Markov Decision Processes

Introduction

This paper looks at the use of three reinforcement learning algorithms on the Grid World Markov Decision Process (MDP) problem. To begin with, reinforcement learning is the science of making decisions by using information on cause and effect, consequences of actions, and achieving goals. The reinforcement learning algorithms tested include two planning algorithms, value iteration and policy iteration, and the q-learning algorithm. An MDP problem consists of a straightforward framing of a problem of learning from interaction to achieve a goal. The agent and the environment interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent. An MDP is used to describe an environment for reinforcement learning, where the environment is fully observable. Most reinforcement learning problems can be formalized as MDP problems.

Methodology

The three reinforcement learning algorithms used in this paper can all solve the grid world MDP that is being explored. The first method, value iteration, uses the Bellman equation to evaluate the utility of different states all the way to a unique optimal converging solution. The downside of value iteration can be the lengthy process it takes to converge to this optimal policy. The second method, policy iteration, looks at iteration over policies rather than values. This method will redefine the policy at each step and compute the value according to the new policy until the policy converges. Like value iteration, policy iteration is also guaranteed to converge to the optimal policy, though it often takes less iterations to reach convergence than it does with value iteration. Both value and policy iteration algorithms are classic ways to solve MDP problems by finding the optimal policy when the agent knows sufficient details about the model's environment.

The third method, Q-Learning, is slightly different than the two previously described. Q-learning is a model-free algorithm that can be used when the agent does not know the environment model well but has to discover the policy by trial and error making use of its history of interaction with the environment. In this case, the agent must actively learn about the environment through the experience of interactions with it. Q-Learning can be more useful than the other two algorithms when less domain knowledge exists or the transition and reward functions are not known.

Grid World Problem Description

The Grid World problem is a common MDP problem as the agent travels through a grid-like square hoping to reach the end in the most rewarding and efficient way possible. There exists a grid world environment which represents many outside world examples to the agent who takes actions, receives observations from the environment that consists of a reward for its action and information of its new state. That reward informs the agent of how good or bad the action taken was, and the observation tells it what its next state in the environment should be. The agent tries to figure out the best actions to take or the optimal way to behave in the environment in order to carry out its task in the best possible way. The end goal state in this case lies in the upper right-hand corner while the agent starts in the bottom left, trying to avoid obstacles along the way. Common examples of real-world grid world problems include many recent attempts at robotic navigation, whether in the form of Tesla's self-driving cars or submarines that manage to navigate the seafloor in exploration of previously untouched underwater landscape. These problems are both interesting as they represent different levels of learning an MDP must reach to navigate increasingly complex environments. This problem is rapidly becoming an extremely important research problem as robotics begin to take over seemingly "simple" human navigational tasks.

The two versions of this problem cover an "easy" grid world and a "hard" grid world defined as a 5x5 grid and a 15x15 grid, respectively. At each different state in the grid world, the reward is -1 with the final goal statement being 100. There also exists a bit of stochasticity within the world in regard to movement, each time the agent takes an action of movement, there is an 80% chance the agent will execute the move correctly in the intended direction and a 20% chance the agent will move in a different direction. If the agent turns into the side of the grid or hits a wall, there is no movement out of the current state. The different grid world problems are displayed below in Figures 1 and 2. I use the BURLAP library with Java to explore these Grid World MDP problems.

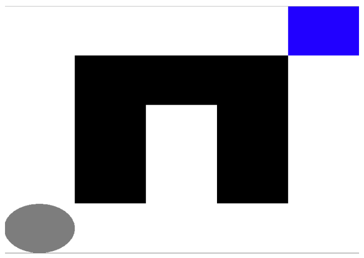
Figure 1

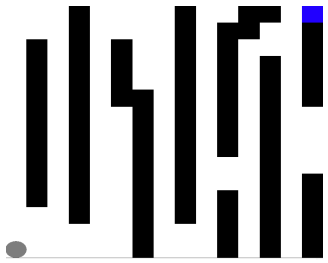
Figure 2

Q-Learning: Choosing a Learning Rate and Epsilon

Before beginning to explore the two different MDP grid world problems, I evaluated the best learning rate and epsilon to use for the Q-Learning algorithm. I did this using the Easy Grid World problem for sake of ease and applied the results to both the easy and hard grid worlds for a comparable analysis. The learning rate (or step size) determines to what extent new information overrides old information with a value of 0 indicating an agent will learn nothing while a value of 1 indicates the agent will consider only the most recent information learned. A learning rate of 0.1 is commonly used with stochastic MDP problems, such as the ones we are working with. The epsilon value determines the importance of future rewards for the agent.

