Georgia Institute of Technology
CS 7641 – Machine Learning
Markov Decision Processes

Santhanu Venugopal Sunitha (ssunitha)

November 25, 2018

The purpose of this paper is to apply Policy Iteration, Value Iteration and Q Learning to two interesting Markov Decision Processes (MDP) and analyze the results.

Grid Worlds

Two grid worlds of different sizes and complexity are chosen to represent the Markov Decision Processes. In both grid world problems, an agent must traverse from the bottom left corner towards the top right corner, avoiding obstacles in the way. The grid worlds are stochastic in nature as the agent's actions does not always execute as per the plan and the same is modeled through a transition probability matrix. There is an 80% chance that the agent moves to its intended direction and 6.67% chance that it moves to the other three directions. Agent receives a reward of 100 for reaching the goal state and the cost function is set as -1, except for the obstacles on the way.

The first grid world problem is an easy 10 X 10 grid world and is designed in such a way that there are two short paths to reach the goal state but one path has more obstacles (-5 red and -1 yellow) compared to other (-3 orange) and hence would be interesting to observe the optimal policy generated by the three algorithms on the same.

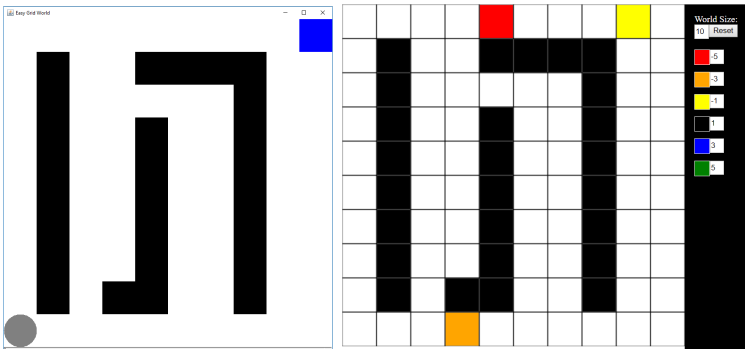


Figure 1: Easy Grid World

The second grid world is a 20 X 20 grid world with more obstacles in the path to terminal, making it a bit complex. There is obviously more no. of states and more negative rewards along the path and the agent has to tactfully plan it's route to maxmize the reward.

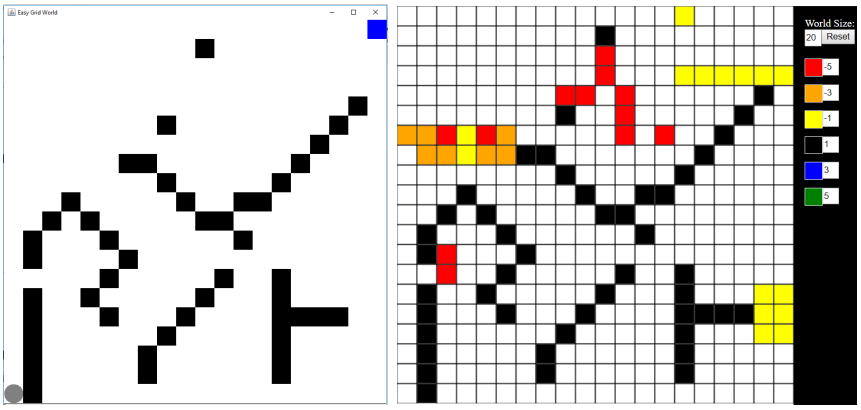


Figure 2: Hard Grid World

Grid world problems are interesting as they represent real world problems such as robot path planning in a warehouse, maze solving etc. Due to the difference in complexity of the two grid world problems, we can hope to throw light on how these three algorithms behave in these two differing MDP scenarios.

Methodology

The above three algorithms were implemented using Burlap software. Both Policy Iterations and Value Iterations were ran over 100 iterations, while Q Learner was ran using higher iterations. The results obtained from the algorithms were saved in excel and plotted later using Matplotlib. Both Policy Iterations and Value iteration convergence threshold was set at $< 1e^{-6}$, while for Q Learner it was < 0.5.

Policy Iteration and Value Iteration

Policy Iteration is a dynamic programming algorithm which find the optimal policy by starting with an arbitrary policy and iteratively improving it via policy evaluation and policy improvement steps.

Value Iteration is a dynamic programming algorithm which iteratively calculates the utilities of each states using the utilities of the neighboring state until the value-function converges, after which calculating the optimal policy would be an easy task.

Easy Grid World Analysis

Policy Iteration algorithm converged to the optimal policy in just 23 iterations while Value Iteration algorithm converged in 60 iterations. Both converged to the same optimal policy as well. As depicted in the below chart for both algorithms, the optimal policy shows path planning by avoiding the red, yellow and orange obstacles along the way, as expected.

