2.3 Questions

2.3.1

The optimal policies are following: (State -> Action)

S -> Right

F (1,2) -> Right

F (1,3) -> Right

F (1,4) -> Right

Now calculating the values to obtain the V(S):

v(F at (1,4)) = 1 * (10 + 0.99*0) = 10

v(F at (1,3)) = 1 * (0 + 0.99*10) = 9.9

v(F at (1,2)) = 1 * (0 + 0.99*9.99) = 9.80

v(S) = 1 * (0 + 0.99*9.80) = 9.70

Therefore, the value at the starting location is 9.70

2.3.2

1) If I am at S(1,1) I would choose terminal state G1 to move because there is 0.2 noise which means the probability of not slipping is 0.8. If we attempt to reach G2, there is a good chance we slip because the route is four locations and the chance of slipping is 0.2. If I am at F(1,3), I would choose to move to terminal state G2, because it is closer than G1. Therefore, the chances of making it to G2 is larger without slipping and also the reward of G2 is greater than G1.

2)

q(S(1,1), LEFT) = 0.8(1 + 1*(0)) + 0.1*(-100 + 1*(0)) + 0.1*(-100 + 1*(0)) = -19.2

q(F(1,4), Right) = 0.8(10 + 1*(0)) + 0.1*(-100 + 1*(0)) + 0.1*(-100 + 1*(0)) = -12

3)

v(F(1,4)) = -12 (going right)

v(F(1,3)) = 0.8(0 + 1*(-12)) + 0.1*(-100 + 1*(0)) + 0.1*(-100 + 1*(0)) = -29.6 (going right)

v(F(1,2)) = 0.8(0 + 1*(-19.2)) + 0.1*(-100 + 1*(0)) + 0.1*(-100 + 1*(0)) = -35.36 (going left)

v(S(1,1)) = -19.2 (which is going left)

2.3.3

1) The optimal policy when gamma is equal to 1 is going to the reward location R. The reason is because when gamma is high, the agent will value the immediate rewards as much as the future rewards. Therefore, the agent will try to explore. If we lower gamma to something close to zero like 0.2, the agent will value immediate rewards more than future rewards. In this case, the agent will go to G without exploring R.

2) The gamma number affects how an agent values the immediate rewards. If gamma is low the agent will try to gain as much rewards immediately than the future. If gamma is high like 1, the agent will value future rewards as much as immediate rewards and therefore try to find more rewards and take more steps than a low gamma agent.

