Assignment 4:
Markov Decision Process (MDP)

Steven Nord
snord3@gatech.edu

1 Introduction

Throughout this report, there were two Markov Decision Process (MDP) problems explored. When choosing MDP problems, the systems needed to meet the basic criterions. First, the system needed to have a well-defined state-space, action-space, and a reward function that assigned a numerical value to each state-action pair. MDPs also needed learnable probabilities for transitioning from one state to another state based on a given action taken. Finally, the system had to abide by the Markov property, which meant that future states only depend on the current state and not on any previous states. This allowed the agents to interact with an environment in a learnable process.

Additionally, each problem was explored with two different state sizes to observe how it impacted parameters for different reinforcement learning algorithms. **Table 1** shows the number of states for each MDP. Value iteration, Policy iteration, and Q-Learning were used to find the optimal policy for each MDP. Since Q-Learning was a model-free algorithm, learning rate, discount rate, and exploritory decay rate were tuned to optimize the algorithm. It was an iterative process of varying only one hyperparameter while setting the others at their current optimal values.

Table 1

MDP	Taxi (S)	Taxi (LG)	Frozen Lake (S)	Frozen Lake (LG)
States	500	2,520	64	400

2 Markov Decision Process (MDP) Problems

The first MDP problem was Taxi. The objective of the challenge was to provide actions to the agent (the taxi) to drive to a predefined location of a potential passenger, pick them up, drive them to their destination, and drop them off. The properties for this problem satisfied all the requirements for an MDP.

The states were defined by the location of the taxi, the passenger, and the destinations on the grid. The actions included directional driving instructions (North, South, East, or West), attempting to pick up a passenger, and attempting to drop off a passenger. The environment also limited some driving instructions from being taken at certain locations with barriers throughout the grid. The

rewards for the system were 20 points for successfully dropping a passenger off at their correct destination, -10 points for attempting to pick up or drop off a passenger in the wrong location or when the passenger was not actually in the taxi, and -1 for all other actions attempted. There were no additional penalties assigned for attempting to turn into a barrier present on the grid. The taxi simply remained in the same grid space if it received instructions to drive into a barrier.

This MDP was deterministic in nature because when given an action, the next state and the reward received were consistent throughout the challenge. If the taxi was given instructions to drive West, it would attempt to drive one unit West 100% of the time. If a barrier was present, the taxi would remain in the same state and accrue a -1 for the time step wasted.

With self-driving vehicles becoming more prevalent each day, this MDP was very intriguing to work through.

The next MDP was the FrozenLake problem. The mission for the agent was to navigate across a frozen lake to the pre-set goal location. This problem met all of the criteria to be formulated into a MDP since it had a state space, action space, rewards, and transition probabilities. The state was simply the agent's location on the grid. Some grid cells were "holes" in the ice. If the agent stepped into a hole, it would fall through the ice and the mission would result in a failure. The actions were the directional instructions (North, South, East, and West). The transitional probabilities were stochastic given the slippery surface. When an agent was given directional instructions, it only took that action 33.3% of the time. The remaining 66.7% was split evenly across the two perpendicular directions. This meant if the agent was instructed to go West, it would go West 33.3% of the time, North 33.3% of time, and South 33.3% of the time.

The rewards for this MDP were altered from the original and the rationale will be discussed later in this report. The original rewards were one point for reaching the goal state and zero for all other actions. The altered rewards were one point for reaching the goal, -1 for falling into the ice, and -0.01 for all other actions.

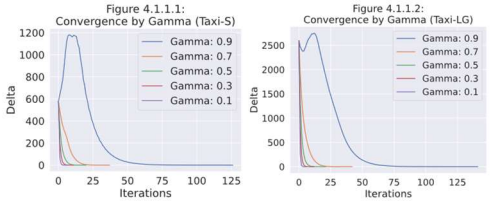
Both MDPs followed the Markov property requirement and both were episodic (i.e., there were predefined termination states and the agent was then reset). The key differences between the Taxi and FrozenLake problems were their transition probabilities (deterministic vs stochastic) and FrozenLake involved avoiding hazard that could end an episode with a loss.

3 Hypothesis

Initial hypothesizes were that policy iteration would converge faster than value iteration. Also, that value iteration and policy iteration would find more optimal policies compared to Q-Learning since the former were model-based algorithms and the latter was model-free. The prior knowledge of the transition model would be helpful for learning the optimal policy.

