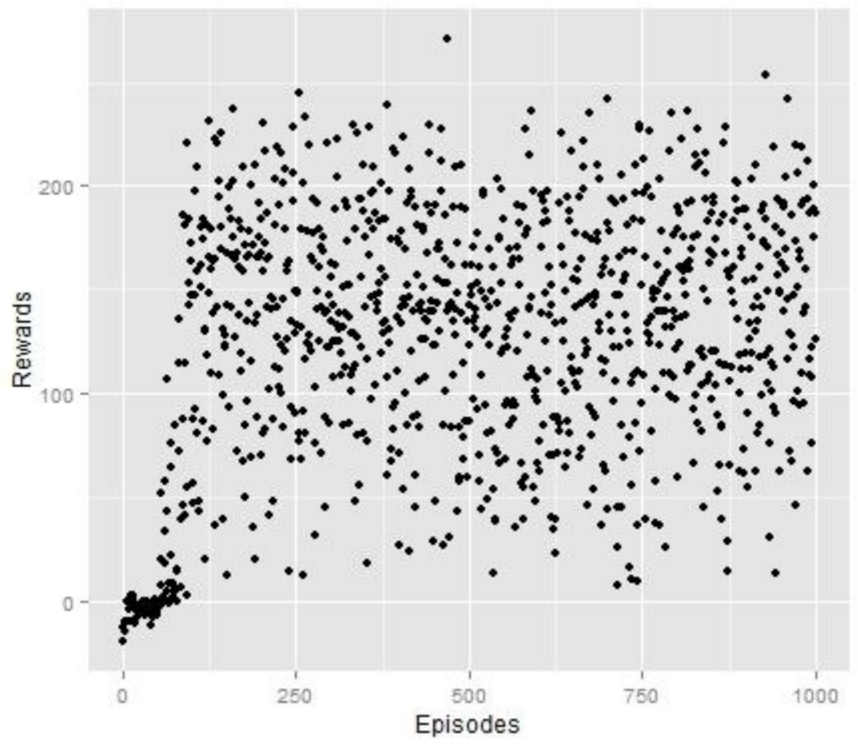
Question 1 (a)

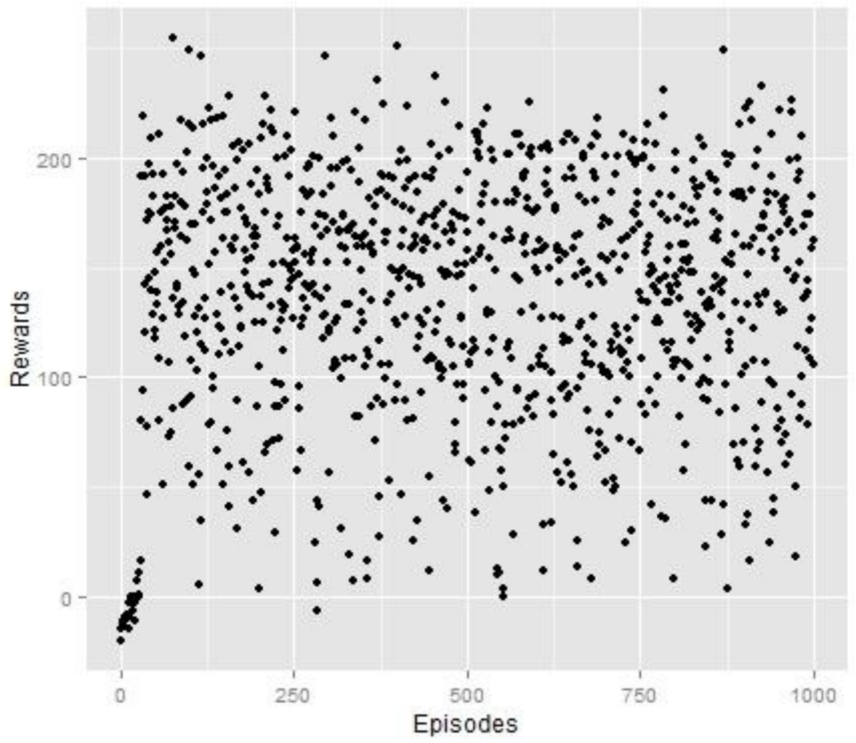
As it can be witnessed **ε=0** causes the agent not to perform any random action at the beginning, the robot will get locked up in the company because it cannot pick any action to continue.



The above figure is plotted diagram of the scenario based on Rewards and Episodes. As we can see at the beginning of the episodes trend, the density of 0 rewards are so high. It will stay like that until agent will learn to follow its way through. And as soon as it learns to open its way, we will notice the random occurrence of the reward values.

Question 1(b)

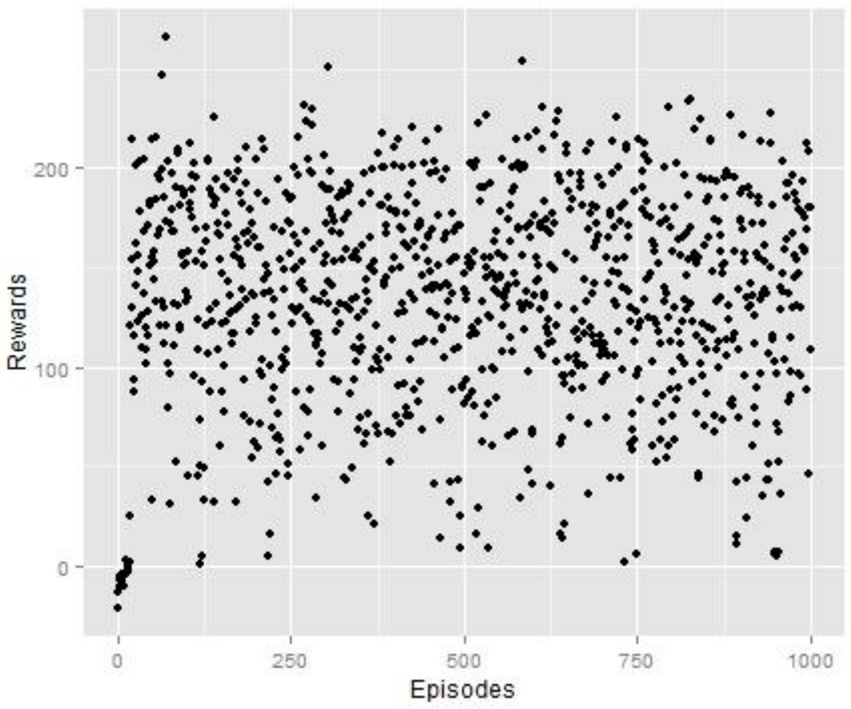
As we minimally increase the randomness, the robot starts moving. However, since the **ε** value is not too high, the randomness occurs scarcely. Therefore the result is successful traversal of the agent to the customers from similar paths and not stocking in loops.



The above figure is plotted diagram of the scenario based on Rewards and Episodes. As we can see at the 0 rewards are notably decreased but the randomness can be witnessed clearly among all the episodes.