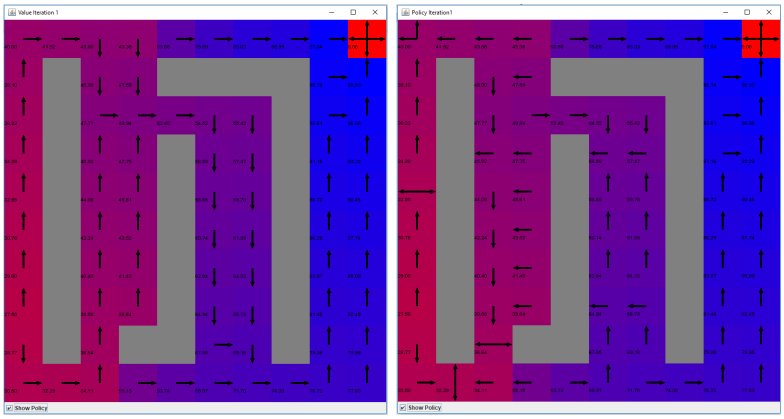


Figure 3: Value Iteration and Policy Iteration at Iteration = 1

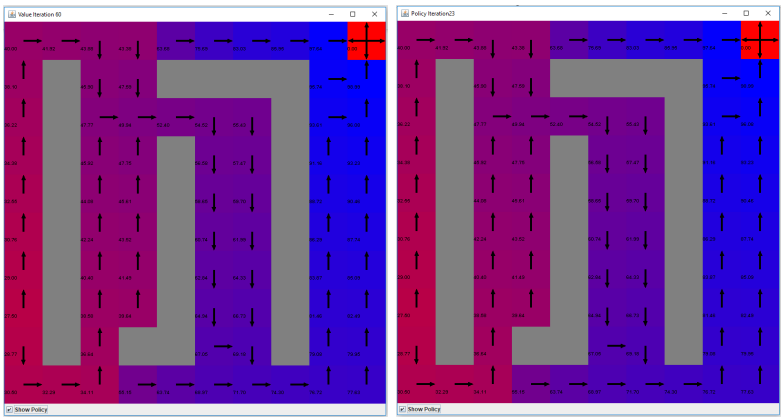
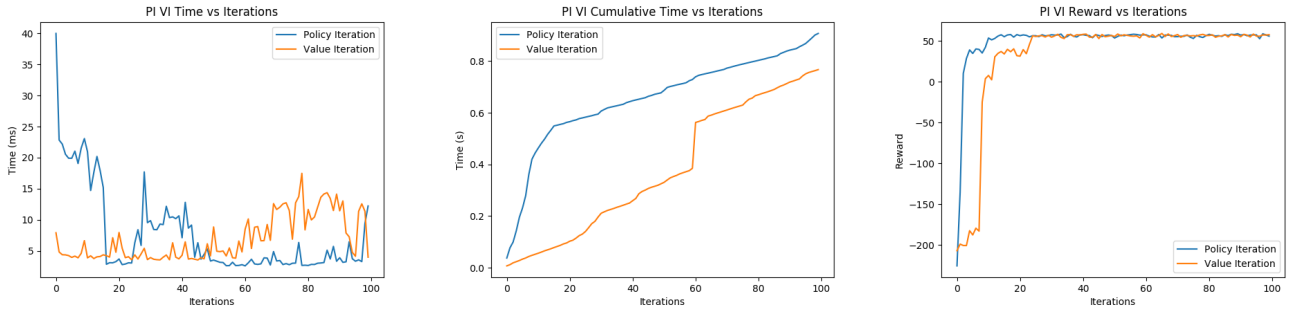


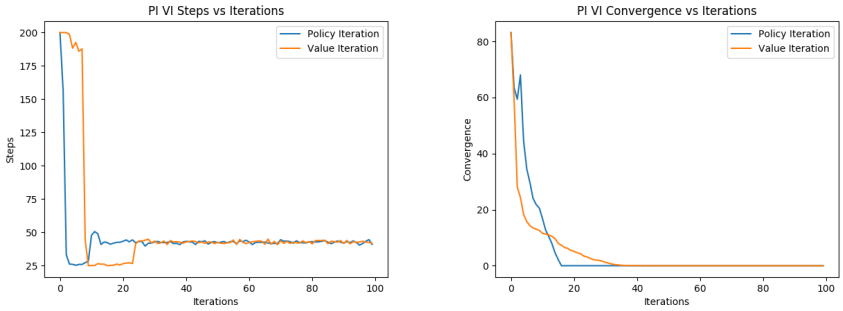
Figure 4: Value Iteration at Iteration = 60 and Policy Iteration at Iteration = 23

Based on the below charts, Policy iteration took more time at each iteration in comparison with Value iteration, until it converged at 23rd iteration. It can be observed that the initial iterations of Policy Iteration algorithm took lot more time than the later iterations. This shows the computational complexity associated with this algorithm at each iteration, till it converges.