4 Experiments

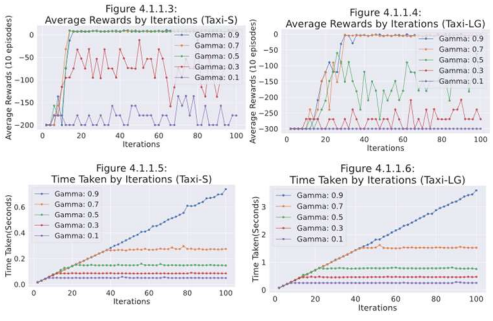
As a way to measure the improvement the algorithms were able to generate, a policy that randomly selected actions from the MDP's action space was used to establish a baseline for each MDP. Convergence for value iteration was checked after each iteration by summing the absolute value of the difference between the current and previous value matrix. For policy iteration, if the current policy remained consistent with the previous policy, then the algorithm was deemed to have converged. Q-learning had a convergence strategy similar to value iteration except it compared the Q-table rather than the value matrix.



4.1 Taxi

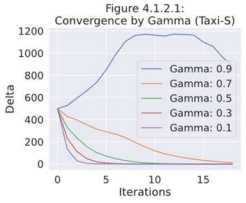
4.1.1 Value Iteration

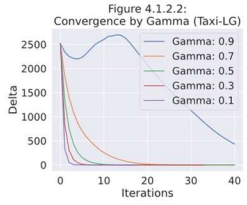
Figure 4.1.1.1 and **Figure 4.1.1.2** show the convergence of value iteration by various gammas for the small (Taxi-S) and large (Taxi-LG) state space of Taxi respectively. There were several points of interest when comparing and contrasting these plots. For both problem sizes, as gamma increased, the number of iterations to converge increased. Intuitively this aligns well with the role of gamma as the discount factor. The higher the discount factor the more emphasis the agent put on long-term rewards. With more rewards included, more iterations were needed for the rewards to propagate through the value table. It was also insightful that value tables' deltas initially diverged for several iterations when gamma was 0.9. This again was due to the initial wave of values spreading throughout the value table before settling on the optimal values and then beginning to converge. The biggest difference between these two problem sizes was the number of iterations before converges. As expected, the large problem required more iterations.



The rewards by iteration for small and large taxi sizes can be observed in **Figure 4.1.1.3** and **Figure 4.1.1.4**. The plots show that smaller gammas may converge faster; too small of a value does not allow the agent to take enough future values into account when making optimal policies. While comparing across problem sizes, gammas lower than 0.5 did not yield optimal policies for Taxi-S and gammas lower than 0.7 did not work for Taxi-LG. Naturally, Taxi-LG would require a larger gamma since it required more steps to get around the map and therefore the agent would need to gather more value from future states. Also, due to the large state space, iterations took longer for Taxi-LG compared to Taxi-S (**Figure 4.1.1.5** and **Figure 4.1.1.6**).

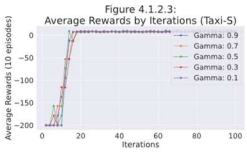
4.1.2 Policy Iteration

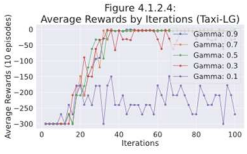


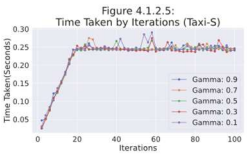


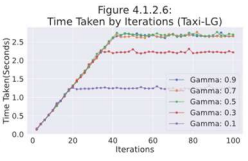
As mentioned previously, policy iteration converged when the policy remained consistent across iterations. This convergence occurred prior to the value table fully converging, which is shown in **Figure 4.1.2.1** and **Figure 4.1.2.2**. Again, Taxi-LG takes more iterations to converge due to the increase in size of the state space.

Policy iteration also converged faster because it involved another step called policy evaluation. This iterative process of policy evaulation and policy improvement helped resolve the exploration versus exploitation dilemma that plagues reinforcement learning. A lower gamma value exploited the policy by placing more emphasis on immediate rewards.





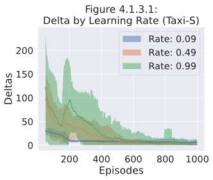


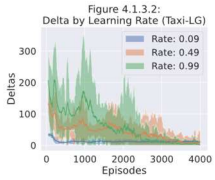


Previously with value iteration, gammas below 0.5 resulted in lower than optimal average rewards. The introduction of policy evaluation in policy iteration allowed for each of the gammas to perform well for Taxi-S (**Figure 4.1.2.3**). **Figure 4.1.2.4** shows gammas of 0.5 and 0.3 improved with policy iteration while also showing 0.1 was still too low of a value for Taxi-LG. Taxi-LG's state space was still too large to only exploit the immediate rewards without considering the long game.