Question 1(c)

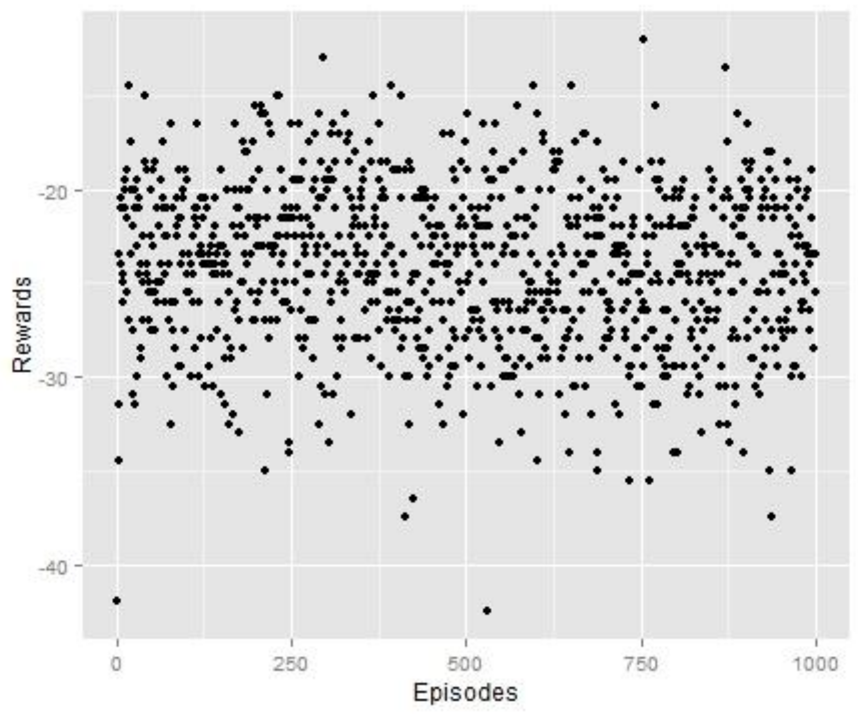
By increasing the epsilon to 0.5 and higher, the randomness increases and relatively the frequency of getting stock in the loops or traversing based on random choices significantly increases.



The above figure is plotted diagram of the scenario based on Rewards and Episodes. As we can see at this scenario also the 0 rewards are notably decreased compared to question (a), but the randomness can be witnessed clearly among all the episodes.

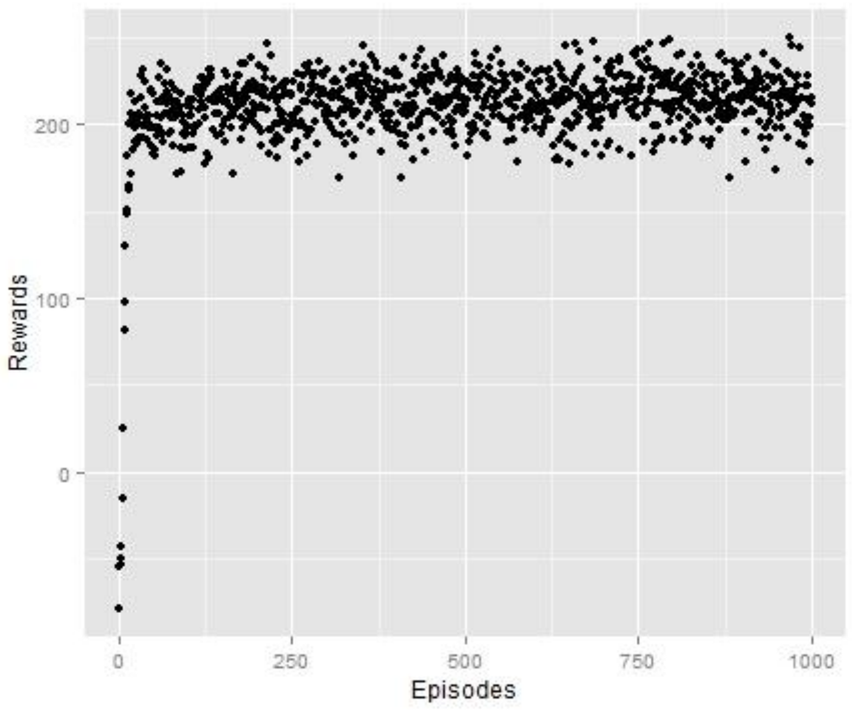
Question 2(a)

Since our world is WorldWithThief, the <knows_thief> argument influences our execution. When we set it to **n**, it means that the robot is not aware of the existence of the thief. That means that the robot will get caught by the thief and no package will get deliver to any of the customers. Therefore the reward value can be nothing but negative. We can also observe this occurrence in the below plotted diagram of Rewards and Episodes. As we can see all of the episodes are ended with negative reward value. This proves should they wish to carry the packages to the customers they will finally get to the thief and receive a penalty.



Question 2 (b)

By setting the <knows_thief> argument to Y, robot will be fully aware of the thief's existence. Therefore it will travel cautiously and successfully delivers the packages to the customers and return to the company and keeps repeating this procedure. The plotted result of the scenario can be seen in the figure below.

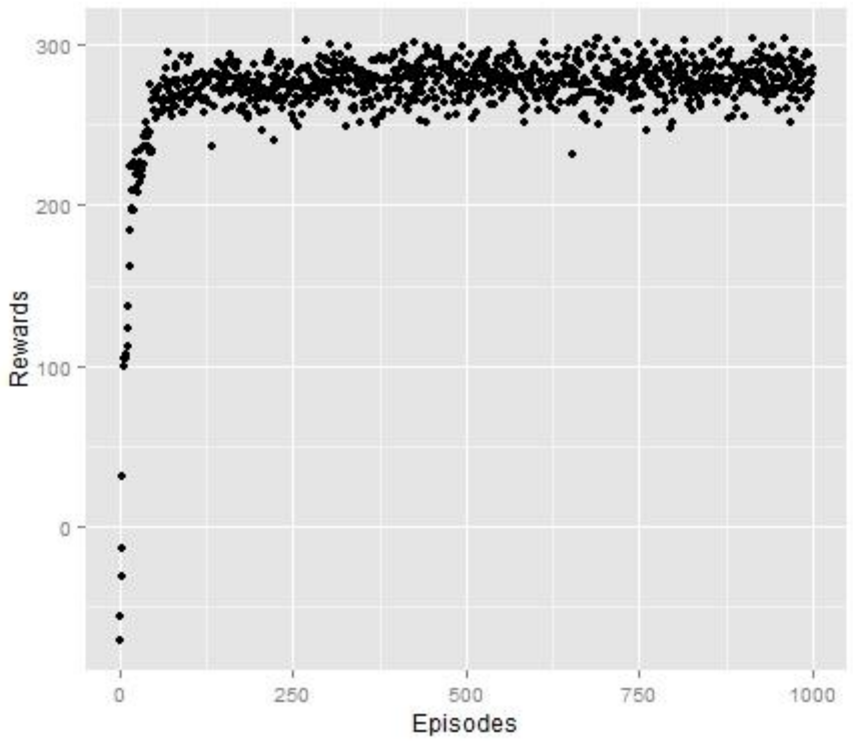


200
100
0
Rewards
0
250
500
750
1000
Episodes

Question 2 (C)