Value iteration algorithm can be seen as having relatively lower time consumption per iterations with less variance, except for a small spike towards the end. As evident from the cumulative time plot below, for 100 iterations, Policy Iteration took more time than Value Iteration. Value iteration algorithm involves starting of with random values for the states and then iteratively improving it until optimal values are reached, followed by a policy extraction process to get the optimal value. Hence, there is just one value improvement step at every iteration till convergence, which accounts for the lesser time interval per iteration. However, Policy iteration algorithm involve starting with a random policy, followed by a policy evaluation step and policy improvement step. These two steps are repeated for every iteration, which accounts for its higher time interval per iterations.

But Policy iteration converges quickly in terms of iterations, due to the above mentioned two steps process. In comparison, Value Iteration takes more time as finding optimal values is more time consuming than policy convergence.





Hard Grid World

As the complexity of the grid world increased, we can observe that both Value Iteration and Policy Iteration algorithms converged at a bit higher iteration, with Policy Iteration taking 25 iteration and Value Iteration taking 62 iterations, which are two more than the previous grid world. They both converged to the same policy at the end. So, the increased grid complexity did not drastically increase the no. of iterations required for convergence.

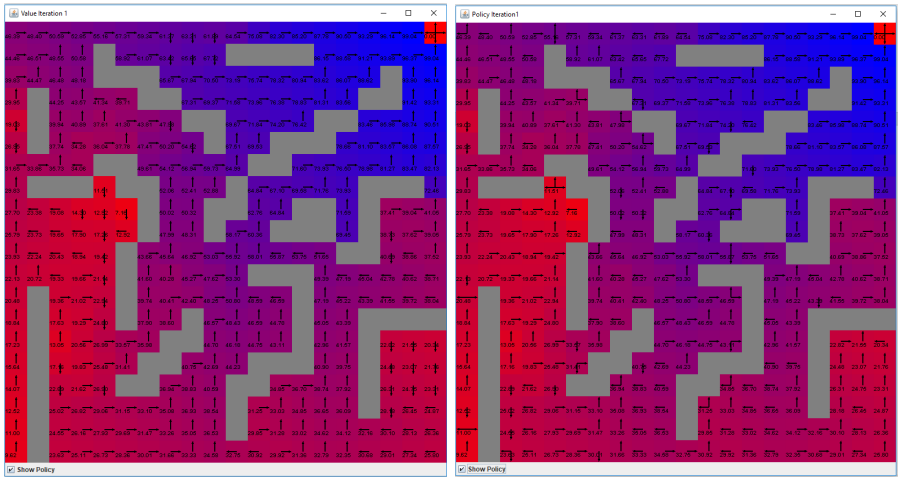


Figure 5: Value Iteration and Policy Iteration at Iteration = 1

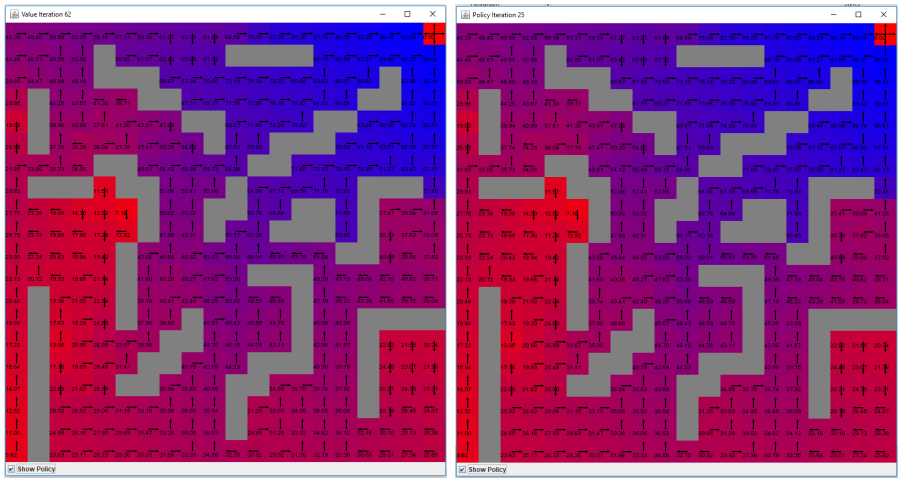
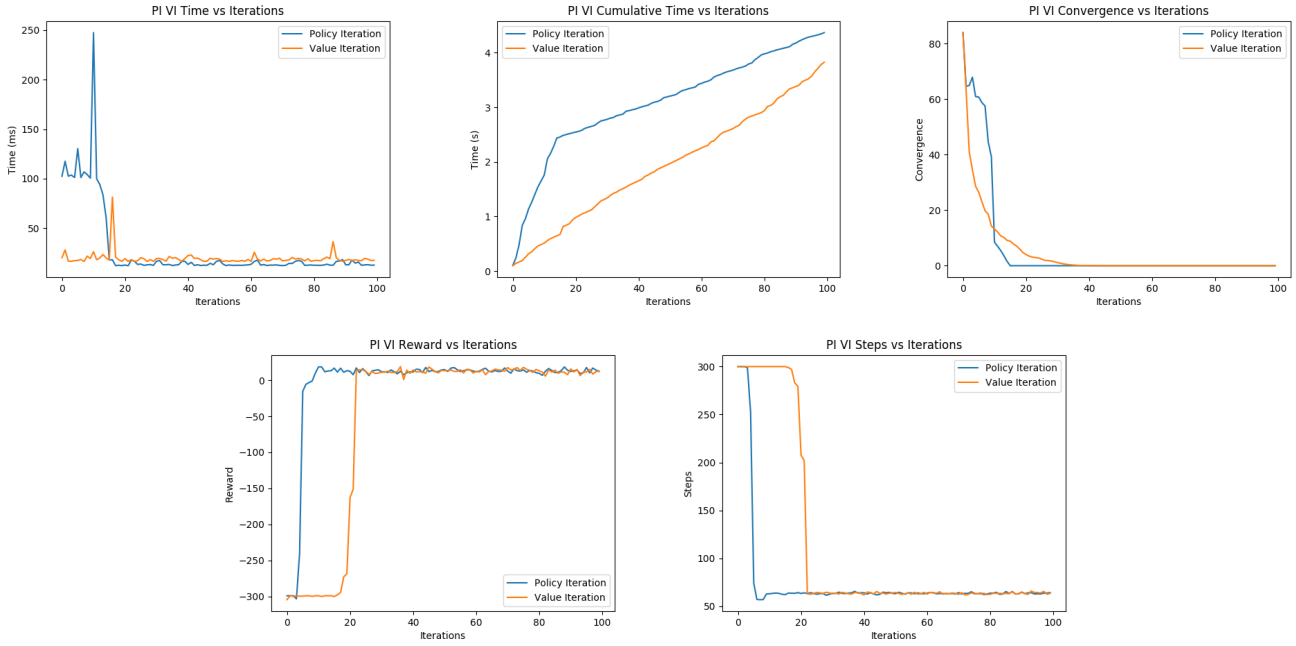


Figure 6: Value Iteration at Iteration = 62 and Policy Iteration at Iteration = 25

However, based on the below plots, increase in grid complexity increased the time complexity for both algorithms, as we have 4 times the no. of states than the previous grid world. While Value iteration took around 3.5secs for 100 iterations, Policy Iteration ran for more than 4 secs.



Effect of Discount Rate γ

Discount Rate determines how much we value the future rewards. A high discount rate would imply that we value future rewards very significantly. In contrast, lower discount would give less importance to later rewards.
Upon implementing the algorithms again after changing the discount factor for both Value Iteration and Policy Iteration from 0.99 to 0.6, we observe that the convergence occurs very quickly for both. However, based on the below plots, the optimal policy at convergence does not seem to be same as the one for discount rate 0.99. Also, few states in the policy does not seem to have the right action as well. The lower discount rate has significantly reduced the total rewards and made it negative. Hence, the discount rate of 0.99 seems suitable for both grid world problems.

