CS 747 : Programming Assignment 2

Hardik Siloiya | 190050047

Task 1)

Created an adjacency list of the transition probability matrix to optimize for space.

Value Iteration: Computed the value function until difference between successive functions is less than $1e-8$ ($L2\,Norm$).

Policy Iteration:

Iterating in a while loop until any of the states do not have any improvable action in our current policy. This policy gives the least amount of error among all of the algorithms I have implemented.

Linear Programming: Used the `PulP` library to solve the system of equations with n variables and nk constraints. We maximize the negative of the sum of the value functions for all the states here.

Task 2)

Created an MDP for the anti-tac-toe using the state of the board as a state in the MDP and then, the agent can perform a move at any of the empty locations and then the environment (other player), makes its move and the final result is some state with probability p and reward r which is 1 if we land in a winning state. I have used 2 end states here, one which is of winning and other which is of draw/loss (can be reduced to 1 end state). If we make a move which causes the opponent to move then we go to the lose end state with 0 reward.

Task 3)

There are a finite number of policies that an agent can adopt while playing the game of Anti-Tic-Tac-Toe since the game board is finite in size and we only have at most 9 possible actions per turn.

There exists a strategy for always losing at a game of Tic-Tac-Toe if one is the second player. Thus, this implies that there exists a strategy for the second player to always win at a game of Anti-Tic-Tac-Toe. We are going to make use of this point, to prove the convergence.

We know that in the min-max game of optimizing the policies of either player the policy reached at the i^{th} iteration will be $\geq$ the policy at the $(i-1)^{th}$ iteration. Thus, a policy from which an agent has learnt in the past cannot beat any of the future policies of the agent, provided the agent and the environment progressively produce optimal policies.

And since there exist only a finite number of policies, the 2^{nd} player will eventually reach one of its optimal policies which guarantee its win. Now, there exist multiple strategies which let the second player guarantee its win, but all these policies are going to have the same value function. (Due to Banach's Fixed Point theorem)

Now at this point, the final policy simply depends on the implementation details and how we break the tie among optimal policies. I simply used the `max` function in python which breaks the tie according to the index and hence it will always give the same policy since the value function of all the optimal policies will be the same. Since, the policy for the second player converges, the policy for the first player also converges since its environment does not change.

Hence, the policies of both the players converges.

Properties:

Since there exists an optimal strategy for the second player, we see that after some iterations when the policies converge, Player 2 always wins.

