Markov Decision Process: A4 - CS7641

Perry Francois-Edwards (pdfe3@gatech.edu)

1 INTRODUCTION

This report will investigate the Markov Decision Process (MDP) problems and how their behavior varies based on problem type, state space and tuning. The problems speed, convergence and policies will be examined to provide information into their differences.

1.1 MDPs

The two MDP problems chosen are FrozenLake, as my large state space problem with a randomly generated 20x20 grid (400), and Blackjack, as my small state space problem. FrozenLake is a grid world problem, whereas Blackjack was a continuous problem that's been discretized by the python library, *bettermdptools,* BlackjackWrapper. Though Blackjack has 290 states, it will be the 'small' state space problem because of its short runtimes. The FrozenLake environment is interesting because of its ability to be random and has a variable state space. Experimentation will be done on both the original large state space, but also on a small, 8x8, state space to compare and confirm observations noted when comparing against Blackjack. FrozenLake also has the built-in function, 'is_slippery', which makes moving from state to state less probable, one-third the probability. FrozenLakes large state space was built with a lower hole probability in order to have feasible runtimes. I'm interested to see the policy effects this has. Blackjack's environment is interesting because it only has 2 actions in comparison to FrozenLake's 4 actions and it's based on strategic luck. Since it is not a grid world, its ability to reproduce optimal policies should be interesting.

1.2 Hypothesis

Policy iteration and value iteration are hypothesized to outperform Q-Learning because they know the transition and reward (T, S) matrix, whereas Q-Learning is model-free. Since it is a model-free algorithm, it will struggle, even with easy problems, to build a policy close to the optimal policies of VI and PI. In order of runtime, value iteration is hypothesized to be the fastest, then policy iteration and then Q-Learning will be the slowest because of its model free nature. In order of convergence, policy iteration is hypothesized to be the fastest because of its policy updates per iteration, then value iteration and then Q-Learning will converge the last. Value iteration and policy iteration are hypothesized to converge to the same values, where Q-Learning will be inconsistent. It's hypothesized that a larger state space will require more iterations than a smaller state space. Q-Learning will be able to perform better with a small state space in comparison to a large state space. Even within the same MDP, the small state space will perform better. Policy maps for a small state space will be closer to optimal.

2 METHODS

Each MDP, FrozenLake and Blackjack, will be solved using value iteration, policy iteration and a model-free reinforcement learning algorithm, Q-Learning, from *bettermdptools*. FrozenLakes large state space was a randomly generated 20x20 grid world and its small state was the predefined 8x8 state space. Blackjack's state space is predefined by the wrapper as 290 states, 29 possible player summations and 10 possible dealer hands.. All MDP environments were built with a random seed to ensure reproducibility. FrozenLakes MDP's activated the 'is_slippery' feature, changed the register from 100 episode steps to 10,000, and reduced the default probability, 'p', of holes by 16%. The FrozenLake-large and Blackjack Q-Learning algorithms were tuned to their optimal exploration-exploitation trade-off.

When comparing these MDPs, it's important to analyze the max V(s) convergence vs iterations, state space size characteristics and policy map optimality for each algorithm, VI, PI and Q-Learning. The policies will be compared using heatmaps to observe optimal states and paths. Hyperparameters and Exploration-Exploitation will also be explored to discuss their importance and effect on convergence and optima.

3 VALUE ITERATION, POLICY ITERATION AND Q-LEARNING

Bettermdptools utilized the Planner algorithms, value_iteration and policy_iteration, and Reinforcement Learning (RL) algorithm, Q_learning.

3.1 Results

Convergence is the max V(s) against iterations for the MDP and the max convergence value, gamma, set for each problem is .99. At this point, the value function, and ultimately the policy, has converged where additional iterations don't change more than the theta, thus stopping further iterations. As some figures, mostly PI, will show, when the threshold of theta is reached, the iterations stop before reaching the defined iterations. As Table 1 shows, the large state space converged at later iterations in comparison to Blackjack. An interesting note here is the comparison of FrozenLakes state sizes, because it's expected that the small FrozenLake would converge sooner than the large state, but it didn't and that is partially due to grid environment complexity. The probability of holes was reduced for FrozenLakes larger state and it was tuned with a faster learning rate, where the smaller FrozenLake has no modifications. Blackjack and FrozenLakes VI's and PI's converged to the same Max V's, which was expected because they know the T,S matrix. However, Q-Learning had different convergence values. It's interesting because without FrozenLake's small state space, I would've analyzed that a smaller state space, 290 (Blackjack) vs 400 (FrozenLake-Large), correlates to better performance which proves not true because the small FrozenLake has 64 states. It brings in the question whether the amount of actions play a larger role in convergence success.

