ASSIGNMENT-3 REPORT

Nalla khyateeswar naidu
2019CS10376

Mukku Vamsi Krishna reddy
2019CS10374

PART-A:

1.a

In this problem each state can be distinguished with the help of three parameters, (i) position of taxi, (ii) position of passenger and (iii) passenger is in/out of taxi (In code we have taken it as pickup variable).

State – S (taxi_pos, Pas_pos, pickup)

For taxi_pos and Pas_pos we can have 25 different values for each when they passenger is not in taxi but when passenger is in taxi, they have only 25 values.

So total possible states = 25*25 + 25 = 650 states.

STATESPACE: Collection of all states of [(x, y), (x^1, y^1), 0] and [(x, y), (x, y), 1] where x, y, x^1, y^1 belongs to [0, 24].

And our destination state will be when taxi_pos, Pas_pos = dest_pos, dest_pos and pickup = 0.

Action Space: {right, left, top, bottom, pickup, drop}

TRANSITION MODEL & REWARD MODEL:

We have defined a function in the MDP class, it gives the list of the next possible states, probability of going to that sate and reward we get by going to that state. So it covers both transition model and reward model.

“T (s, a, s1) is the transition probability for going from state “s” to its one of the successor states, s1 by performing action a. At every state, the probability when the actions pick up or put down is taken is 1. For other actions, say north, it has a probability of 0.85 to go to north if north is present and 0.05 each to go to other directions. If north is not permitted (there is a wall), the 0.85 will be used to be in the same state. Similar is the case for east, west, and south. This is the Transition model.”

"Reward model, R (s, a, s1) is the reward given while going from state "s" to its successor state, s1 by performing action a. Now, if action is north, east, west, or south, the reward will be -1. If the action is 'pick up' or 'put down', and the passenger is not in the same place as the taxi, it gives -10 reward. If the 'put down' action leads to an end state, it gives +20 reward. In other cases, the reward is -1."

2.a

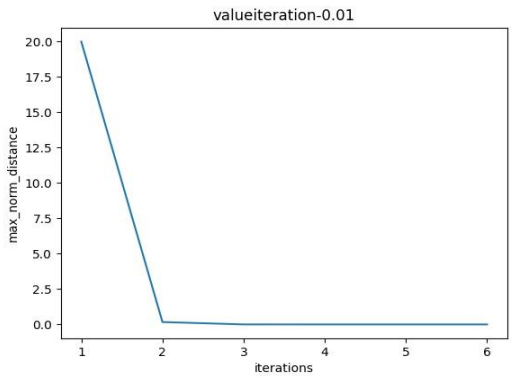
Epsilon used →1e-10

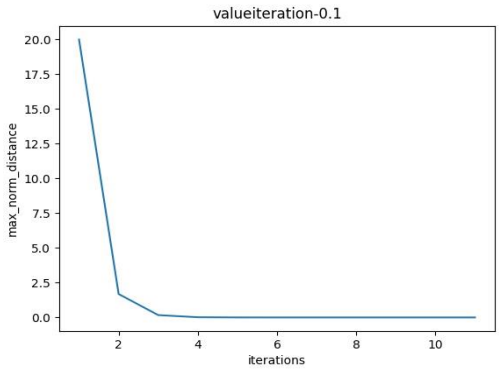
Number of iterations = 60

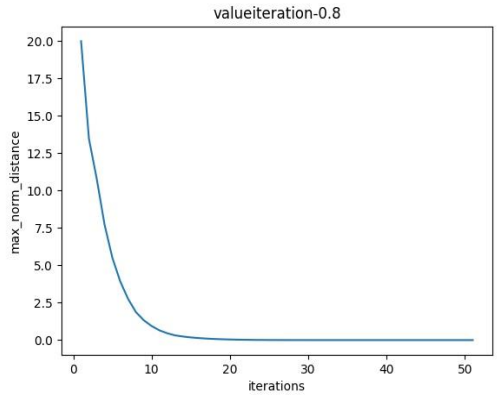
```
[4, 4] [4, 1] 0          -4.221981176490496       top
[4, 4] [4, 2] 0          -4.252094557987019       top
[4, 4] [4, 3] 0          -0.6204951293429393      left
[4, 4] [4, 4] 0          -0.43889925092540816     pickup

total iterations = 60
```

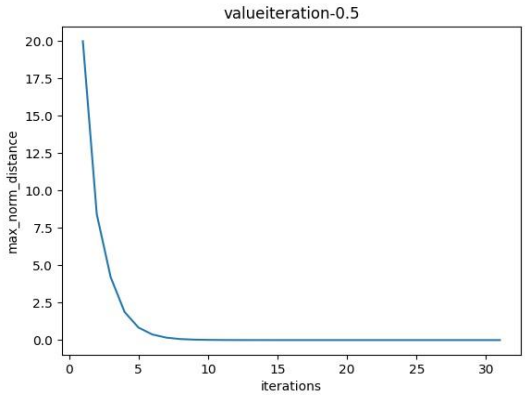
2.b

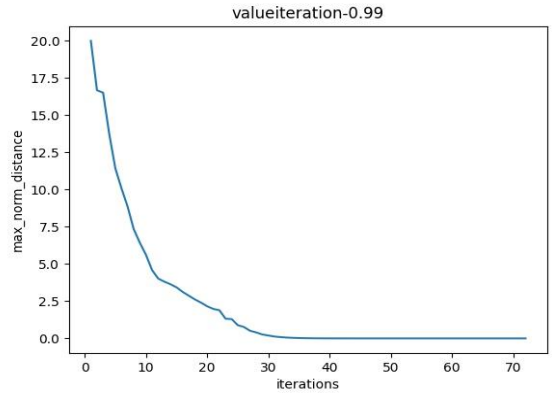
The plotted graphs are max_norm_distance vs iterations.











We can observe that for increase in gamma it takes larger iteration to converge this is because as we increase gamma the importance that we will give to the neighbours increases, so it will take more iterations, for all the sates to achieve the convergence combiningly.

2c.