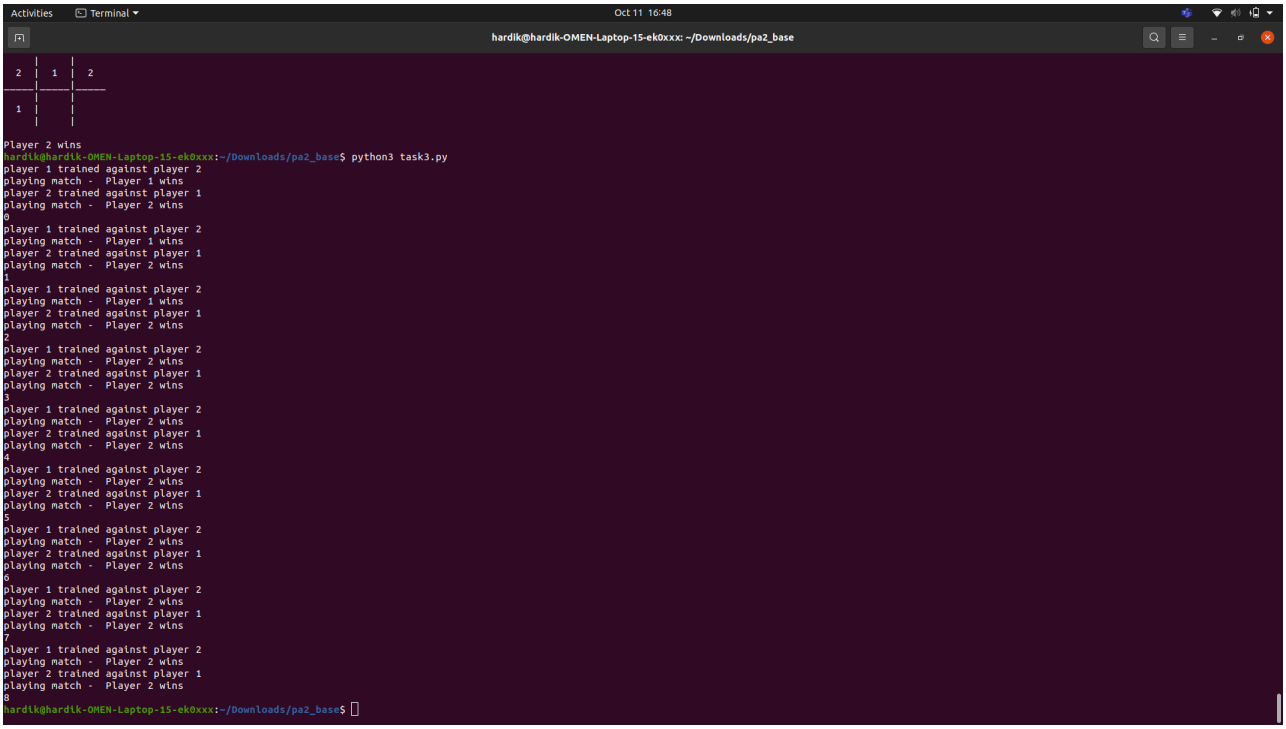

Figure 1: The result of the players playing against each other in successive iterations

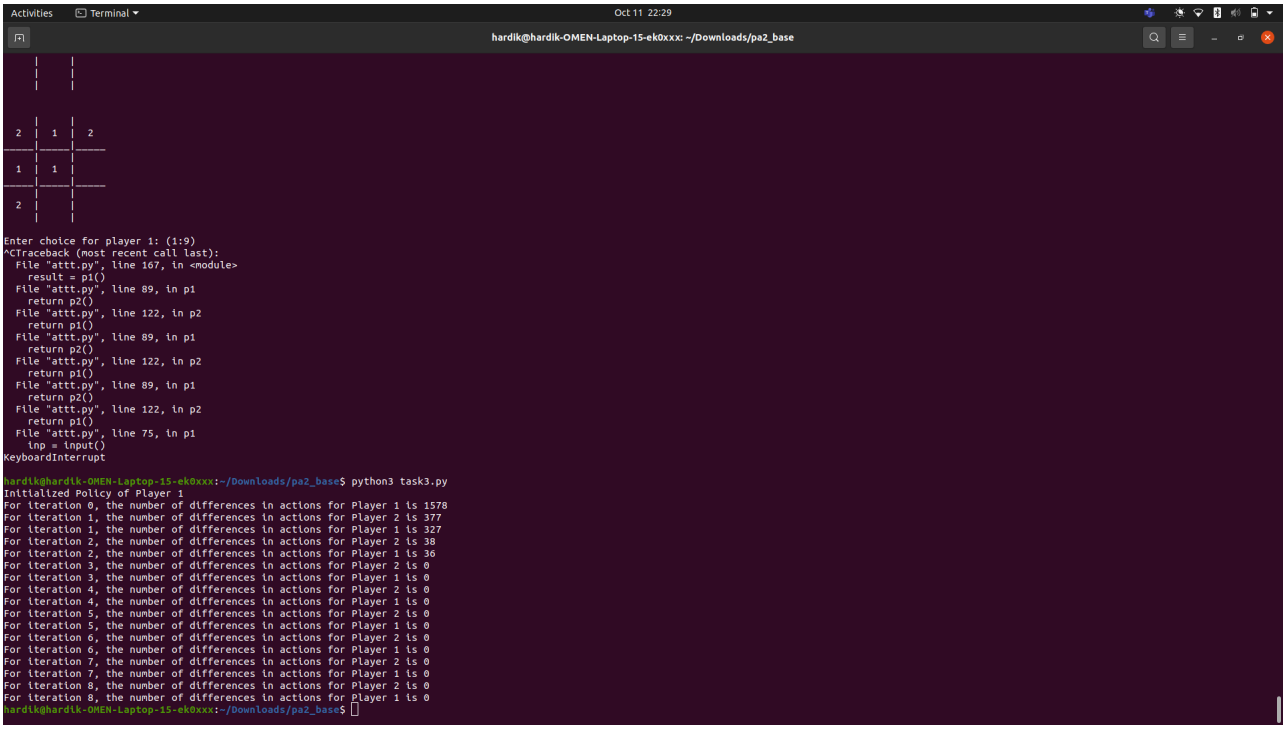

Figure 2: Result of `Task3.py` (differences in the policies of a player)

We can see here the number of changes in the policy generally decreases as the number of iterations increase and eventually becomes 0 for both the players.