Table 1—Convergence value and iterations: VI, PI, Q-Learning

MDP / Run Time (s)	VI:Value	VI:Iters	PI:Value	PI:iters	Q-Learning:Value	Q-Learning:Iters
FrozenLake - Large	.99	46	.99	2	~.95	50,000
FrozenLake - Small	.99	206	.99	3	~.88	30,000
Blackjack - Small	.99	1	.99	1	~.99	6154

Table 1 also gives insight into the memory complexity of Q-Learning based on the problem. 50,000 iterations is time expensive and it may not always converge.

Table 2 analyzes the state spaces using the runtime parameter. The large state space does have the longest run time, as hypothesized, versus the small Blackjack space, which is confirmed by the FrozenLake small space. Thus, VI converges to the max V with the fastest, consistent, runtime. VI and PI are deemed to converge faster than Q-Learning, model-free, because they are privy to the T,S matrix. VI performs faster because its algorithm is simpler to implement through an iterative process while PI tries to update the policy through each step making it have fewer iterations. Also note that increased exploration is time expensive.

Table 2—Run Time: VI, PI and Q-Learning.

MDP / Run Time (s)	VI	PI	Q-Learning
FrozenLake - Large	.47	1.16	57.89
FrozenLake - Small	.35	.41	24.12
Blackjack - Small	.02	.03	.61

Within Q-Learning, exploration-exploitation was done by altering epsilon-greedy and learning rates with the hyperparameters: initial epsilon, epsilon decay ratio, initial alpha (learning rate) and alpha decay. This naturally includes annealing if the decay ratio is less than one. The min_alpha and min_epsilon parameters were explored, but the outputs did not yield as much change. Altering iterations and theta did not improve performance, but rather allowed the policy to continue learning as they were increased.

As shown in Figure 1, depending on the parameter chosen, like learning rate or alpha decay, the max defined V(s) can be reached, but at what cost. Less exploration but faster convergence (exploitation), larger rates, or more exploration at a lower convergence that could cost more time and iterations. Determining the best trade off is risky, but finding the balance is essential. The Q-Learning algorithm for FrozenLake-Large chose the exploitation route and Blackjack chose the exploration route, based on learning rate since its other parameters did not show sharp contrast, as seen in Figure 2.

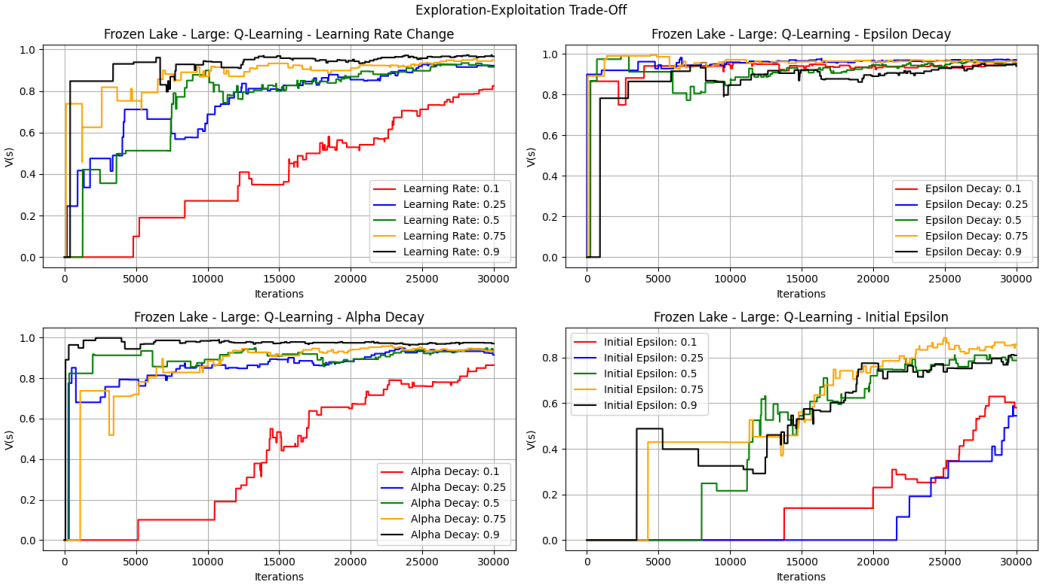

Figure 1—Q-Learning Exploration-Exploitation hyperparameter tuning on FrozenLakes - Large.