Taxi initial : [1,1] pas initial : [0,0] des_pos = [4,3]

Iteration 11

s	V(s)	pi(s)
[0, 0] [0, 0] 1	-1.1111103305066095	bottom
[0, 0] [0, 0] 0	-1.1111110330630225	pickup
[0, 0] [0, 1] 0	-1.111111040913894	right
[0, 0] [0, 2] 0	-1.1111111111	right
[0, 0] [0, 3] 0	-1.1111111111	right
[0, 0] [0, 4] 0	-1.1111111111	right
[0, 0] [1, 0] 0	-1.1111110371577584	bottom
[0, 0] [1, 1] 0	-1.1111110368379638	right
[0, 0] [1, 2] 0	-1.1111111105244038	bottom
[0, 0] [1, 3] 0	-1.1111111105244038	bottom
[0, 0] [1, 4] 0	-1.1111111111	right
[0, 0] [2, 0] 0	-1.1111110412352727	bottom
[0, 0] [2, 1] 0	-1.1111110330630225	bottom
[0, 0] [2, 2] 0	-1.1111110330630225	bottom
[0, 0] [2, 3] 0	-1.1111110330630225	bottom
[0, 0] [2, 4] 0	-1.111111110492426	bottom
[0, 0] [3, 0] 0	-1.1111111106136042	bottom
[0, 0] [3, 1] 0	-1.1111111105244038	bottom
[0, 0] [3, 2] 0	-1.1111111105244038	bottom
[0, 0] [3, 3] 0	-1.1111110330649037	bottom

Iteration 71

s	V(s)	pi(s)
[0, 0] [0, 0] 1	9.518845444275236	bottom
[0, 0] [0, 0] 0	8.423656989832484	pickup
[0, 0] [0, 1] 0	8.212204698415615	right
[0, 0] [0, 2] 0	3.1371212652637634	bottom
[0, 0] [0, 3] 0	3.1732699120366745	bottom
[0, 0] [0, 4] 0	1.2085151306816224	bottom
[0, 0] [1, 0] 0	8.286726311412448	bottom
[0, 0] [1, 1] 0	8.349097853684743	right
[0, 0] [1, 2] 0	5.69255467447363	bottom
[0, 0] [1, 3] 0	5.694503711275745	bottom
[0, 0] [1, 4] 0	3.5846881655925102	bottom
[0, 0] [2, 0] 0	8.134212650846536	bottom
[0, 0] [2, 1] 0	8.423656989832484	bottom
[0, 0] [2, 2] 0	8.423656989832482	bottom
[0, 0] [2, 3] 0	8.422447035496889	bottom
[0, 0] [2, 4] 0	5.9608706029515375	bottom
[0, 0] [3, 0] 0	5.3715357567918245	bottom
[0, 0] [3, 1] 0	5.739022938267601	bottom
[0, 0] [3, 2] 0	5.742345280446993	bottom
[0, 0] [3, 3] 0	8.411335074094412	bottom

Taxi initial : [3,3] Pas initial : [0,4] des_pos : [4,3]

Iteration 11

s	V(s)	pi(s)
[0, 0] [0, 0] 1	-1.1111103305066095	bottom
[0, 0] [0, 0] 0	-1.1111110330630225	pickup
[0, 0] [0, 1] 0	-1.111111040913894	right
[0, 0] [0, 2] 0	-1.1111111111	right
[0, 0] [0, 3] 0	-1.1111111111	right
[0, 0] [0, 4] 0	-1.1111111111	right
[0, 0] [1, 0] 0	-1.1111110371577584	bottom
[0, 0] [1, 1] 0	-1.1111110368379638	right
[0, 0] [1, 2] 0	-1.1111111105244038	bottom
[0, 0] [1, 3] 0	-1.1111111105244038	bottom
[0, 0] [1, 4] 0	-1.1111111111	right
[0, 0] [2, 0] 0	-1.1111110412352727	bottom
[0, 0] [2, 1] 0	-1.1111110330630225	bottom
[0, 0] [2, 2] 0	-1.1111110330630225	bottom
[0, 0] [2, 3] 0	-1.1111110330630225	bottom
[0, 0] [2, 4] 0	-1.111111110492426	bottom
[0, 0] [3, 0] 0	-1.1111111106136042	bottom
[0, 0] [3, 1] 0	-1.1111111105244038	bottom
[0, 0] [3, 2] 0	-1.1111111105244038	bottom
[0, 0] [3, 3] 0	-1.1111110330649037	bottom

Iteration 71

s	V(s)	pi(s)
[0, 0] [0, 0] 1	9.518845444275236	bottom
[0, 0] [0, 0] 0	8.423656989832484	pickup
[0, 0] [0, 1] 0	8.212204698415615	right
[0, 0] [0, 2] 0	3.1371212652637634	bottom
[0, 0] [0, 3] 0	3.1732699120366745	bottom
[0, 0] [0, 4] 0	1.2085151306816224	bottom
[0, 0] [1, 0] 0	8.286726311412448	bottom
[0, 0] [1, 1] 0	8.349097853684743	right
[0, 0] [1, 2] 0	5.69255467447363	bottom
[0, 0] [1, 3] 0	5.694503711275745	bottom
[0, 0] [1, 4] 0	3.5846881655925102	bottom
[0, 0] [2, 0] 0	8.134212650846536	bottom
[0, 0] [2, 1] 0	8.423656989832484	bottom
[0, 0] [2, 2] 0	8.423656989832482	bottom
[0, 0] [2, 3] 0	8.422447035496889	bottom
[0, 0] [2, 4] 0	5.9608706029515375	bottom
[0, 0] [3, 0] 0	5.3715357567918245	bottom
[0, 0] [3, 1] 0	5.739022938267601	bottom
[0, 0] [3, 2] 0	5.742345280446993	bottom
[0, 0] [3, 3] 0	8.411335074094412	bottom

We can observe that for gamma 0.1 the values are negative indicates that it doesn't look up to the destination, it is just nearer to the greedy approach. And For gamma 0.99 the values becomes positive indicates it looks up to destination and takes the good policy. So for gamma 0.1 it took longer distance to reach destination than the 0.99 gamma.