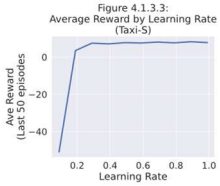
Figure 4.1.2.5 and **Figure 4.1.2.6** show the training time for policy iteration. These times were lower than the times provided in **Figure 4.1.1.5** and **Figure 4.1.1.6** for value iteration. Essentially, policy iteration found a policy and was able to bypass infinite number of value tables that would have generated the same results.

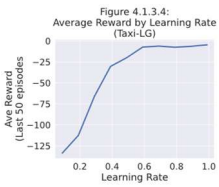
4.1.3 Q-Learning



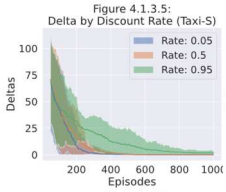


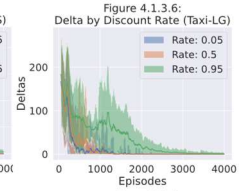
Other learning rates were evaluated in hyperparameter tuning, but **Figure 4.1.3.1** shows only a few learning rates' convergences of the Q-table. The solid-colored line indicated the rolling average over the last 50 episodes and the shaded-colored area identified the spread for two standard deviations. The plot displayed how the larger the learning rate, the wider the standard deviation bands. This was due to the Q-values taking more of the new values into consideration and therefore taking on more variance.





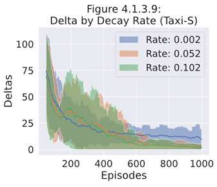
A learning rate of 0.09 resulted in the Q-values taking on a smaller fraction of the new rewards that were returned from the environment. With smaller changes to the values, this resulted in a faster convergence. **Figure 4.1.3.3** shows this quick converge resulted in a suboptimal policy. By the plot, the optimal learning rate for Taxi-S needed to be set higher than 0.3. **Figure 4.1.3.4** shows a similar pattern for Taxi-LG, but the learning rate for this problem size required a rate greater than 0.6. Too low of a rate for this size of problem does not pull enough information from each new reward. This resulted in the agent not learning enough during the initial exploritory phase of the algorithm.

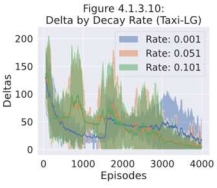


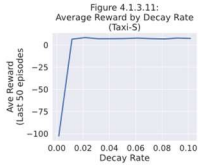


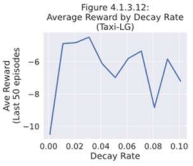
The next hyperparameter tuned was discount rate. The larger the value, the larger the spread for the standard deviation (**Figure 4.1.3.5** and **Figure 4.1.3.6**). This additional fluctuation was introduced due to more of the future rewards being taken into consideration.

As was the case before in the value iteration and policy iterations, a discount rate too small can result in suboptimal results. The agent had a short-sighted view of the rewards and missed out on future opportunities. For Taxi-S and Taxi-LG, the discount should be no less than 0.6 (**Figure 4.1.3.7** and **Figure 4.1.3.8**)



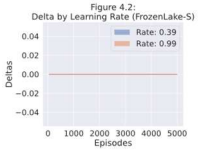






The final hyperparameter tuned was the exploritory decay rate. The exploritory rate was initially set to one and had a minimum threshold of 0.01. The role of the decay rate was to reduce the amount of exploring of the environment as the number of episodes increased. Too low and the algorithm will

take longer to converge and over explore the environment. On the other hand, too large could result in a suboptimal policy being returned as the algorithm will quickly prioritize choosing greedy actions. **Figure 4.1.3.9** and **Figure 4.1.3.10** show the convergence for various decay rates. Values below 0.01 for Taxi-S resulted in suboptimal policy based on the plot from **Figure 4.1.3.11**. **Figure 4.1.3.12** shows that too low of decay rate for Taxi-LG caused the agent to over explore before convergence. Also, from this plot it showed too large of a decay rate and the algorithm abandoned exploring the environment too early. A decay rate between 0.01 and 0.03 was optimal for Taxi-LG.

