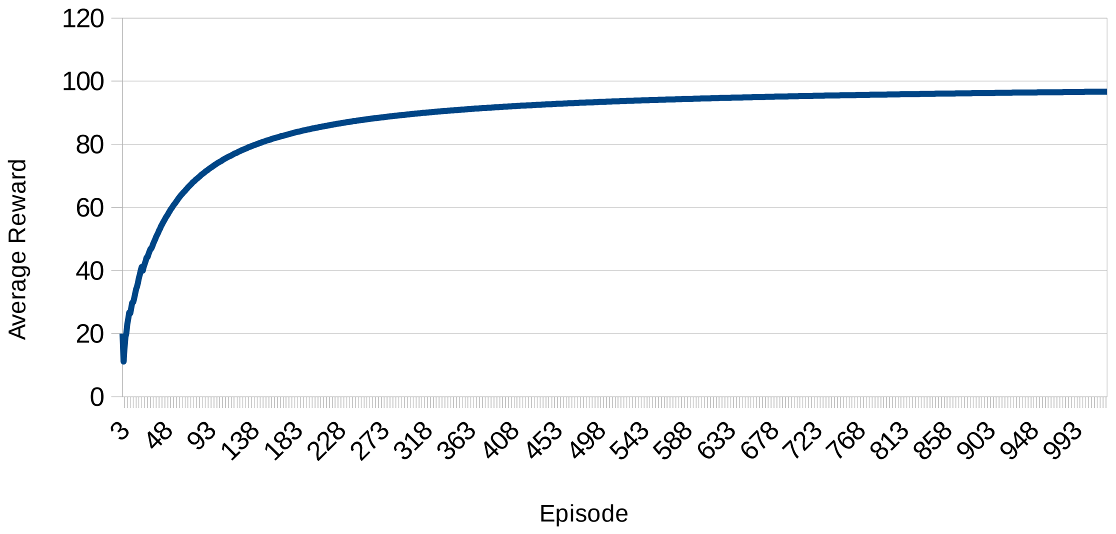
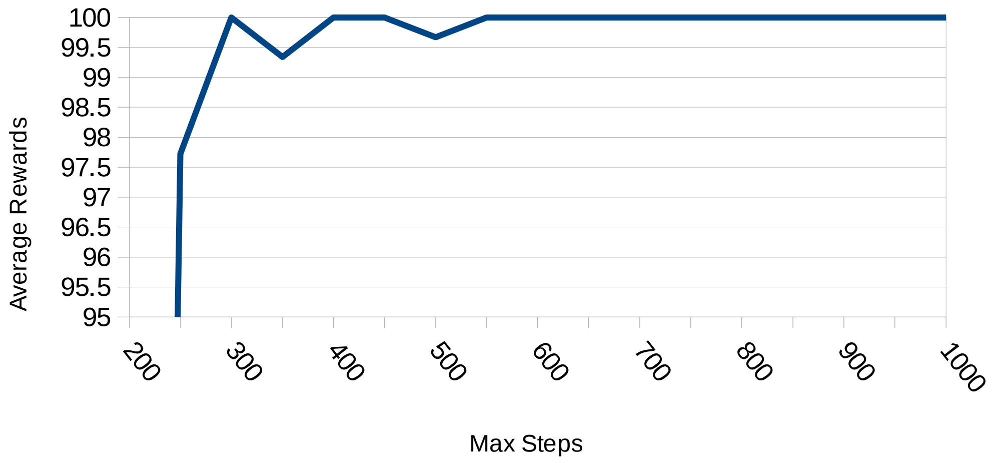
Jacob Vargo
CS 425
Project 6

To start this project I first created a function to generate some grids for which the ‘robot’ could wander around. I defined open spaces with the value ‘0’, walls with ‘1’, and the goal with ‘2’. I then saved these as comma separated files for later use. I generated two randomized grids to test with. Grid one had a sparse density of walls with the density set to 0.1. Grid two had a moderate density of walls with the density set to 0.25 which was enough to start seeing lengths of walls appear. Grid three had no walls in it. Grid four was a custom built grid that was similarly to rooms in a house with a mixture of small and large rooms.