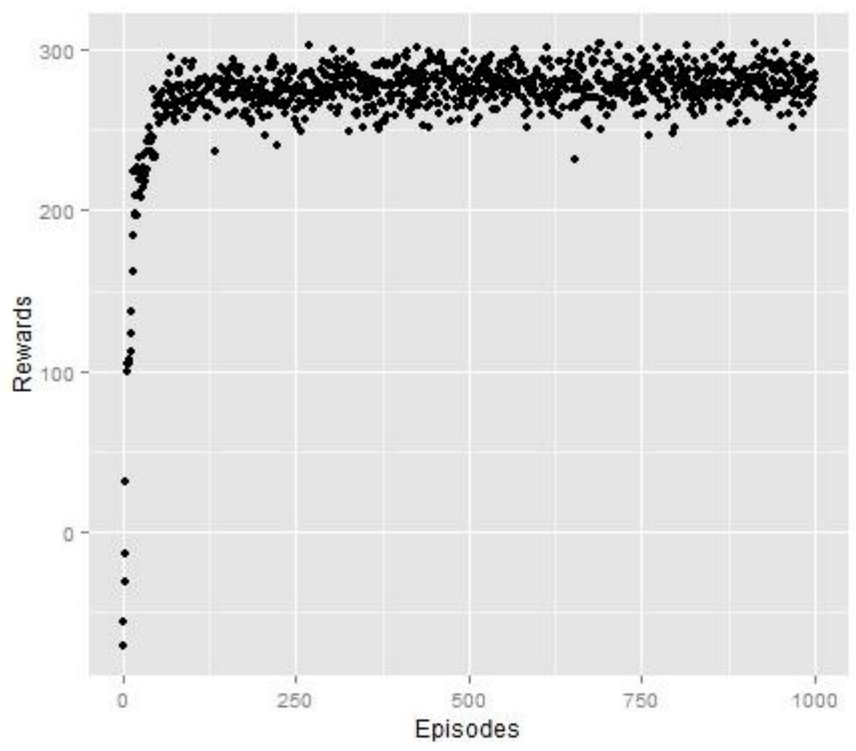
I have tried different values for **learning rate** and **ε**. And the results are plotted as below.

