Planning in the Taxi Domain

Deepanshu Yadav (2019CS10343)

Harikesh (2019CS10355)

PART A (Computing Policies):

1 a) Formulated the Taxi Domain problem as MDP:

The State space(S) is the set of all possible states of the agent in the MDP. In the Taxi Domain planning we can totally define the MDP using three things. The taxi location I.e. (Xt , Yt) , the passenger Location i.e. (Xp, Yp) and the destination of the passenger i.e. (Xd, Yd). Using only this information we can define the next state of our MDP. Now, the taxi can be in any of the 5*5 positions. The passenger is either picked up or not and hence in total there are only 5 such positions of the passenger. The destination of the passenger is only of the 4 random depots. Hence, in total there are 5*5*5*4 = 500 states in our 5*5 taxi domain problem.
S = {(Xt, Yt), (Xp, Yp), (Xd, Yd)}

The action space(A) is the set of all possible actions our taxi driver can take which includes four stochastic actions which are North, South, East, West and two deterministic actions which are Pickup and Putdown. Hence, in total the action space consists of these 6 actions.
A = {North, West, South, East, Pickup, Putdown}

The transition model i.e. T(s, a, s') is a table containing all possible transitions where each entry implies the probability that action a taken on state s leads to state s'. This is nothing but the conditional probability P(s' |s, a). A few of the conditional probabilities are shown below:
P((0, 1), (4,0),(0,3) / (0,0) , (4,0) , (0,3) , East) = 0 because there is a wall in between (0,0) and (0,1) states.
P((2, 0), (4,0),(0,3) / (2,1) , (4,0) , (0,3) , East) = 0.85 because there is no wall in between (2,0) and (2,1) states (and also moving in the direction given) and also the passenger is not picked up yet by the taxi driver.
P((0, 1), (4,0),(0,3) / (4,0) , (4,0) , (0,3) , North) = 0 because it is not possible to move from (0,1) to (4,0) in one move by taking an action North.
P((2, 0), (4,0),(0,3) / (2,1) , (4,0) , (0,3) , West) = 0.05 because there is no wall in between (2,0) and (2,1) states and also the passenger is not picked up yet by the taxi driver and there are three other possibilities in which the taxi can move randomly and hence 0.15/3 = 0.05.
P((0,0) , (0,1) , (0,3) / (0,0) , (0,0) , (0, 3) ,Pickup) =1 as both taxi and the passenger are in same position and pickup is a deterministic action.
Some of the entries of the Transition model are shown below:
T({(0,0) , (4,0) , (0,3)} , East, {(0, 1), (4,0),(0,3)}) = 0

T({(2,1) , (4,0) , (0,3)} , East, {(2, 0), (4,0),(0,3)}) = 0.85
T({(4,0) , (4,0) , (0,3)} , North, {(0, 1), (4,0),(0,3)}) = 0
T({(2,1) , (4,0) , (0,3)} , West, {(2, 0), (4,0),(0,3)}) = 0.05
T({(0,0) , (0,0) , (0,3)} , Pickup, {(0, 0), (0,1),(0,3)}) = 1
………….

The reward model tells about the instantaneous reward obtained by the agent by performing an action on the environment. It is basically represented as R(s, a, s') and it shows the reward obtained by the agent on performing an action a in state s and finally the agent ends up in state s'.
Some of the entries of the reward model are shown below:
R({(0, 0) , (4,4) , (0, 0)} , Pickup, {(0, 0) , (4,4) , (0,0)}) = -10 as taxi and passenger are not in the same position and the Pickup action is attempted.
R({(0, 0) , (4,4) , (0, 0)} , Putdown, {(0, 0) , (4,4) , (0,0)}) = -10 as taxi and passenger are not in the same position and the Putdown action is attempted.
R({(0, 0) , (0,0) , (4, 4)} , Pickup, {(0, 0) , (0,0) , (4,4)}) = -1 as taxi and passenger are in the same position and the Pickup action is attempted. (It is the cost of breathing)
R({(0, 0) , (0,0) , (0, 0)} , Putdown, {(0, 0) , (0,0) , (0,0)}) = +20 as taxi and passenger are in the destination position and the Putdown action is attempted.
R({(2, 0) , (4,4) , (0, 0)} , North, {(2, 1) , (4,4) , (0,0)}) = -1 as it is the cost of breathing.

b) In this part we are implementing a simulator for out Taxi domain problem which takes as input the state of our MDP, the action taken on that state and returns the instantaneous reward as well as the next possible state of our MDP. The stochastic effect is taken care in transition model by taking probabilities of reaching the particular state by taking a particular action on a particular state.

2

a) In this part of the assignment, we are implementing the value iteration method for our taxi domain. We have used the max-norm distance in successive value functions to determine convergence. The discount factor(gamma) is taken as 0.9. The formula used in updating the successive value functions is

The instance used for this data is Starting depot = (0, 0), destination depot = (3,0) and the initial position of taxi = (4,4).

V(s) = max on all actions (Summation of all s'(T(s,a,s')*[R(s,a,s') + gamma*(V*(s'))]))

The value of epsilon chosen = 0.01, iterations for convergence = 34

The value of epsilon chosen = 0.005, iterations for convergence = 36

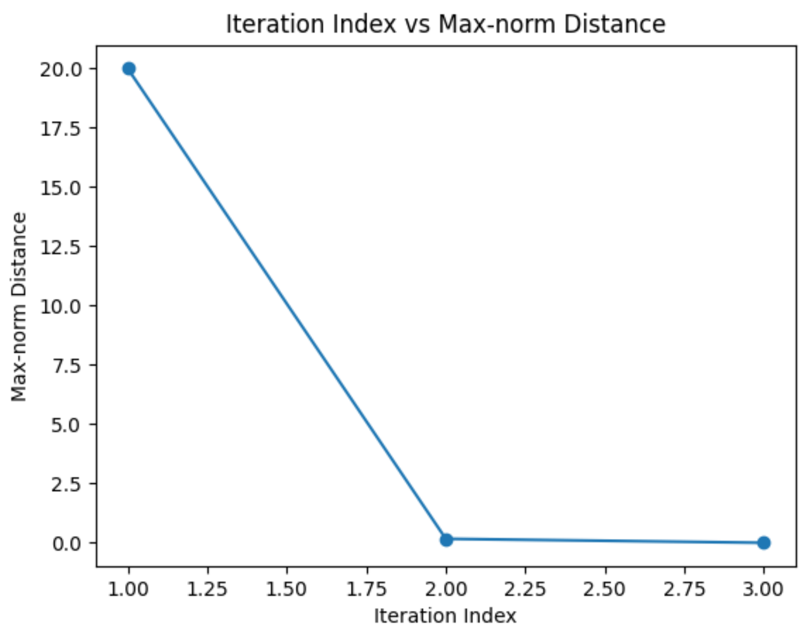
The value of epsilon chosen = 0.001, iterations for convergence = 39

The value of epsilon chosen = 0.0005, iterations for convergence = 40

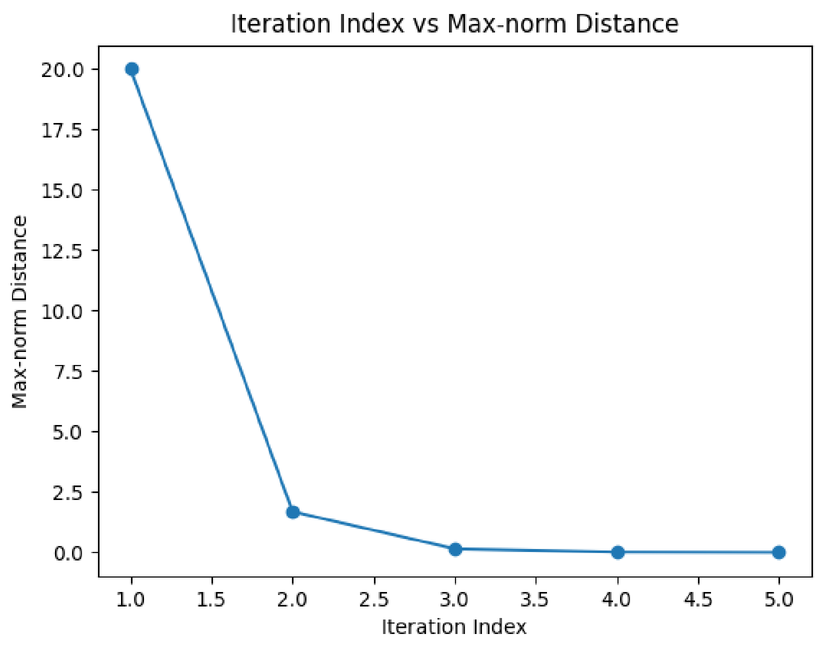
The value of epsilon chosen = 0.0001, iterations for convergence = 43

Which is correct because epsilon is nothing but the maximum error allowed in the value of any state and as we will decrease the value of epsilon, we are more and more wanting more accuracy i.e. we are wanting less and less error in value of any state and to decrease the error the number of iterations must definitely increase.