With Q-Learning, traditionally large state spaces take more time, memory and iterations to converge, than a small state space. Exploration-exploitation tuning is imperative to improve performance. Tuning has the ability to take longer amounts of time because of the larger state spaces and a smaller 'p', which could produce more holes in the FrozenLake environment.

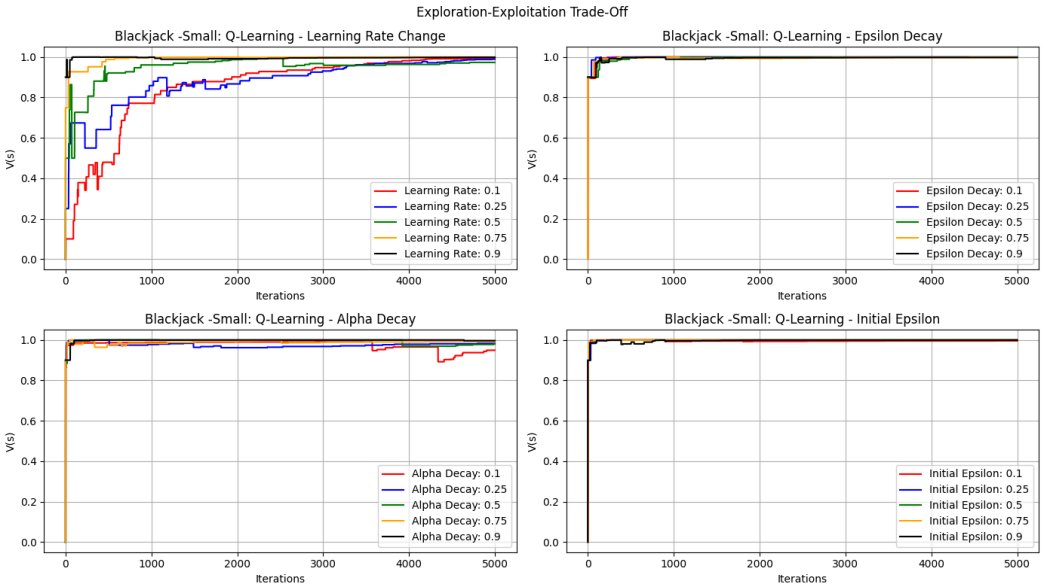

Figure 2—Q-Learning Exploration-Exploitation hyperparameter tuning on Blackjack.

3.2 FrozenLake

Below are the two convergence Figures, 3 and 4, that relate to FrozenLake. Here, it is visually apparent how the problem space complexity and tuning can improve speed, performance and variability.

