CS 7641 – Markov Decision Processes

mgupta318

Abstract

The paper analyzes the behavior of two real world MDP problems - Forest Management is chosen as the non-grid problem and Frozen Lake has been taken as the Grid problem. For understanding the behaviors of the Value iteration, Policy iteration and Q-learner (reinforcement learning algorithm), forest management is compared by running for small and large states and then Forest Management is compared to Frozen Lake problem to understand the behavior of non-grid large state problem to a grid world small state problem. The paper is structured into four sections where section1 provides a brief overview of the MDP problems, states why these problems are interesting, section2 which describes the technical approach taken for each MDP problem, section3 which describes the value iteration and policy iteration and their comparisons for small and large problems, section4 which describes the Q-learner algorithm including the exploration and exploitation strategies used. At the end there is a conclusion section, which provides a brief summary of the paper. The paper describes the comparison between – one between Forest Management (small vs. large states) and one between (forest management non-grid problem (large) vs. frozen lake grid world (small). mdptoolbox hiive package has been used to run the policy iteration, value iteration and Q-learner algorithms and matplotlib library for generating the plots.

Section1: MDP problems

1. **Forest Management** – Forest Management is non-grid world problem to manage a forest with two objectives including preserving the old forest for wildlife and getting profit from selling the cut trees. This is an interesting problem since this is a real world problem and challenge here is to maximize the objective function of rewards with preserving wildlife with constraints like the age of the trees determined based on the number of states, discount probability i.e. should the trees be cut now vs. in the future to gain maximum reward and also take into consideration the fire probability of burning the forest in which the state of the forest resets to its youngest state. There are two actions in this problem – "wait" and "cut". This is a stochastic problem since the states chosen are random. The reason which inspired me to chose this problem is it can be great for wildlife preservation and helping the nature.
2. **Frozen Lake** – Frozen Lake is an interesting problem, which presents an environment for the grid problem. This problem consists of the 4 actions to move in the grid - LEFT, DOWN, RIGHT, UP which has multiple states consisting of starting point and objective function is to reach the goal. There are various constraints like falling into the hole where the ice is melted which results in negative rewards. Once the agent reaches the goal he gets the reward of 1. The algorithms finds the optimal policy using the rewards at every state which are fixed in case of value iteration and policy iteration and

are re-calculated multiple times for Q-learner. Discount probability is also taken into consideration to evaluate whether the rewards should be taken now or in future. Based on this problem, a large number of problems can also be solved like Taxi, Mountain car in the similar way once we define the objective function, number of states, actions and the constraints. This is a stochastic problem since the states chosen are random. The reason which inspired me to chose this problem is our famous tic-tac-toe game which might help me in beating my opponent some day by finding the optimal policy.

The problems can be solved by calculating the maximum reward values for an objective function but MDP makes it interesting since it also calculates the optimal policy for maximum reward. MDP uses the concept of dynamic programming to calculate value iteration and policy iteration, which avoids recursion and makes the calculations much faster than other algorithms.

Section2: Problems approach:

Forest Management

Forest problem is first run for small states – 100 and then large states – 1000 and then policy iteration is run to get the optimal policy, value iteration is run to get the value of optimal policy and then reinforcement algorithm Q-learner is run using some initial state as input and then find optimal value. The comparison of the small vs. large states for forest management provides an insight about the convergence behavior of small and large problems for these algorithms.

The algorithm was first run for the default states age-class 0-20 years (state 1), 21-40 years (state 2), more than 40 years (state 3). The state 3 correspond to the oldest age-class. The forest management problem was then run for 100 states (small problem size) and 1000 states (large problem size) and value iteration, policy iteration and q-learner were compared to analyze the convergence behaviors. This resulted always in the same policy being returned.

Using mdptoolbox hiive, the transition probability matrix of the form P (a, s, s') and the reward matrix of the form R(s, a) are constructed by passing the number of states and rewards r1=50 (forest in oldest state and "Wait" action) and r2=25 (forest in oldest state and "Cut" action)

