Task 1:

1) **Implemented Eval_Policy , value_iteration , Howard policy Iteration, linearProgrammingFormulation.**
2) **Used np.linalg.solve() to solve the linear equations for finding the values for a given policy.**
3) **Default Algorithm is set to value_iteration**
4) **Howard's Policy Iteration:**
    a) **Switch actions in every improvable state.**
    b) **Takes action that gives us maximum Value on particular state.**

5) **Value Iteration:**
    a) **Took an arbitrary value vector**
    b) **Computed Improved Values and Policy**
    c) **If the improved values are approximately equal to old values, return them or loop again**

6) **Linear Programming:**
    a) **Using Linear Programming Formulation to compute the values given the constraints and variables and Value**
    b) **Maximized -1*sum(values)**

Task 2:

Algorithm:

For n states in the statefile, I considered n+2 states where the last 2 states represent win state and lost state and on going to win state we get a reward of 1 otherwise 0. There are no transitions from win state or lost state. For each state B1, the probability of going to other state B2 is computed as follows

⇒ A will have a transition to B2 state , if he can strike again in that state after striking in B1 state, i.e. A should strike in the state B1 and then B strikes for all the balls in between and then A strikes again in the state B2.

⇒ probability of going to win state is calculated by A strikes then wins or A strikes, batting goes to B , B keeps striking and then wins.

⇒ probability of going to the lost state can be simply found by subtracting all those transmissions from 1.

⇒**`def check(balls1,balls2,A_score,B_score)`**

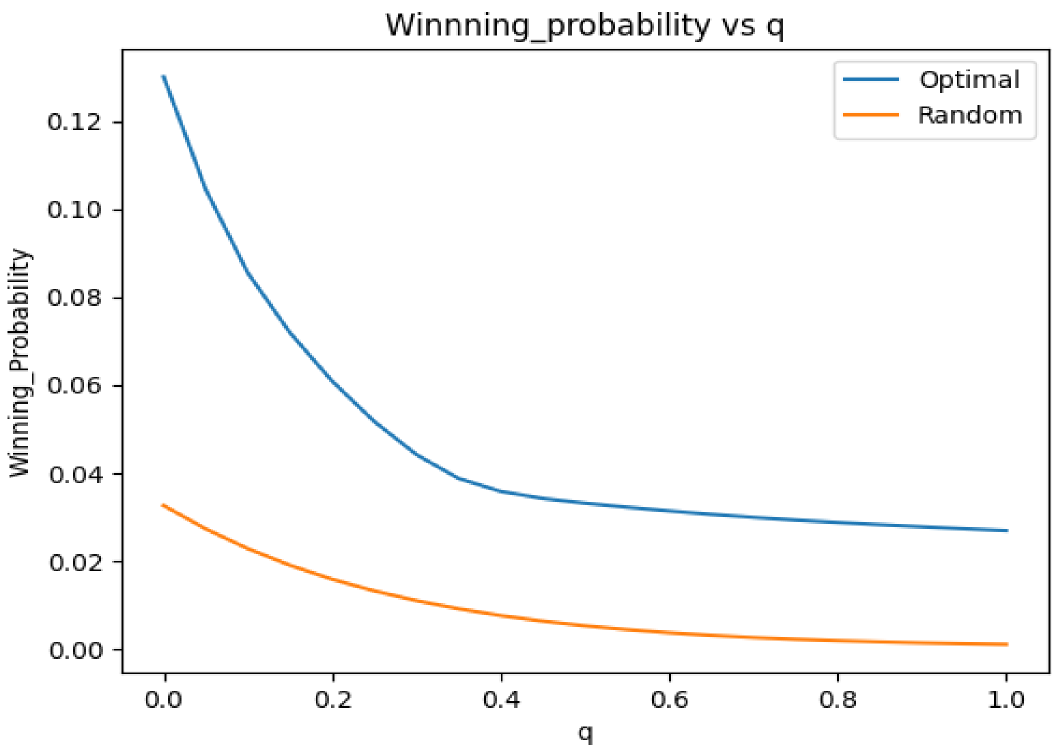
⇒This function checks whether it is possible to achieve such a transition or not, A strikes in balls1 and again strikes in balls2 given A scored A_score and B scored B_scored

⇒ This probability will be added to the transition

⇒ def Bwin(start_ball, runs_needed,q)

⇒ This function returns the probability of B winning given the conditions before giving the striker back to A.