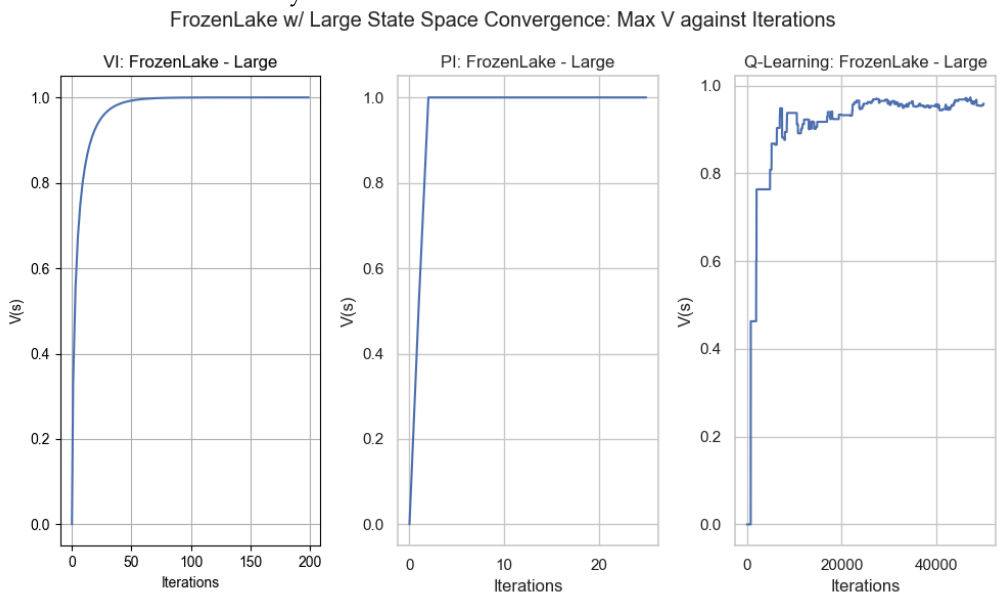

Figure 3—FrozenLake-Large convergence: VI, PI and Q-Learning.

It was expected that the small state space would perform better. However, the Q-Learning convergence in Figure 4 intuitively shows that a default baseline is not ideal for performance and can be improved, based on exploration-exploitation trade-off.

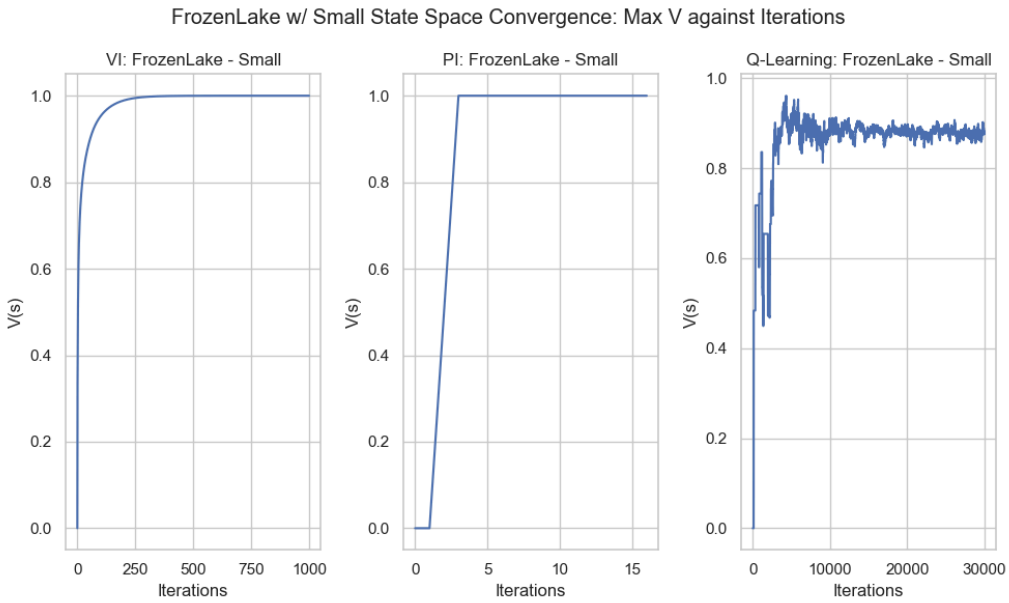

Figure 4—FrozenLake-Small convergence: VI, PI and Q-Learning.