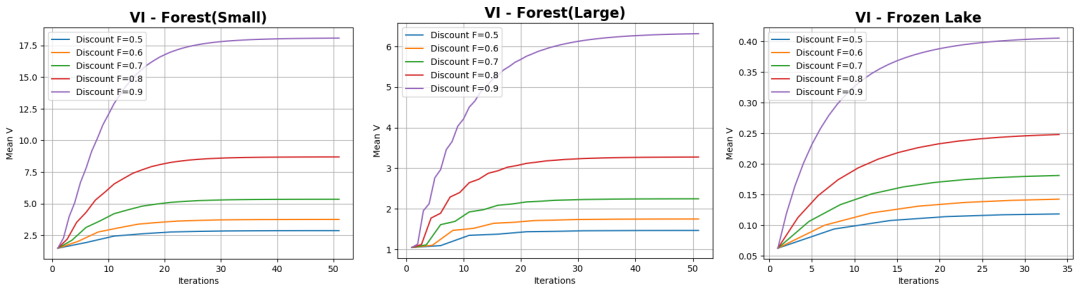
Frozen Lake

Frozen lake is a grid problem was run for a 4*4 grid problem. It has four actions – LEFT, DOWN, RIGHT, UP and consists of the states "SFFF","FHFH","FFFH","HFFG" where the states are defined as S - starting point, F - frozen surface, H - hole, G - goal. The states S and F are considered safe. The objective function of this problem is to reach the state G where the Frisbee is located starting from S. The iteration ends when either when the agent reaches G or falls into H. Reward of 1 is attained if the agent reaches the goal G, else its zero in any other state [5].

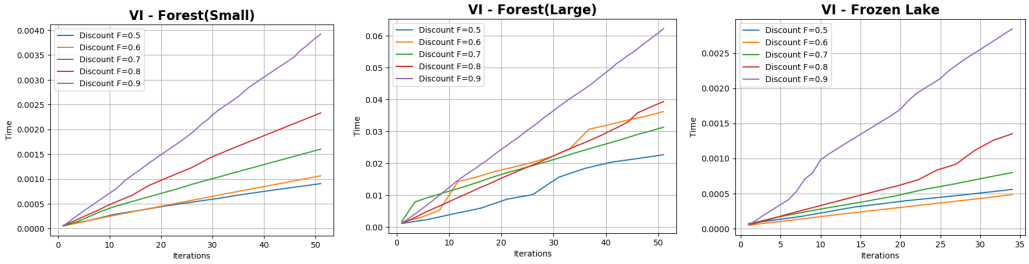
OpenAI gym package was used to create the environment for this problem using the default grid of 4*4. The environment was created in the form [transition probability, next state, reward, done], which was embedded within the dictionary of dictionary data structure values of states and actions. This is transformed to the transition probability matrix form P (a, s, s') and the reward matrix of the form R(s, a) which are then passed to the mdptoolbox hiive algorithms as input to run value iteration, policy iteration and Q-learning algorithm. The probabilities for each row in the transition probability matrix should add up to 1, which was computed correctly for the states while preparing the P matrix.

Section3: Value Iteration and Policy Iteration:

Value Iteration - Value iteration starts by selecting a random value function and then finds an optimal policy and converges to a optimal value policy by iteratively calculating the value function and using the optimal value function. Value iteration was run for the below three problems using Forest management (Small), Forest Management (Large) and Frozen lake problem and the below plots show the "Mean V vs. Iteration" and "Time vs. Iteration" plots. Value iteration was run for different values of discount factors to find the optimal policy.