Given the fact that stochastic MDP problems normally perform best with a learning rate of 0.1, I choose to use that for my analysis and in determining an appropriate epsilon. In Figure 3, the epsilon values of 0.3, 0.5, and 0.7 are shown in their performance of convergence vs. iterations. From this graph, the epsilon value of 0.7 seems to perform best as it seems to converge in the fewest number of iterations out of the three epsilon values. It would also seem that the 0.7 epsilon curve is smoother with fewer spikes indicating that more randomness is actually better in this case when an agent is seeking out its best policy option. Thus I used an epsilon value of 0.7 when evaluating the Q-Learning algorithm in both the easy and hard grid world problems.

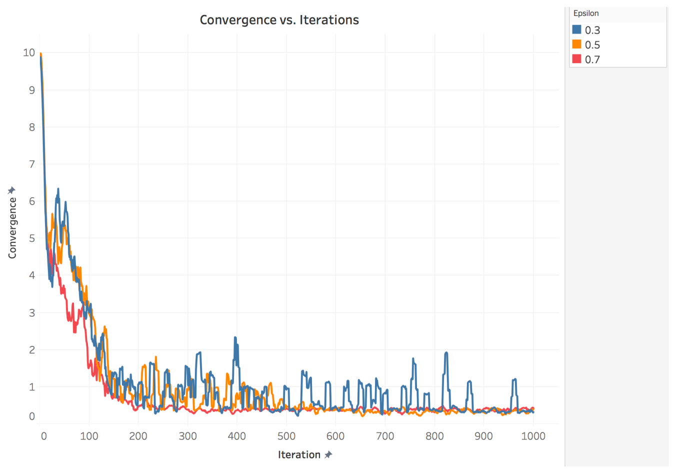

Figure 3

Easy Grid World

In the first look at the grid world problem, the “easy” version is explored which encompasses the 5x5 grid world setup. The value iteration and policy iteration algorithms were run with 100 iterations while the Q-Learning algorithm was run with 1000 for easier comparison. The learning rate for Q-Learning is 0.1 with an epsilon value of 0.7. The main focus of this exploration is to determine the optimal policies produced by each of the three different algorithms, and to evaluate their similarities and differences. Other points of evaluation include iterations to convergence, length of runtime for each algorithms’ iterations and convergence, as well as the interactions between iterations, convergence, runtime and reward values.

The optimal policies for each of the three algorithms all reached the same maximum reward value of 93, although it took each Q-Learning a far different route to get there than the value and policy iteration methods. Below in Figures 4, 5, and 6 are the optimal policy routes for each of the three algorithms. Value and policy iteration algorithms only required 100 iterations to reach their optimal policies whereas Q-Learning took much more with 1000 iterations being used. The outputs for the value iteration and policy iteration grids are the same with only 100 iterations, but this was much too few to use for Q-Learning. This could be due to the fact that the reward function and domain knowledge is known for the first two algorithms, thus requiring fewer iterations to find the optimal policy. Begin unable to converge in only 100 iterations, the optimal policy for Q-Learning reveals itself after 1000 iterations are tested.