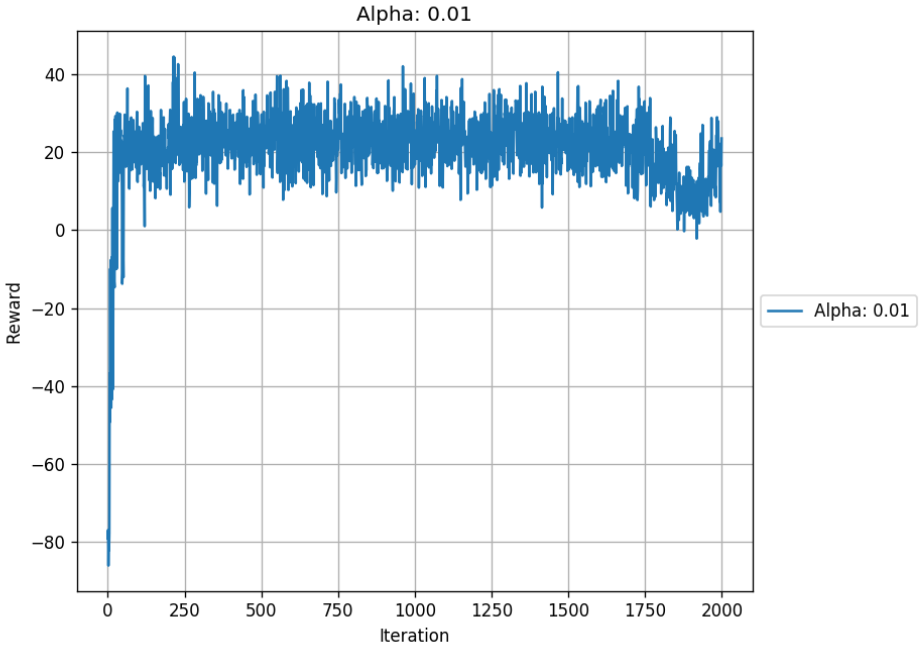
2.3.4

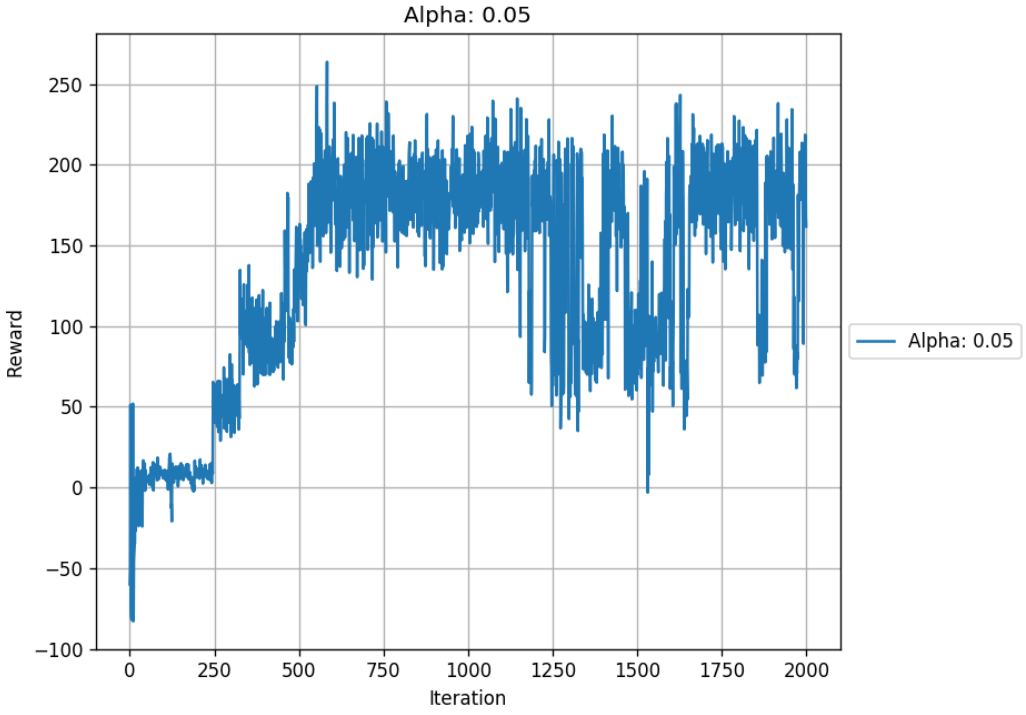
My exploring strategy was first exploring the edges of the map to know a little more about the board. I started exploring until I found useful locations such as R1 which gave me rewards. Then

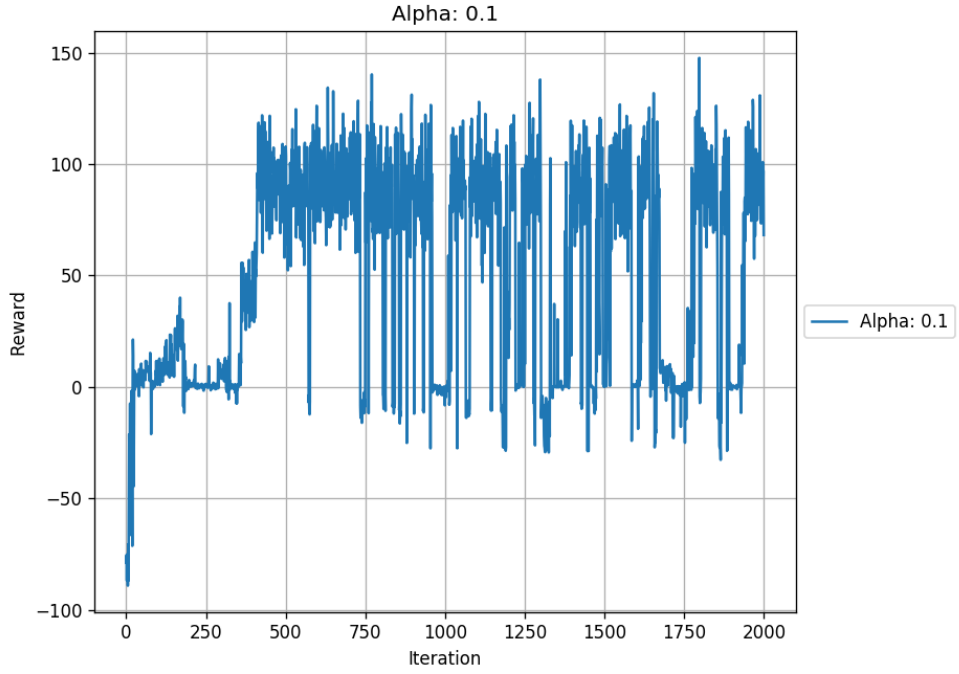
I started to find paths that involved R1 and reaches G. After few attempts, I was able to find a route that involved R2 that reaches the goal terminal G. To sum it up, my exploration strategy was exploring as much of the maps to find useful locations like R1 and then finding continue exploring with exploring that location until the terminal location is found.