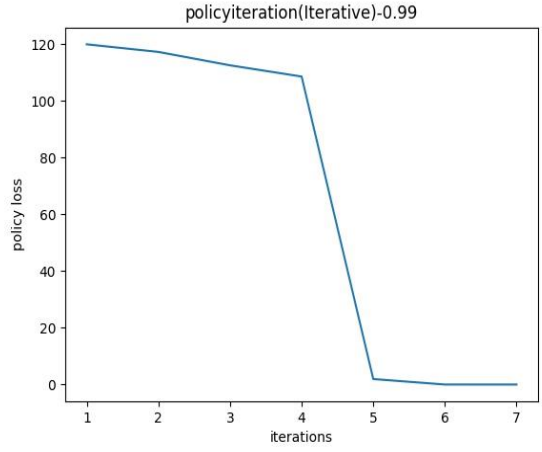
3.a

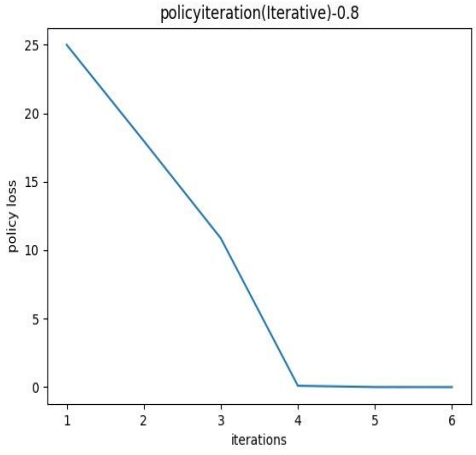
At higher gamma values it is better to calculate the next policy with linear algebra method instead of Iterative method, because for higher gamma it takes more time for convergence of values for a policy. And at low values of gamma usage of both the methods are good enough but usage of Iterative will give us result easily (result in just low number of iterations) than the calculating solutions for the entire matrix.

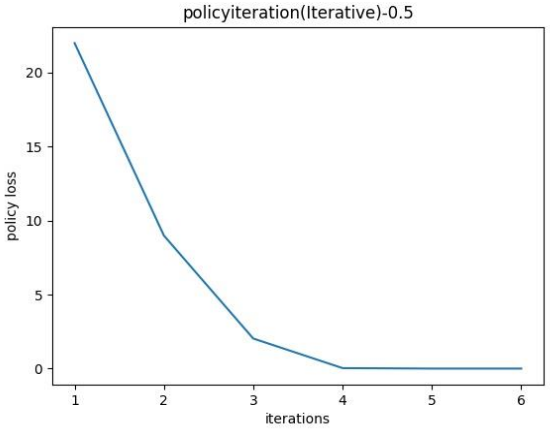
So, low gamma – Iterative High gamma – linear algebra

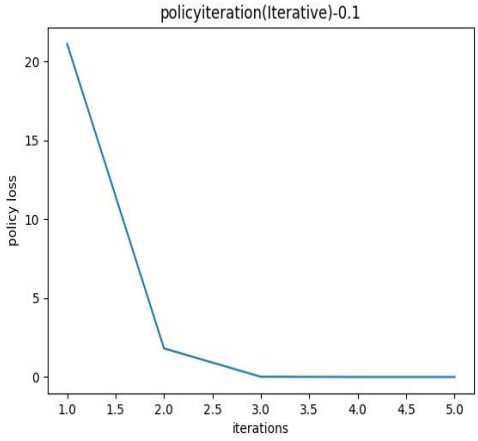
3.b

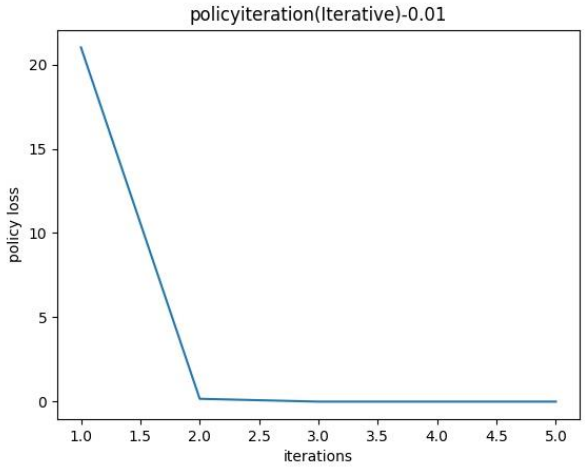
We have drawn graphs for "policy loss vs number of policy changes"











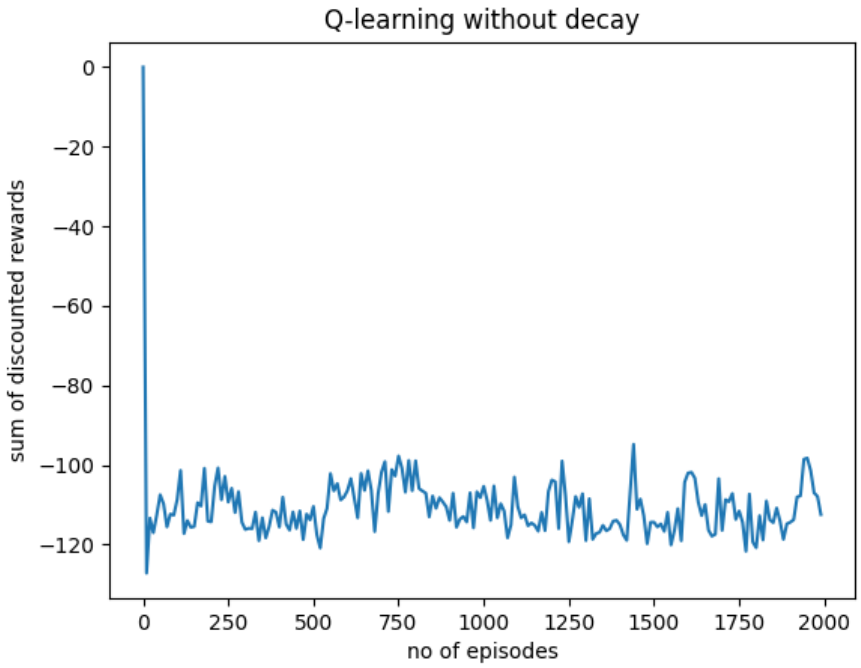
Here we can observe that policy loss is less when it approaches to final policy (as expected because we are trying to converge policies).

And at lower values of gamma the policy loss decreases with in less policies, but higher gamma has more policies with the higher policy loss. At lower values of gamma we may not achieve good policy initially and finally but at higher values of gamma we will be getting better and better policies than the previous and at starting we didn't have much good policies. So these will have higher policy loss at starting stage.