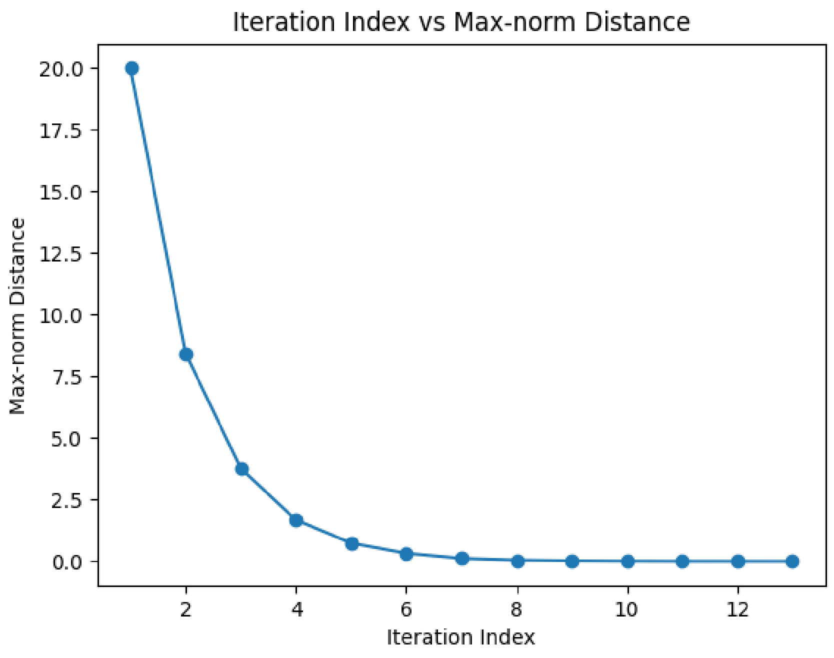
b) Now, we have to study the effect of changing discount factor on rate of convergence. The values of discount factor used are = 0.01, 0.1, 0.5, 0.8, 0.99
The value of epsilon used in this case is 0.001.
Discount factor = 0.01



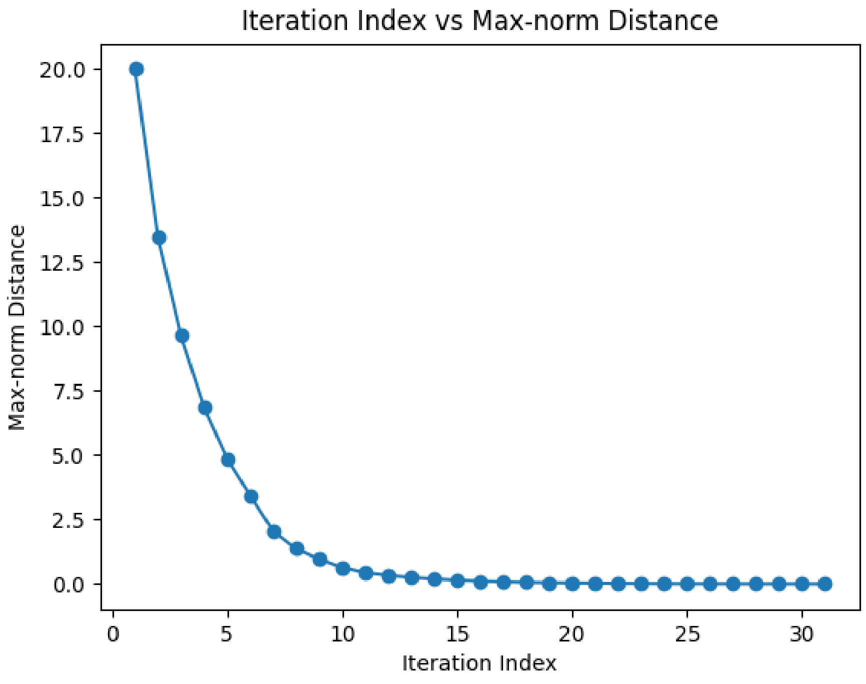
Discount factor = 0.1



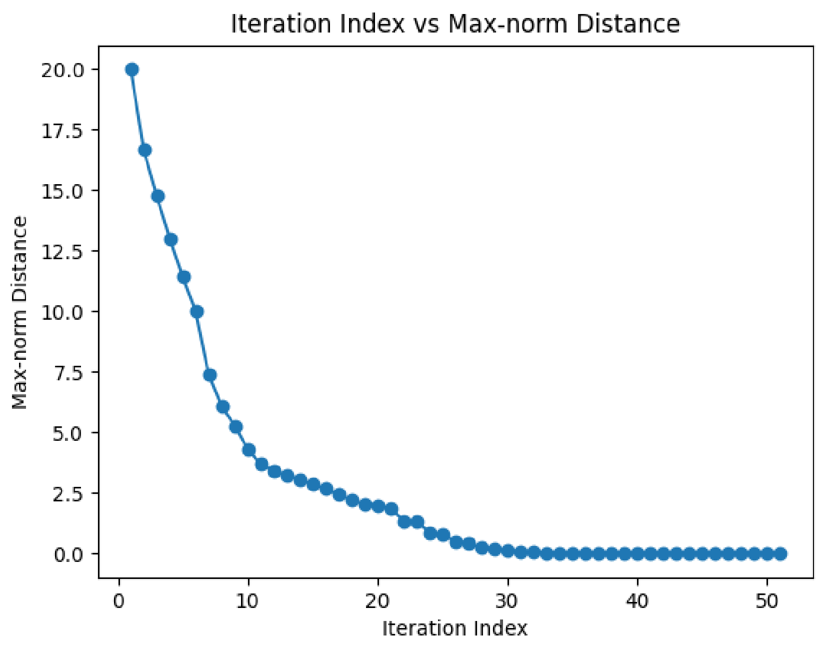
Discount factor = 0.5



Discount factor = 0.8



Discount factor = 0.99



From the above graphs and the variation in the values of discount factor we can clearly see that as the value of discount factor increases, we are getting a greater number of iterations and also the rate of convergence decreases. Because initially when the discount factor was very less, we are propagating less effect of values to the adjacent states and hence, our error bound was reached quickly and as the value of discount factor increases the increase in the number of iterations can be explained as the more effect of adjacent states and hence the error bound will be loose in this case.

c) The instance chosen is:
Y (passenger initial location) = (0,0)
G (passenger destination location) = (3,0)
R (taxi initial location) = (4,4)
We have simulated the policy obtained for discount factor gamma as 0.1 and 0.99. Our state number is based on the position of the taxi i.e. from 0 to 24 and in case if the passenger is also present in the taxi our states are from 25 to 50. Hence, >25 means the passenger is in the taxi and otherwise.

Case 1: Discount factor = 0.1
In case the discount factor is very less the value iteration algorithm is not much influenced by the values in the predecessor states and it has to maximize from the rewards mainly and hence it will just want to reach the destination state and put the passenger there. Here the agent will try to receive reward early and then at last will try to receive negative rewards because the effect of previous values is very less.
The first 20 states (if possible) are: [24, 29, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28]

The first 20 actions (for above states) are: ['North', 'West', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown', 'Putdown'].
Case 2: discount factor = 0.99
In case the discount factor is near to 1 the effect of collection of rewards earlier is not as much as compared to previous case and hence it this case it will be more simple (migrating towards goal) as compared to previous case.
The first 20 states (if possible) are: [24, 23, 18, 17, 12, 11, 10, 5, 0, 25, 30, 35, 36, 37, 38, 33, 28, 3]
The first 20 actions (for above states) are: ['West', 'South', 'West', 'South', 'West', 'West', 'South', 'South', 'Pickup', 'North', 'North', 'East', 'East', 'East', 'South', 'South', 'Putdown'].

Now we will change the initial positions of the passenger and our taxi keeping the state same and then will try to obtain some these states and the preferred actions.
Y (passenger initial location) = (0,4)
G (passenger destination location) = (3,0)
R (taxi initial location) = (2,3)

Case 1: discount factor = 0.1
The first 20 states (if possible) are: [17, 22, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27]
The first 20 actions (for above states) are: ['North', 'North', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup', 'Pickup']

Case 2: Discount factor = 0.99
The first 20 states (if possible) are: [17, 12, 11, 16, 21, 20, 45, 40, 41, 36, 37, 38, 33, 28, 3]
The first 20 actions (for above states) are: ['South', 'West', 'North', 'North', 'West', 'Pickup', 'South', 'East', 'South', 'East', 'East', 'South', 'South', 'Putdown']

On changing the initial passenger position and taxi position we can see that in this case also when the gamma is very much then the discount in the rewards will be very less and hence it will try to take the passenger and then go to the destination but when the discount factor is very less then the effect of rewards will keep on decreasing and the adjacent state values will have more effect. Hence, once the taxi reaches to initial or destination depot then it will keep on doing pickup here also as the rewards are decreasing and the value will have less effect on the next value iteration. Hence, it will keep on doing actions there only.

3. a) We will be implementing the policy iteration algorithm in this part and for that we will be implementing the Policy evaluation step. We will start with any random policy

for our taxi domain and then in every iteration we will be checking if the utility value is maximum corresponding to action of our policy or not. If it is then we will be not changing our argument else we will be taking the argument which correspond to maximum of utilities possible from that state. This is what we will be doing in iterative method and for linear algebra method we will be using the standard library from python. The policy estimation approach is usually slower than the value estimation approach as we are repeating the calculation every time until there is no change in policy in the policy estimation algorithm.

b) Now we will be running our policy iteration till convergence and then we get our optimal policy. To check if it is the case or not, we can compare our result with the policy we obtained using Value Iteration algorithm. We will repeat the above process again to find the policy loss at every iteration between the policy of corresponding iteration and our optimal policy calculated above. We will be then changing the discount factor to check the behaviour.