Learning Rate=0.9 , ε=0.9



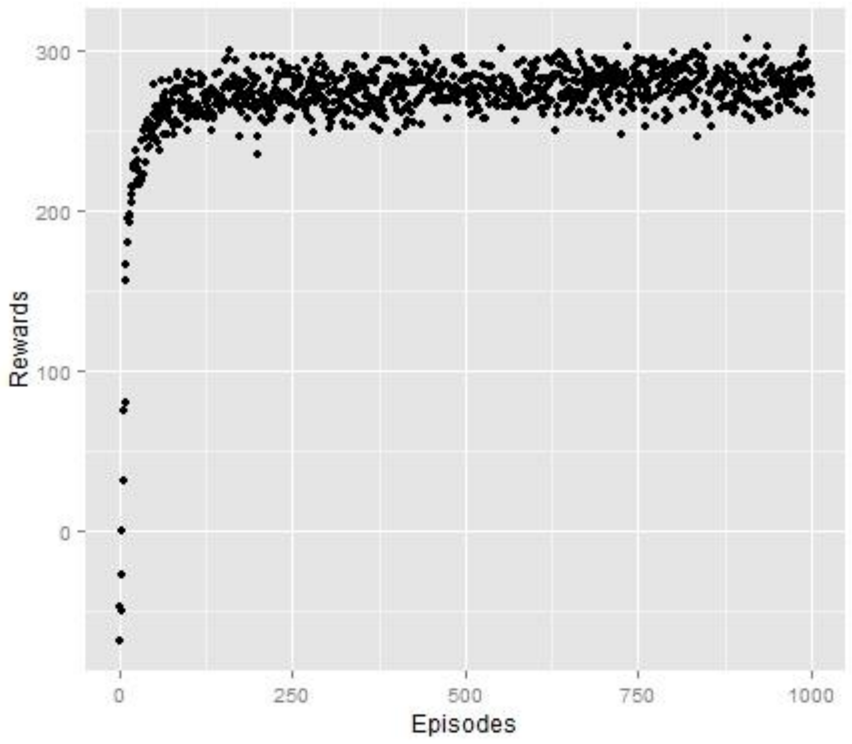
Calculated Sum of Rewards: 272324.5

Learning Rate=1 , ε=1



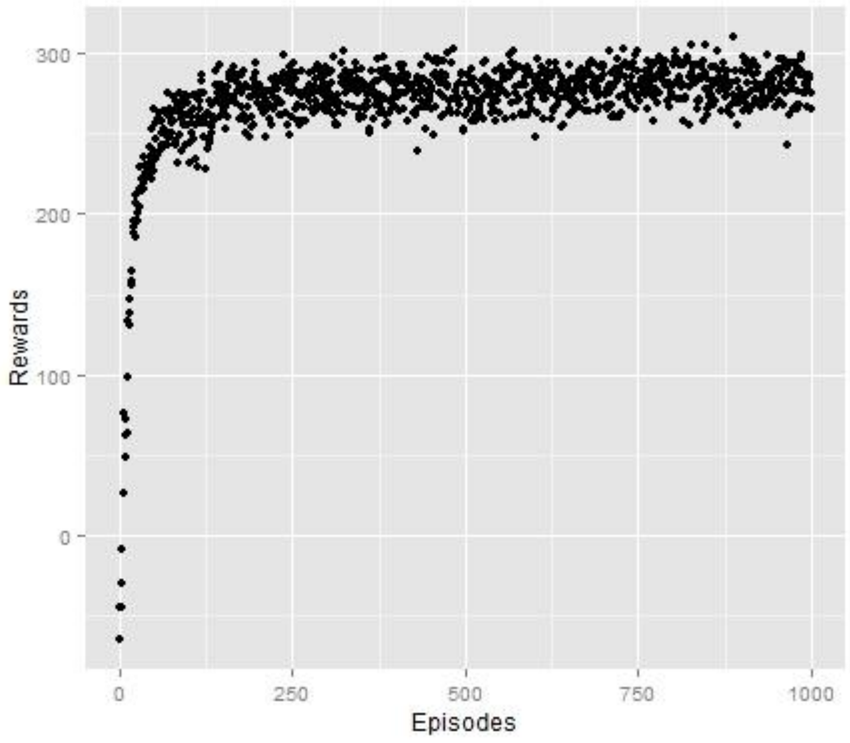
Calculated Sum of Rewards: 272542

Learning Rate=0 , ε=0



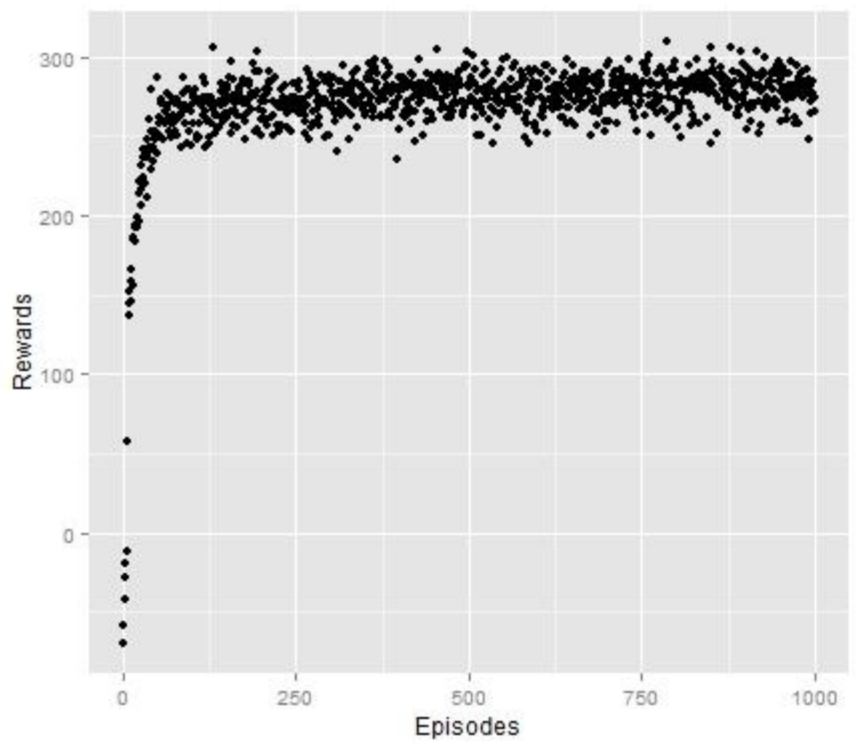
Calculated Sum of Rewards: 271360

Learning Rate=1 , ε=0



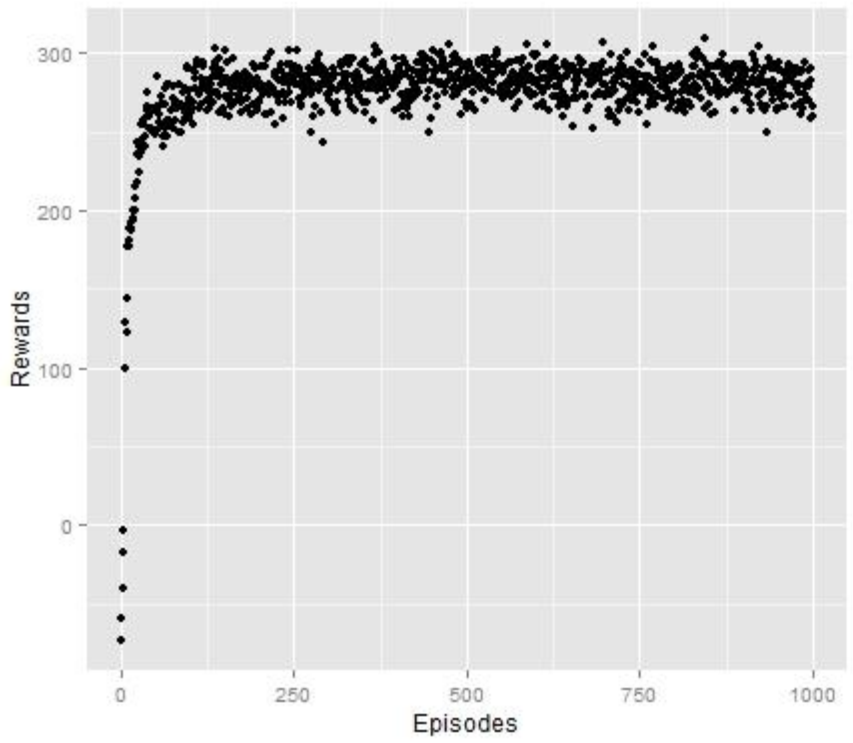
Calculated Sum of Rewards: 269680

Learning Rate=0 , ε=1