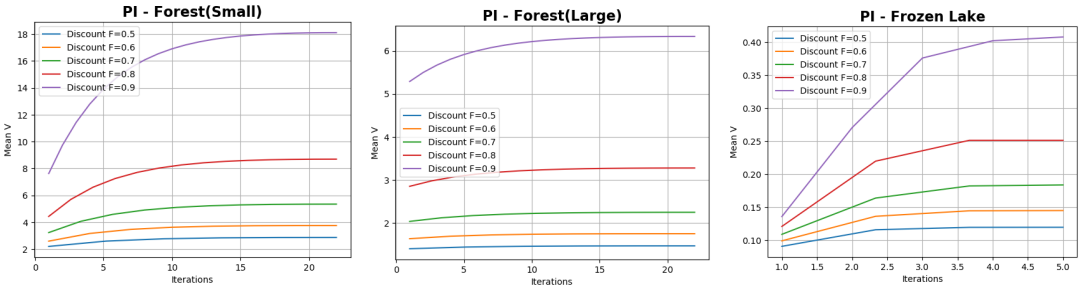


Mean Reward Value vs. Iterations

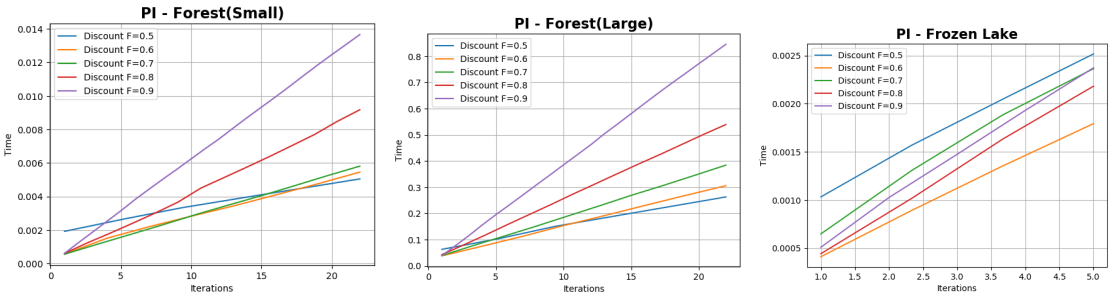


Time vs. Iterations

Policy Iteration - Policy iteration starts by selecting a random policy and then finds an optimal policy and converges to a optimal value policy by iteratively calculating the optimal policy function based on previous value function. Policy iteration was run for the below three problems using Forest management (Small), Forest Management (Large) and Frozen lake problem and the below plots show the "Mean V vs. Iteration" and "Time vs. Iteration" plots. Policy iteration was run for different values of discount factors to find the optimal policy.



Mean Reward Value vs. Iterations



Time vs. Iterations

Comparison between value iteration and policy iteration:
Convergence criteria were chosen as the point where it gives the optimal value of the policy and stops iterating the algorithm taking into consideration both the mean reward value and time taken to converge. The results of the algorithm run are as follows:

Forest Management (Small – 100 states vs. Large – 1000 states)

	Iterations			
Discount Factor	VI - Forest(Small) Mean V	PI - Forest(Small) Mean V	VI - Forest(Large) Mean V	PI - Forest(Large) Mean V
0.5	11	6	11	6
0.6	15	8	15	8
0.7	21	10	21	10
0.8	31	14	31	14
0.9	51	22	51	22

Both the small and large problems for Forest Management converged in same number of iterations in case of both Value Iteration and Policy iteration, which shows that problem size did not matter much for this non-grid world problem. The highlighted row for discount factor = 0.9 got the maximum reward which shows it basically gives more preference of taking rewards in the future. Policy iteration converged in less number of **iterations** as compared to Value iteration and had a better **reward**. Policy iteration takes more **time** than Value iteration *(Observations can also seen in the plots above for VI and PI)*. The VI and PI do not converge to the same value but converge in same number of iterations.

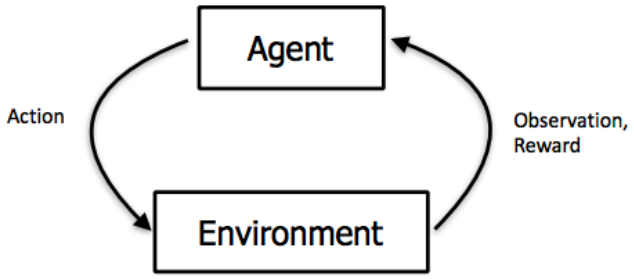
Forest Management (Large non-grid world – 1000 states) vs. Frozen Lake (Small Grid World – 16 states)

	Iterations			
Discount Factor	VI - Forest(Large) Mean V	PI - Forest(Large) Mean V	VI - Frozen Lake Mean V	PI - Frozen Lake Mean V
0.5	11	6	6	4
0.6	15	8	7	4
0.7	21	10	10	4
0.8	31	14	15	4
0.9	51	22	34	5

The large problem for Forest Management takes more iterations than small states in Frozen Lake to converge for both value iteration and policy iteration. The highlighted row for discount factor = 0.9 got the maximum reward gives the maximum reward which shows it basically gives more preference of taking rewards in the future. Policy iteration converged in less number of **iterations** as compared to Value iteration and had a better **reward**. Policy iteration takes more **time** than Value iteration *(Observations can also seen in the plots above for VI and PI)*. The VI and PI do not converge to the same value and do not converge in same number of iterations.

Section4: Q-learner:

Q-learner was used as the reinforcement-learning algorithm. This algorithm learns the model as compared to value iteration and policy iteration, which already know the model. Q-learner takes the initial state of transition probability and rewards matrix and re-calculates rewards by re-visiting every state multiple times to find the optimal policy also taking into consideration the negative rewards. Q-learner was initially run with default parameters and it was observed that it got stuck at local minima since we were directly using exploitation. Then before running the Q-learner again, parameters such as epsilon and epsilon-decay are tuned to make sure Q-learner picks up the optimum policy by choosing right balance between exploration and exploitation.