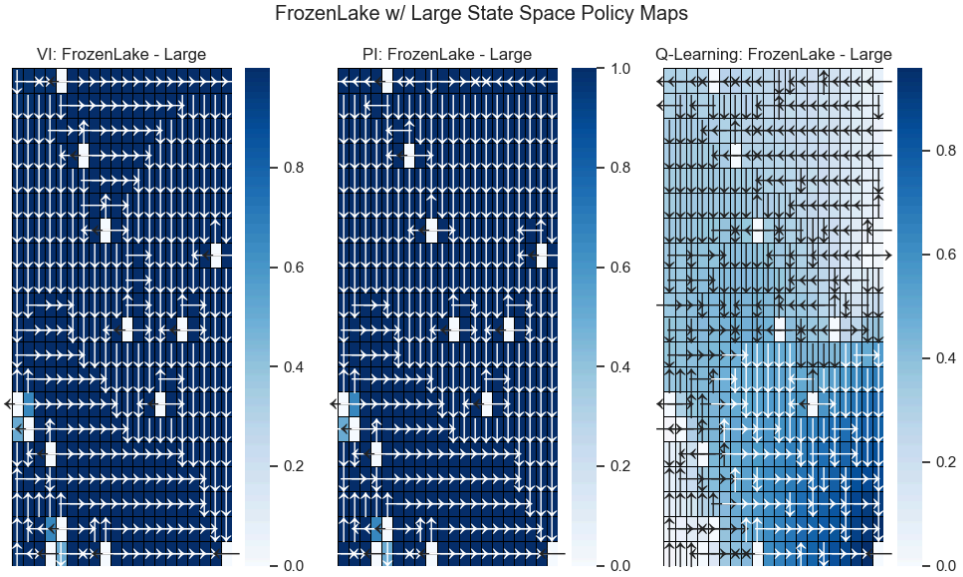

Figure 5—FrozenLake-Large policy: VI, PI and Q-Learning.

Within policy maps, the darker the state's color represents how preferred that action is. Due to a lower probability of holes in the FrozenLake-Large state space, the VI and PI policy maps were easier to reach an optimal policy with highly preferred actions at most states, as shown in Figure 5. VI and PI also produced very similar heatmaps for FrozenLake and Blackjack, as expected. Though the policy maps are similar, I expect PI's to be the most accurate because its execution evaluates and improves the policies sequentially and its goal is to converge to the optimal policy, where VI's goal is to converge to the optimal value function[1]. If PI's map is expected to be optimal, Q-Learning is striving to perform closely to these and it's shown in Figure 6 and 8 that the smaller state space problem reaches a closer optimal policy than the Larger state space shown in Figure 5, even though it did perform quite well for high exploitation.

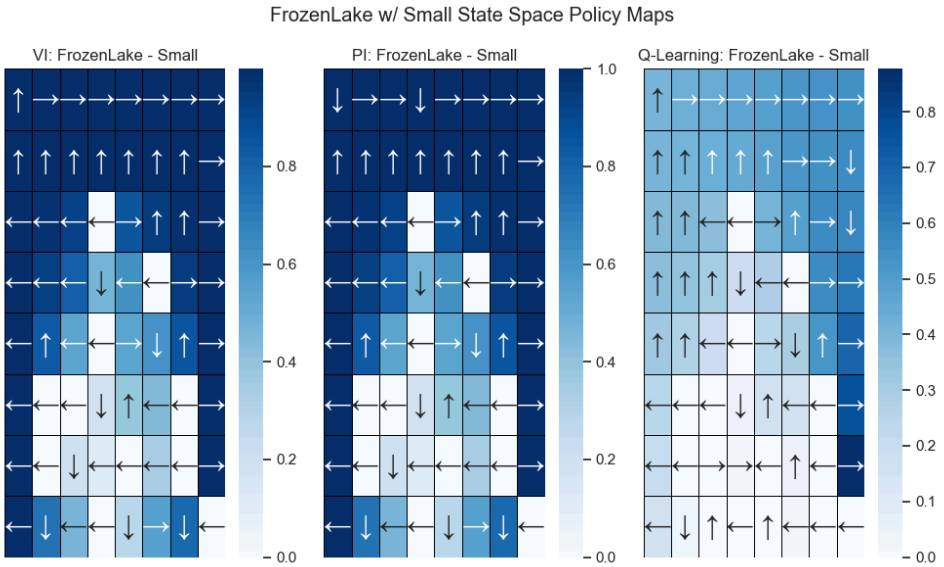

Figure 6—FrozenLake-Small policy: VI, PI and Q-Learning.

3.3 Blackjack

Blackjack as the small state space performed as expected. Its convergence reached the maximum V(s) value, run times were the fastest, and because it was small the optimal policy was able to perform well. Visually, its aligned performance to an algorithm that knows the T, S matrix meets expectation.

The amount of iterations needed to converge the small state space seemed large, but considering it took a more explorative learning rate and was still lower than FrozenLake, it proved to be the only Q-Learning convergence to reach its expected max.

The policy map for Blackjack is not as intuitive as FrozenLakes' directions to a goal, but its actions are either H, hit-where the player draws another card, or S, stand-player takes no more cards. The Q-Learning heatmap is interesting because it shows the hand summation versus the dealer's open face card.