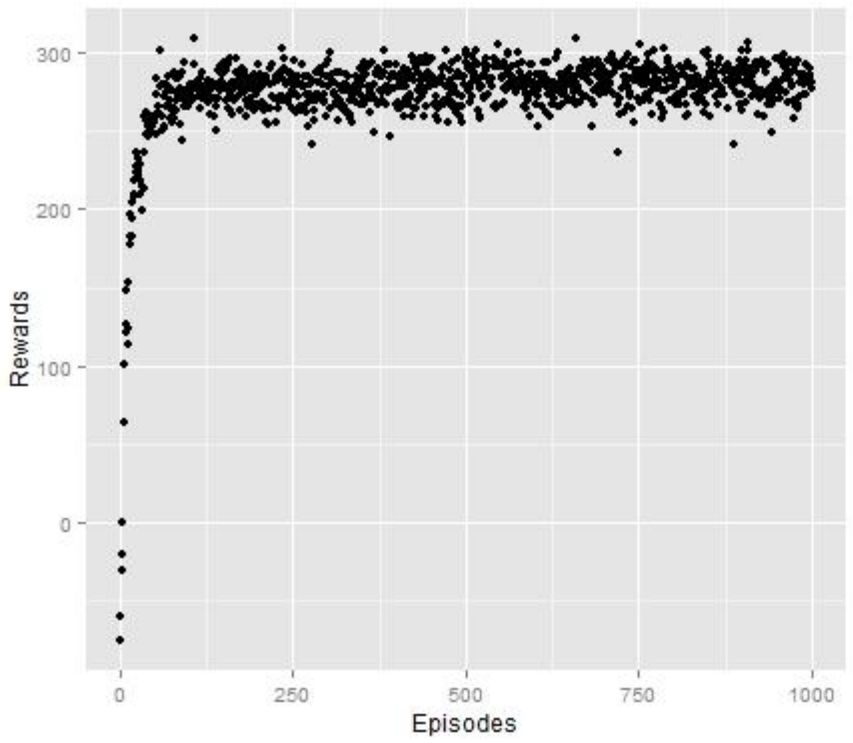
Calculated Sum of Rewards: 270409.5

Learning Rate=0.5 , ε=0.5



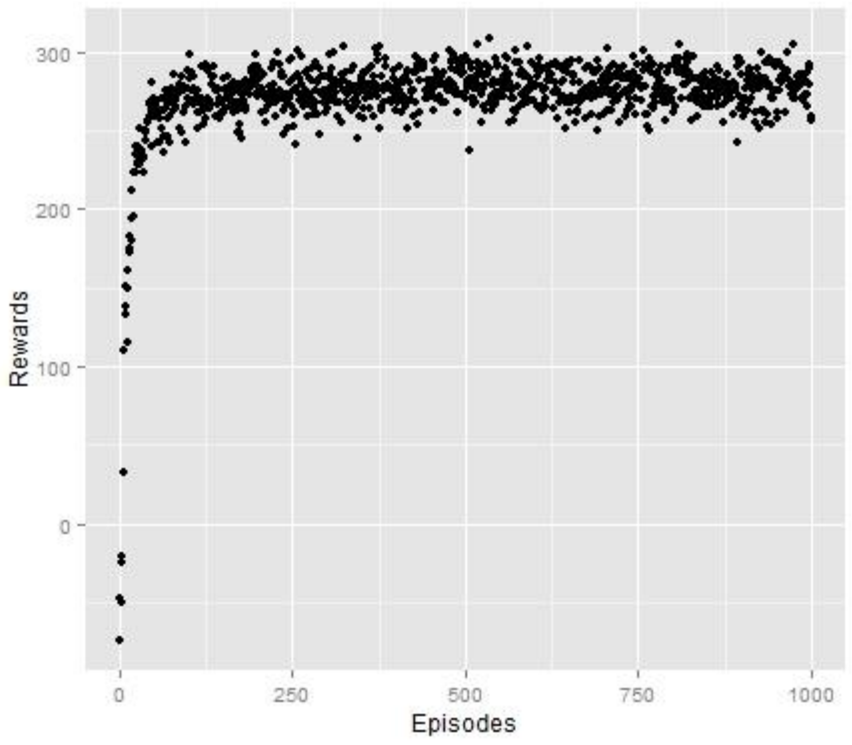
Calculated Sum of Rewards: 276009

Learning Rate=0.5 , ε=0



Calculated Sum of Rewards: 274352

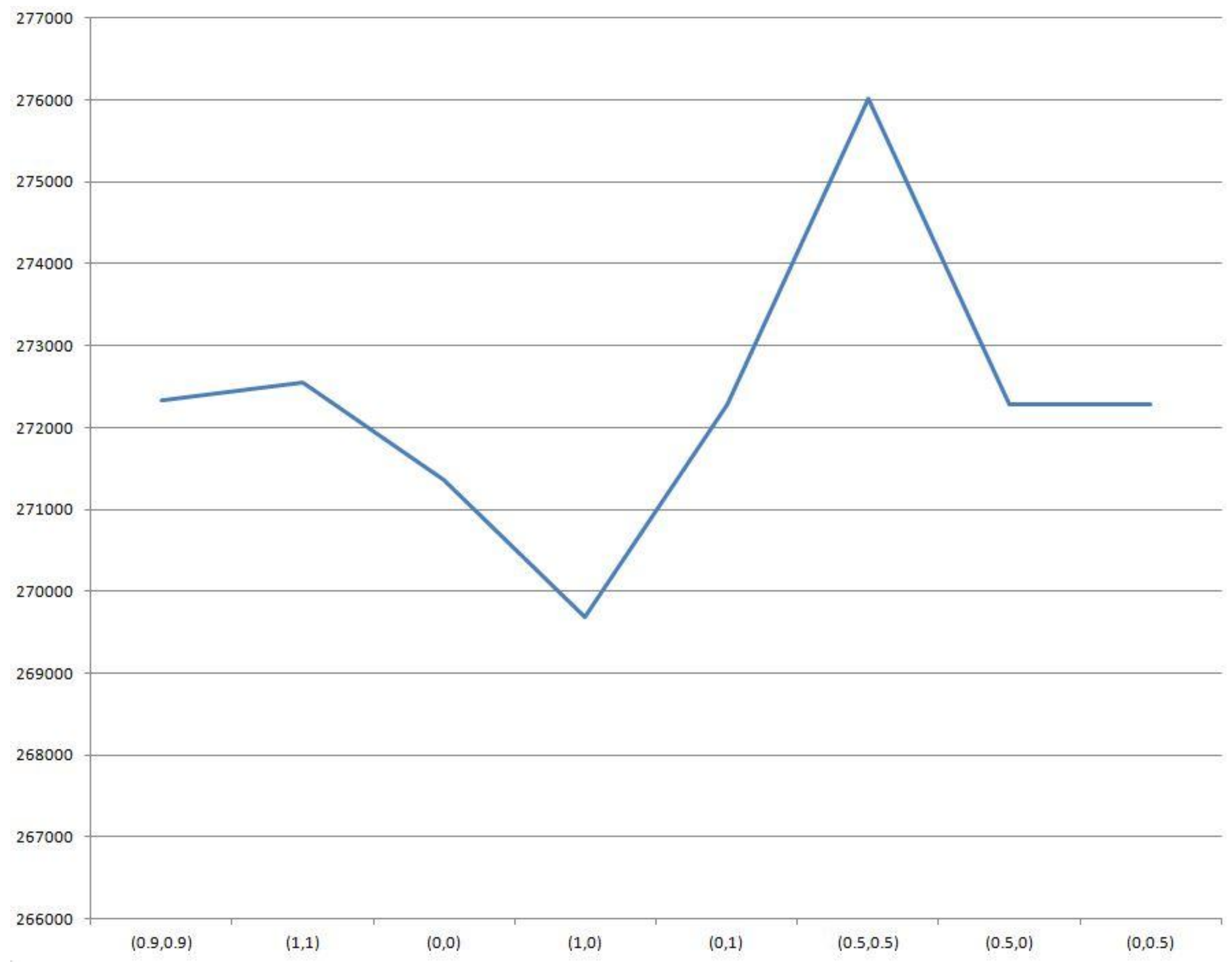
Learning Rate=0 , ε=0.5



Calculated Sum of Rewards: 272288

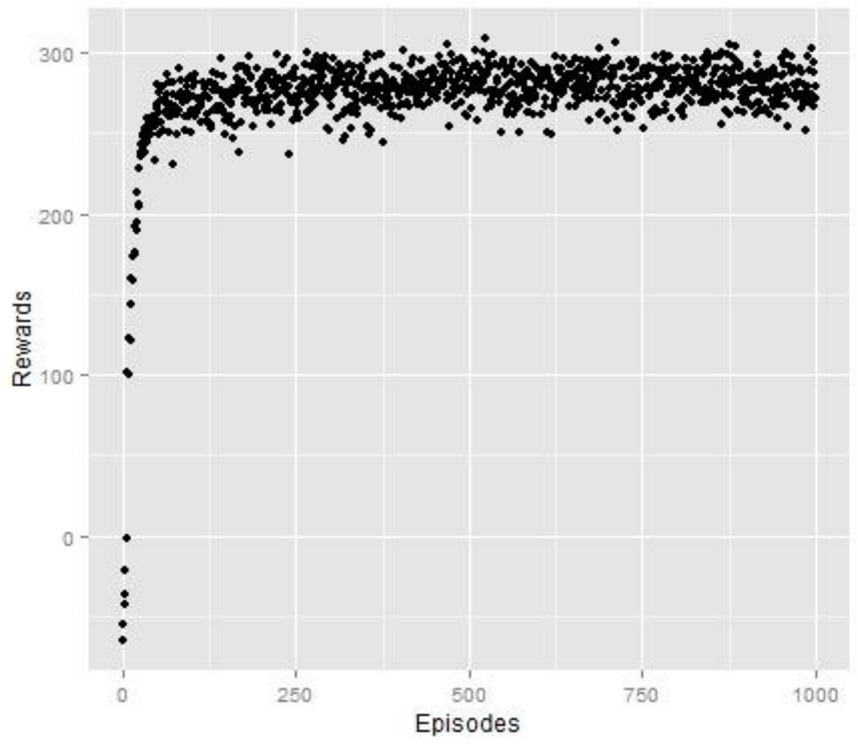
As we can clearly see, the best value for the requested variables is:

Learning Rate=0.5 , ε=0.5



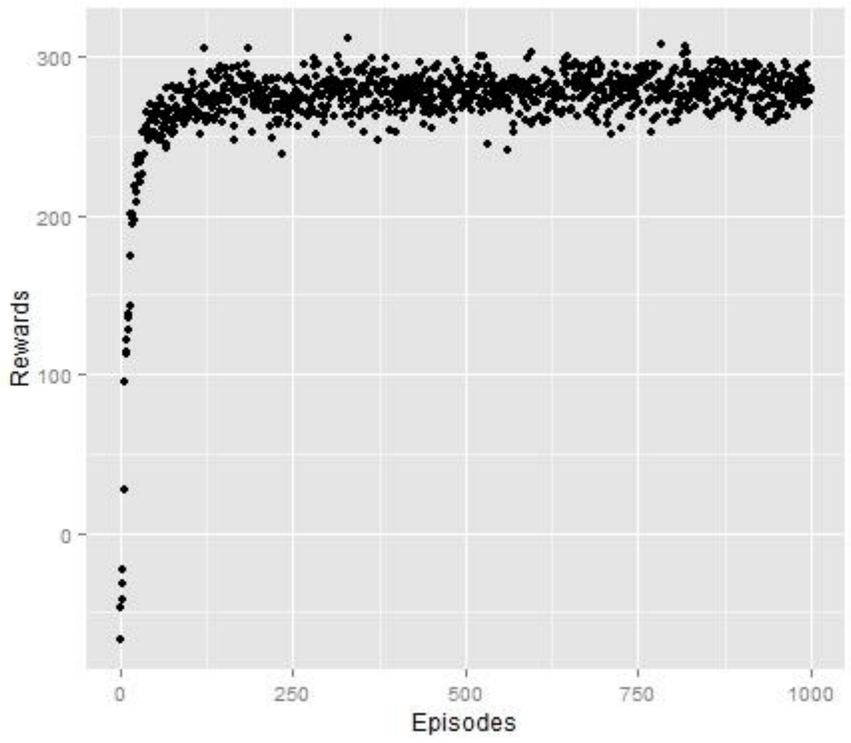
Question 3

First Time

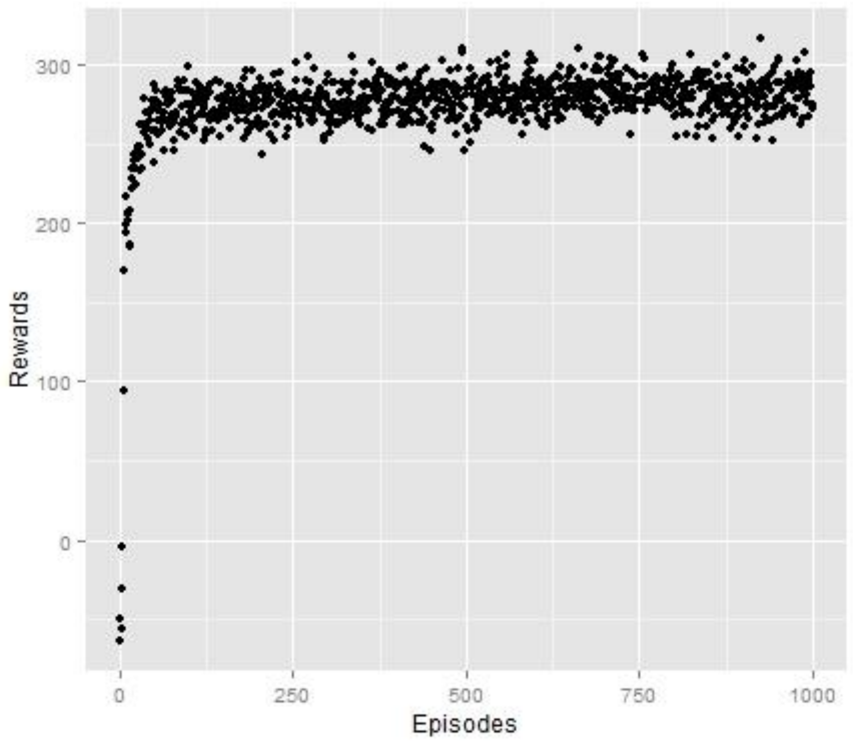


One of the noticeable things upon execution of this scenario is that the agent learning ability helps him to correct himself and get close to the optimal traversal which cost him the minimum possible penalty. As we get to the end of the episodes we see that the mistakes are noticeably decreased and the rewards are more concentrated.

