

Q-Learning: 9.08

We can find that the reward of Value Iteration is less than Q-Learning’s. Because Q-Learning updated its strategy in each simulation, so it has a higher reward. We can learn Q-Learning is much reliable than value Iteration.