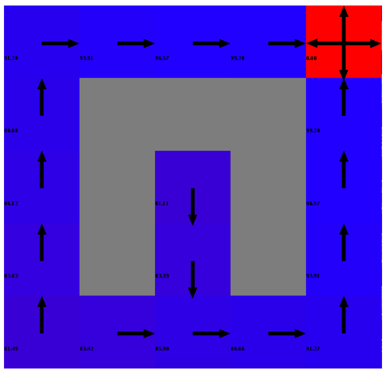

Figure 4 – Value Iteration

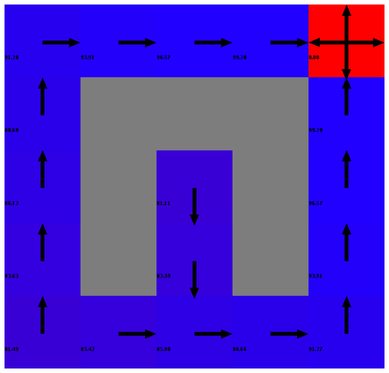

Figure 5 – Policy Iteration

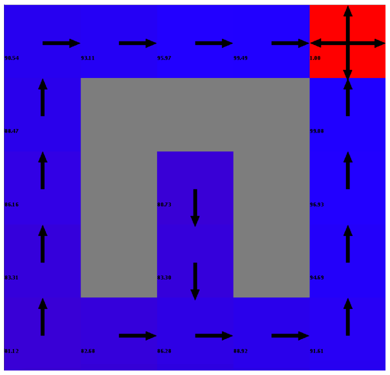

Figure 6 – Q-Learning

In order to analyze these algorithms further, looked at the clock time of convergence for all three as shown in Figure 7. In this graph it can be seen that Q-Learning is by far the quickest of the three to converge as it uses a constant speed to calculate at each step what its next move will be. The value iteration algorithm attempts to improve the value function at each iteration until it converges, which could be an explanation for the lengthier time as well as the sharp increases in time throughout the process. Policy iteration also takes a longer time to reach convergence as it is redefining the policy at each step. This difference in time can also be seen in Figure 8, as both the policy and value iteration algorithms take far longer in only 100 iterations (used for comparison) than the Q-Learning algorithm. When looking at the Q-Learning algorithm over its full 1000 iterations, the time does not ever increase significantly. Overall, the time to converge and iterate is much shorter for Q-Learning though the total number of iterations it takes to reach the optimal policy is shorter for value and policy iteration.

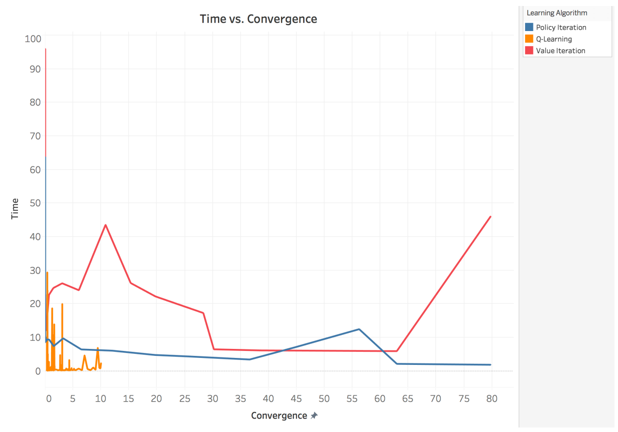

Figure 7

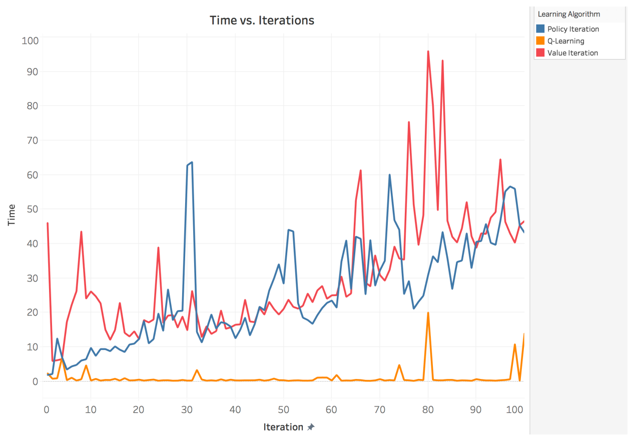

Figure 8

I also looked at the reward values versus the number of iterations for each of the three algorithms as shown in Figure 9. It can be seen that the value and policy iteration methods take fewer iterations (around 2-3) to reach maximum rewards, whereas Q-Learning seems to need 11 or more to hit the same high reward number. This could again be due to the way each method works to find its optimal policy.