Instance of taxi domain used:

Y (passenger initial location) = (0,0)

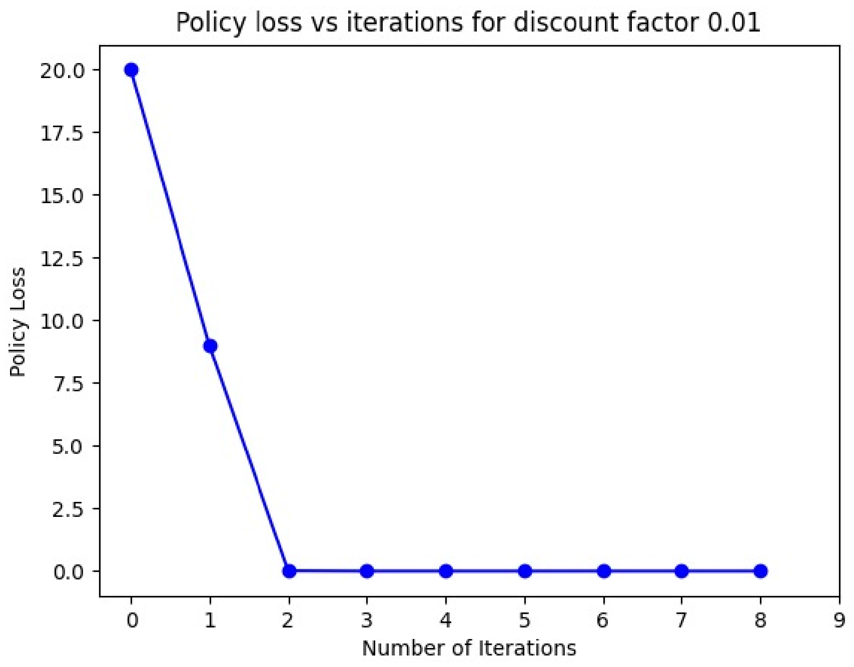
G (passenger destination location) = (4,4)

R (taxi initial location) = (2,4)

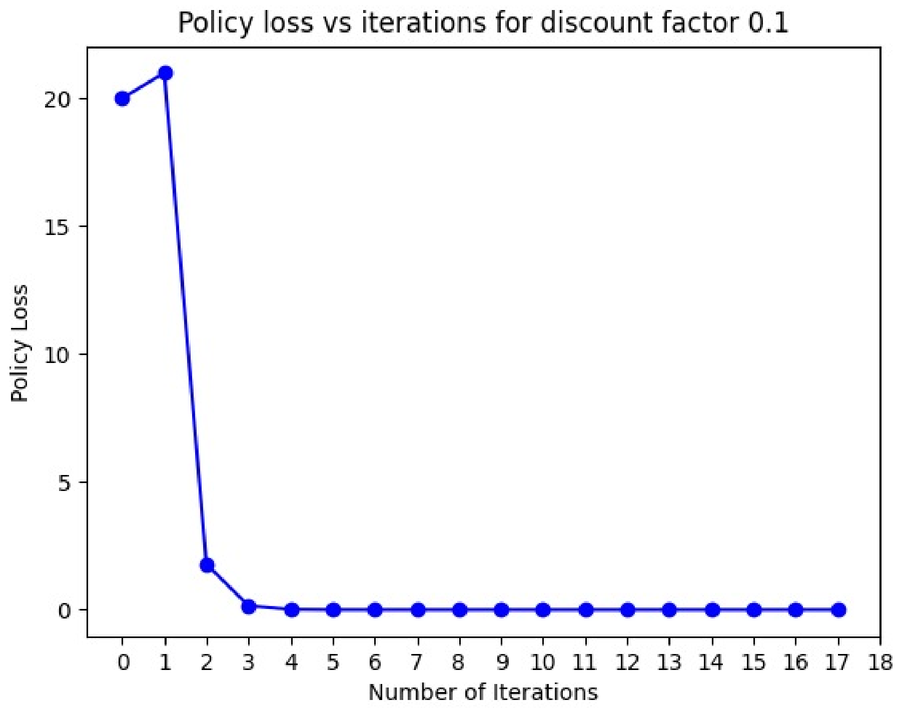
Epsilon = 0.001

Plots on varying the discount factor are as follows:

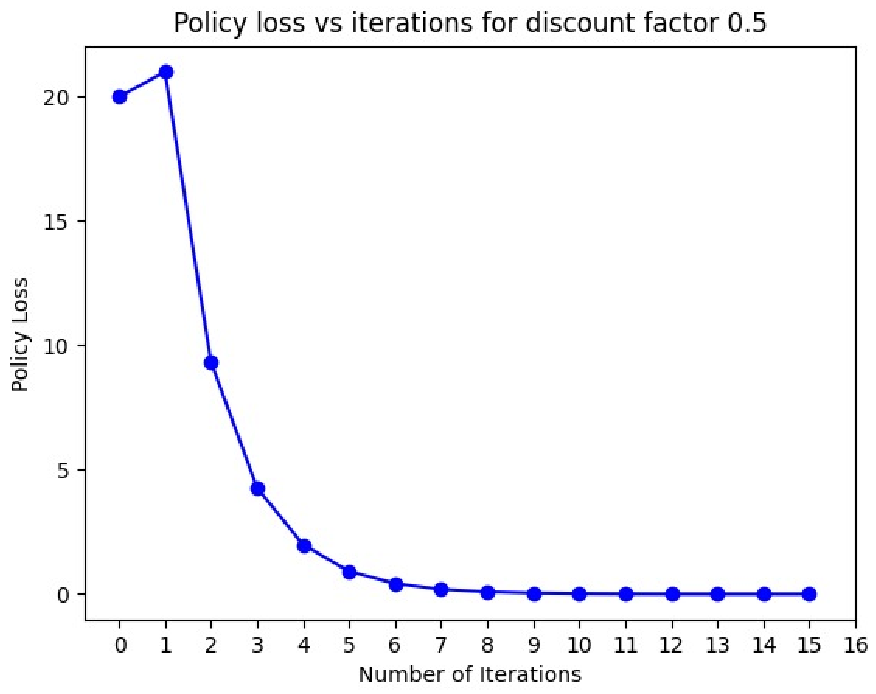
Discount factor (gamma) = 0.01



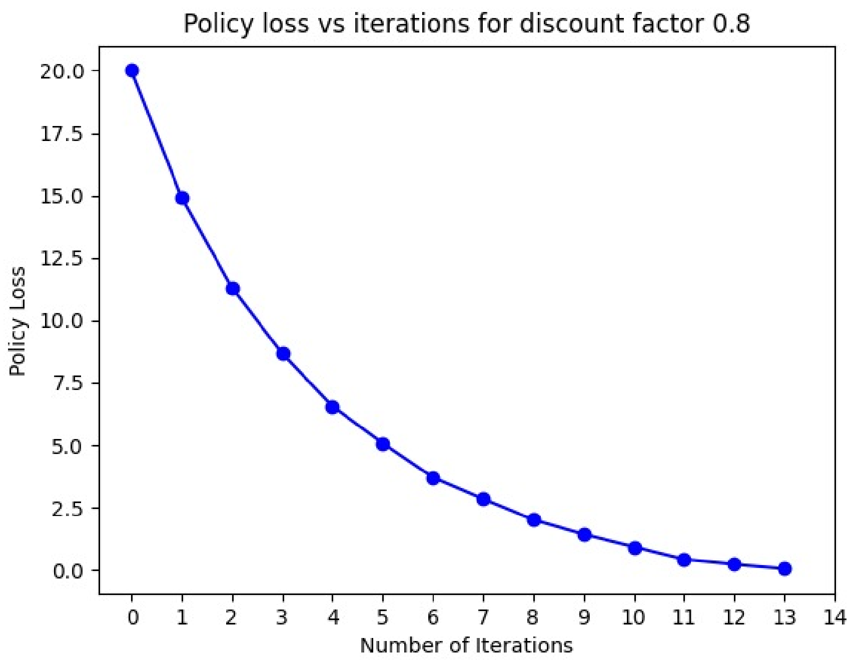
Discount factor = 0.1



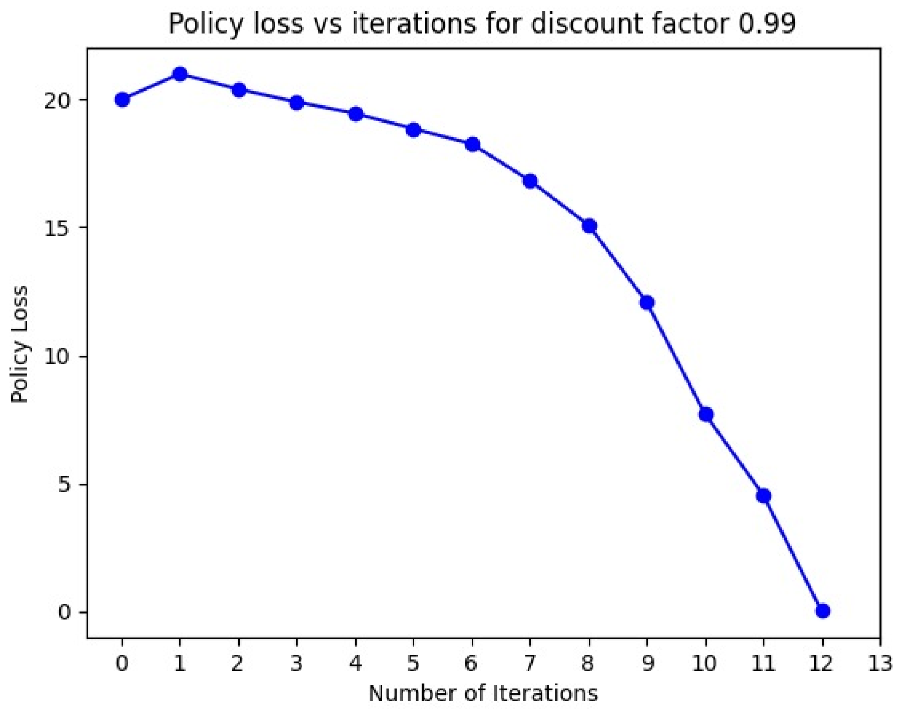
Discount factor = 0.5



Discount factor = 0.8



Discount factor = 0.99



From the above graphs we can conclude as we will be as the iteration number increases our policy will tend to move towards the optimal policy and then after a number of iterations it will become optimal policy i.e. after converging. Now, the optimal policy loss is 0 finally after convergence and it decreases as we will be going to next iteration and our policy will be coming closer and closer to optimal policy. We can also study its relation with the discount factors and can see that as the discount factor increases it will converge after more iterations i.e. for low discount factor it will converge immediately because the later rewards will be highly penalized and it will tend to receive rewards earlier.

PART B (Incorporating Values):

In part B we will be dealing basically with unknown Taxi Domain MDP. The Reward Model and the Transition model are unknown. Hence, to do the reinforcement learning here we will be learning from the past experiences. In this case the next state and instantaneous reward will be provided by the environment when our taxi executes an action and which will be simulated by Q learning.

1.
   a) Implemented the model free approach of Q-learning using 1 step greedy look-ahead policy. We have used an epsilon-greedy policy for exploration with fixed exploration rate (epsilon) = 0.1.
   b) Now we updated the previous Q-learning model to an epsilon-greedy policy and this time the value of epsilon is not constant and is decayed to a value inversely proportional to the iteration number.
   c) We again go back to the implementation of part a i.e. fixed epsilon value and this time while updating the previous value we will be using the SARSA concept i.e. we will also be looking at the previous action which has taken place.
   d) This part includes adding the SARSA method along with the decaying epsilon values for each iteration (here also the decay rate of epsilon is same as part b).

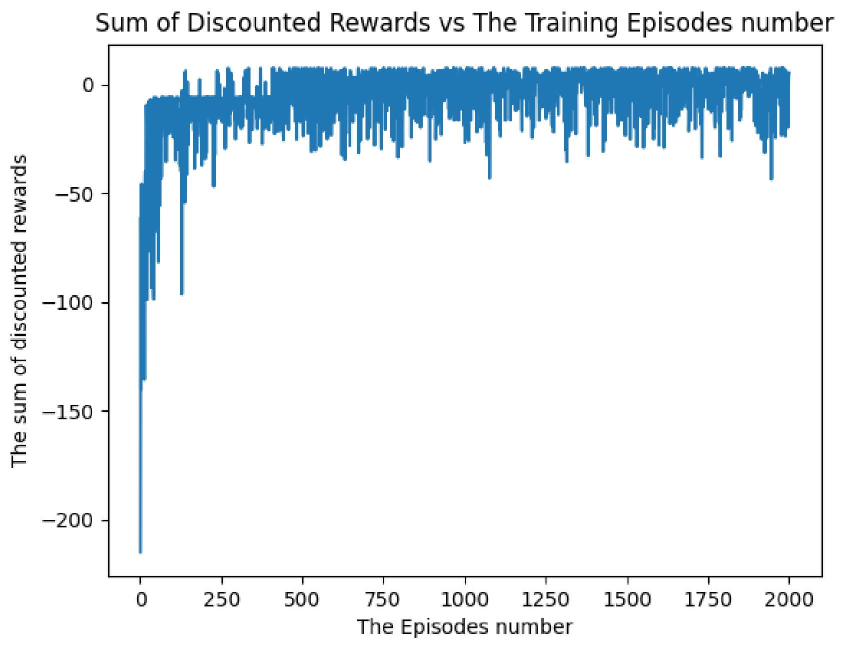
2. Episodes used = 2000
   Learning rate (alpha) = 0.25
   Discount factor (gamma) = 0.99
   We are using the termination condition for our episode as either the passenger dropped at the destination position by the taxi or the episode reaches a maximum length of 500.
   The given graphs show the sum of discounted rewards (Y-axis) and the number of training episodes (X-axis):
   Case a: Q-learning epsilon-greedy policy with fixed exploration rate(epsilon) =0.1

The instance chosen is:

Y (passenger initial location) = (4,4)

G (passenger destination location) = (0,4)

R (taxi initial location) = (3,3)



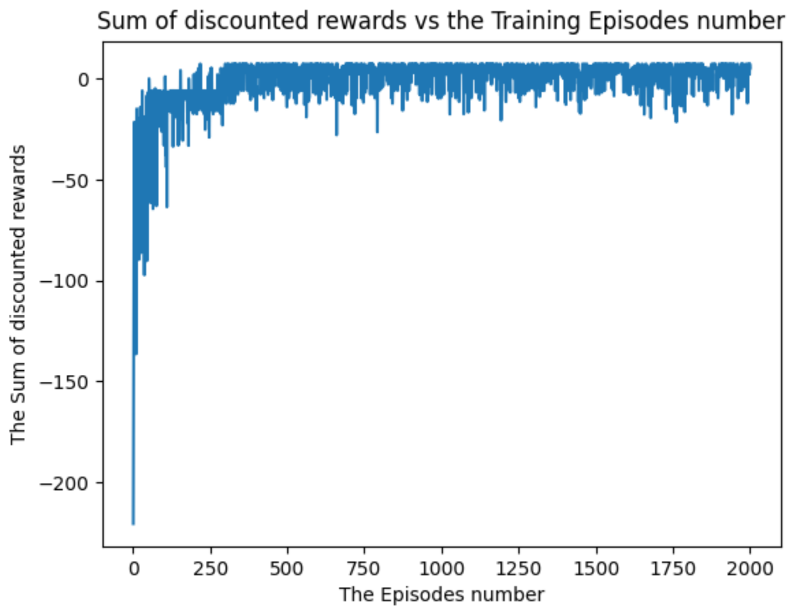
Case b: Q-learning epsilon-greedy policy with decaying exploration rate(epsilon) w.r.t iteration number

The instance chosen is:

Y (passenger initial location) = (4,4)

G (passenger destination location) = (0,4)

R (taxi initial location) = (3,3)



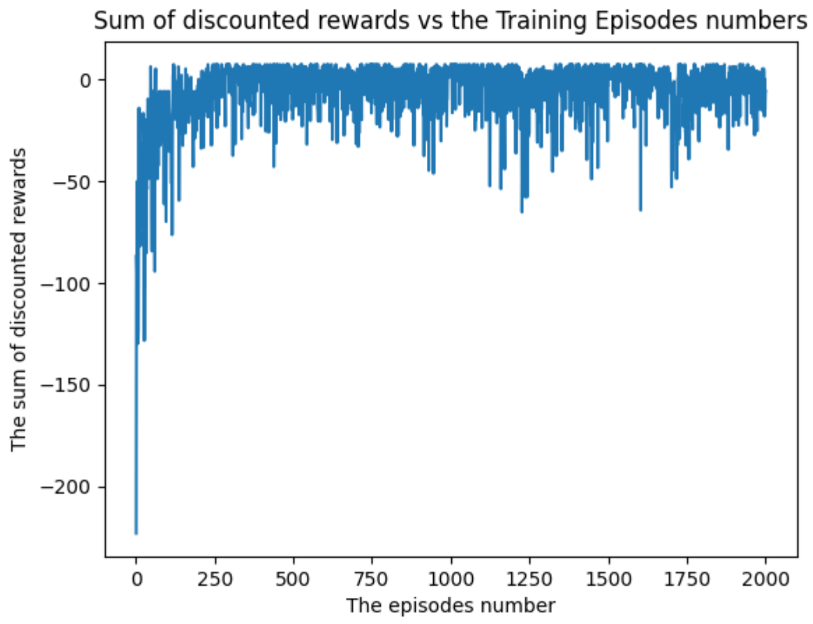
Case c: SARSA epsilon-greedy policy with fixed exploration rate(epsilon) =0.1

The instance chosen is:

Y (passenger initial location) = (4,4)