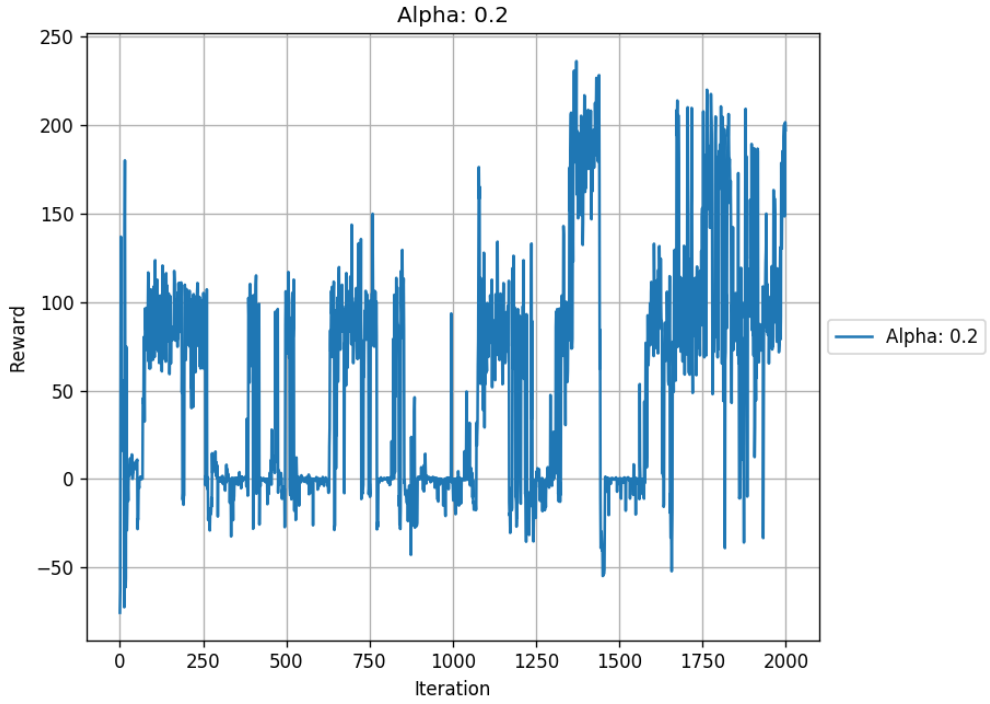
Q Learning Task

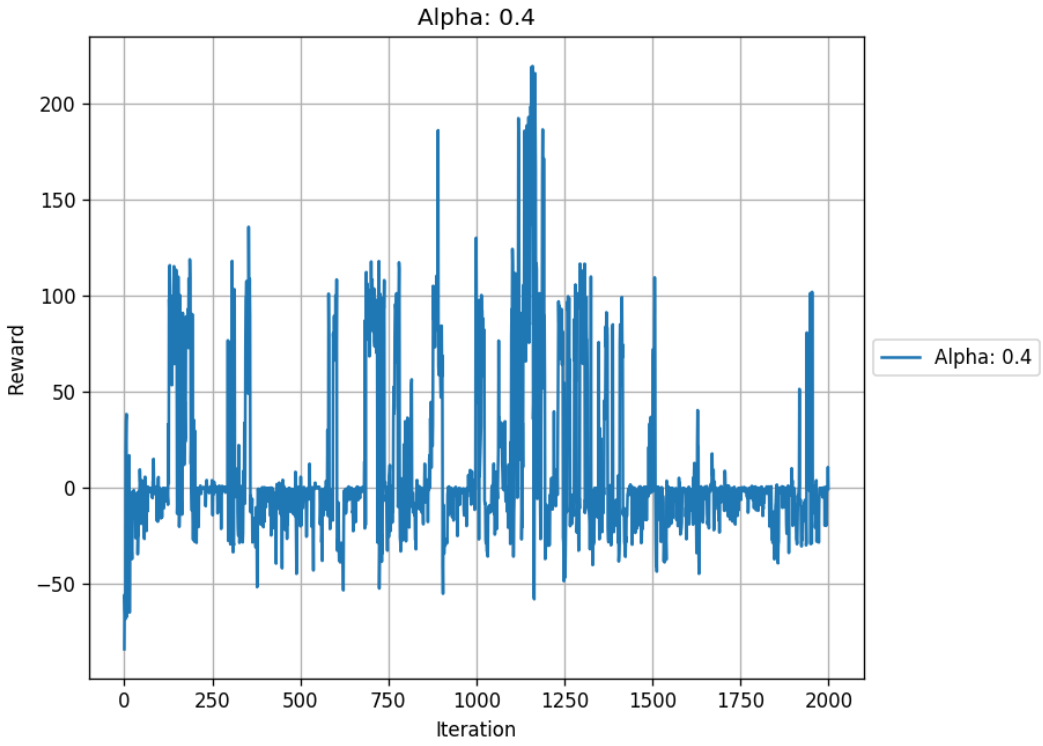
Subtask 1: Experiment with α






Alpha: 0.1
Reward
Iteration
150
100
50
0
−50
−100
0
250
500
750
1000
1250
1500
1750
2000
Alpha: 0.1


Alpha: 0.2
Reward
Iteration
250
200
150
100
50
0
−50
0
250
500
750
1000
1250
1500
1750
2000
Alpha: 0.2



Now running Value Iteration with the same parameters:

α = 0.01 Result: Mean Reward 276.409, Standard Deviation 440.529
α = 0.05 Result: Mean Reward 279.326, Standard Deviation 442.099
α = 0.1 Result: Mean Reward 274.562, Standard Deviation 438.897
α = 0.2 Result: Mean Reward 271.918, Standard Deviation 438.803