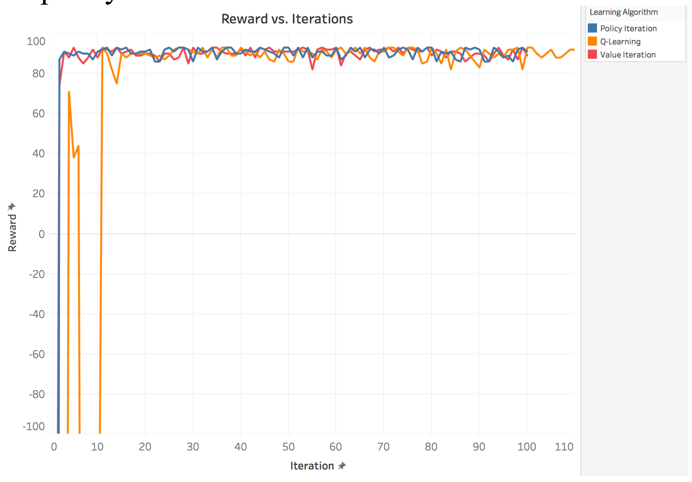

Figure 9

The graph in Figure 10 also confirms this idea of value iteration and policy iteration converging to an optimal solution over a fewer number of iterations. Here we see that within 100 iterations, the value and policy methods quickly reach convergence, whereas the Q-Learning continues on well past this 100 iteration mark.

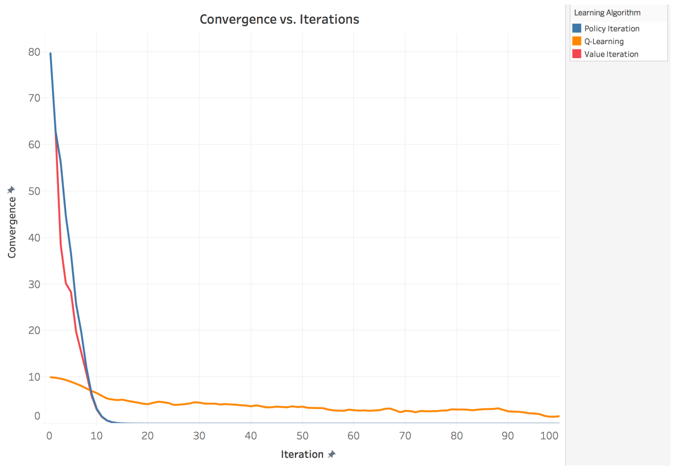

Figure 10

Finally, when evaluating the number of steps to reach the maximum reward of 93, each algorithm performs differently with value and policy iteration reaching this reward after only about 9 steps under 100 iterations whereas for Q-Learning to reach the same number of steps and equal max reward, it takes over 300 iterations.

<u>Hard Grid World</u>

In the second look at the grid world problem, the "hard" version is explored which encompasses the 15x15 grid world setup. Differently than before, the value iteration and policy iteration algorithms were run with 200 iterations while the Q-Learning algorithm was run with 500 iterations for the best comparison. The learning rate for Q-Learning is 0.1 remains the same with an epsilon value of 0.7. The optimal policies for value iteration and Q-Learning reached max rewards of 43, while policy iteration was able to reach a max reward of 44. The optimal policies are shown below in Figures 11, 12, and 13.