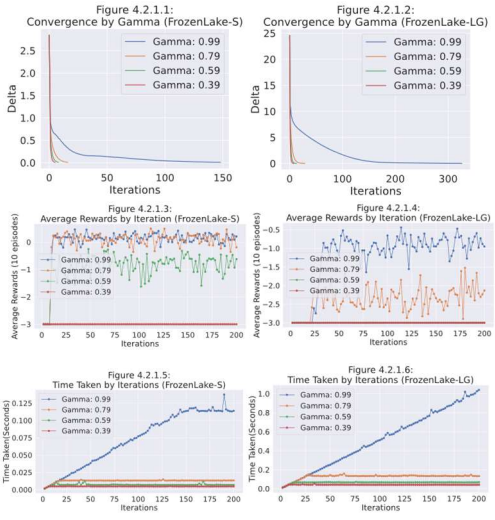


4.2 FrozenLake

As previously mentioned, the FrozenLake reward structure was altered. With access to the transition probabilities, value iteration and policy iteration were able to build policies that were effective at solving this challenge. As for Q-Learning, the rewards were too sparse to gather any meaningful insight into the environment since it was generally receiving rewards of zero for all actions (**Figure 4.2**). The transition probabilities gave model-based algorithms sufficient awareness of where the goal state was that was not available for the model-free approach.

The following results are based on the altered reward structure for FrozenLake.

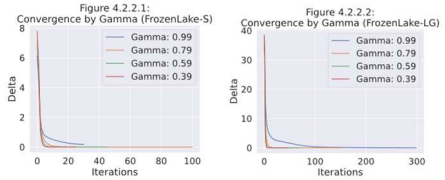
4.2.1 Value Iteration



The deltas for FrozenLake held onto some of the key principles discussed for the Taxi problem. The larger gammas took more iterations. The deltas for this problem were much smaller than for Taxi. This was due to the reward range for this problem being much smaller (-1, -0.1, and 1) as opposed to (-10, -1, and 20).

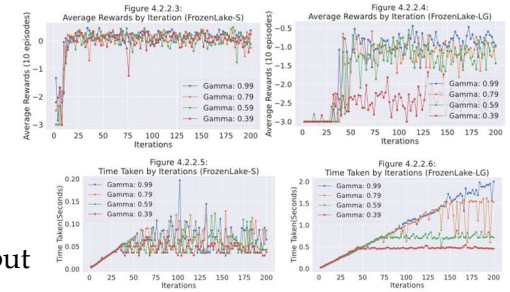
Figure 4.2.1.3 and **Figure 4.2.1.4** show rewards over the last 10 iterations for four different gammas. Gamma 0.39 did not perform well with such a short-sighted view for either environment size. For FrozenLake-S, gamma of at least 0.79 provided enough information for each state to navigate to the goal state. While only 0.99 worked to provide a strong policy for FrozenLake-LG. The average rewards from the large problem set were also lower than the small because it took more steps to reach the goal. With a grid size of 20 by 20, it took a minimum of 40 steps to reach the goal state and that would

have a very small probability of being the case given the agent was more likely to slip (66.7%) than to step in the intended direction (33.3%). Iterations through the large problem set also took much longer since it had more than six times the number of states to process (**Figure 4.2.1.5** and **Figure 4.2.1.6**).

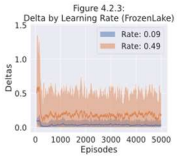


4.2.2 Policy Iteration

Figure 4.2.2.1 shows gamma of 0.99 converged before value table converged, which was similar to the case for Taxi. This was not the case for the other gammas, especially 0.79. Since the problem was stochastic, it caused the policy to oscillate and therefore take longer than value iteration.

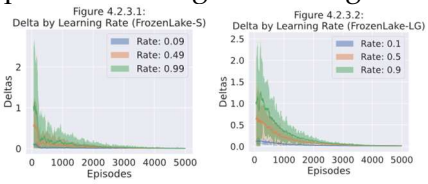


It was hard to distinguish a difference between gammas, but the largest gamma had an average reward of 0.09 higher than the other gammas. As was the case for the Taxi problem, the additional policy evaluation step helped the lower gammas perform better for policy iteration than for value iterations. For FrozenLake-LG, gamma of 0.39 did not look far enough into future states to generate optimal rewards (**Figure 4.2.2.4**).