Analysis 1: 15 balls, 30 runs

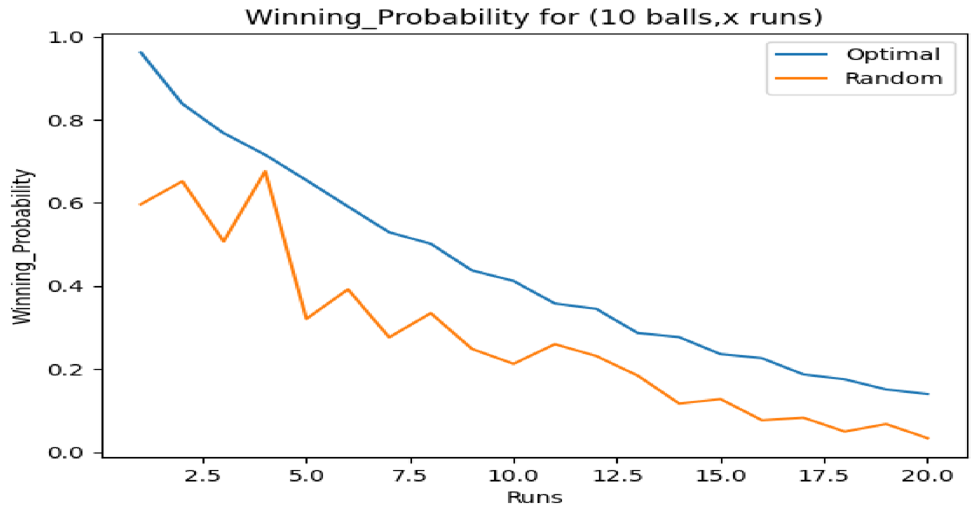


⇒ Optimal policy performs better than Random policy

⇒ As we increase q , probability of losing the match increases because q is the out probability of b

⇒ As we decrease q, probability of winning the match increases.

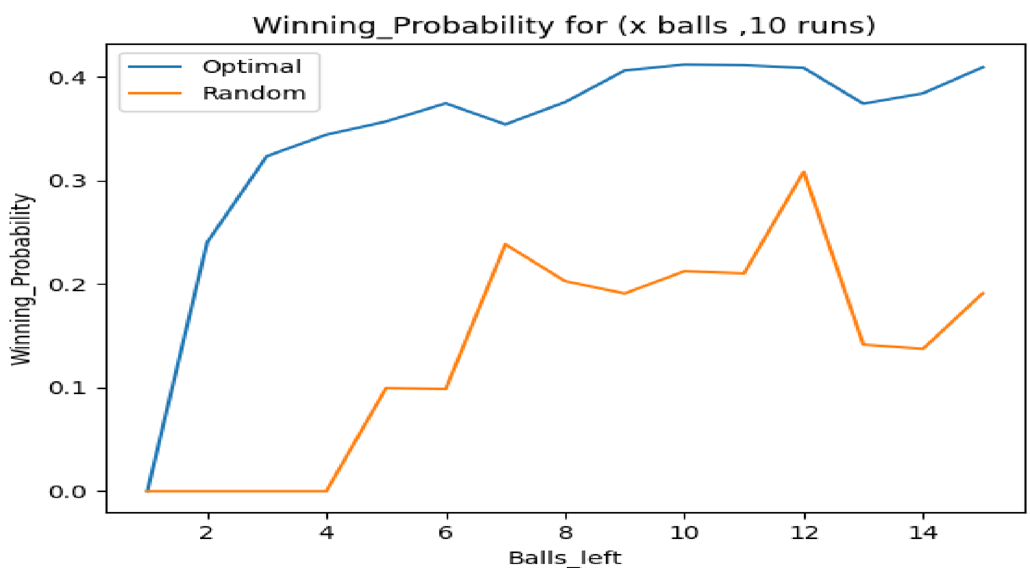
Analysis 2:



⇒ decreasing the number of runs increases the probability of winning

⇒ optimal policy performs better than random policy

Analysis 3:



⇒ optimal policy performs better than random policy

⇒ as the number of balls increases, probability of winning increases.