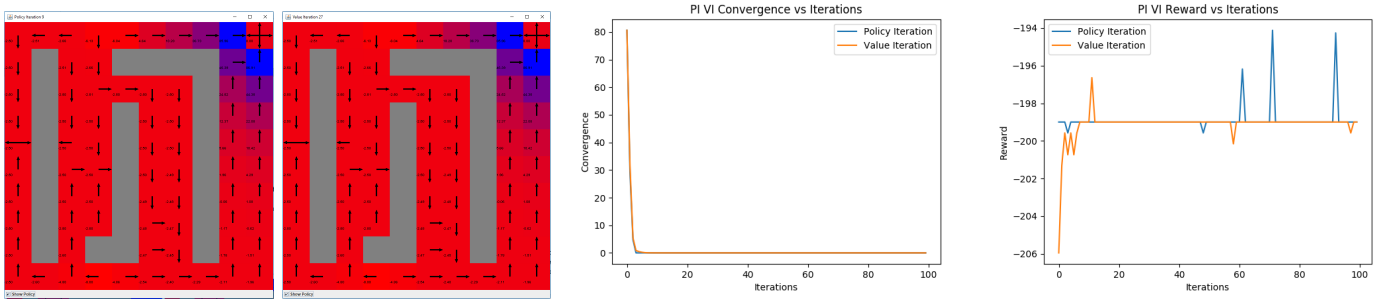


Figure 7: Easy Grid World

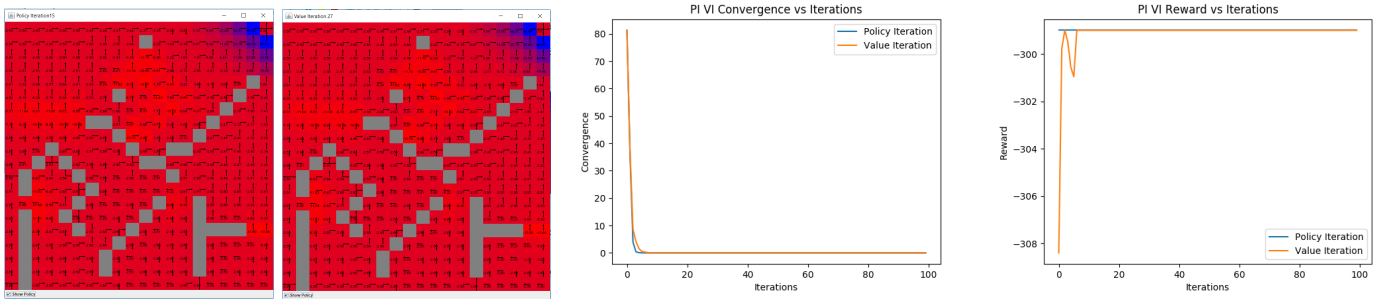


Figure 8: Hard Grid World

Q Learning

With a model free approach like Q Learning, as the agent does not have the knowledge of the state transition probabilities and rewards, it has to find an optimal policy by exploration of the environment and exploitation of the rewards received as a result. This algorithm, hence takes more iterations to converge to an optimal policy.

Hyperparameter Tuning

The hyperparameters of Q Learner includes learning rate, epsilon and Q Initial values. Below are the range of values used for tuning the Q Learner.

Hyperparameter	Range
Learning Rate	0.1, 0.9
Epsilon	0.1, 0.3, 0.5
Q Initial Value	-100, 0, 100

The algorithm was implemented with an epsilon decay strategy of 0.1.

Easy Grid World

Q Learning algorithm achieved convergence less than 0.5 around 60 iterations. However, the rewards were very negative and hence it did not achieve the optimal policy at 60 iterations. After running the algorithm for over 10K iterations, it was observed that the optimal policy was obtained at around 600 iterations for the below hyperparameter combination.

Hyperparameter	Optimal Value
Learning Rate	0.1
Epsilon	0.5
Q Initial Value	100

It is also observed that the optimal policy differed at different iteration and it does not look similar to the ones from Policy Iteration and Value Iteration.

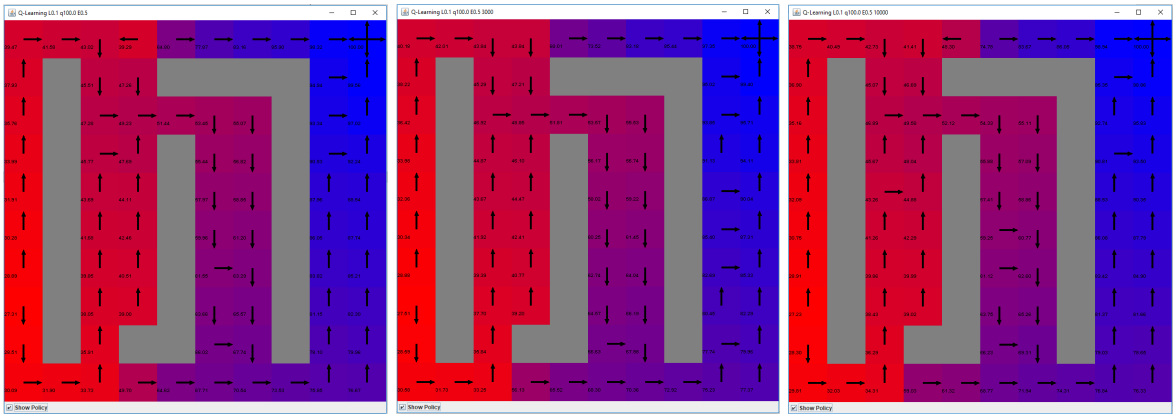
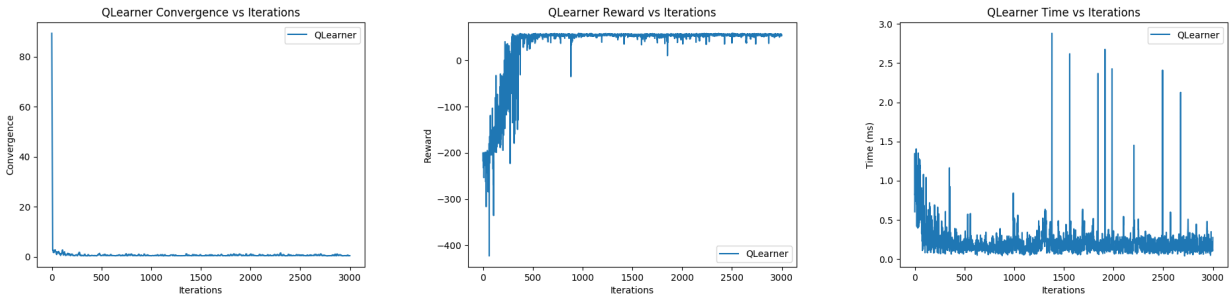
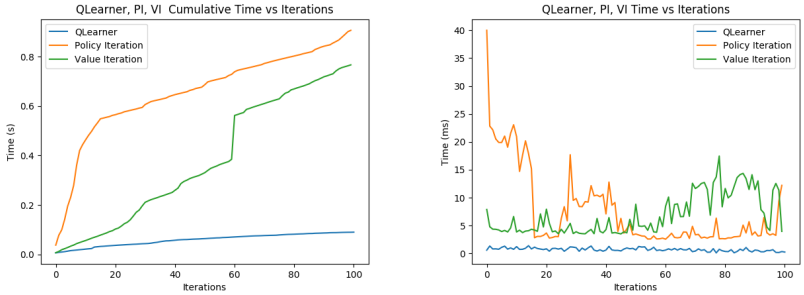


