Assignment 2

Dhvanit Beniwal
200050035

October 11, 2022

Task 1

When reading MDP values from a file, we could assume that a particular transition s, a, s' will only be listed once. But I have implemented planner.py assuming that a transition s, a, s' may appear more than once with different values of reward, probability since this allows me to later write the encoder more conveniently. In the implementation, all the transition probabilities of s, a, s' will be added up and the reward would be calculated according to Bayes. It was not absolutely necessary to calculate reward this way as for cricket the reward only depends on s' and is the same in these multiple instances. But anyway

For episodic MDPs I have simply added self loops of probability 1 and reward 0 for end states which seems to have worked.

For the condition for convergence in value iteration, I check if the norm of difference between successive iterations is less than 1e-8.

While making helper functions for π^* given V^*, or for calculating B^* etc 2 helpful observations with respect to numpy arrays were that these would be max() and argmax() of the action value Q and that in the calculation for Q the numpy interpretation for matrix multiplication of 3D array (T) with 1D array (V) lets me write a clean expression.

For Policy iteration, it was simpler to perform an argmax() on action value Q^π so that all improvable states get improved (in accordance with Howard).

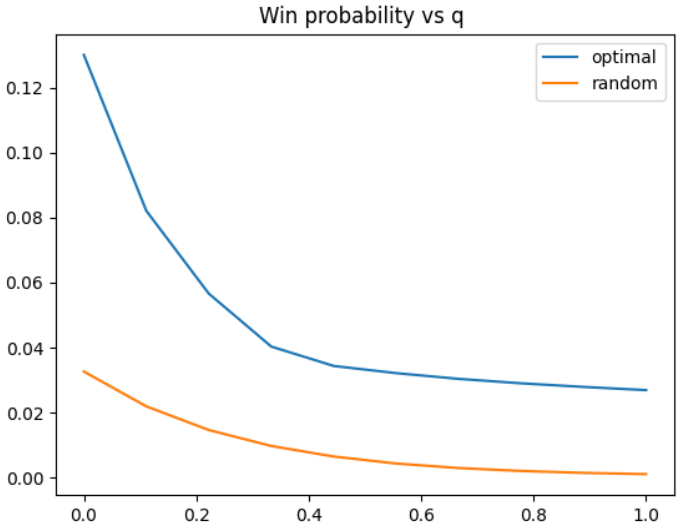
Task 2

We receive BR number of states where we need to score R runs in B balls. My MDP will have a total of $2(BR) + 2$ states and the same 5 actions $0, 1, 2, 4, 6$. The first BR states correspond to player A being on strike and the next BR states correspond to player B being on strike. Then there is a win state and loose (or draw) state. These will be the end states and mdp will be episodic.

For the states where B is on strike, I still pretend that some action $0, 1, 2, 4, 6$ is done (by A supposedly) but since A can't influence B, whenever I add a transition state, probability and reward I add a copy for all 5 actions so that all states end up with the same 5 actions. While decoding, I will not need to look at the optimal action for these states (or the end states either for that matter). The first BR values of the optimal policy will suffice. Here the optimal policy may assign any action for states with B on strike and it wouldn't change the optimal value.

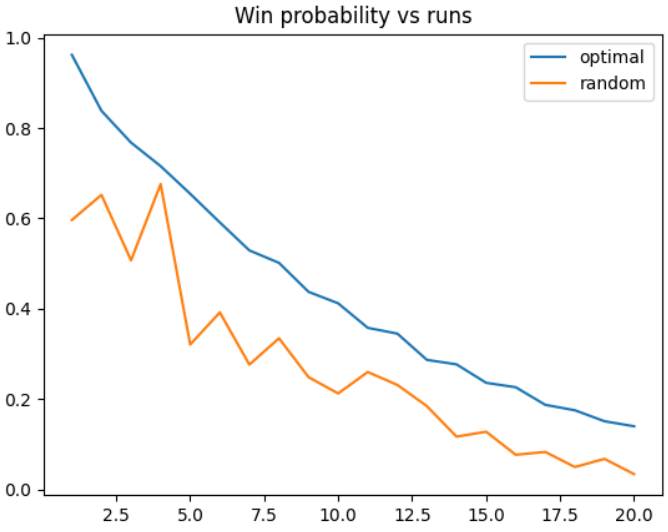
For calculating transition values (s',T and R given s,a) I just iterate over states($2 + 2BR$), actions(5) and possible outcomes (7 if A's on strike, 3 if B's on strike) and add a transition probability, reward for the state s' I calculate (taking care of win and lose conditions and strike-change, over-finished conditions). As stated before, I allow myself to add multiple transitions for some s, a, s' if it can 'happen in more than one way' because my implementation of planner.py will add the probabilities.

Graphs (Task 2)



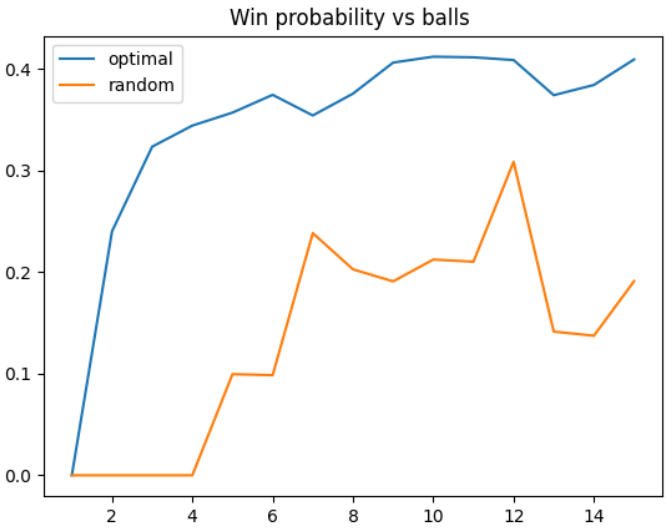
Win probability vs q (weakness of B)

As expected, we are less likely to win if B is more weak. Optimal policy is seen to perform better than a random policy. It is interesting to note that at even at high values of q optimal policy maintains a difference over random policy instead of converging to become equal. If B is guaranteed to end the match by coming on strike at any time there is still a policy A may follow to significantly improve the chances



Win probability vs runs needed

Clearly we are less likely to be able to make more runs. The advantage of optimal policy is more prevalent in hard to win situations (more runs needed) since a random policy compares fairly ok otherwise for lower values.



Win probability vs balls left

Clearly we are more likely to be able to win if we have more balls to play. However, after some point the danger of running out of balls while having runs to make is outweighed by the danger of getting out before winning and thus the graph is flat after a while. This means that even with an infinite number of balls to play we can only improve our chances so much since we have non-zero probabilities of getting out at any point.