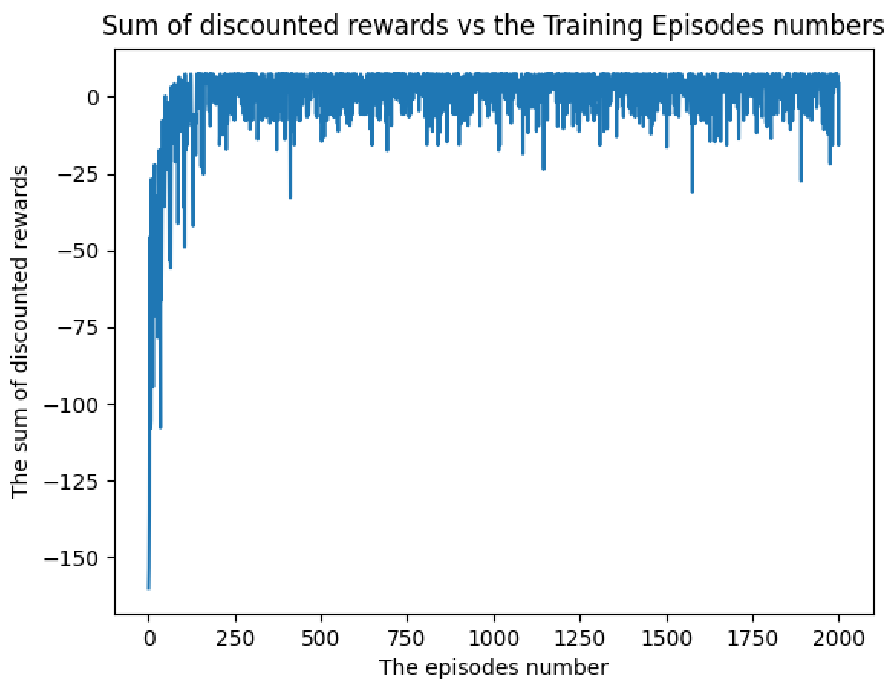
G (passenger destination location) = (0,4)

R (taxi initial location) = (3,3)



Case D: SARSA epsilon-greedy policy with decaying exploration rate(epsilon) w.r.t iteration number

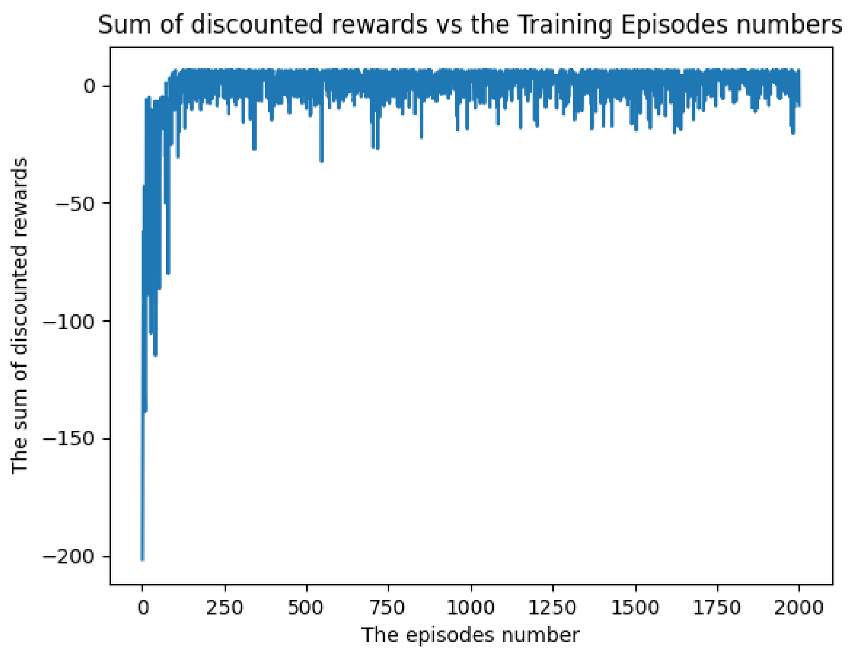
The instance chosen is:
Y (passenger initial location) = (4,4)
G (passenger destination location) = (0,4)
R (taxi initial location) = (3,3)



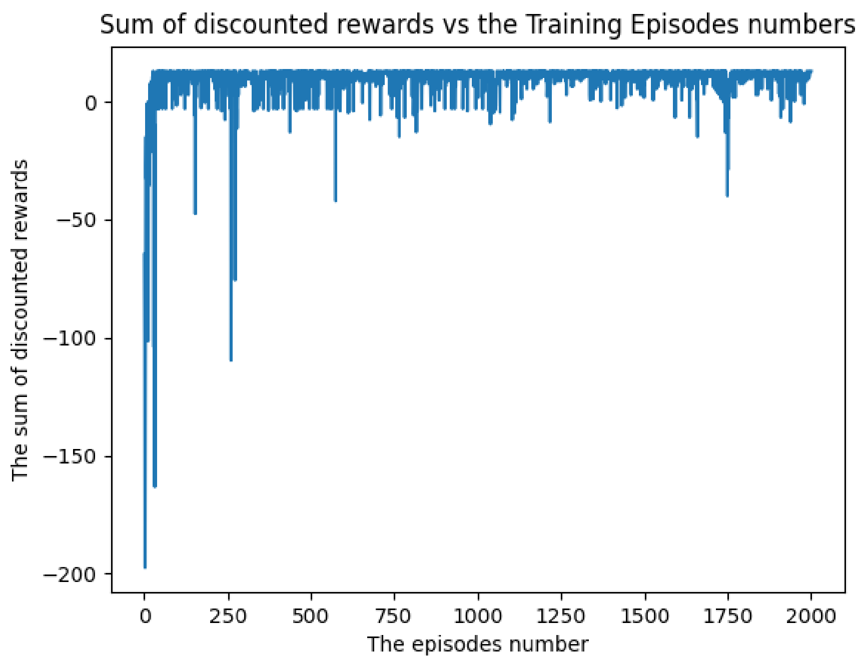
Now as far as the question of convergence arises, we know that because initially the agent is trying to learn from the environment, we can see that it will receive negative rewards initially and once the agent is learning from its past we can see that the sum of discounted rewards is increasing in that case and after a large number of iterations the maximum sum reward will converge in each case and it converges either at small negative value or nearly 0.

3. From the above part we have to find the algorithm which converges to the highest accumulated reward. Almost all of them converge to nearly same reward but the SARSA-learning algorithm with decaying epsilon (initially 0.1) is converging to maximum reward. Now, we will be changing the initial passenger depot and taxi location, keeping the goal same. The value of various hyperparameters and the destination goal used are: G (passenger destination location) = (0,4), initial epsilon = 0.1, alpha = 0.25 and gamma = 0.99.

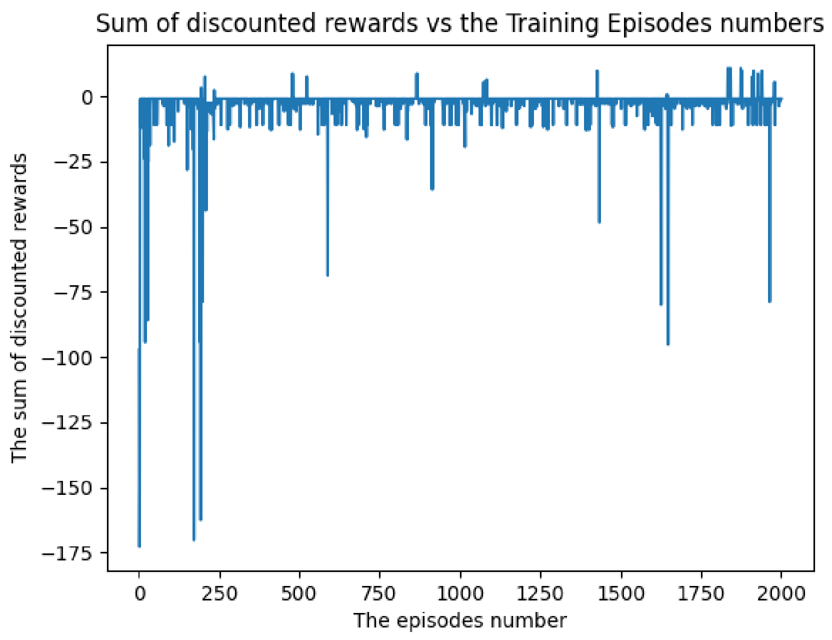
Y (Initial passenger depot) = (4,4) and R (Initial taxi location) = (3,2)



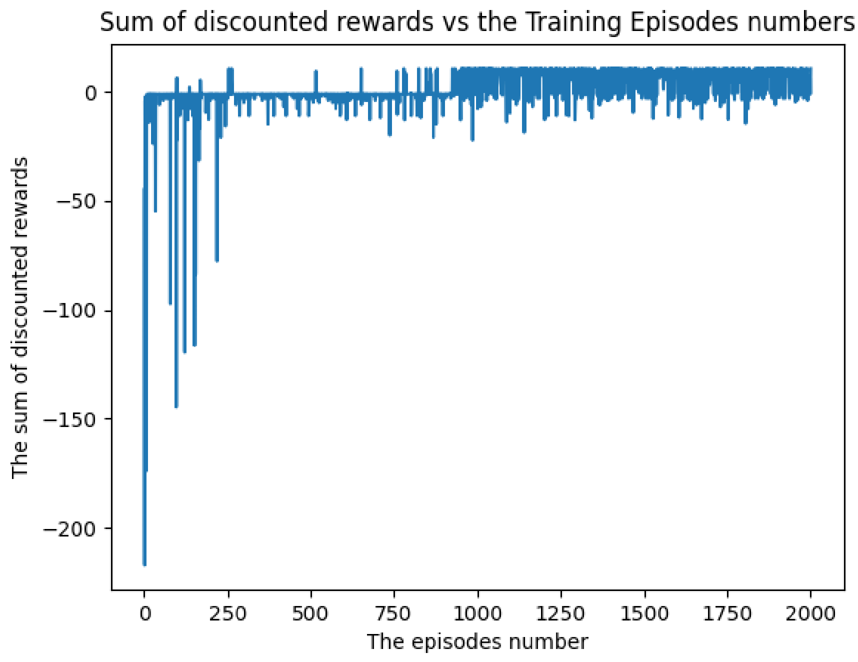
Y (Initial passenger depot) = (0,0) and R (Initial taxi location) = (0,1)