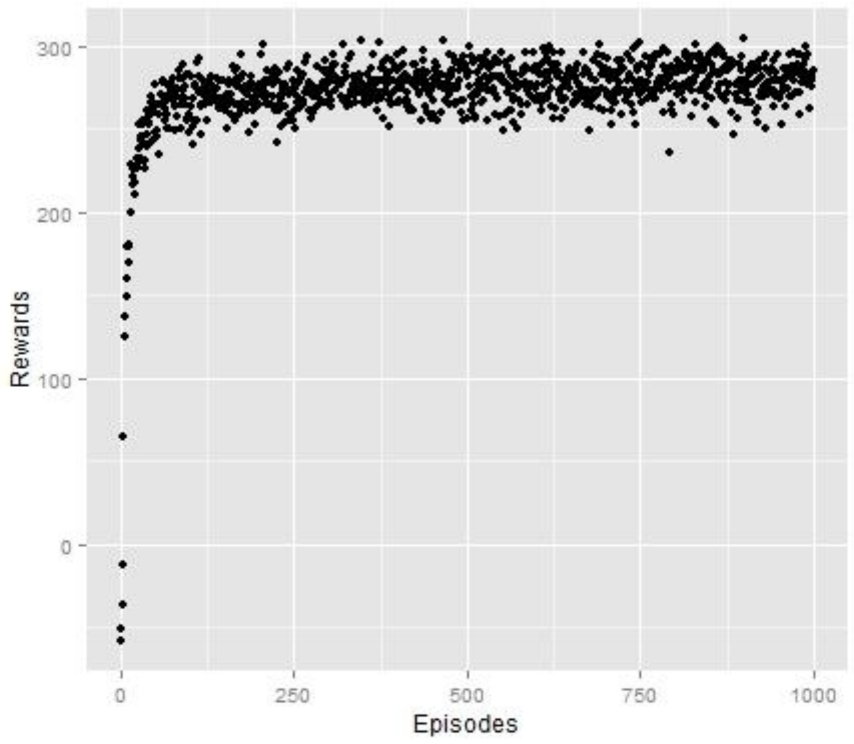
Second Time



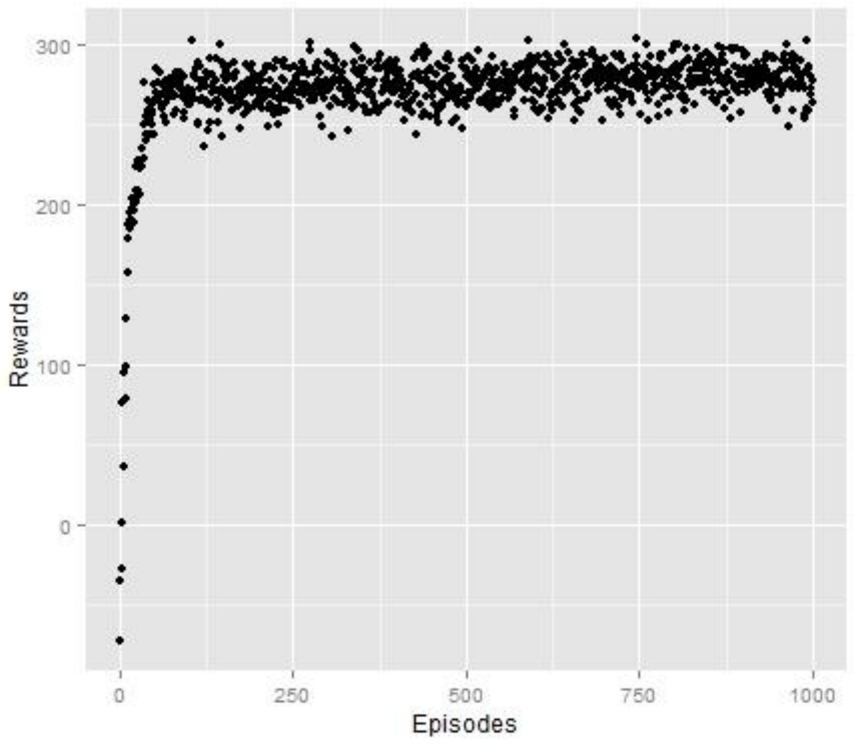
Third Time



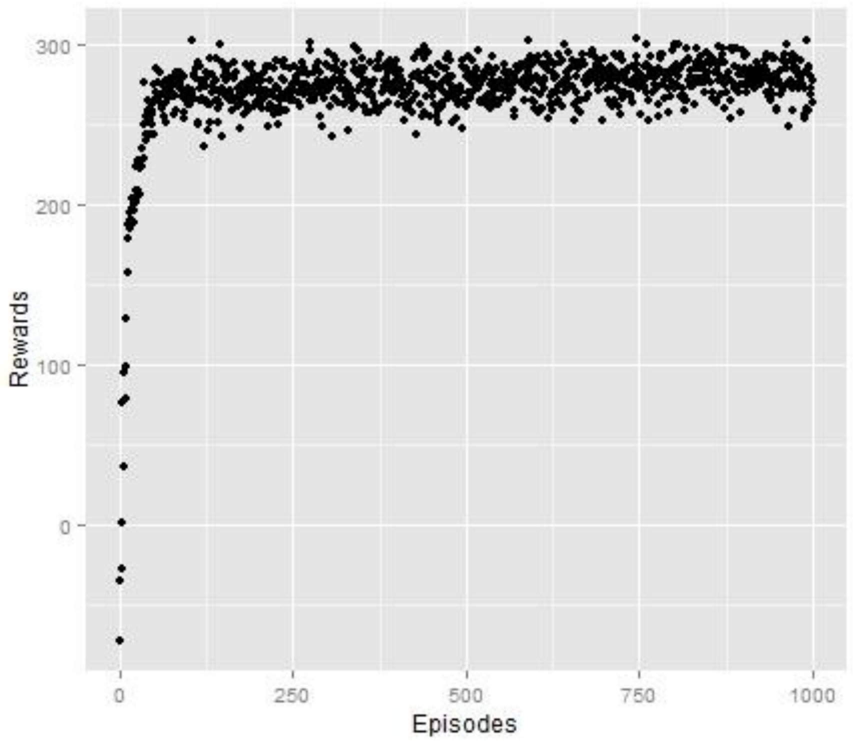
Fourth Time



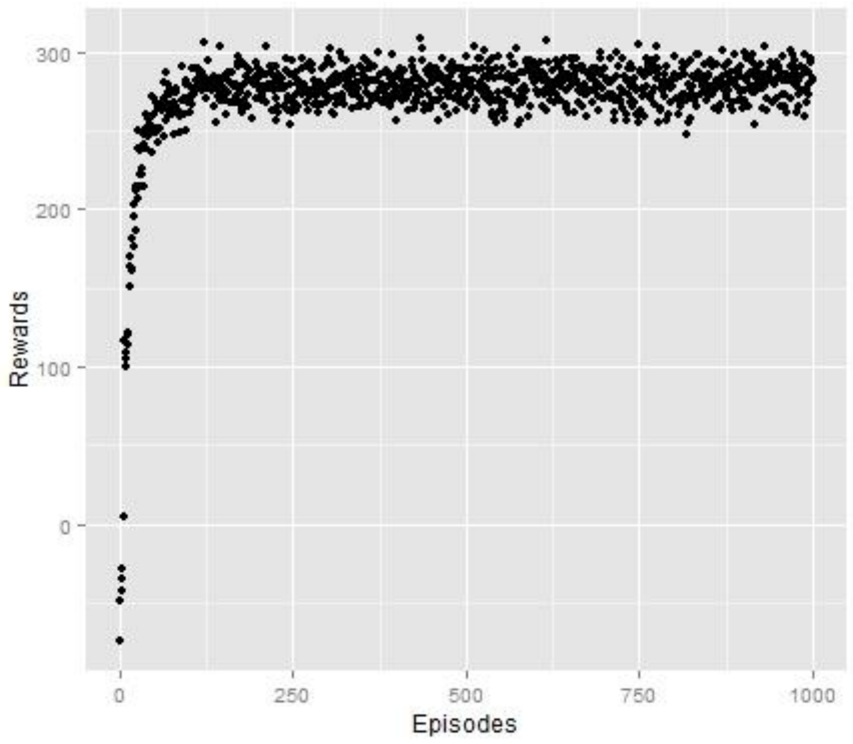
Fifth Time



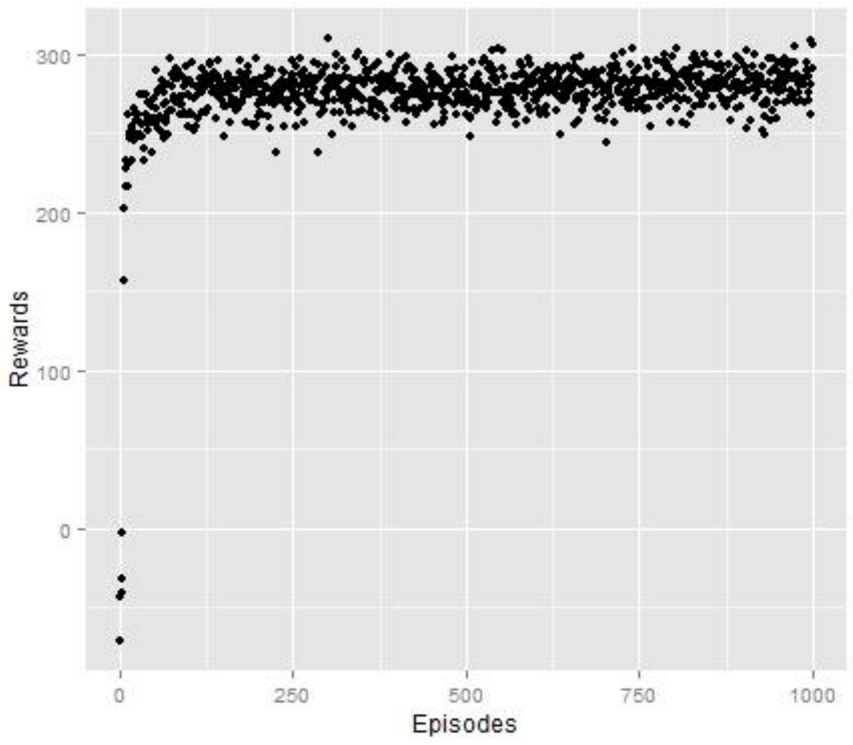
Sixth Time



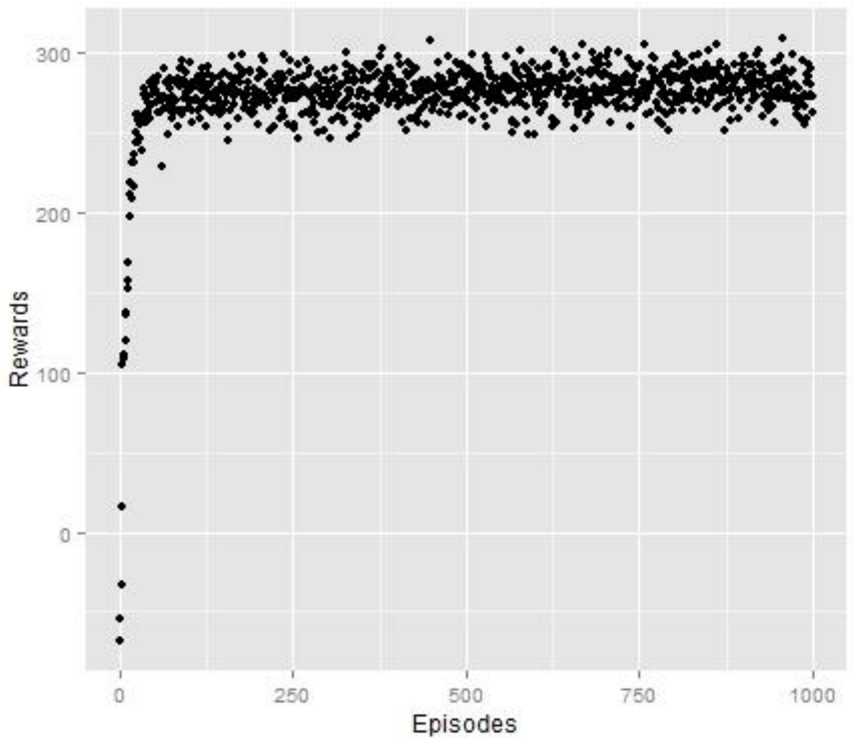
Seventh Time



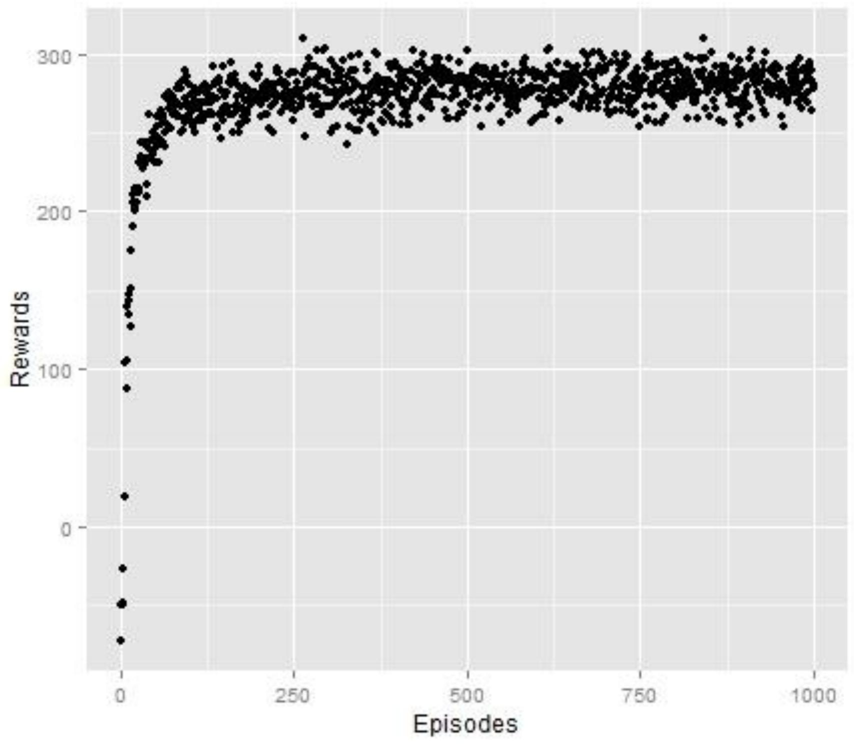
Eighth Time