PART B

Part 2

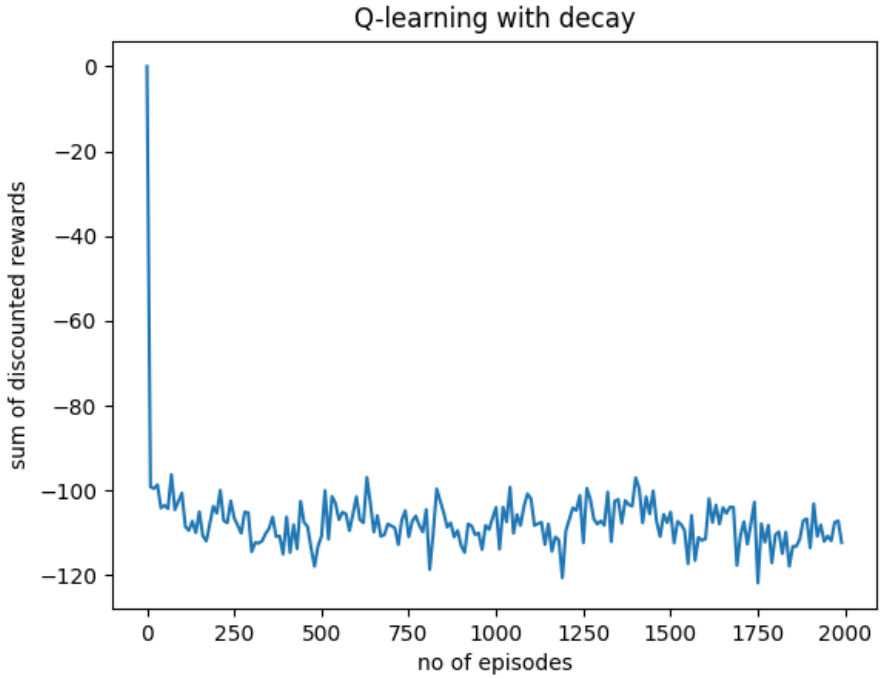
Q-learning without epsilon decay



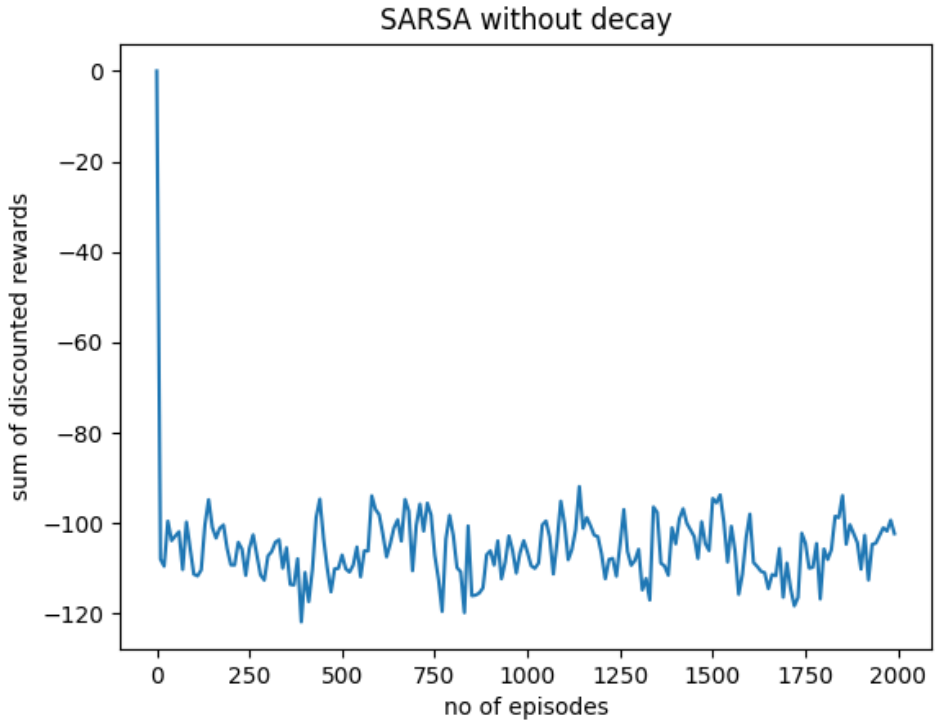
The randomness in the graph is due to change of taxi position with every episode. Usually the reward is -1 for most cases so the discounted reward is (-1)/(1-0.99) is approximately -100 so if the discounted reward is greater than -100 means that is reaching the "destination state". As there is some random is picking action so there is a possibility of getting "-10" reward so the discounted reward is <-100.As there is no decay in the epsilon value there is more variance in the discounted rewards even in the later episodes

Q-Learning without epsilon decay

It is also similar to Q learning because they have randomness like it but the difference comes when the episodes values increase and the epsilon value decrease with iteration we see convergence in the value of discounted reward.

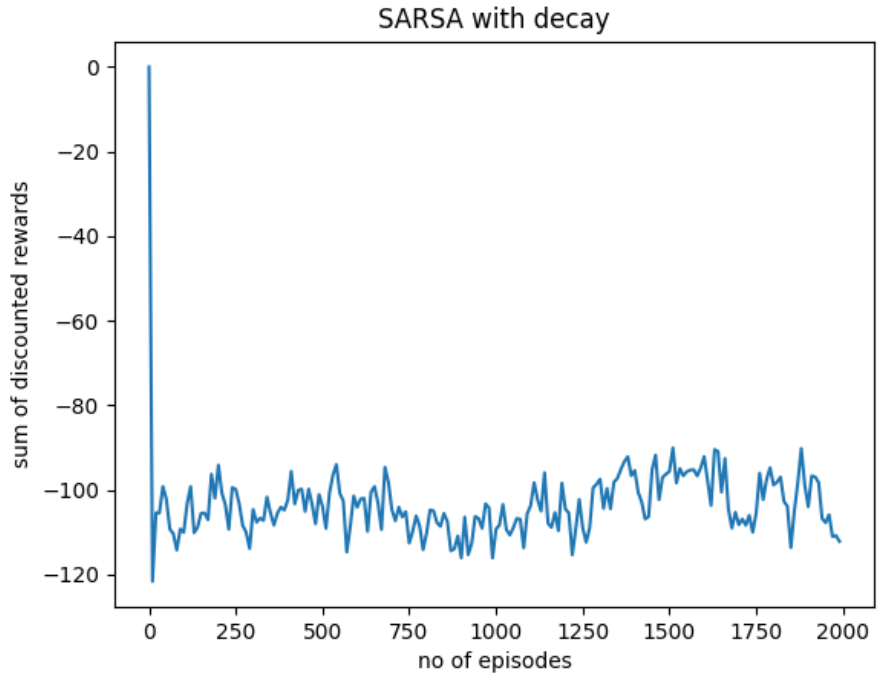


SARSA without decay



It is also similar to Q learning but the difference is it takes the epsilon greedy approach in the next state also so there is more exploration in the next state also so there is more exploration compared to Q learning so there is more variance and it reaches destination faster but doesn't converge well due to more exploration .