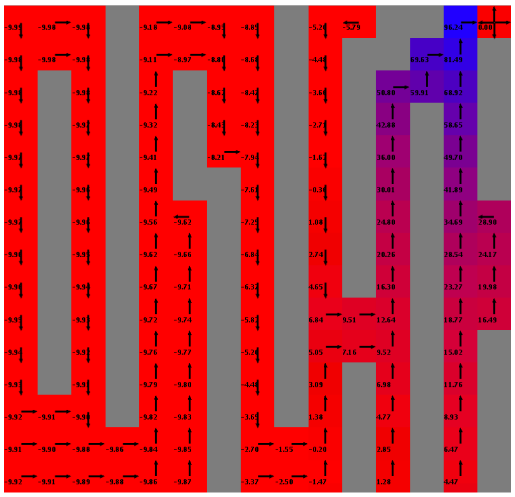

Figure 11 – Value Iteration

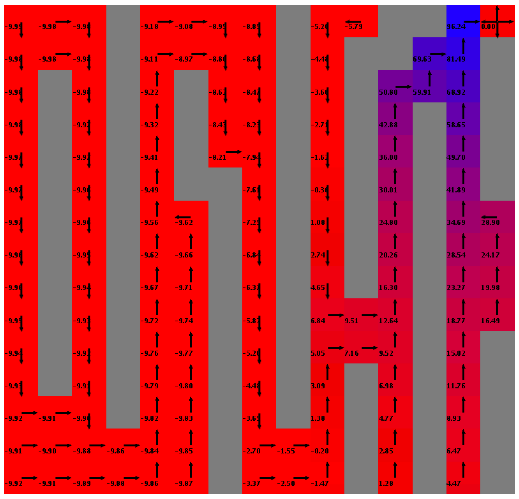

Figure 12 – Policy Iteration

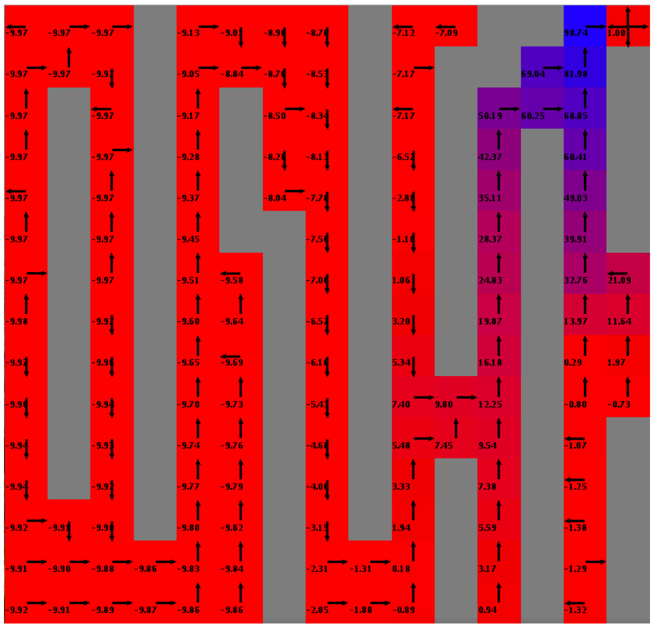

Figure 13 – Q-Learning

As with the easy grid world problem, the value and policy iteration optimal policies are the same using 200 iterations, whereas the Q-Learning policy required 500 iterations to reach an optimal policy. Similar to the analysis of the easy grid world, the hard grid world analysis of time to convergence and over iterations shows similar results. In Figure 14, it can be seen that Q-Learning again is by far the quickest of the three to converge with value iteration taking the longest, followed closely by policy iteration. The different calculation methods may explain this difference. This difference in time can also be seen in Figure 15, as both the policy and value iteration algorithms take far longer in only 200 iterations (used for comparison) than the Q-Learning algorithm. When looking at the Q-Learning algorithm over its full 500 iterations, the time does not ever increase significantly. Overall, the time to converge and iterate is much shorter for Q-Learning though the total number of iterations it takes to reach the optimal policy is shorter for value and policy iteration. In comparison to the easy grid world, the time for value and policy iteration is drastically higher, which makes sense as the hard grid world is a much larger environment for the agent to navigate. However, the Q-Learning clock time is similar for convergence and iteration, which could also makes sense as the agent does not try to learn explicit models of the environment state transition and reward functions possibly explaining the shorter times.