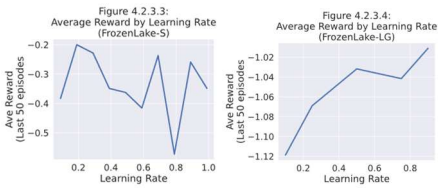


4.2.3 Q-Learning

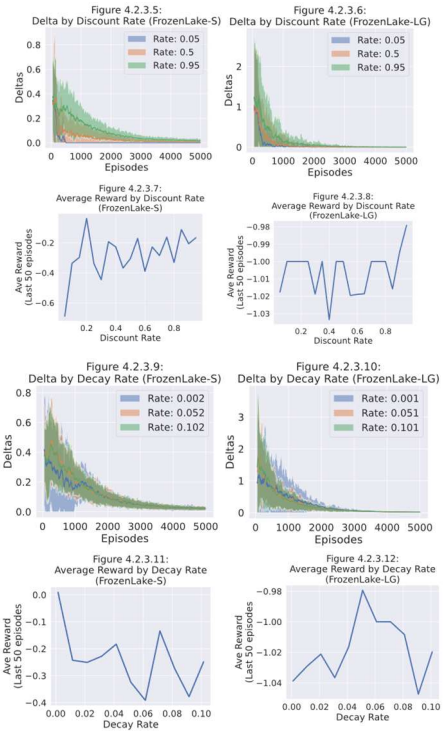
Since FrozenLake transition probability was stochastic, it made convergence more difficult. **Figure 4.2.3** shows that the Q-values oscillated because of the randomness in rewards returned from the state-action pairs even after the exploritory rate (epsilon) was reduced. A learning rate decay rate was introduced for this MDP to assist in the convergence of the algorithm. This meant that early episodes would take on more of the new rewards while later episodes would stick more to the Q-value. This strategy helped reduce the randomness from the environment by putting less weight on the new values received from the environment. The improved convergence strategies' results can be seen in **Figure 4.2.3.1** and **Figure 4.2.3.2**.



The learning decay rate remained consistent across both problem sizes at 0.001. The initial learning rate was tuned to determine the optimal policy; results can be seen in **Figure 4.2.3.3** and **Figure 4.2.3.4**. For FrozenLake-LG, the higher the initial learning rate the better the algorithm performed. The larger rate allowed the Q-values to be altered enough in earlier



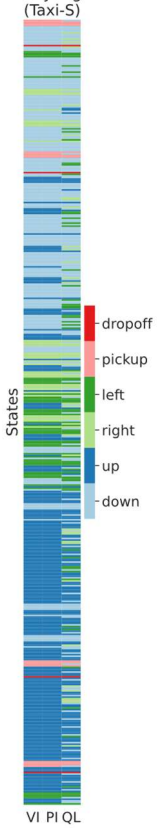
episodes by new rewards from the environment before the rate was reduced. This allowed the Q-values to be altered long enough for the agent to explore the large environment.



Aside from 0.2, the average rewards were trending upwards as the discount rate was increased for FrozeLake-S (**Figure 4.2.3.7**). As for FrozenLake-LG, most of the smaller discount rates resulted in an average of about -1 (**Figure 4.2.3.8**). This coincided with the reward for falling into a "hole". This meant that unless the discount was set to 0.99, the agent would not have a long enough horizon to see the goal. The agent would rather step in a hole and receive the negative reward rather than wandering around and accruing -0.01 rewards.

The pattern for exploritory decay rate for FrozenLake-S (**Figure 4.2.3.11**) was different than it was for Taxi-S (**Figure 4.1.3.11**). The smaller decay rate encouraged the agent to do more exploring compared to how Taxi-S encouraged the agent to start exploiting the Q-values in earlier episodes. This was a direct result of the deterministic versus stochastic nature of the two problem sets. This pattern did not hold for FrozenLake-LG and will be discussed further in the results section.

Figure 5.1:
Policies by Algorithm
(Taxi-S)



5 Results

The policies for Taxi-S can be observed in **Figure 5.1**. The policies generated from value and policy iteration were 100% match and out of 1,000 trails, all resulted in wins (**Table 2**). The policy from Q-learning was only about a 70% match to the other policies, but still resulted in a win 99% of the time. Most of the discrepancies were directional. VI policy said to go up and then left while QL policy said to go left and then up. These would not amount to the lower winning percentage. The red areas of **Figure 5.1** were the key actions (dropoff) for this MDP. It was the only action that generated a positive reward or a win. All the policies agreed on these 4 key states that the correct action was to drop the passenger off. In order to actually receive credit for dropping a passenger off, they would have to be in the taxi prior to attempting to drop them off. The policies agreed on the 16 pickup states 75% of the time.

