CS 747: Programming Assignment 2

Dhakne Ajay Sopan
190050033

Task1- MDP Planning Algorithms

Value Iteration, Howards Policy Iteration, and Linear programming are implemented in the file planner.py.

Value Iteration:
Initialize one vector V which stores value function values for all states (initially all zero). Calculate the value function according to the value function formula in value iteration until the new vector is less than some tolerance value (1e-12) compared to old V.
Value Iteration gives better performance on MDP which has a large number of states.

Howards Policy Interation:
A policy vector with random values is assigned initially. Then Calculate the action value function. Then calculate the new policy by comparing the action value functions of states. Calculate new policy until a new policy is equal to the old policy or there are no improvable states remaining. Performs better than VI but takes time to give an output when the MDP size is high.

Linear Programming:
It is implemented using Pulp for solving linear equations. Value function constraints are given to PULP_CBC_CMD solver. Gives better performance than VI but is slower due to a large number of constraints to the solver in case of high MDP size.
Based on the discount factor different Value functions and policies are obtained.

Task2 - MDP for Cricket Game

Each state is represented as bbrrs.
bb= balls remaining
rr=runs required
s={0 when player A is on strike; 1 when player B is on strike}

Whenever **player A** chooses to do one of [0,1,2,4,6] actions his next state will be determined by probabilities. So each action will result in 7 possible transitions (transitions with zero

probabilities included). Based on actions and probabilities bb and rr will change . Depending on runs scored and balls faced bb will decrease and rr will decrease.
The reward for such transitions is assigned 0
Ex: once hit a six bb -= 1 and rr -=6

When **bb%6 == 0** a player will change the value of s.
We have valid states for those when s == 0 i.e when our agent or player A is in control.

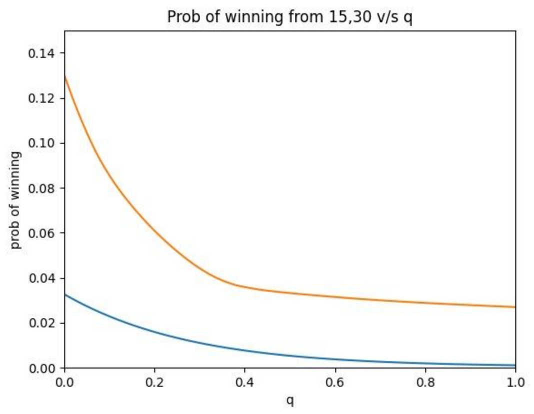
Once **bb == 0 and r>0** such states will be a loss state - with reward 0
The transition probability will be the sum of such action probabilities which reach a state s from some state s_prime and result in some bb or rr change.
The next state will be a loss-trap state which will have no outgoing transitions.
With the state number as ‘0’

Once **bb >=0 and r<=0**

Such states will be a win states. With reward = 1 .
The next state will be a win-trap state which will have no outgoing transitions.
With the state number as ‘901’
A Dictionary is used whose key is “bbrrs” and the state number as a key
Eg. Dict[“00000”] = 0 and so until all possible states which are 900 +2(trap states)
So the number of states will be 902.

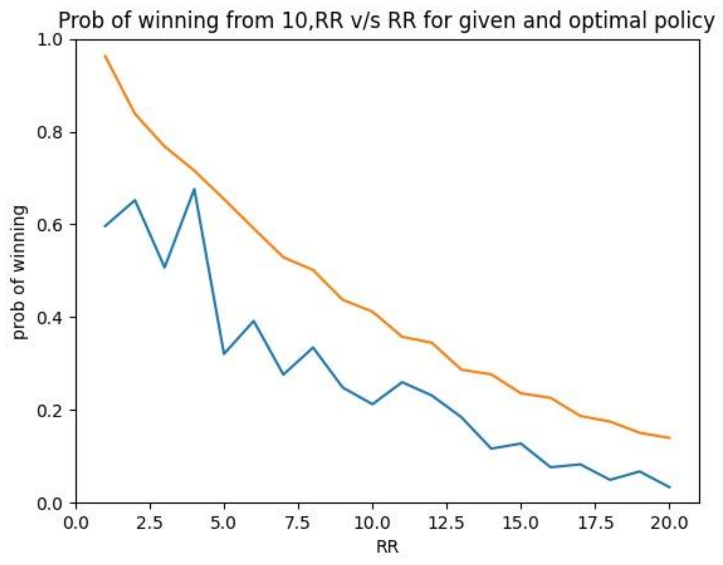
Plots:

1. Clearly, as the q value increases the probability of winning decreases for both policies. Clearly the optimal policiy will perform better since it will take only those actions where reward is positive or runs_scored >=0.Since B can only have [-1,0,1] as its sample space mean reward will decrease if we give more strength to B or q is high.

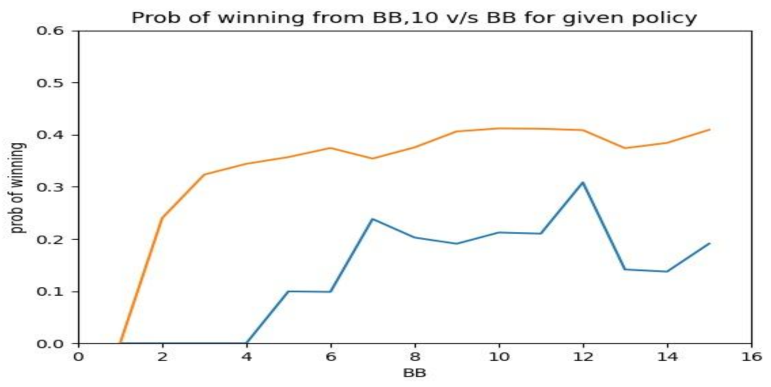


2.

The probability for both policies decreases as for both policies' runs to score increases.Since a player has the same set of actions and probability of happening of the actions as runs to score increases. The expected value of runs scored by player A will remain constant so as the runs scored increase player has less control over winning and hence is likely to loose. As Expected value will be less than the runs scored if runs scored are high and players probability distrubution wont lie in that range.



3. As the number of balls remaining increases the player has more chances of failure as even if player chooses to defend the ball he will have same number of balls as compared to earlier MDP .Hence his probability of winning will be atleast as same as of the MDP with(number of balls -1).


Prob of winning from BB,10 v/s BB for given policy
prob of winning
BB