CS7641 Project 4 Report
Markov Decision Processes

Juejing Han, jhan446@gatech.edu

***Abstract*—**This paper investigates Blackjack and Frozen Lake problems using model-based and model-free algorithms. The diverse nature of problems results in distinct parameter settings, including exploration-exploitation trade-off strategies. In scenarios with a larger state size, the demands for more computational resources, increased episodes for convergence, additional efforts on parameter tuning, and the crucial need for a well-defined reward structure are evident. Model-based algorithms exhibit superior performance in an environment with known dynamics, characterized by faster convergence, shorter runtime, reduced parameter tuning efforts, and the guarantee of convergence to the optimal policy.

1 EXPERIMENT DESIGN

1.1 Markov Decision Processes (MDPs)

1.1.1 *Blackjack (non-grid world)*

Blackjack is a popular card game in casinos with players playing against a dealer. Its goal is to win the game by achieving a higher hand total than the dealer's without exceeding 21 – face cards count as 10, Aces count as either 1 or 11 towards the player's hand, and all other cards count at face value. At the start of each round, both player and dealer are dealt two cards, and only one of the dealer's cards is shown to the player. If the player has less than 21, he/she has two options – "hit" to receive a random card, or "stand" to stop drawing cards. If the player exceeds 21, he/she goes "bust" and loses the game. The player wins if he/she has exactly 21 or is closer to 21 than the dealer.

There are 290 discrete observable states (**which is considered a small size**) in the Blackjack problem studied in this paper: 29 player hands (H4-H21, S12-S21, BJ) and 10 dealer cards (2-9, T, A)[1]. In the reward structure, there are 4 rewards for the player, and in units relative to the bet size: -1 for a loss, 0 for a tie, 1 for a win, and 1.5 for a win with a "blackjack", which corresponds to a starting hand of an Ace and a 10-valued card. **The reward structure is considered sparse because the agent only receives rewards at the end of each game.**

Blackjack problem is interesting because finding an optimal strategy is attractive and can "maximize financial return in the long run while avoiding gambler's ruin"[2]. From a machine learning perspective, Blackjack's stochasticity and reward structure offer insights into agent behavior in a dynamic environment. The focus on the current round underscores the significance of immediate rewards and new information. **This property makes the algorithm sensitive to the choice of discount factor and learning rate.**

1.1.2 *Frozen Lake (grid world)*

Frozen Lake is a classic grid world reinforcement learning problem, where the goal is to teach an agent to navigate in a frozen lake environment and reach the goal without failing through the holes. The agent can take four actions by moving up, down, left, or right. Two map sizes are examined: 4 by 4 (16 states, which is considered a small size) and 16 by 16 (256 states, **which is considered a large size**). There are 3 types of rewards: 1 for the goal, -1 for a hole, and -0.04 otherwise. **This reward structure may become sparse for a large size and pose convergence issues.**

Frozen Lake problem is interesting because it provides a standardized benchmark for comparing and testing different approaches. Moreover, its environment can be deterministic or stochastic – setting slippery ice leads to unintended directions. It also allows for easy scalability to larger grid worlds, providing a basis for testing reinforcement learning algorithms that generalize to more complex scenarios. Its emphasis on long-term outcomes

[1] https://github.com/rhalbersma/gym-blackjack-v1/blob/master/src/gym_blackjack_v1/envs/blackjack_v1.py
[2] https://arxiv.org/pdf/2308.07329.pdf

highlights the importance of both immediate and long-term rewards, as well as current and new information. **This property makes the algorithm sensitive to the choice of discount factor, learning rate, and exploration strategy.**

Additionally, Frozen Lake prioritizes long-term rewards as the agent aims to maximize cumulative rewards over the entire process until reaching the goal. In contrast, Blackjack emphasizes short-term rewards as the player seeks to maximize immediate rewards in each round. **Combining the two problems provides a distinctive contrast.**

1.2 Algorithms

1.2.1 *Value Iteration (model-based)*

Value iteration (VI) is a model-based dynamic programming algorithm, which finds the optimal value function by solving the Bellman equations. VI uses the transition model of the environment to iteratively improve a value function until it converges to the optimal value function[3]. **VI convergence criteria**: the change in the value table is below a certain threshold, i.e., VI converges when the value table stabilizes.