SARSA with epsilon decay

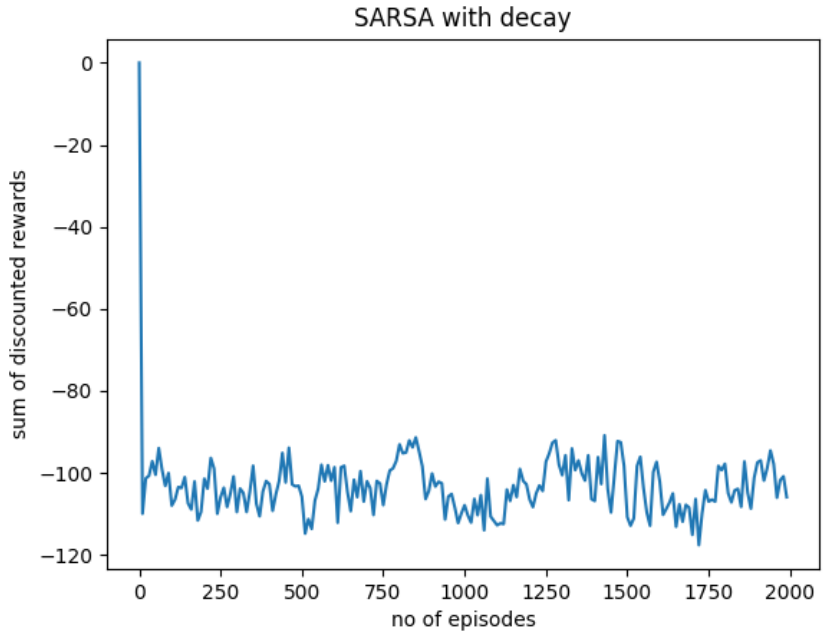


It is similar to SARSA but with increasing in episodes it has the better convergence because it has less exploratory at the start and exploits further so it reaches destination state more often after some episodes .

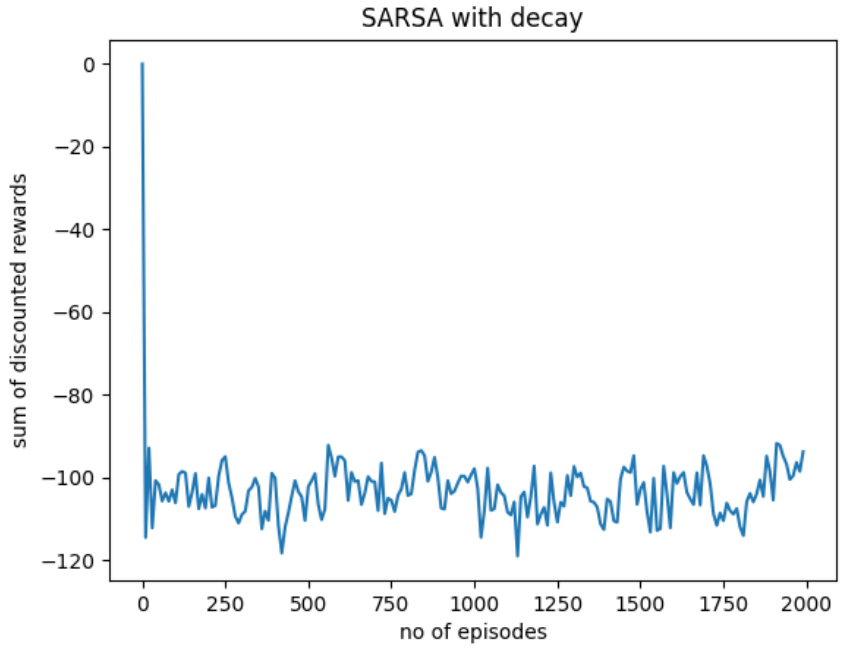
So I feel SARSA with decay is better than other learning techniques mentioned above. We can see that SARSA with decay converges to higher discounted rewards than others in the later episodes.

Part 3

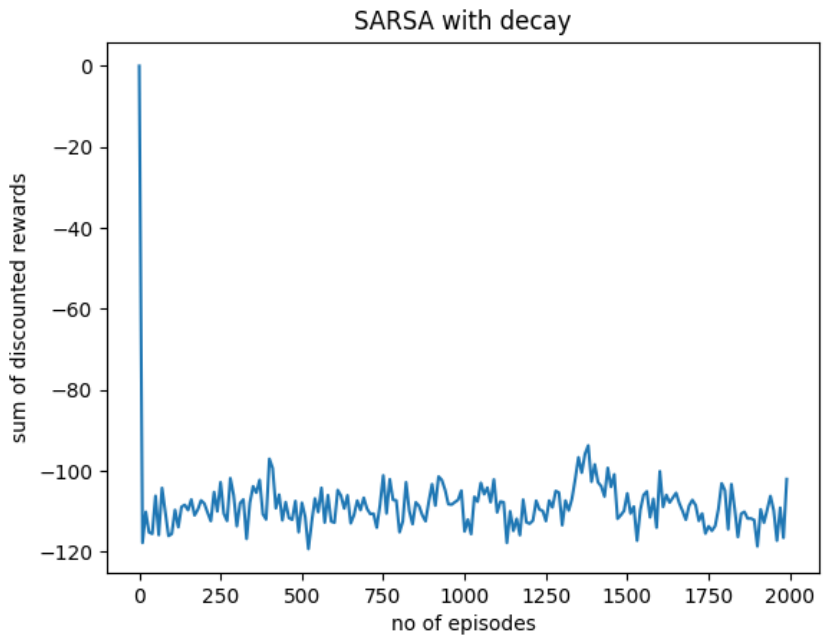
First instance with passenger position = (0,0)



Second instance with passenger position= (0,4)