The policy for Taxi-LG was too long to provide, but it is most important to note that the policies generated from value and policy iterations had a 99.96% match. They differed on 1 state out of 2,520 which was only a directional state. Q-learning had a 62% match to the policy generated from value iteration, 75% of the pickup states, and agreed on the drop-off states. This resulted in a success rate of 97% for Taxi-LG over 1,000 runs.

Figure 5.2:
Policies by Algorithm
(FrozenLake-S)

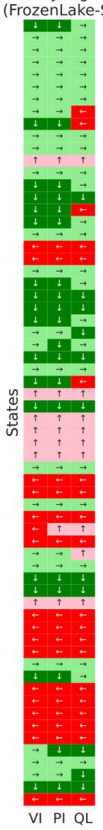
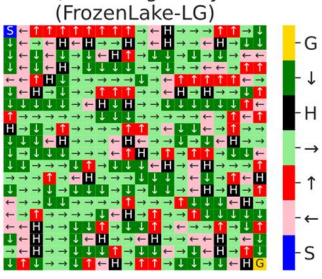


Figure 5.2 shows the policies generated by the three algorithms for FrozenLake-S. Greenish colors indicated steps taken heading in the direction toward the goal-state and reddish colors were actions that led the agent away from the goal. The policies created from value and policy iterations matched on 61 of the 64 state-actions (95.31%) and the three they differed on were just differing opinions for actions that led toward the goal (greenish). Both policies performed well given the slippery surface of the frozen environment (99.6% and 99.5%). The policy from Q-learning matched on 76.56% of the state-action pairs with value iteration. Most of the differences were differing paths that were still in the direction of the ultimate goal; however, there were 5 states that took steps away from the goal while the policy from value iteration would have taken a step toward the goal. This only caused a slight drop in the win percentage for the Q-learner (98.8% from **Table 2**).

The highlights for the FrozenLake-LG will be provided since the table was too large to provide much detail. The policies from value and policy iteration were almost a match (99.75%); they only differed on a single state out of the 400 possible. The Q-learning policy only agreed on 67%. The results for the three algorithms against FrozenLake-LG can be found in **Table 2**. Value iteration and policy iteration where slightly lower given the increased chance to slip into a hole along the way, but still managed to reach the goal-state 93.3% of the time. Q-learner did not complete the challenge at a very high success rate, only 25% of the time.

Figure 5.3:
Q-Learning Policy
(FrozenLake-LG)



To get a better understanding for the low success rate, **Figure 5.3** shows the policy generated for FrozenLake-LG. With a decent amount of green, it was challenging to see why the agent struggled to get to the goal. Losses were made up of two different categories: terminating and truncating. Terminating happened when the agent fell into a hole and truncating was when the agent took more than 300 steps. If truncated

steps were not taken into consideration, then the agent reached the goal state 54.72% of the time.

6 Conclusion

From the training time displayed in **Table 2**, policy iteration did indeed converge faster than value iteration. This supported the initial hypothesis of policy iteration converging faster than value iteration. The only exception was FrozenLake-LG. That environment's randomness and large state space caused the policy to oscillate and therefore converge slower than value iteration.

The second hypothesis was that the prior knowledge provided by the transition matrix would be an advantage for value iteration and policy iteration over Q-learning. The former both had better win percentages and shorter training times when compared to Q-learning. This shows that if a transition model is available then it makes reinforcement learning an easier task to tackle. However, the transition model is not always available and that is when Q-learning becomes extremely helpful for learning an effective policy to solve an MDP.

Table 2

MDP	Random Policy			Value Iteration			Policy Iteration			Q-Learning		
	Win %	Ave. Reward	Training Time (s)	Win %	Ave. Reward	Training Time (s)	Win %	Ave. Reward	Training Time (s)	Win %	Ave. Reward	Training Time (s)
Taxi 500	3.8	-776.9	N/A	100	7.843	0.368	100	7.843	0.248	99.0	0.443	1.478
Taxi 2520	0.6	-1190.4	N/A	100	-4.508	2.872	100	-4.335	2.221	97.2	-15.221	4.803
FrozenLake 64	0.2	-1.343	N/A	99.6	0.165	0.153	99.5	0.164	0.076	98.8	-0.004	15.703
FrozenLake 400	0.0	-1.327	N/A	93.3	-0.893	1.905	93.3	-0.893	2.830	27.4	-2.422	19.174