1.2.2 *Policy Iteration (model-based)*

Policy iteration (PI) is a model-based dynamic programming algorithm, which uses the transition model of the environment and alternates between policy evaluation and policy improvement. PI starts with an initial policy, iteratively evaluates the value function by solving the Bellman equations, then improves and updates the policy until meets the **convergence criteria:** the policy no longer changes between iterations. Both VI and PI are guaranteed to converge to an optimal policy in the end[4].

1.2.3 *Q-learning (model-free)*

Q-learning (QL) is a model-free reinforcement learning algorithm because, unlike VI and PI, it does not require a model of the environment. QL computes the expected rewards (known as the Q-value) for an action taken in a given state, and the Q-value is updated iteratively using the Bellman equation until it meets the **convergence criteria**: the Q-values stabilize between iterations. Given unlimited exploration time and a partially random policy, QL can find an optimal policy for any finite Markov Decision Process[5].

2 RESULTS & ANALYSIS

2.1 Blackjack

2.1.1 *Model-based Algorithms (Value Iteration & Policy Iteration)*

Discount factor (gamma) determines how much weight the agent assigns to immediate versus future rewards. When gamma=1, the agent values future rewards equally, while a lower value prioritizes short-term rewards. In Fig.1, lower gammas yield higher state values because of the limited horizon of the Blackjack problem, where each round is relatively short and independent. The player's objective is to maximize rewards within each round.

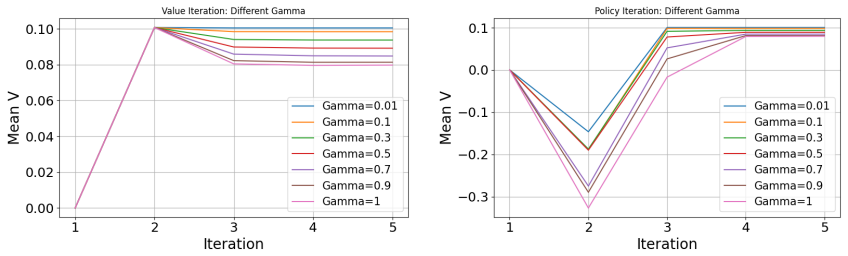


Figure 1—VI (left) and PI (right) with Different Gammas

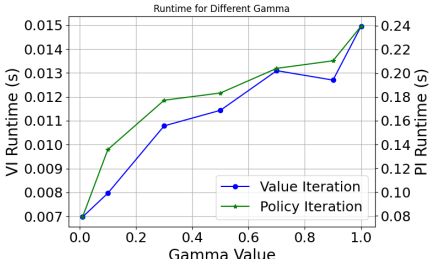


Figure 2— Wall Clock Time for VI/PI

[3] https://gibberblot.github.io/rl-notes/single-agent/value-iteration.html
[4] https://www.educative.io/answers/value-iteration-vs-policy-iteration
[5] https://en.wikipedia.org/wiki/Q-learning

Meanwhile, Fig.2 illustrates that a larger gamma leads to longer runtime due to 1) the algorithm needs to consider a larger search horizon as the agent must account for more possibilities in the future and 2) potential delayed convergence with sparse rewards. Varying theta (convergence threshold) from 10^{-4} to 10^{-8} yields the same policy for both VI and PI, suggesting the current choice of theta (10^{-4}) is optimal. A smaller theta in VI results in more iterations and a longer runtime due to its convergence criteria, while PI maintains the same number of iterations because PI converges when the policy stabilizes even if the value function continues to change.

Stabilized value function and policy are the convergence criteria for VI and PI, respectively. Cumulative rewards, delta (change in value function or policy between consecutive iterations), and state value (expected cumulative reward from a given state under a specific policy) are metrics to evaluate convergence in both VI and PI.

With gamma = 0.2 and theta = 10^{-4} (value chosen based on the analysis of Fig.1, 2, and grid search), VI converges in 10 iterations (Fig. 3), while PI converges in 2 iterations (Fig. 4). **PI converges faster than VI because** 1) PI performs a full policy evaluation to update the value function comprehensively; 2) PI incorporates policy improvement after policy evaluation and selects the greedy policy with respect to the current value function; and 3) PI converges when the policy stabilizes even if the value function continues to change.

VI and PI converge to the same policy (Fig. 5) **because** 1) both algorithms rely on the Bellman equations, which characterize the relationship between the value function and the optimal policy; and 2) despite the difference in convergence criteria and update schemes, both algorithms target the same optimal policy derived from the shared underlying mathematical problem (MDP model).