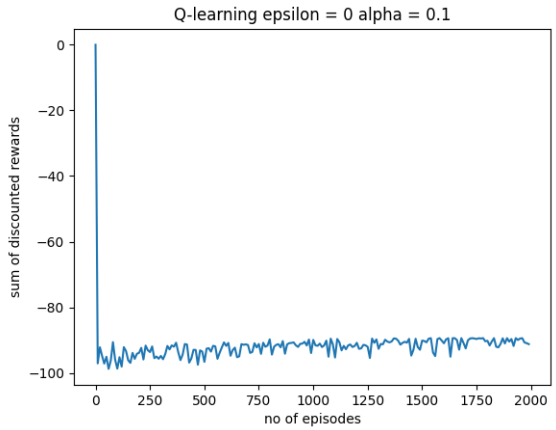
Third instance with passenger position = (4,0)

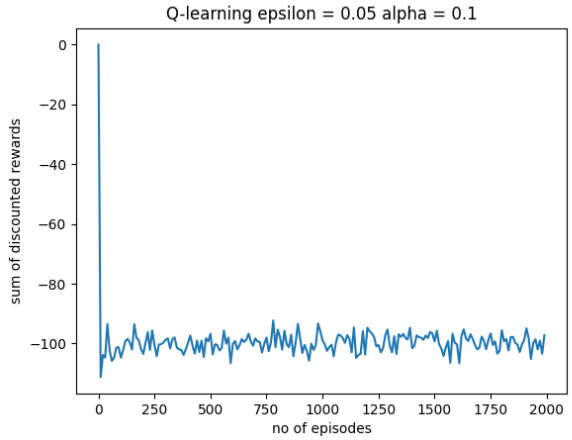


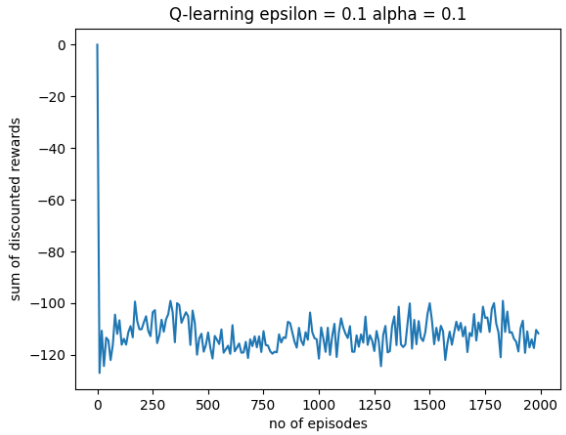
We can see that from the above 3 graphs the discounted rewards are almost similar because the passenger position and taxi position doesn't matter much if the destination position is fixed as these are already covered in the states but there is slight advantage to those who reach the destination state early because the positive reward makes the other episodes converge more to the destination so it helps reaching destination state more compared to others.

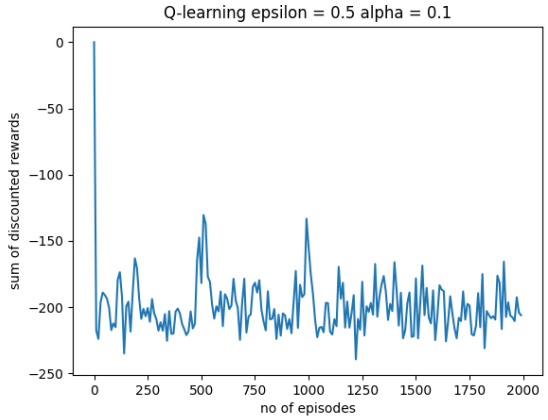
Part4

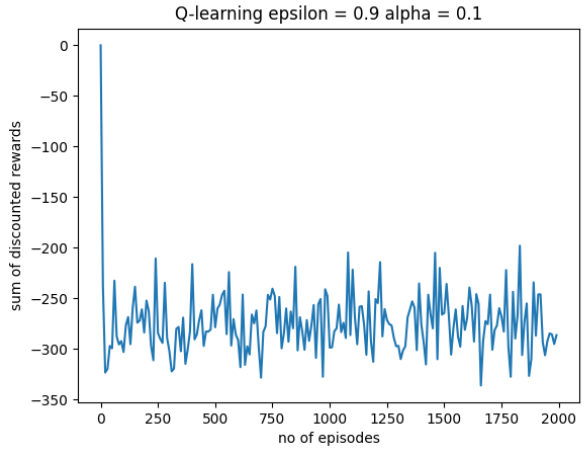
Varying epsilon with alpha = 0.1







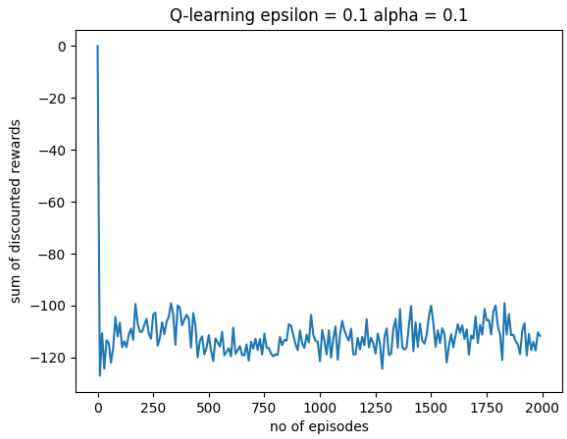


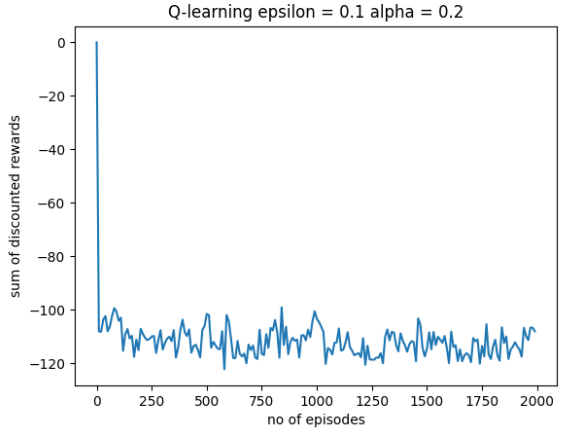


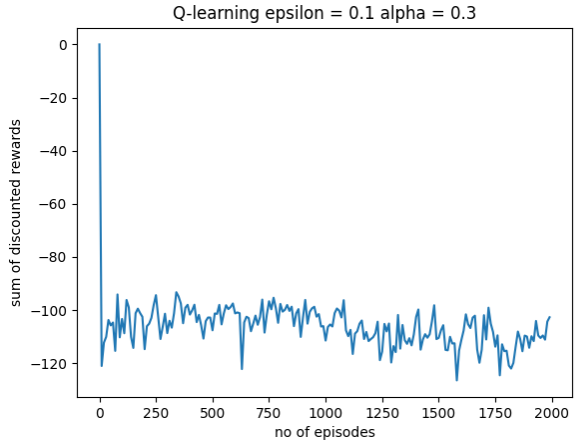
We can see that as the epsilon in increasing with alpha fixed we observe more negative discounted reward i.e due to more exploration compared to exploitation takes actions which are more negative reward again after exploring them once so have less discounted rewards with increase in epsilon.