The trade-off between learning fast (high learning rate) and learning well (low learning rate) is most apparent with Blackjack and it shows in Figure 2 and Figure 8. In order to learn a more optimal rate, a lower init_alpha is expected. However, there is also the trade-off between learning time and it could be costly. For Blackjack, it's worth it to have a lower learning rate whereas for FrozenLake its memory and time costs are too expensive, so a higher learning rate is preferred.

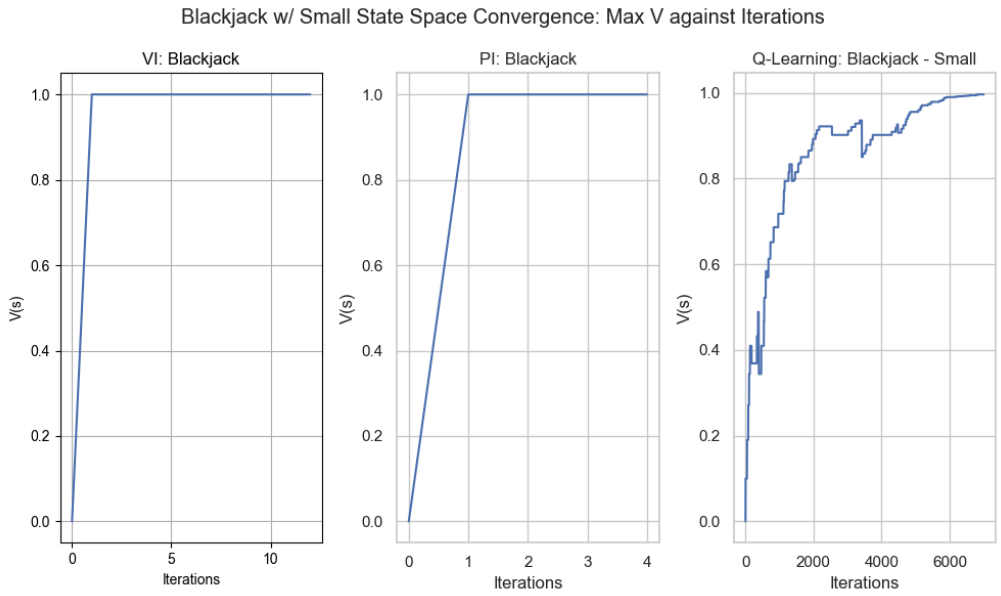

Figure 7—Blackjack-Small convergence: VI, PI and Q-Learning.

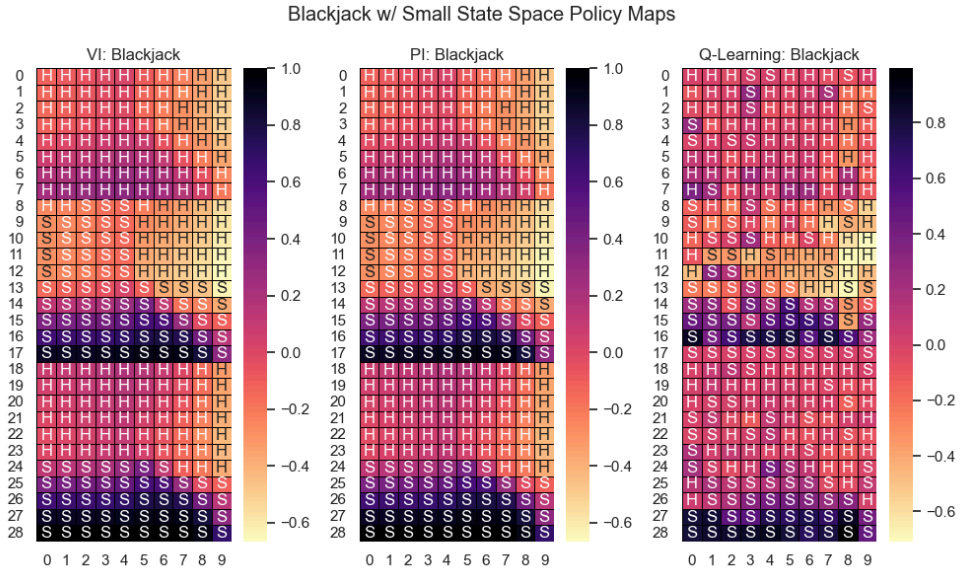

Figure 8—Blackjack-Small policy: VI, PI and Q-Learning.