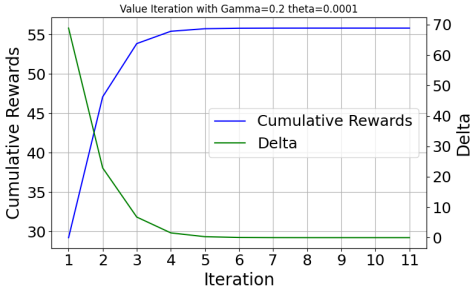


Figure 3—Value Iteration Convergence

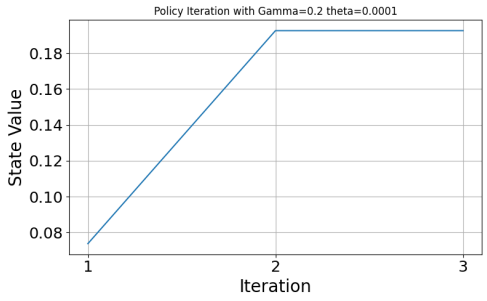


Figure 4— Policy Iteration Convergence

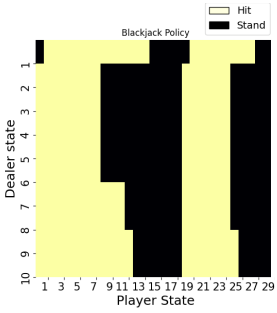


Figure 5— VI/PI Policy Map

In terms of wall clock time, VI converges in 0.01 s while PI takes 0.17 s to reach the same optimal policy. **PI requires longer runtime because** 1) PI performs full policy evaluation and iterative policy improvement, and 2) VI updates the state value more selectively based on the current action and the maximum expected cumulative reward.

During 1000 rounds of tests, the agent achieves 431 wins, 485 losses, and 84 draws. Employing 100 random seeds to test the policy to address the variance between different runs, the average win rate is 43.5%.

2.1.2 *Model-free Algorithm (Q-learning)*

Learning rate (LR) controls the influence of new information on updating Q-values (expected cumulative rewards), **striking a balance between new experiences and existing knowledge**. A higher LR enables the agent to rapidly adapt to new information. Parameter tuning suggests a preference for a high LR, resulting in large max Q values and fast convergence. Conversely, a low LR corresponds to small max Q values and delayed convergence (Fig. 6). A decaying schedule allows LR to shift from initial exploration to gradual exploitation.

Epsilon governs the **exploration-exploitation trade-off**. A high value encourages more exploration to discover potential rewards, while a low value emphasizes exploitation to maximize immediate rewards. A decaying epsilon allows the agent to prioritize exploration in the early stages and exploitation as it gains more knowledge. The differences among epsilon choices are small but a moderate value (0.3-0.5 with a decaying rate of 0.9) is recommended by parameter tuning (Fig. 7). Setting a high (0.9) or low (0.1) epsilon value yields suboptimal results

(average win rate < 42%), and epsilon=0.5 yields the optimal results (43.5%). This implies that a moderate exploration strategy at the initial stage and gradually shifts towards exploitation works well in Blackjack.

As discussed in section 2.1.1, a small gamma is selected based on domain knowledge of Blackjack, and it is also supported by experiments with parameter tuning (Fig. 8).