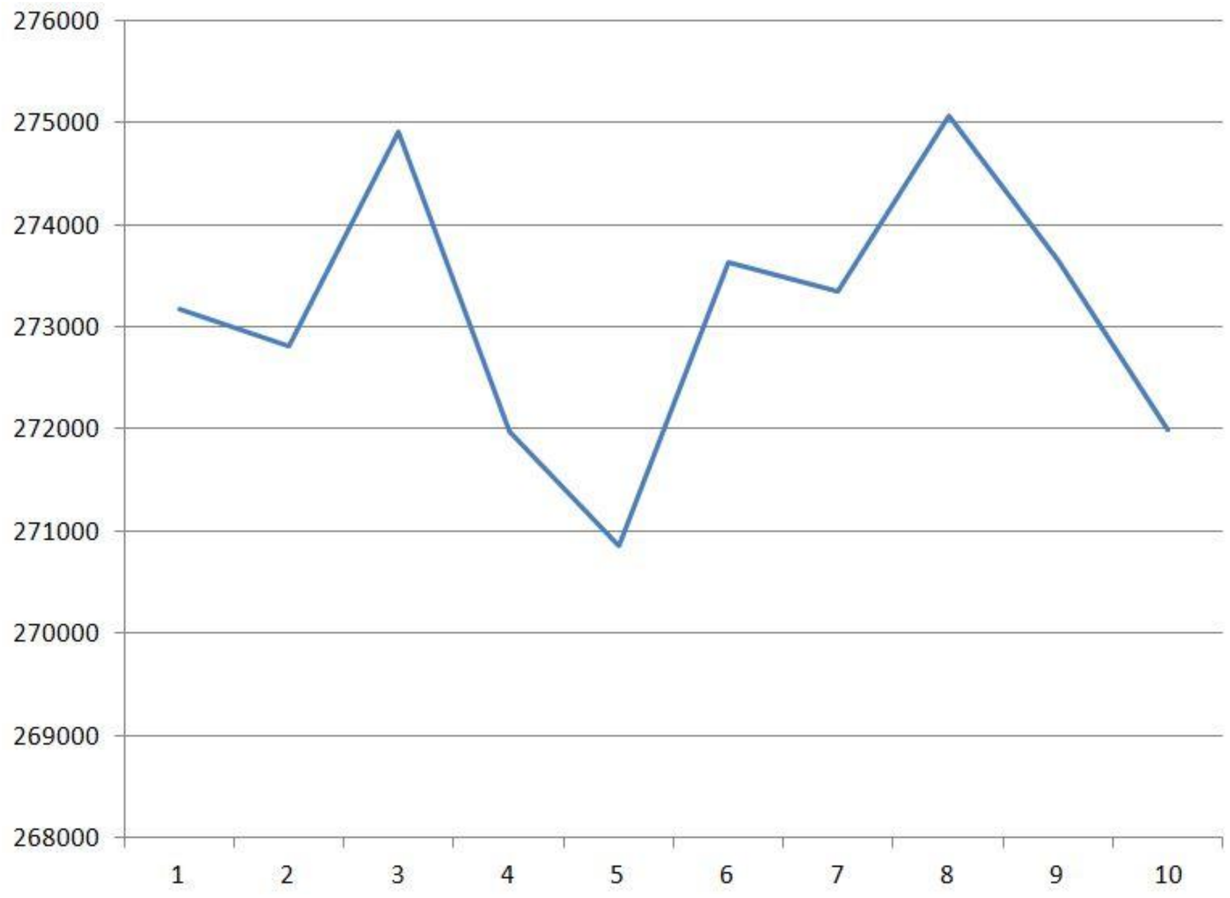
Ninth Time



Tenth Time



The sum of the rewards for each execution is on the below chart.



As we can see, although in each scenario the agent gradually learn how the optimize itself, but due to the existence of epsilon in our function and the randomness it creates, we get different results in different reward range each time we execute or function.