α = 0.4 Result: Mean Reward 278.744, Standard Deviation 439.963

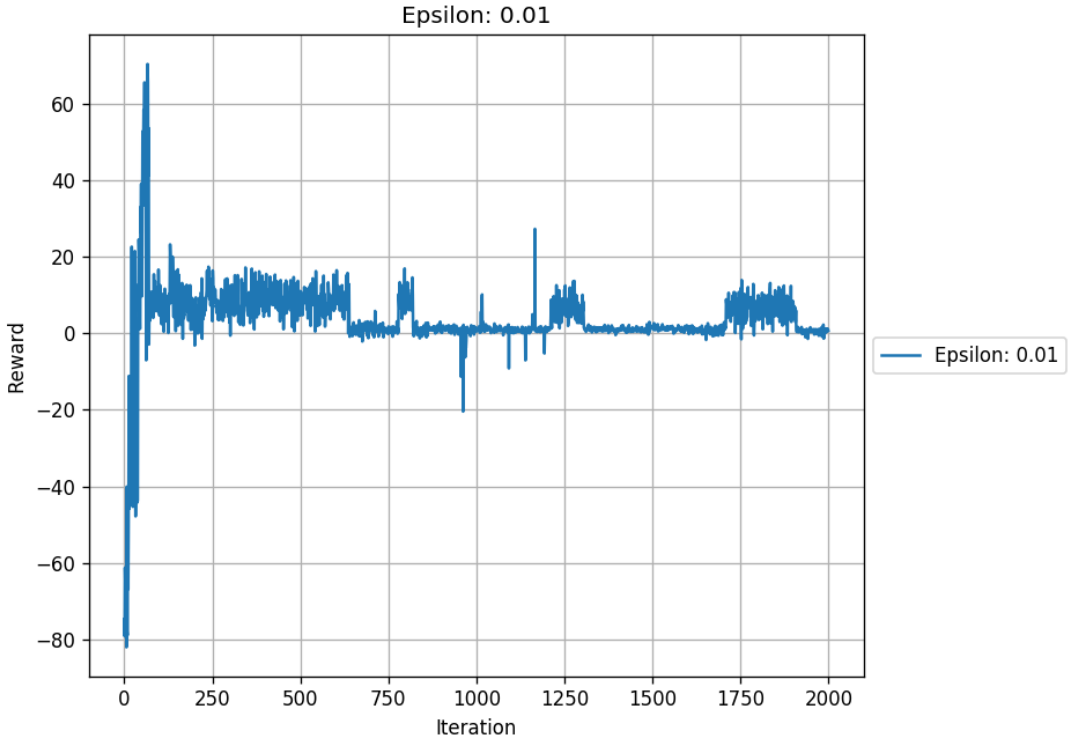
When α = 0.01, the q learning only reached discounted reward between 20 and 40 on average while value learning reached a mean of 276. When α = 0.05, q learning reached discounted reward between 150 to 250 while value learning reached mean reward of 279. In this alpha, the q learning outputted very close results as the value learning. When α = 0.1, q learning reached reward between 100 and 150 while value learning achieved a mean of 274. When α = 0.2, q learning achieved an average of total reward between 100 and 200 while value learned achieved