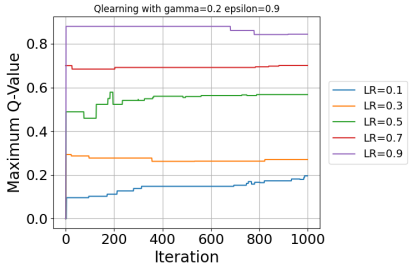


Figure 6—Q-learning with Learning Rates

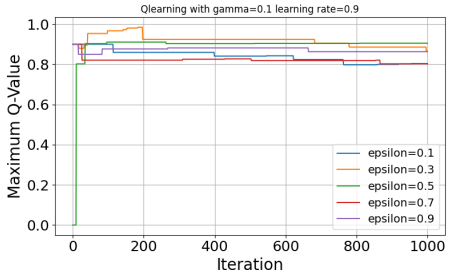


Figure 7— Q-learning with Epsilons

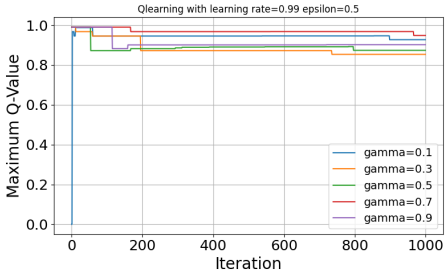


Figure 8— Q-learning with Gammas

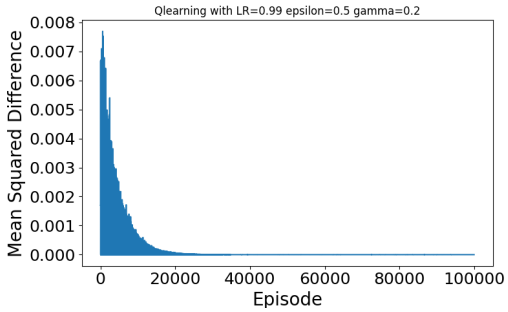


Figure 9— QL Convergence with Lr=0.99, Ep=0.5, Ga=0.1

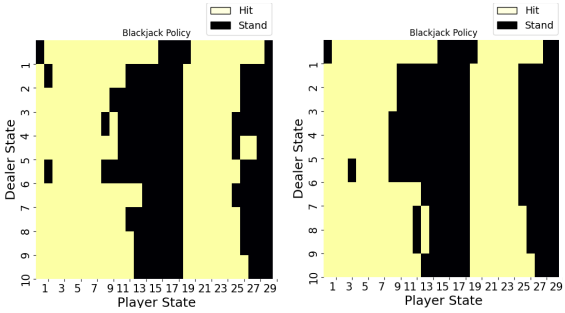


Figure 10—QL Policy Map (left: 10^5 episodes, right: 10^6 episodes)

With LR=0.99 (decaying ratio of 0.4), epsilon=0.5 (decaying ratio of 0.9), and gamma=0.1 (selected based on the analysis of Fig. 6-8 and grid search), Q-learning (QL) converges around 20,000 episodes in a run of 10^5 episodes as the change of the Q table stabilizes (Fig. 9).

Testing the QL policy (Fig. 10 left) over 1000 rounds yields 428 wins, 490 losses, and 82 draws. Across 100 random seeds, the average win rate is 43%. The difference between QL policy (Fig. 10 left) and VI/PI policy (Fig. 5) is obvious. QL is designed to converge to an optimal policy as the number of episodes approaches infinity under certain conditions. Extending to 10^6 episodes, QL achieves an average win rate of 43.5%, aligning with VI/PI results. Fig. 10 illustrates the updated optimal policy with increased training episodes, converging to VI/PI policy after 10^6 episodes (Fig. 10 right, compared to Fig. 5) **with a runtime of 70 s**.

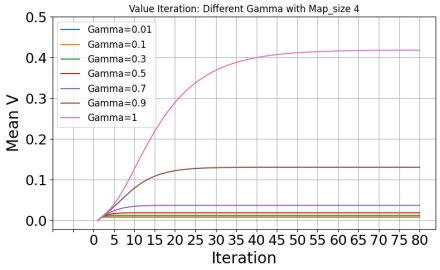
Given that VI and PI know the model, they converge faster than QL, which requires environmental interaction for learning. VI/PI can directly use the Bellman equation for value update. In the case of Blackjack, VI/PI takes 0.01/0.17 seconds to converge to an optimal policy, while QL demands 70 seconds to converge to a similar policy. Meanwhile, VI and PI require less tunning effort as they have fewer parameters than QL. Blackjack problem is sensitive to the choice of discount factor (Fig. 1) and learning rate (Fig. 6), and the reason is mentioned in section 1.1.1. Based on their respective performances, **VI and PI demonstrate superiority over QL in solving the Blackjack problem**.

2.2 Frozen Lake with Stochasticity (is_slippery = True)

2.2.1 *4 × 4 Map Size with Model-based Algorithms (VI &PI)*

With different gammas ranging from 0.01 to 0.7, VI and PI yield similar state values (Fig. 11). It is because the simplicity (only 16 states) of the 4 × 4 Frozen Lake makes it less sensitive to variations in the discount factor. Gamma=1 achieves the highest state values. This is different from what we observe in the Blackjack problem (lower gamma corresponds to higher state values). It is because of the different nature of these problems. In the Frozen Lake environment, the agent's objective is to reach the goal state, which relies on both immediate and future

rewards. Changing theta (convergence threshold) from 10^{-4} to 10^{-8} yields the same policy for both VI and PI, suggesting the current choice of theta (10^{-4}) is optimal.