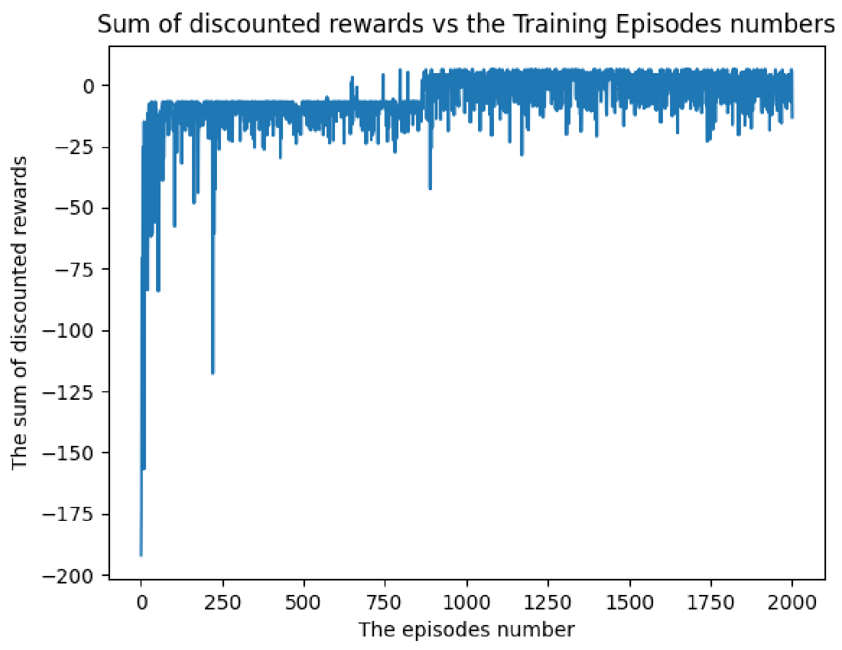
Y (Initial passenger depot) = (0,0) and R (Initial taxi location) = (0,3)



Y (Initial passenger depot) = (4,0) and R (Initial taxi location) = (3,3)



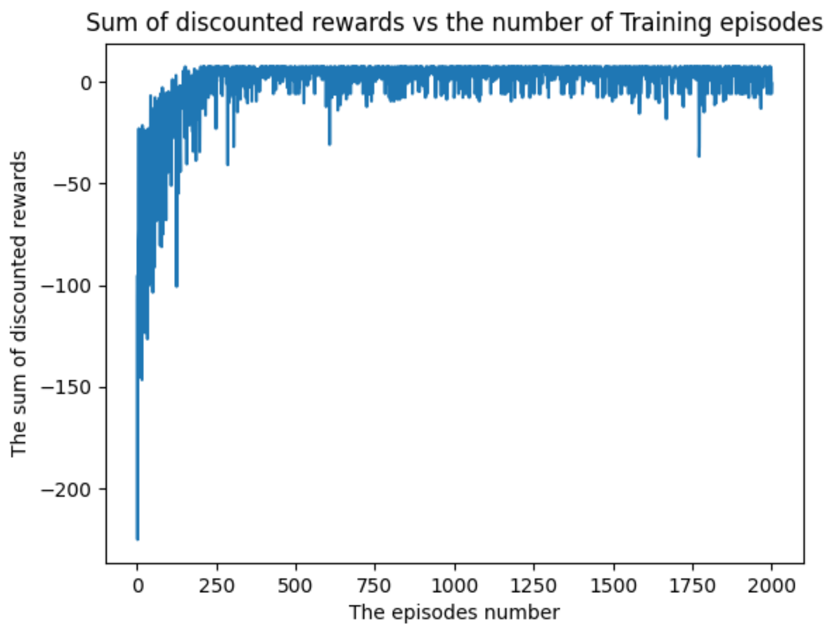
Y (Initial passenger depot) = (0,0) and R (Initial taxi location) = (3,0)



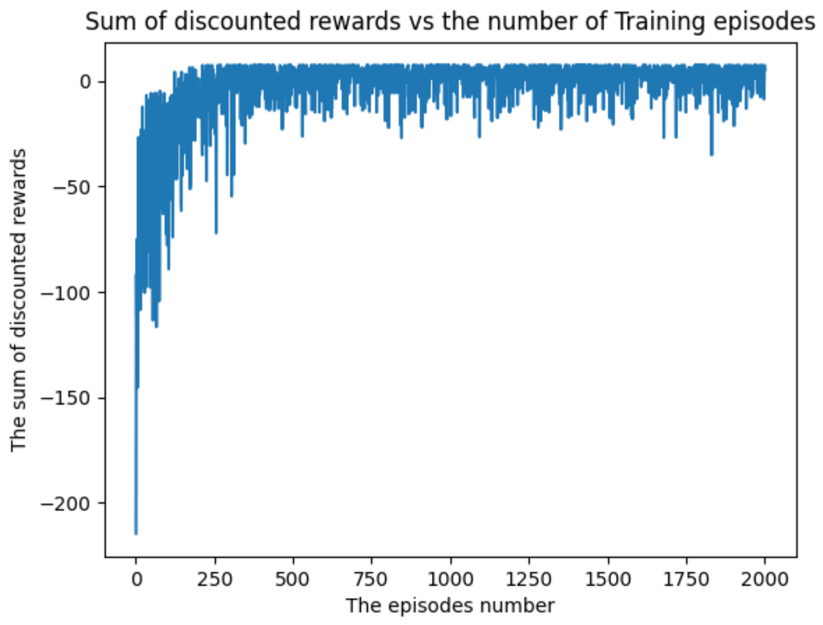
Now, from the above graphs we can clearly see that even if we try to change the initial passenger depot and initial taxi location the sum of discounted reward we are getting is also slightly changing. We can clearly see that the sum of reward where the algorithm is converging is different in these cases. Also, the episodes taken for learning and then increasing the discounted reward is also different in all these cases. That also depend on the initial passenger depot and initial taxi domain. In some case we can also see that it is converging to a negative reward i.e. the taxi is living too much in the domain. But most of the time the reward to which it converges is either slightly negative or nearly 0.

4. In this case we will be dealing with the Q-learning method of Case 1 i.e. Q-learning policy with fixed epsilon.
   Learning rate (alpha) chosen = 0.1
   The instance chosen is:
   Y (passenger initial location) = (4,4)
   G (passenger destination location) = (0,4)
   R (taxi initial location) = (3,3)
   Exploration rate is varying as {0, 0.05, 0.1, 0.5, 0.9}

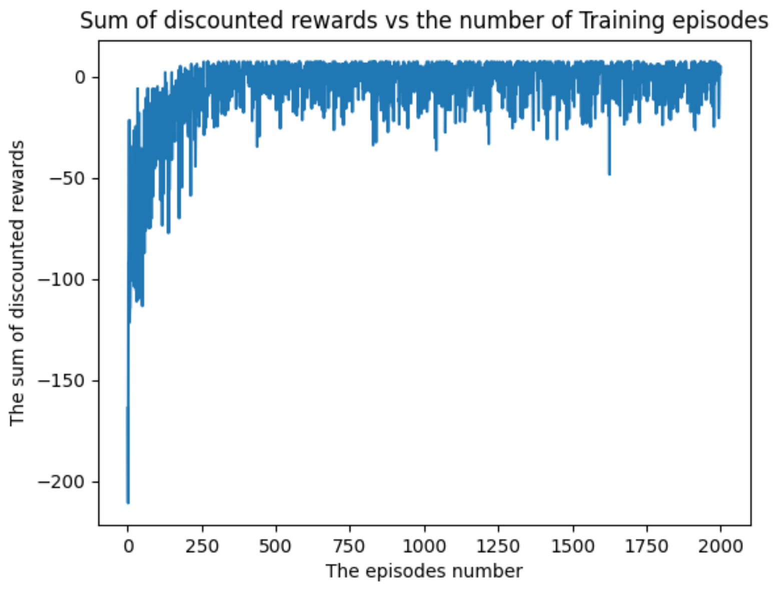
   The plots of all these cases are shown below:
   Exploration rate (epsilon) = 0