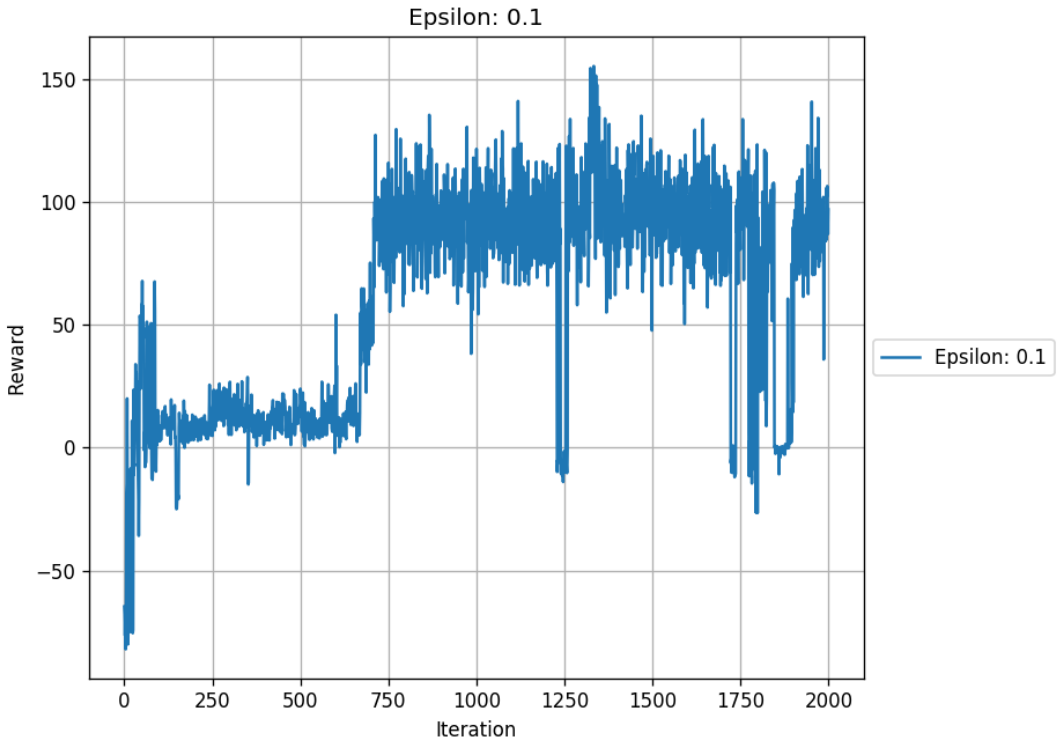
271. Lastly, when $\alpha = 0.4$ q learning achieved reward between 50 and 200 on average and value learning achieved an average of 278 reward. The results are close.