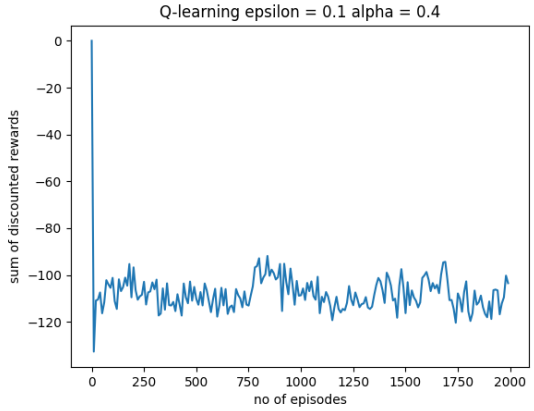
More exploration also causes more variance in the graph as we can see that in the above graphs because without proper balance between exploration and exploitation we can see there is no convergence in values so more variance.

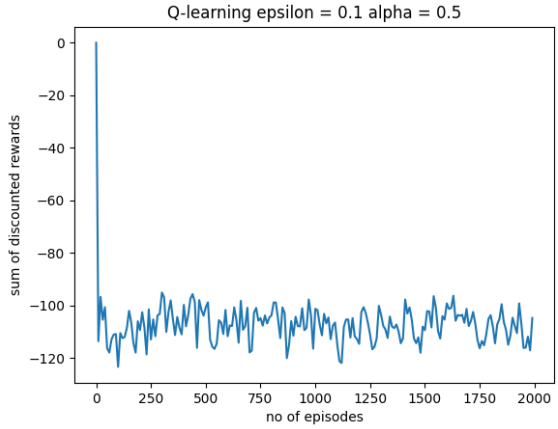
Varying alpha with epsilon = 0.1









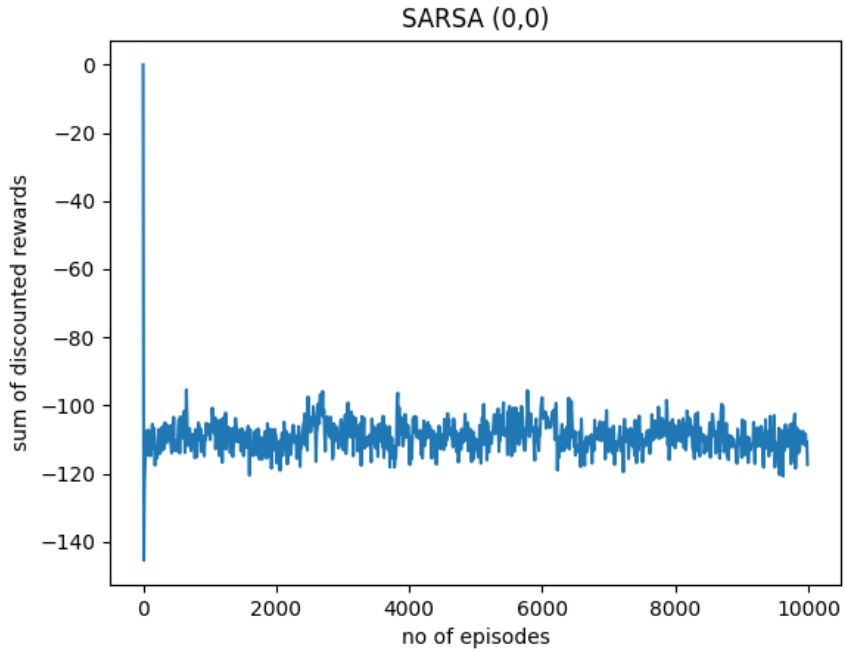


Here we kept the epsilon value fixed but the learning rate “alpha” is varied from 0.1 to 0.5 in the above 5 graphs.

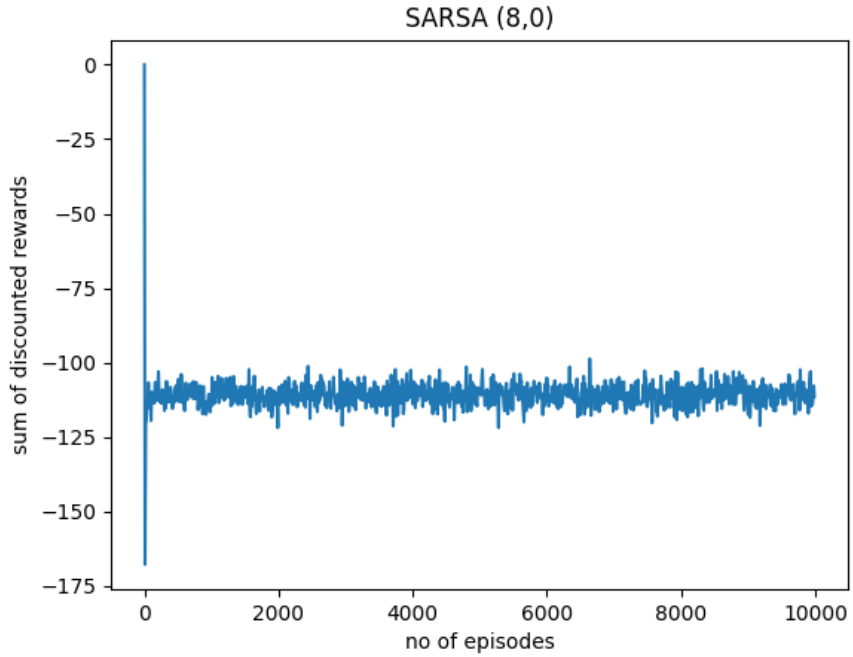
We can observe that alpha =0.3 have the higher discounted rewards than others. This is because it thas the best balance compared to others there is no learning or over learning i.e with lesser alpha we see slower learning but with higher alpha wee see rapid learning and we can say that balance is achieved near 0.3.