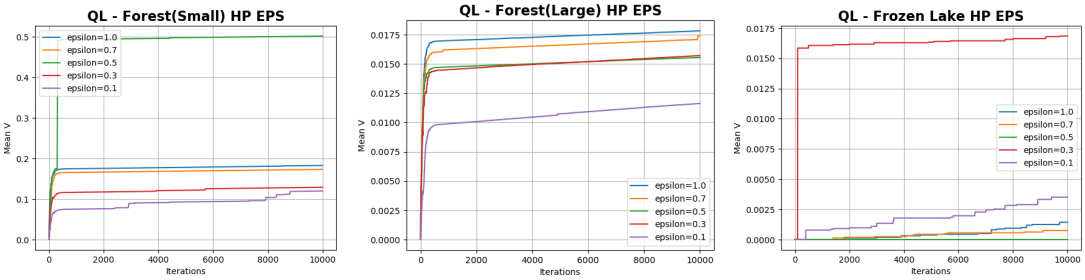


Q-learner interaction [4]

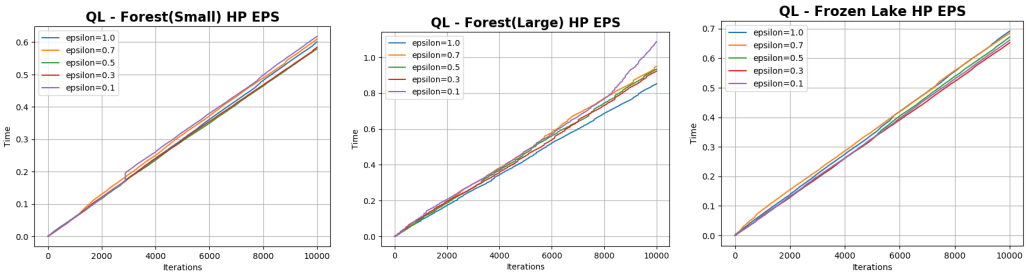
Exploration strategies:

1) Finding the optimal value of Epsilon – Epsilon provides an indication of how the model should explore before starting to exploit. Finding the right value of

epsilon helps in determining the level of exploration model should do on this problem. Learning rate parameter was also explored but did not have effect on the performance of Q-learner.

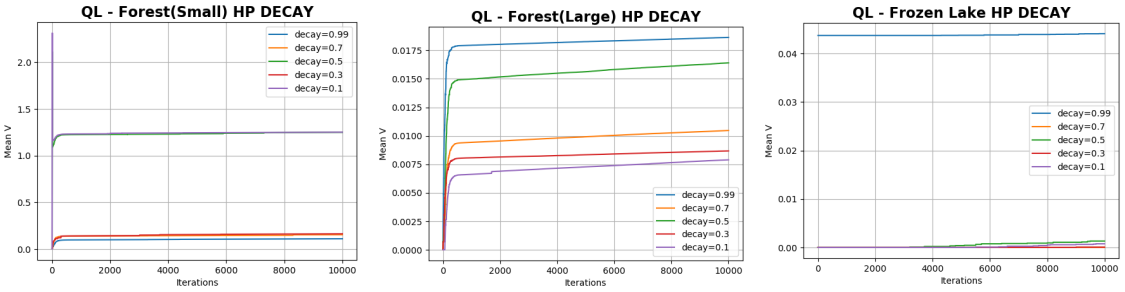


Mean Reward Value vs. Iterations

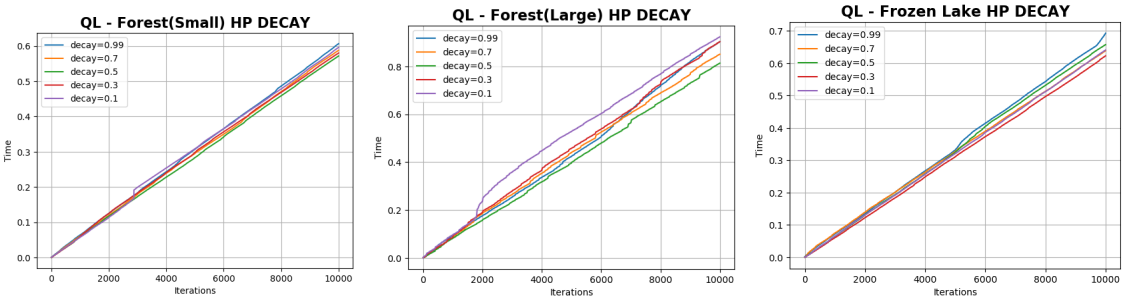


Time vs. Iterations

2) <u>Finding the optimal value of Epsilon Decay</u> – Epsilon Decay parameter was hyper tuned since initially the convergence was taking a long time. By using an optimal decay parameter value helped model to converge