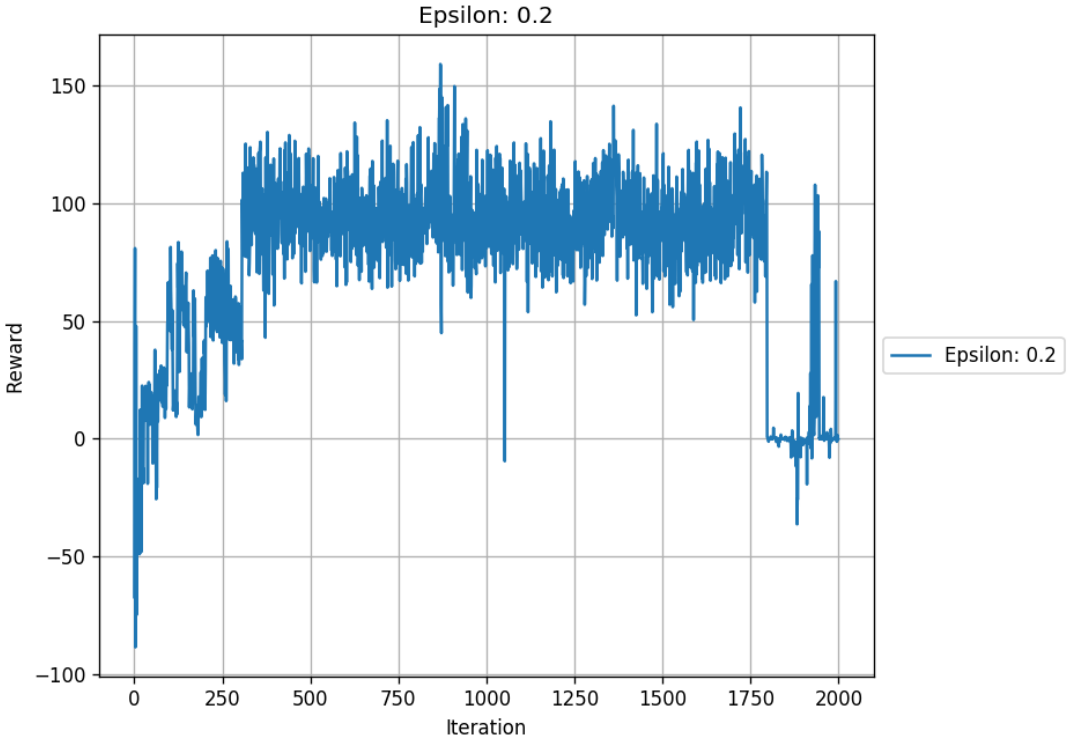
In summary, alpha is the learning rate of the agent. The learning rate controls the speed of the agent learning. After running these experiments, low alpha seems to make the agent converge or oscillate quicker than larger alphas. For example, when alpha is equal to 0.01 the agent oscillated between 20 and 40 quickly. As we increase, the agent seems to taking more risks and exploring more because it reaches more rewards. Lastly, when alpha is the largest in this experiment (0.4), the agent is oscillating around zero and then achieving the optimal rewards.