Part 5

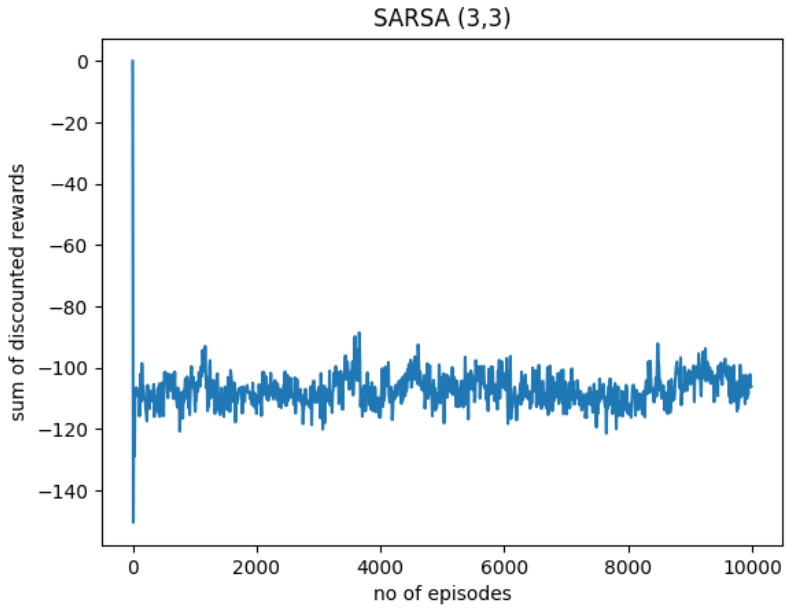
First instance : passenger position = (0,0)



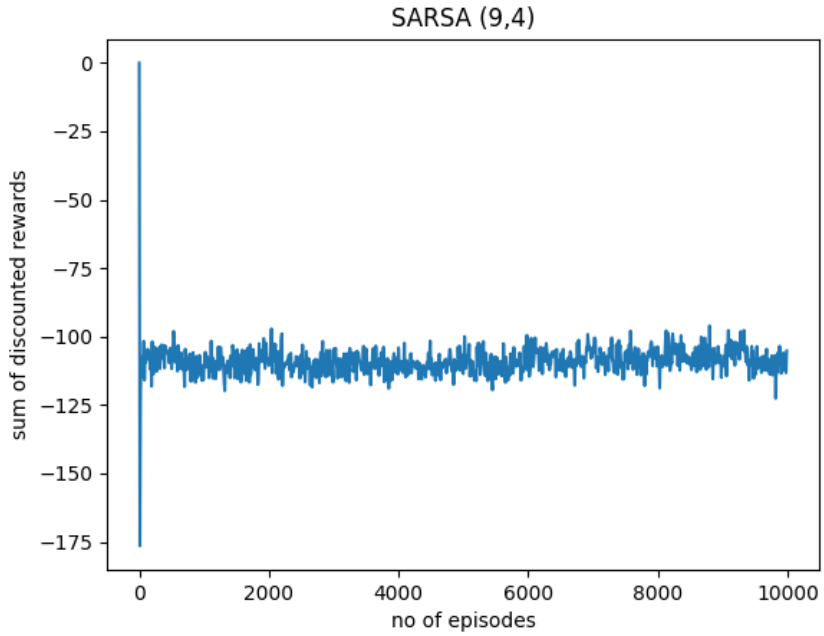
Second instance : passenger position = (8,0)



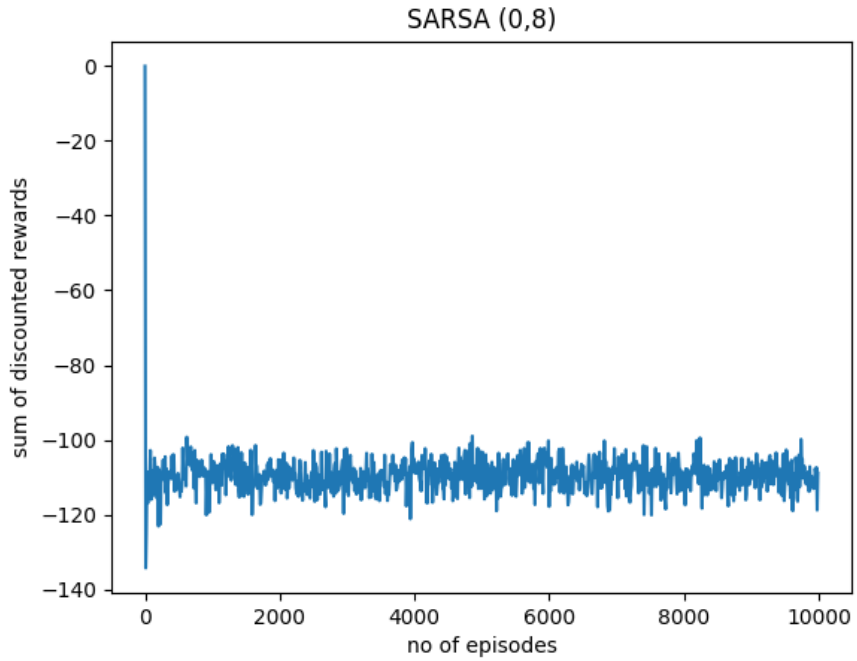
Third instance : passenger position = (3,3)



Fourth instance: passenger position = (9,4)



Fifth instance : passenger position = (0,8)



We can observe that all of them are same because there will be no change because even with change in passenger destination with fixed destination state doesn't have much change and it is approximately (-1)/(1-0.99). The randomness is already explained is due to varying taxi position and the random action selection due to the exploration of the epsilon policy.

We also observed some peaks in (3,3) and (0,0) is because they reached destination state in those and couldn't reach destination state in other instances. As we kept the iteration limit to each episode as 500 and the no of states in the state space is around 10000 so the chance of reaching destination state is low .So we saw converged values in the other instances