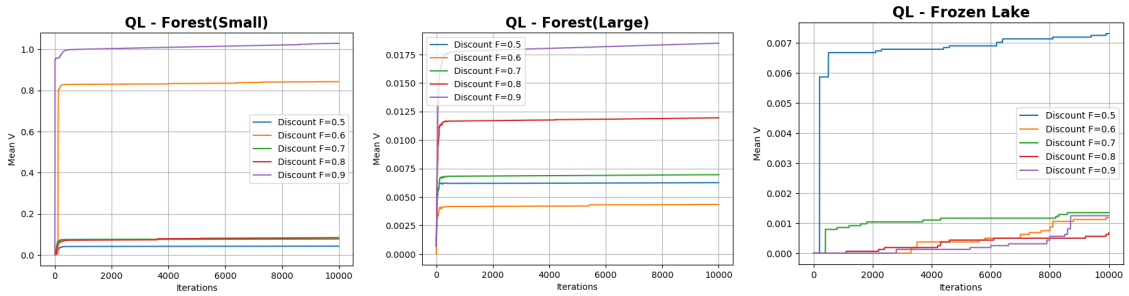
Mean Reward Value vs. Iterations



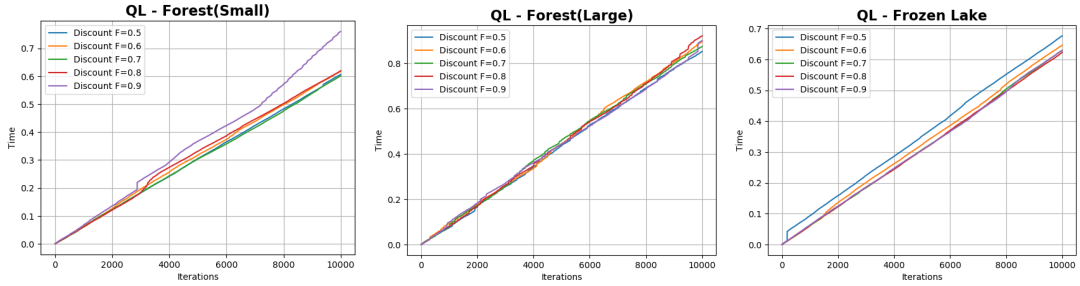
Time vs. Iterations

Q-learner using tuned exploration parameters:
Small problem Forest Management exploration – epsilon=0.5, epsilon_decay=0.5
Large problem Forest Management exploration – epsilon=1.0, epsilon_decay=0.99
Frozen Lake exploration - epsilon=0.3, epsilon_decay=0.99

Q-learner was run for different values of discount factors to find the optimal policy.



Mean Reward Value vs. Iterations



Time vs. Iterations

Q-learner for the small problems provides maximum reward and takes least amount of time to converge. Discount factor of 0.9 gives the optimal policy in case of Q-learner for Forest Management whereas for Frozen lake discount factor of 0.5 gives the optimal policy as compared to discount factor of 0.9 always provided the optimal policy for Value iteration and policy iteration.

Conclusion:

Based on the run of overall experiments, it can be concluded that value iteration took the least amount of time and policy iteration had the best reward as compared to Q-learner. This can be attributed to the fact that both Policy iteration and Value iteration already knew the model and has the transition probability matrix and reward matrix already known whereas Q-learner re-calculates the rewards and transition probability for every state multiple times before finding the optimal policy.

Citation:

[1] https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html
[2] https://stackoverflow.com/questions/37370015/what-is-the-difference-between-value-

iteration-and-policy-iteration#:~:text=In%20Value%20Iteration%20%2D%20You%20randomly,—%2D%3E%20Policy%20improvement"
[3] https://www.linkedin.com/pulse/frozenlake-dynamic-programming-rob-van-putten
[4] https://reinforcement-learning4.fun/2019/06/09/introduction-reinforcement-learning-frozen-lake-example/
[5] https://gym.openai.com/envs/FrozenLake-v0/
[6] Georgia Tech CS 7641 lecture videos and Piazza forum