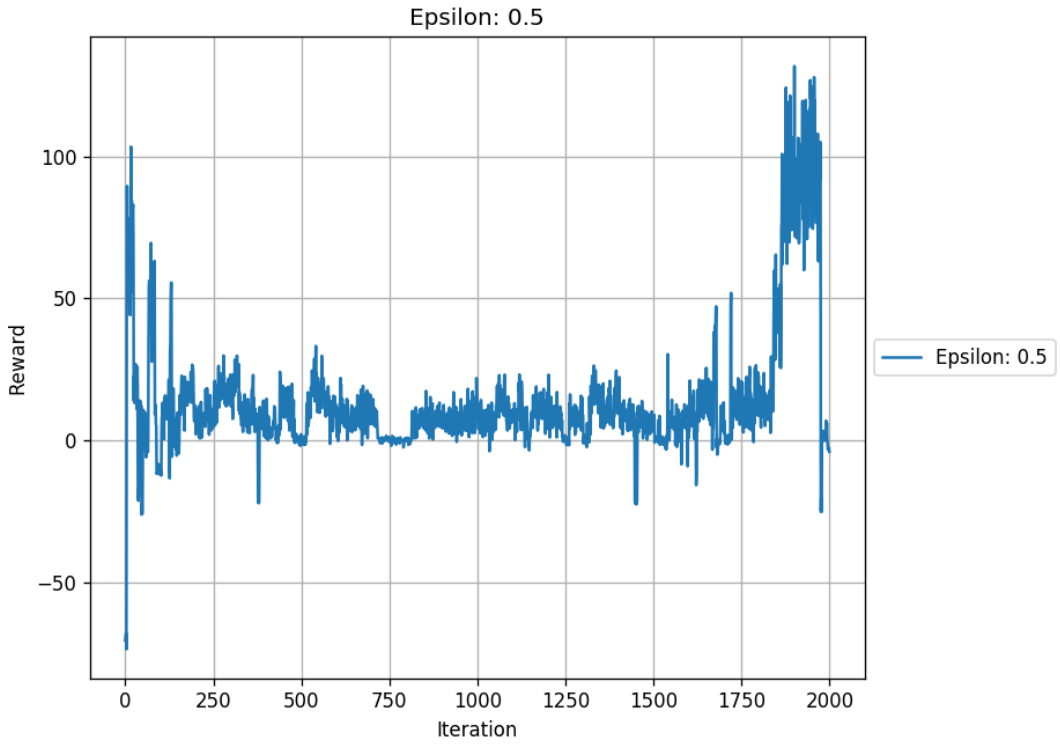
Q Learning on large maps is difficult because the agent has to take a lot of risks to explore the map. This can be time costly and inefficient. When the map is smaller, the agent takes less risks and therefore a higher chance converging to the optimal while larger maps require more time and risks to obtain the optimal rewards which is hard.

Subtask 2: Experiment with ε




Epsilon: 0.1
Reward
Iteration
150
100
50
0
−50
0
250
500
750
1000
1250
1500
1750
2000
Epsilon: 0.1