Figure 9: Q Learner Policy at Iteration = 600, 3000, 10000

By observing the Convergence plot in conjunction with the reward plot, it is evident that Q Learning algorithm have converged at around 600 iterations to a maximum reward of 58. At higher iterations, we can see less variance in the reward. Time plot shows that Q Learner is relatively stable time wise at each iteration, with the exception of few spikes.



By comparing the same with Policy Iteration and Value Iteration for the first 100 iterations, Q learner took the least amount of time, both at each iteration and overall. This shows that the computational complexity at each iteration for Q Learner is lesser than other two algorithm, as at each iteration it is just taking an action based on the current Q table value and then updating the Q table based on the reward receievd for taking that action.



Hard Grid World

Q learner algorithm was executed over 500K iterations, but it could not achieve a near optimal policy as depicted below. Below plots are for Epsilon ϵ-0.1, Q Initial of 100 and Learning Rate α of 01.There are many states which do not show the right action.

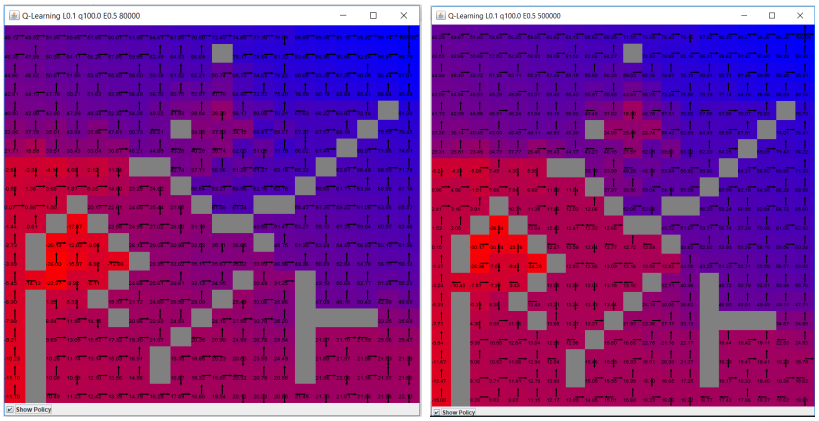
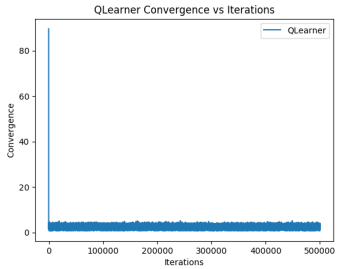
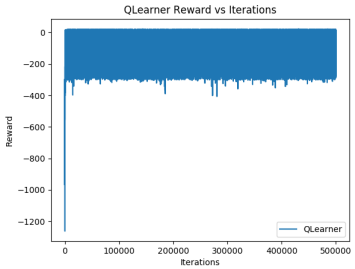


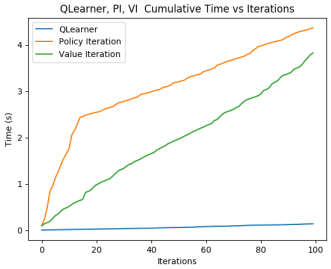
Figure 10: Q Learner Policy at Iteration = 80K, 500K