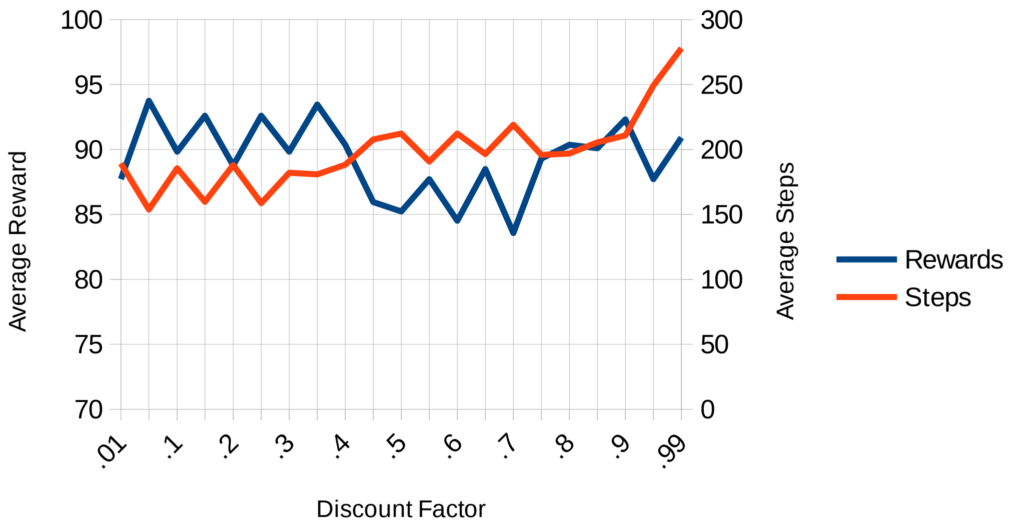
To begin testing, I needed to decide on how many successful episodes should be repeated to train the V matrix. I ran the test using grid one and moderate hyper-parameters as a starting point: discount=0.5, epsilon=0.5, update factor=0.5, trace decay=0.5, max steps=500.



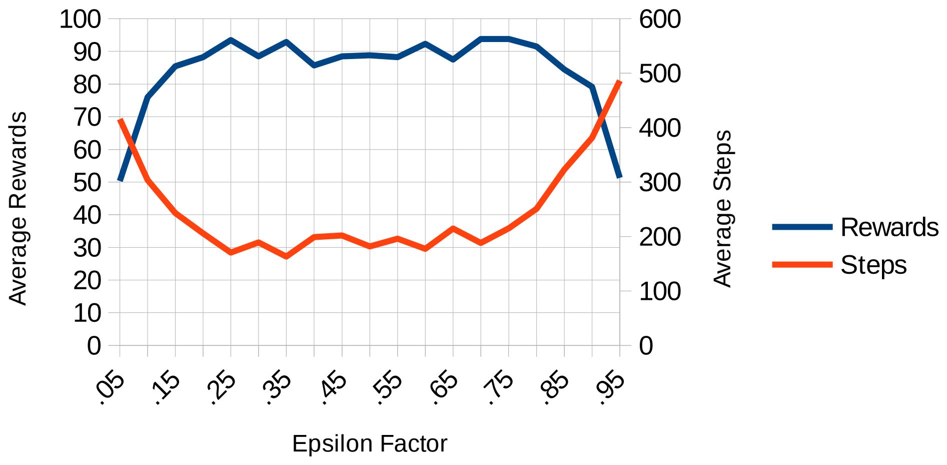
Based on my results I concluded that a larger number of episodes would be good. However, to keep run time manageable for a large test I decided to use 300 episodes in a test.