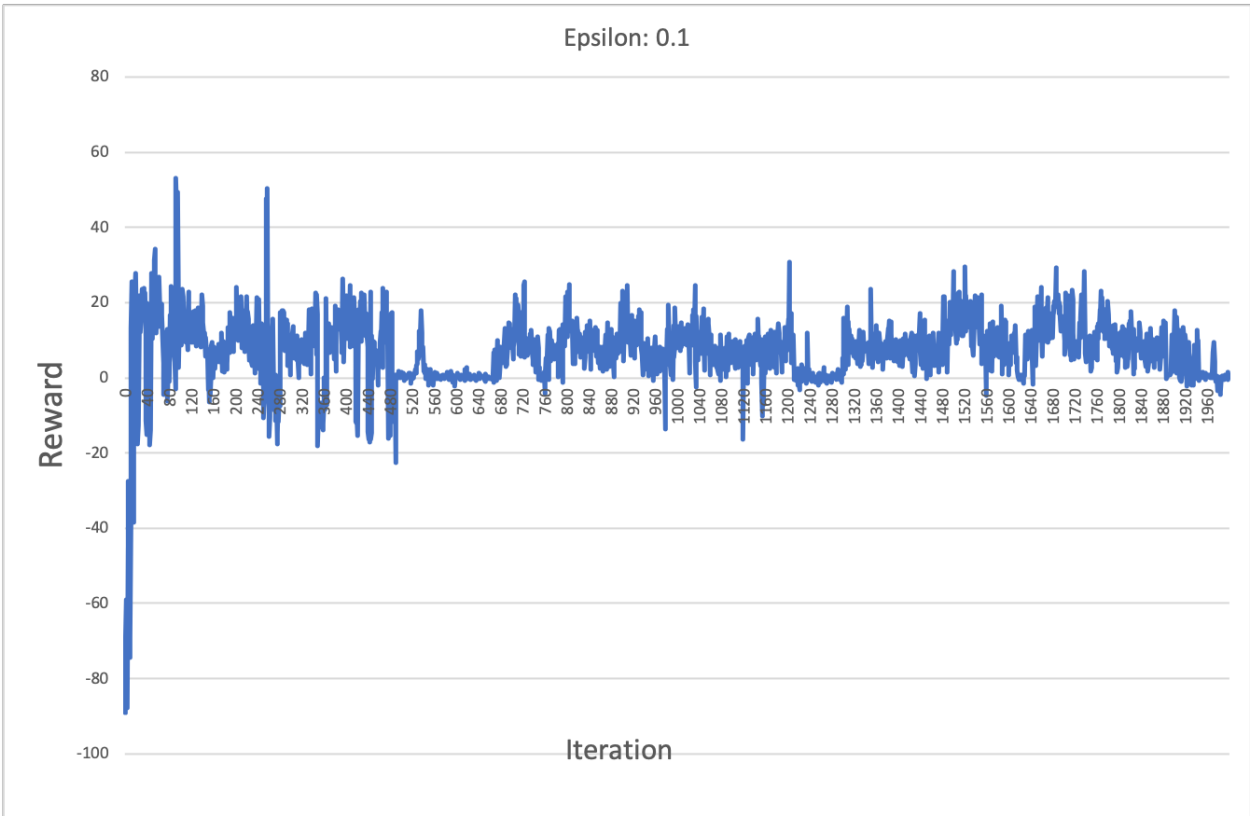

Epsilon: 0.2
Reward
Iteration
150
100
50
0
−50
−100
0
250
500
750
1000
1250
1500
1750
2000
Epsilon: 0.2



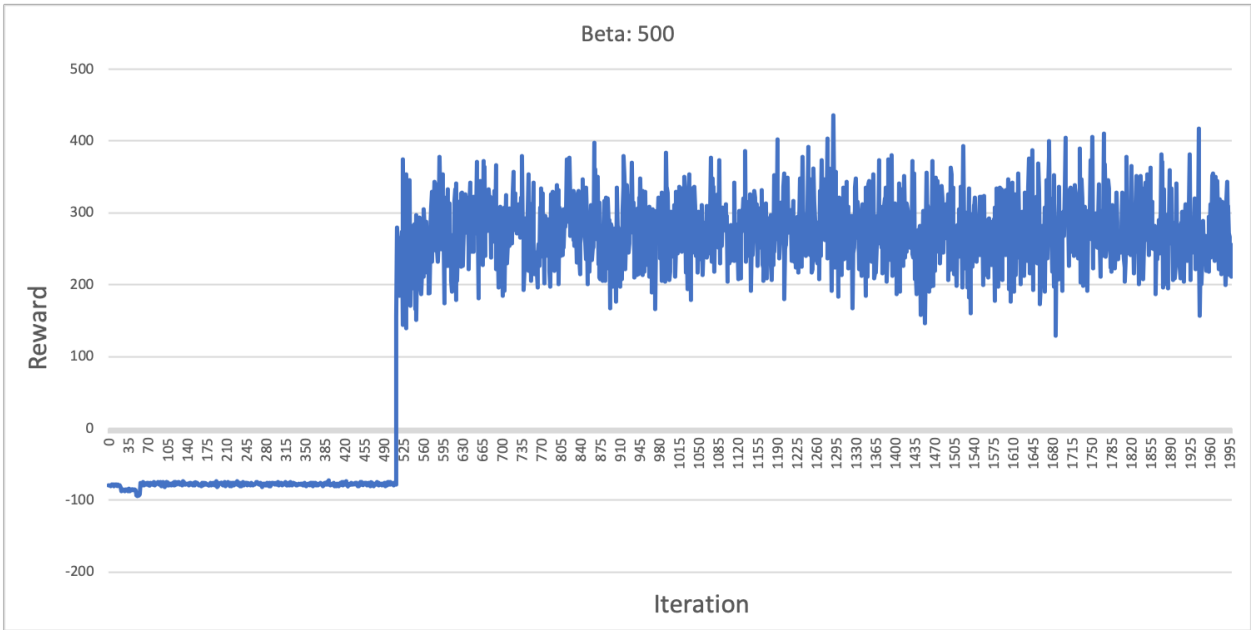
The epsilon determines if the agent should exploit or explore. The higher the epsilon number the higher the chances the agent will try to explore paths rather than exploiting. The lower the epsilon is the higher the chances the agent will exploit rather than exploring. Adjusting the epsilon value will change how the agent is picking actions at a time/state.

Subtask 3: Experiment with Counting-Based Exploration

Q Learning Epsilon Greedy Algorithm with Epsilon set to 0.1



Q Learning Counting-Based Exploration with Beta set to 500



The more the agent explores the better but if the agent isn't exploring efficiently then it will be costly and not efficient. The exploration strategy finds an optimal balance between exploring and exploiting. The agent will be exploring a lot but at the same time exploiting already known information. This way it is efficient and fast while obtaining the optimal rewards.