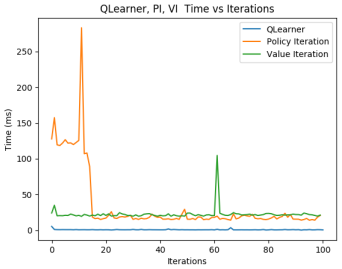
Above plots shows the policy at 80K and 500K iterations, which indicates that even more no. of iterations may be required for the learner to converge. As shown the below plots, the total rewards are negative and shows high variance. The convergence plots shows a bit high variance as well. Thus, increase in the MDP states and complexity would significantly increase the no. of iterations required for convergence as more no. of states and higher complexity implies more exploration and freqeuent updates of the Q table, which is very time consuming. A technique like Dyna-Q may also help here as the agent can hallucinate the Q values over some no. of iterations , which can help in faster convergence.





Below time complexity comparison between all three algorithms for 100 iterations of the hard grid world depicts that while Policy Iteration and Value Iteration took more time, Q Learner took almost the same time as the easy grid world, which shows that computational complexity within an iteration doesnot increase with increase in MDP model complexity like the other two.

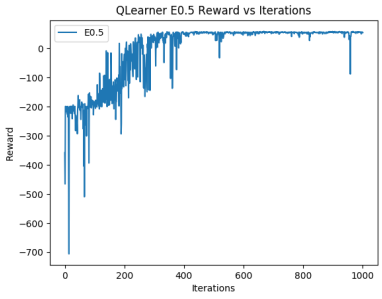


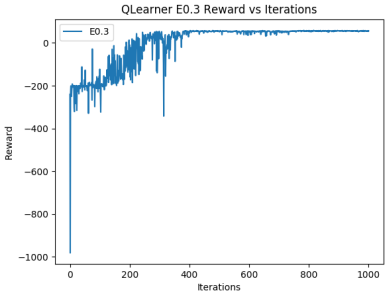


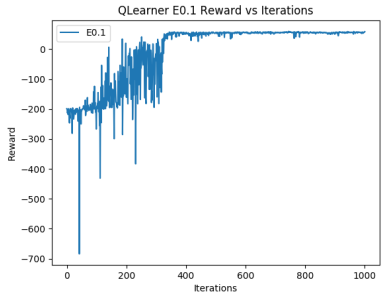
Effect of Epsilon ϵ, Q Initial and Learning Rate α

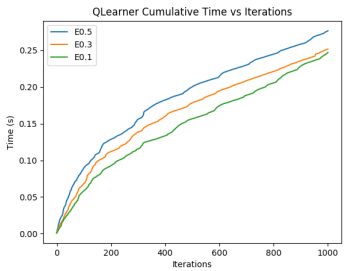
Success of reinforcement learning largely depends on the exploration - exploitation trade off. There are different exploration strategies used in reinforcement learning like ϵ-greedy, softmax etc. Here, we will study the effect of Epsilon ϵ, Q Initial and Learning Rate α on Q Learning algorithm.
Learning rate α determines how quickly new information is propagated to the Q table and override old information. Low value of learning rate will cause the learning to be slower. Q Initial values also encourage exploration when it is initialized to high values. However, over estimating the initial values can cause the convergence to be lot slower. Epsilon ϵ determines the probability that a random action will be taken instead of a greedy action. This parameter is decayed at a constant rate until it diminishes and greedy action is chosen all the time by the learner. Here, we had set a decay rate of 0.1 for the learner.

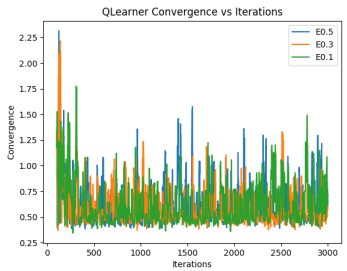
Firstly, **at a learning rate α of 0.1 and Q Initial of 100**, for different rates of Epsilon, Q learner performance was plotted over first 1000 iterations for further analysis.