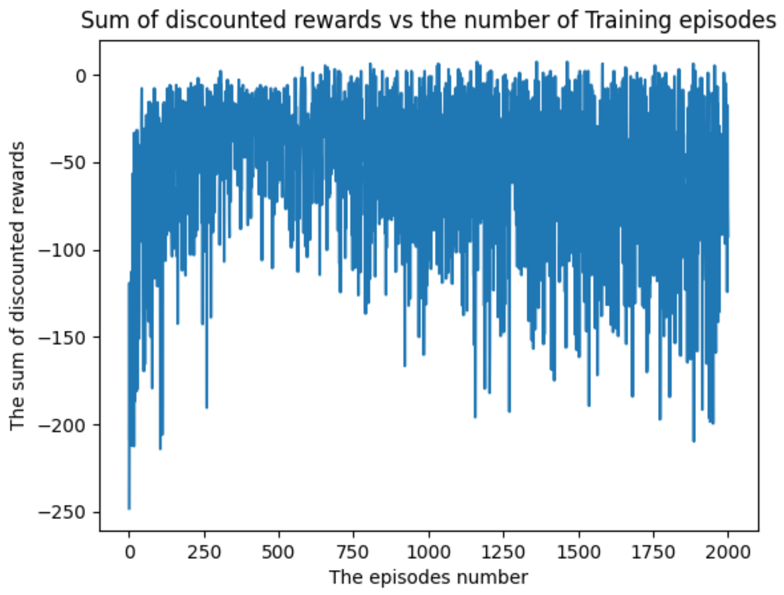
Exploration rate(epsilon) = 0.05



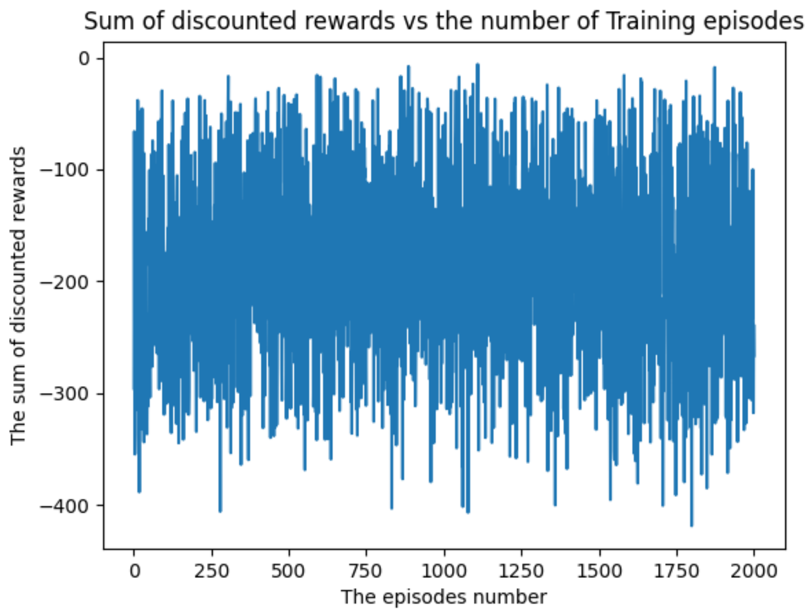
Exploration rate (epsilon) = 0.1



Exploration rate (epsilon) = 0.5



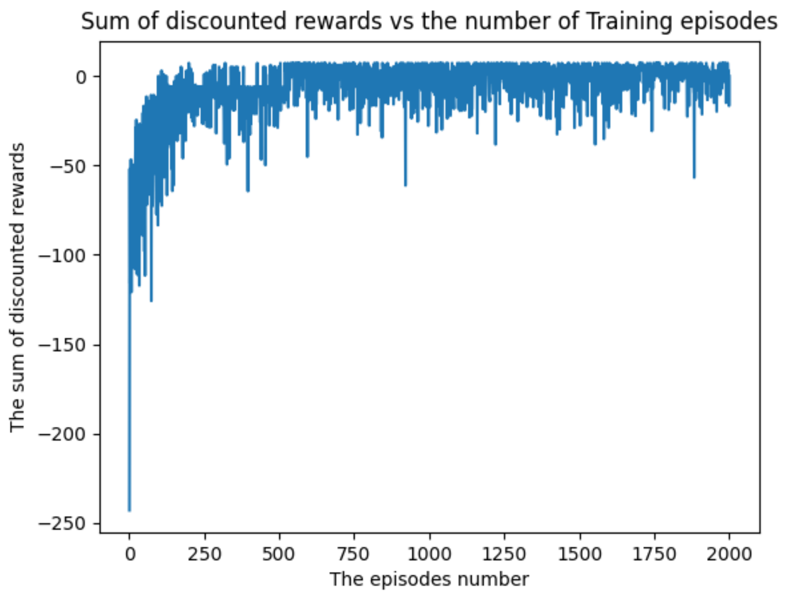
Exploration rate (epsilon) = 0.9



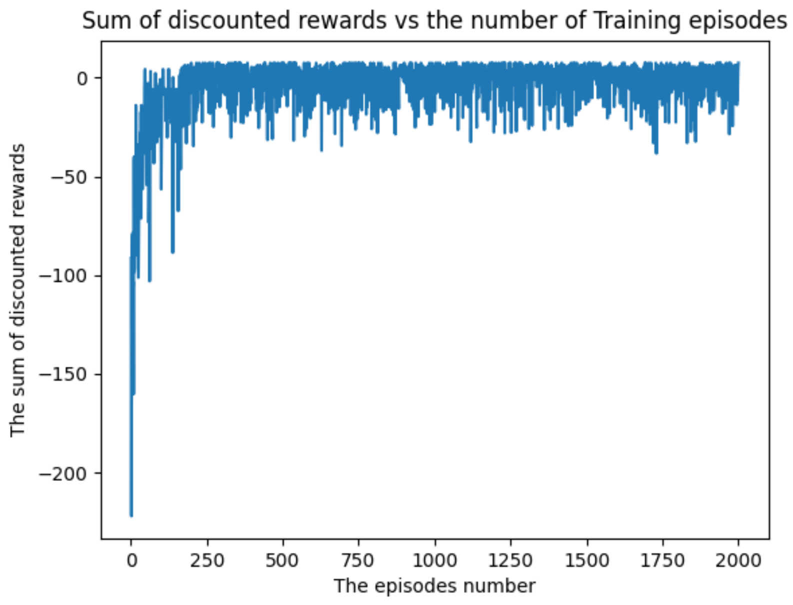
Now we can clearly see from the above graphs that as we will be increasing the exploration rate we are basically randomizing our actions with more probability and because of that we can clearly analyse that on increasing the exploration rate the reward we are getting is very low and as the epsilon values reaches a high value (0.5) the episodes taken to learn from the past increase and in case the value is 0.9 it is so randomized that the reward we get on converging in 2000 episodes is very much less i.e. highly negative.

Next, we will be varying the learning rate alpha as (0.1, 0.2, 0.3, 0.4, 0.5). We will be keeping the instance of the Taxi domain problem same and exploration rate (epsilon) is fixed to 0.1.