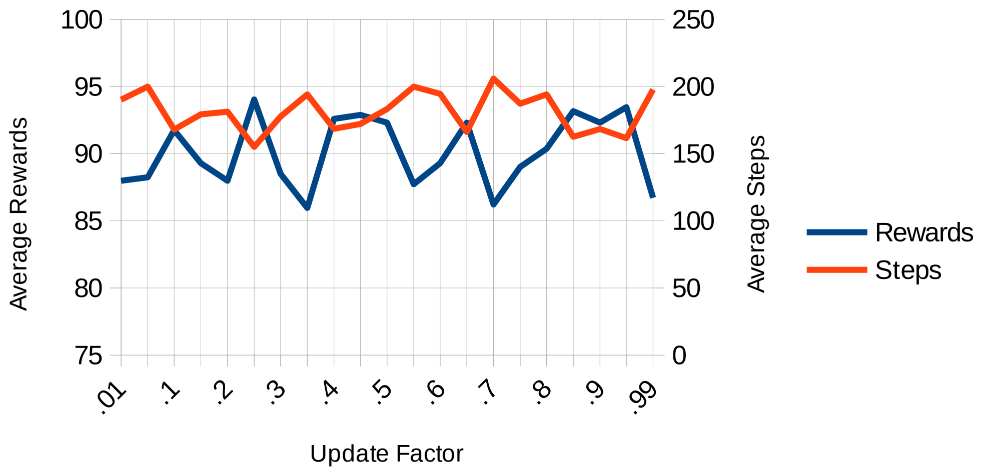
Based on my results, I decided to use 400 as the maximum number of steps allowed. I chose this figure because the grid size is 20x20 so 600 is enough steps for the ‘robot’ to visit every state at least one before reaching the goal. By increasing the max step to 600, I hope to encourage the robot to explore more in initial episodes so as to allow shorter episodes later on. After that I tested the discount factor.



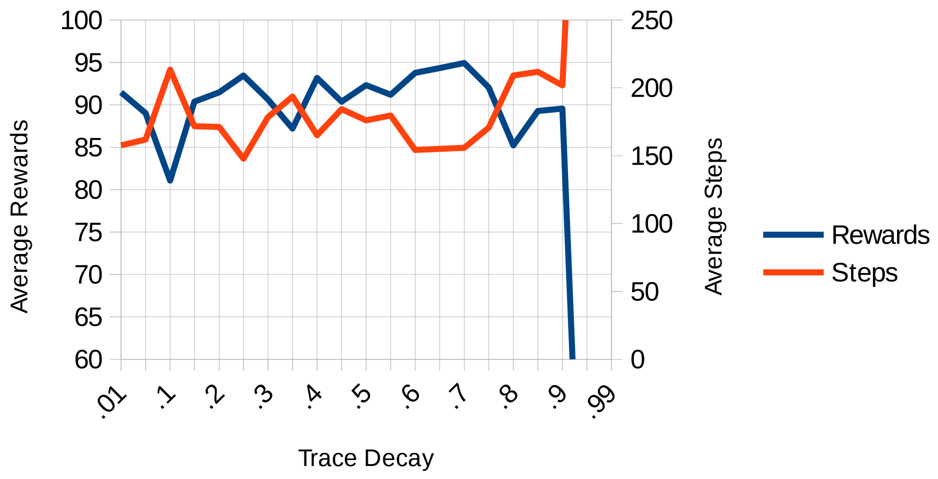
Based on my results, I decided to use 0.35 as my discount factor due to it giving a higher reward along with fewer steps. After that I tested the epsilon factor.



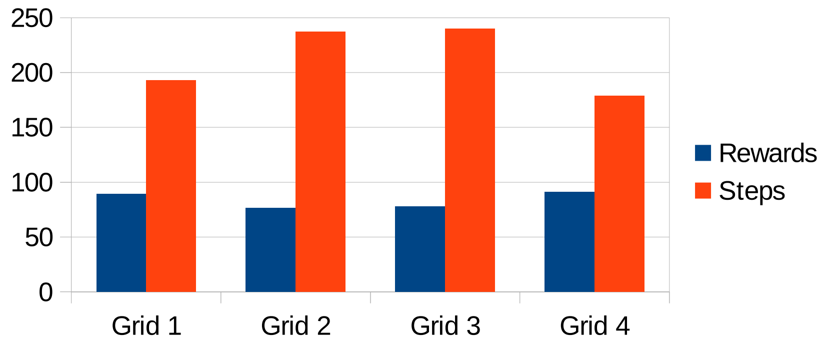
Based on my results, I decided to use 0.35 as my epsilon factor because it had a high reward as well as the lowest number of steps. After that I tested the update factor.



Based on my results, I decided to use 0.25 as my update factor because it had the highest average rewards as well as the lowest average steps. After that I tested the trace decay.



Based on my results, I decided to use 0.7 as my trace decay because it gave the highest rewards along with a low average step count. After tuning my hyper-parameters, I tested my algorithm on the different grids to see how it reacted.


Grid 1
Grid 2
Grid 3
Grid 4
Rewards
Steps