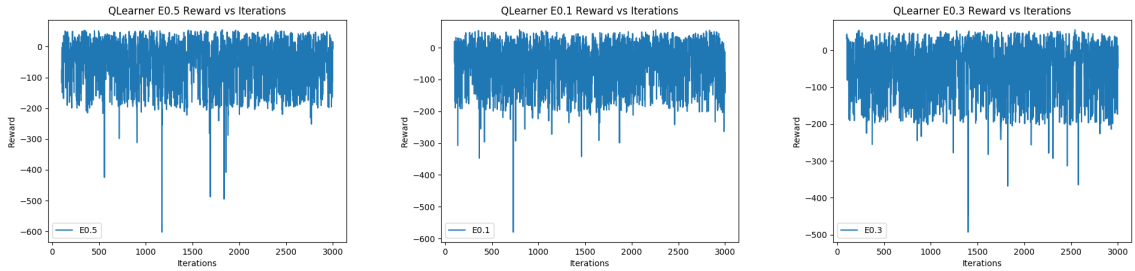




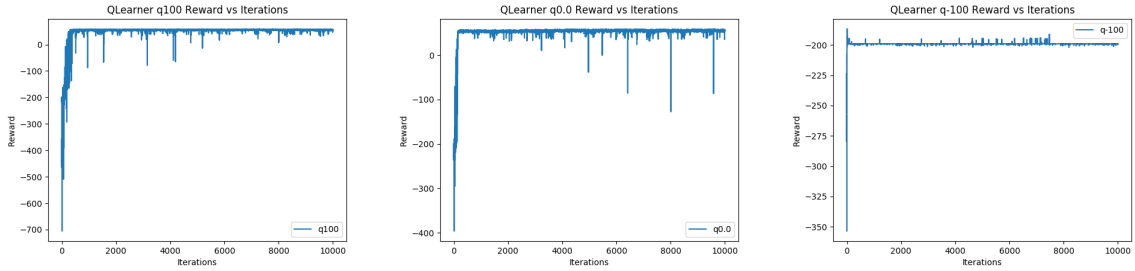




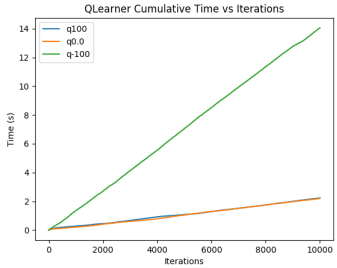
Based on the above plots , higher value of Epsilon ϵ resulted in faster convergence of rewards but resulted in increase in time complexity. Higher value of Epsilon ϵ caused the learner to explore more at the beginning of the learning phase, hence arriving at the optimal policy quickly.
When the **learning rate α was increased to 0.9**, it was observed that the learner failed to converge and was amassing negative rewards as shown below.



Upon varying the **Q Initial values between -100, 0 and 100 for learning rate α of 0.1 and Epsilon ϵ 0.5**, it can be observed that reward values converge around ˜58 for Q Initial values of 0 and 100, while Q Initial of -100 converged arounf ˜-199 rewards. However, only Q Initial of 100 seems to obtain a near optimal policy while others had incorrect states in it's optimal policies. A higher Q Initial of 100 , which is somewhat close to the actual total reward encouraged the learner to explore more at the beginning. With a high Epsilon ϵ 0.5 and low learning rate α of 0.1, causing further exploration and slow learning made the learner to reach the optimal policy for the easy grid world problem in around ˜600 iterations as mentioned in previous section.



As per the below plot, Q Initial of -100 took lot of time for completing 1000 iterations , due to minimal exploration and hence the computational complexity increased causing higher runtime.



Conclusion

Policy Iteration and Value Iteration algorithms are guaranteed to find the optimal policy of a MDP, provided the model and state transition probabilities are known beforehand. Among both, Policy Iteration takes less iterations to converge but takes more time than Value iteration for the same. When the complexity of the MDP environment increases, thought a drastic increase in iterations for convergence is not observed, the time complexity definitely increases for both, with Policy Iteration showing a bit higher increment due to the policy evaluation and policy improvement steps.

Q Learner, on the other hand, can help to find the optimal policy but is hugely dependent on the MDP model complexity. As observed above, it was able to converge at very higher iterations in comparison with Policy Iteration and Value Iteration, but took very less time than both of them. Hence, the computational complexity per iteration associated with this algorithm seems to be lesser than the other two. However, Q Learner failed to converge to an optimal policy, even after

500K iterations for the hard grid MDP, which indicates that increase in MDP model complexity drastically increase the amount of iterations needed for convergence. Have we let Q Learner run for even more no. of iterations, it should converge at a later stage.

Exploitation-exploration dilemmna is an important factor in Reinforcement Learning and by varying Epsilon ϵ, Learning Ratye α and Q Intitial values, we found that a low learning rate in conjuction with an ϵ-decay startegy and High Q Initial value closer to actual reward helped the learner to converge to the optimal policy successfully.

Easy Grid World Summary

Algorithm	Iterations for Convergence	Total time (s)	Computational Complexity
Policy Iteration	23	0.70	High
Value Iteration	60	0.271	Medium
Q Learner	600	0.214	Low

Hard Grid World Summary

Algorithm	Iterations for Convergence	Total time (s)	Computational Complexity
Policy Iteration	25	2.4	High
Value Iteration	62	1.76	Medium
Q Learner	NA	NA	Low

References

[1] Brown-UMBC Reinforcement Learning and Planning (BURLAP) java code library http://burlap.cs.brown.edu/
[2] Sutton & Barto Book: Reinforcement Learning: An Introduction