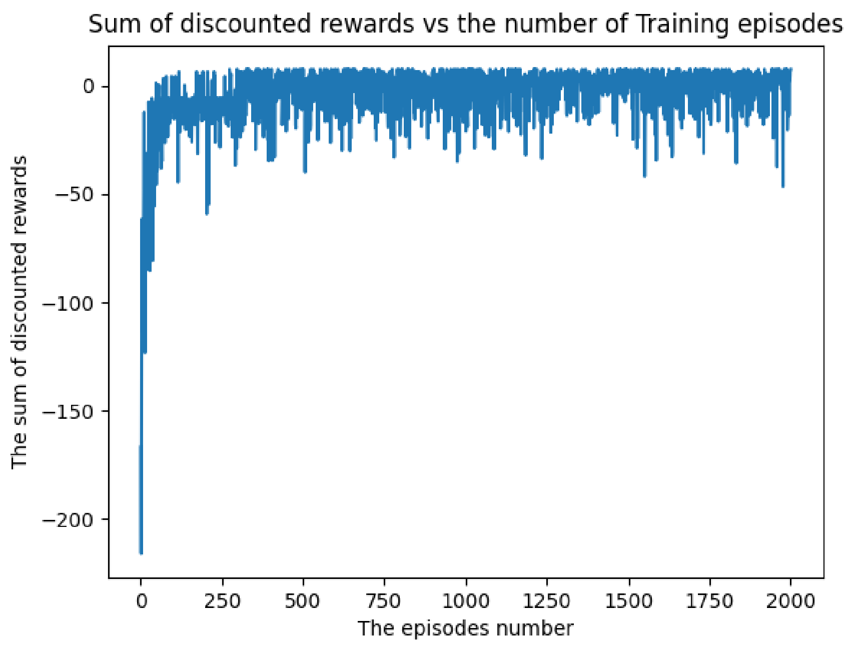
Learning rate (Alpha) = 0.1



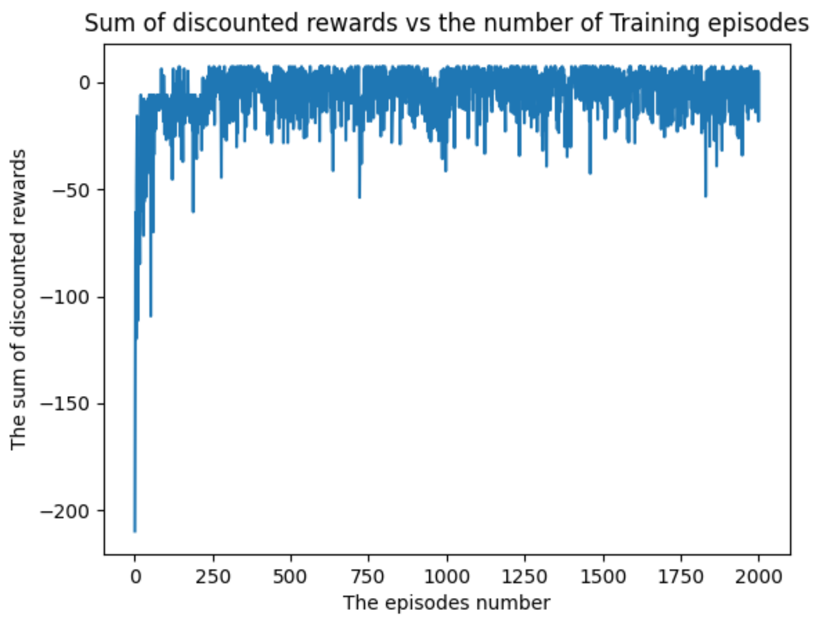
Learning rate (Alpha) = 0.2



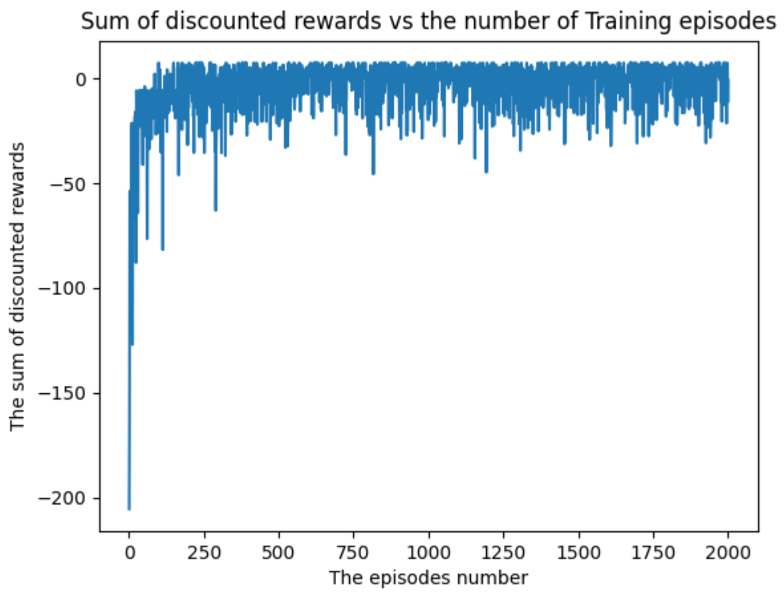
Learning rate (Alpha) = 0.3



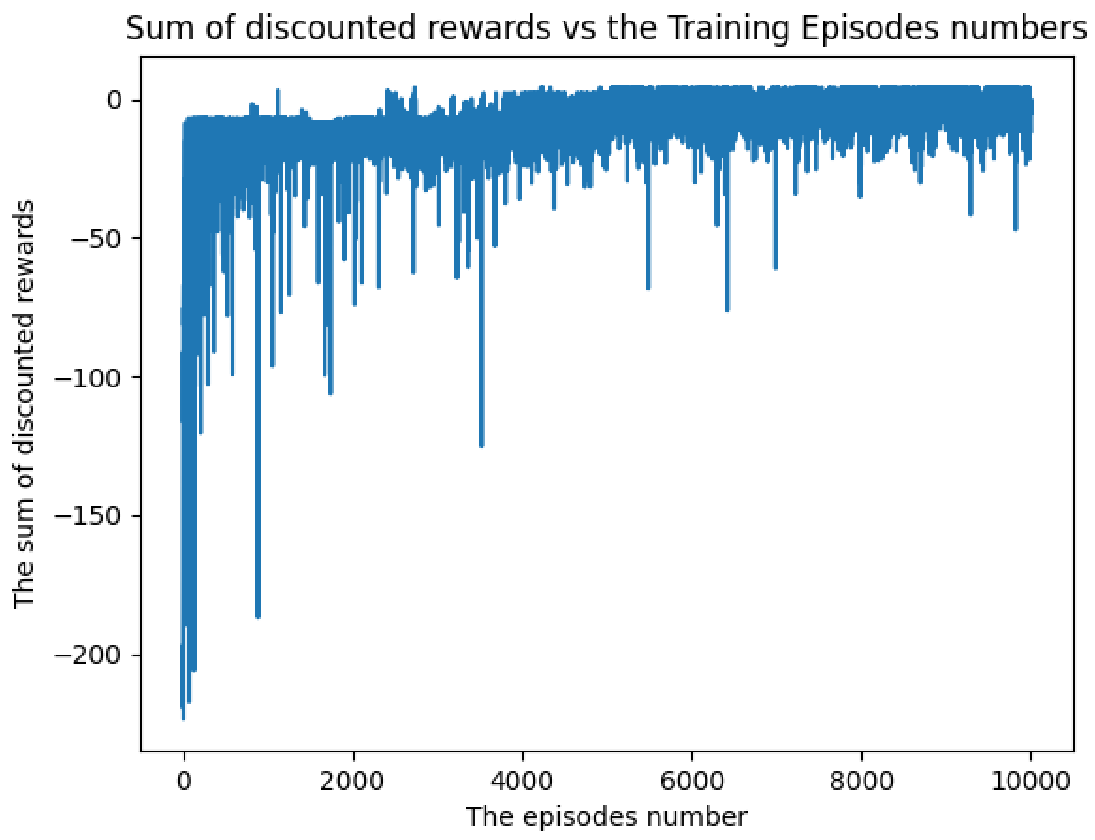
Learning Rate (alpha) = 0.4



Learning rate (alpha) = 0.5



5. Now we will be extending our taxi domain to 10*10 extension. As we have seen that in our previous algorithms SARSA with decaying epsilon is the one that is giving the best rewards and is hence the best learning algorithm. Here we are also changing our learning iterations to 10000. Our hyperparameters we are taking the same as in the previous case which are: epsilon initially = 0.1, alpha =0.25 and gamma = 0.99.
   The instance taken in our new taxi domain is:
   Y (passenger initial location) = (4,0)
   G (passenger destination location) = (6,5)
   R (taxi initial location) = (2,4)
   The plot obtained is as shown below:



As we are seeing from the graph that the sum of discounted rewards in this new domain is going very less as compared to our previous domain because here the grid size is more and hence the living reward will be obviously more accumulated (which is negative). Thus, this leads to a low sum of discounted rewards in our domain.

Now we will be averaging it over 5 instances and then will be sampling it by changing the initial passenger depot and the initial taxi position and the destination position. The five instances used by us are:

Y (passenger initial location) = (4,0)
G (passenger destination location) = (6,5)
R (taxi initial location) = (2,4)

Y (passenger initial location) = (0,1)
G (passenger destination location) = (0,9)
R (taxi initial location) = (3,9)

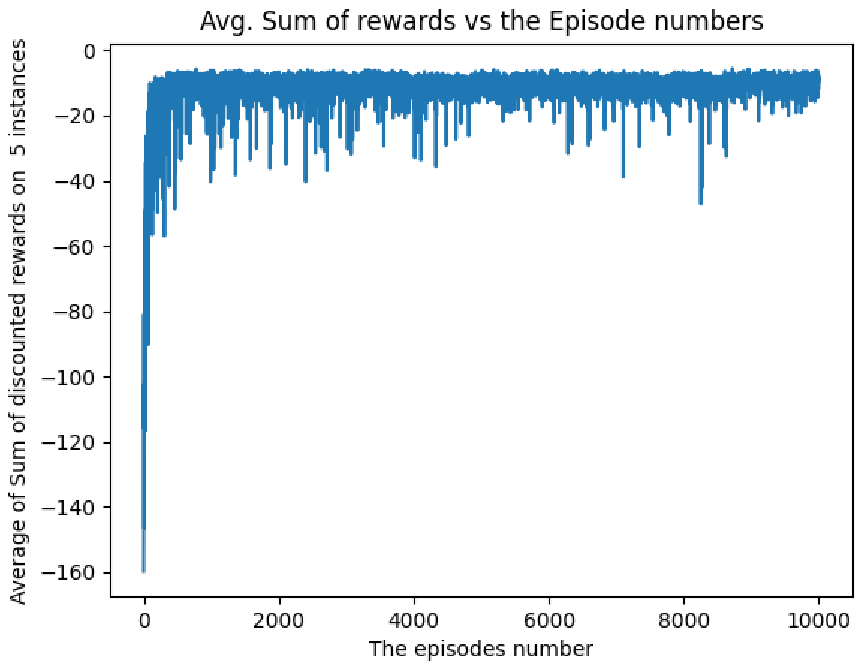
Y (passenger initial location) = (6,5)
G (passenger destination location) = (0,1)
R (taxi initial location) = (2,8)

Y (passenger initial location) = (9,0)
G (passenger destination location) = (8,9)

R (taxi initial location) = (6,6)

Y (passenger initial location) = (8,9)
G (passenger destination location) = (6,5)
R (taxi initial location) = (8,1)

The plot obtained by averaging of these instances is as follows:



From the above graph obtained by averaging the sum of discounted rewards over 5 instances we can see that it also kind of converges. The graph is highly towards negative rewards because the state space has increased significantly here and hence the taxi will be living more and hence the rewards will be more negative as compared to our previous taxi domain which was smaller. (Rewards here are same compared to previous one).