3.4 Gamma and Theta Tuning

For FrozenLake, VI and PI have similar outcomes when varying Gamma and Theta. For Blackjack, varying the gamma did not affect convergence because of its high performance at low iterations. The discount factor takes into account the reward in the near future to evaluate the value function later, which is why FrozenLakes values converged sooner for low gammas [3]. The theta affects how soon the value function will stop learning based on a threshold.

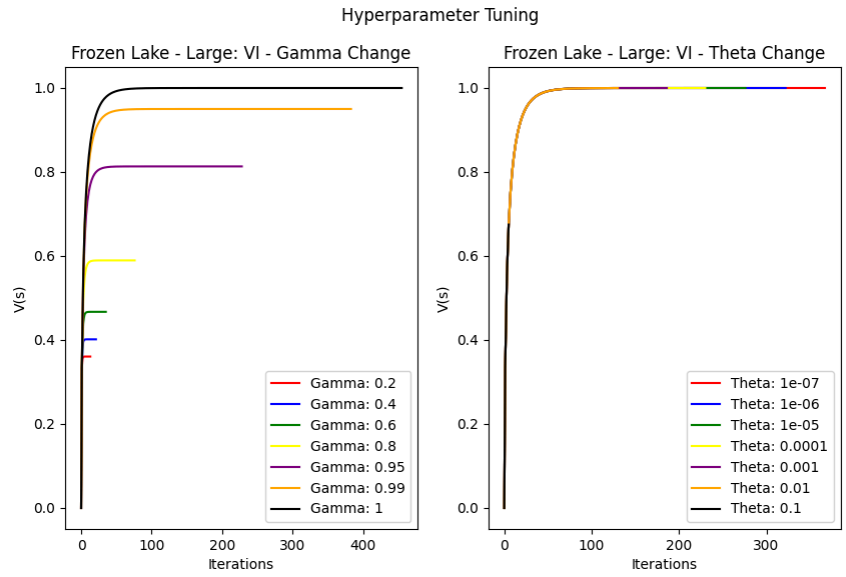

Figure 9—Gamma and Theta Hyperparameter tuning: FrozenLake.

4 CONCLUSION

Overall, this last assignment was fun, aesthetically pleasing and exciting to diagnose the algorithm's visually. Though a great exercise, it still proved challenging. As expected, PI and VI outperformed Q-Learning. In order of runtime and convergence, VI and PI won respectively, while Q-Learning was the slowest. As hypothesized, VI and PI did converge to the same values whereas Q-Learning varied depending on complexity, size and tuning. It was disproven that state size requires more iterations and that was the comparison of the 'small' Blackjack to the 'small' FrozenLake. This starts to beg the question of what should be considered state space? It was also disproven that Q-Learning will always perform better than a small state space, because FrozenLakes'-small did not outperform FrozenLake-large. When making a statement of this nature, it's important to include complexity, scalability of features (holes), runtime and exploration-exploitation trade offs. Q-Learning's ability to pick the best action possible sometimes forces the algorithm to exploit more than explore, which may lead to less desirable policies [2]. However, it was true that the smaller policy map for Blackjack performed better than FrozenLake, even though it did have a lower learning rate. To improve upon further exploration, I would have investigated other environments to view how their actions and nuances contribute to problem difficulty, created contrasting policy heat maps that show the difference between a low and high learning rate to observe improvement in policy, compared multiple model-free algorithms like sarsa, included reward shaping and investigated altering gamma and theta with Q-Learning algorithm.

5 REFERENCES

1. GeeksforGeeks. (2024b, February 9). *What is the difference between value iteration and policy iteration?*https://www.geeksforgeeks.org/what-is-the-difference-between-value-iteration-and-policy-iteration/
2. D, A. (2022, December 17). *Reinforcement learning model for blackjack*. Medium. https://medium.com/@Andrew_D./reinforcement-learning-model-for-blackjack-a29817218d53
3. Kim, M., Kim, J. S., Choi, M. S., & Park, J. H. (2022). Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty. *Sensors (Basel, Switzerland)*, *22*(19), 7266. https://doi.org/10.3390/s22197266