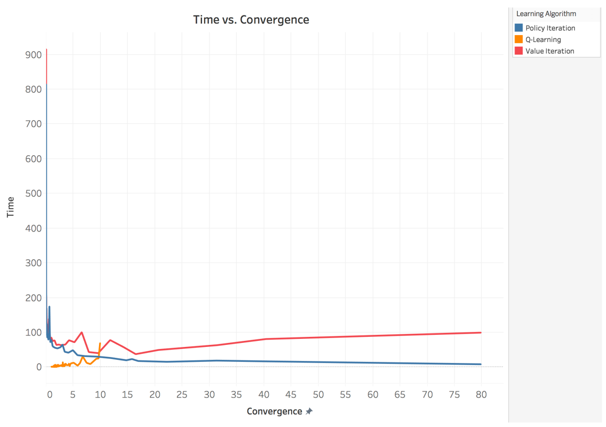

Figure 14

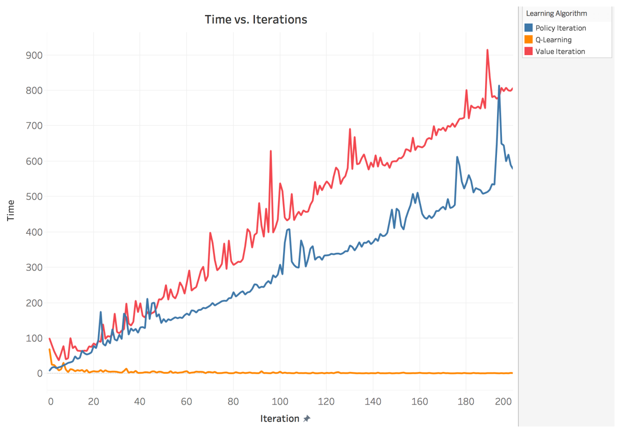

Figure 15

In Figure 16, the rewards vs. iterations is again explored with similar results as the easy grid world analysis. It can be seen that the value and policy iteration methods again take fewer iterations (around 80-90) to reach maximum rewards, whereas Q-Learning seems to need around 250 or more to hit the same high reward number. This could again be due to the way each method works to find its optimal policy. This also shows the need to use Q-Learning with higher iterations if you want to reach the same max rewards value and policy iteration can provide.

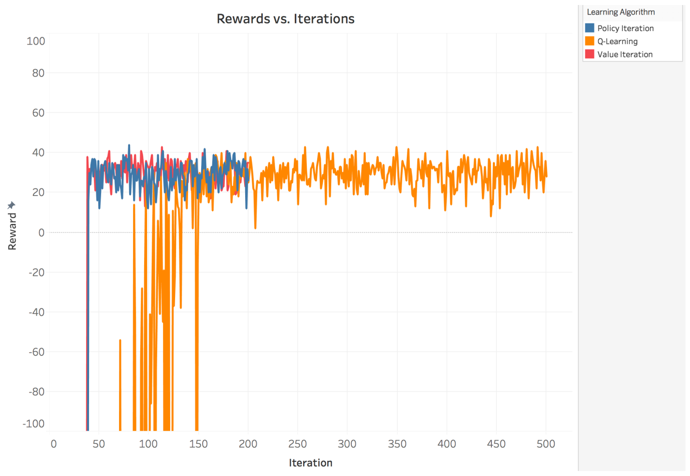

Figure 16

Finally, the graph in Figure 17 again confirms this idea of value iteration and policy iteration converging to an optimal solution over a fewer number of iterations. Here we see that within 200iterations, the value and policy methods quickly reach convergence needing only about 30 iterations to do so, whereas the Q-Learning continues on well past this 200 iteration mark.

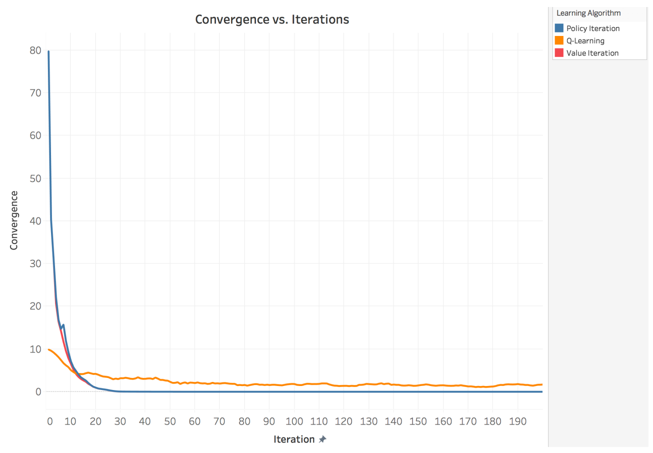

Figure 17

When evaluating the number of steps to reach the maximum rewards of 43 or 44, each algorithm performs differently. Value iteration and Q-Learning each take 59 steps to reach their max reward value of 43, while policy iteration takes 58 steps to reach the higher max reward value of 44. However, policy iteration performs the best in the sense that it takes far fewer iterations than either of the other two algorithms to reach this max reward point with value iteration followed by Q-Learning taking many more iterations to accomplish this same task.

Conclusion

In both the easy and hard grid world problems, policy iteration seemed to perform the best in regard to requiring the fewest number of iterations to reach the maximum reward. Value iteration had somewhat similar results, but it took longer and was not able to reach the higher reward value in the hard grid world problem. Overall, Q-Learning was by far the quickest method as its time stayed consistently low whereas both policy and value iteration methods often took much longer. If the goal is to find quick convergence to an optimal policy, then value iteration and policy iteration should be used. But if keeping time low is a crucial aspect in your analysis, the Q-Learning is more useful. Perhaps using different learning and epsilon values could improve the timing of this method to make it better.