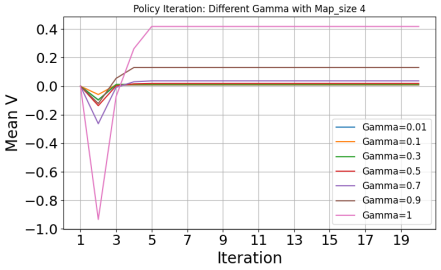


Figure 11—VI (left) and PI (right) with Different Gammas

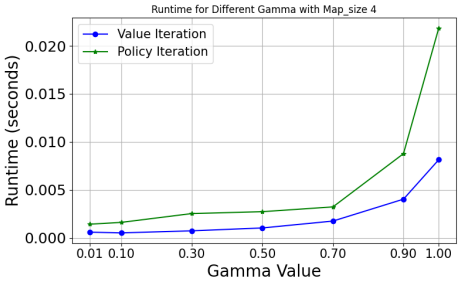


Figure 12— Wall Clock Time for VI/PI

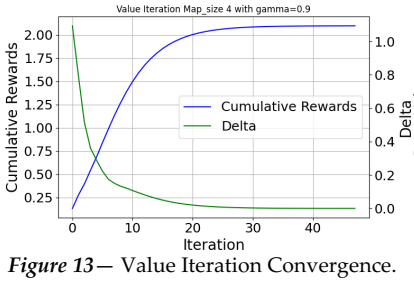


Figure 13— Value Iteration Convergence.

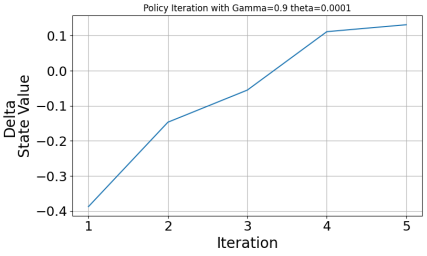


Figure 14— Policy Iteration Convergence

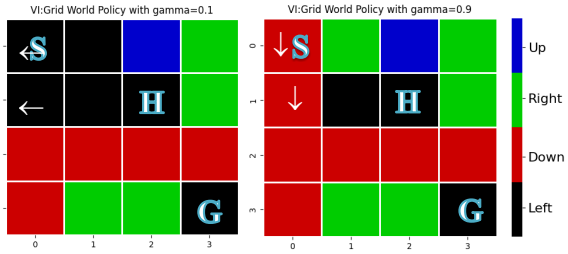


Figure 15— VI Policy (left: gamma=0.1, right: gamma=0.9)

Tuning gamma from 0.1 (resulting in 4 iterations to converge) to 0.9 (47 iterations to converge, as shown in Fig. 13 with a runtime of 0.004 s) in value iteration leads to a policy improvement (Fig. 15). Actions for the start state (S) and the state below change from left (inconsistent with their leftmost position) to down, guiding towards the goal state. Policy iteration (gamma=0.9) shares the same optimal policy as shown in Fig. 15 (right) and converges within 4 iterations (Fig. 14) with a runtime of 0.012 s. Testing the policy with 100 random seeds for 1000 rounds yields an average success rate of 75.4%.

VI and PI yield identical policies. **PI converges faster than VI but requires higher computational costs.** These results parallel those observed in the Blackjack problem, and the underlying reasons are detailed in section 2.1.1.

Compared to Blackjack (290 states), the 4 × 4 Frozen Lake has significantly fewer states. While the nature of the two problems differs, it still offers an opportunity to compare state sizes in simple scenarios. **A larger state size results in higher computational costs** – VI and PI on Blackjack (0.01 – 0.17 s) demands longer runtime than on 4 × 4 Frozen Lake (0.004 - 0.012 s) because algorithms iterate over all states. Besides, it also requires additional memory to store information for new states, such as state values.

2.2.2 *4 × 4 Map Size with Model-free Algorithm (Q-learning)*

Fig. 16-18 demonstrate the sensitivity of Frozen Lake to parameter tuning in Q-learning (QL). It stems from the agent's dependence on immediate and long-term rewards (associated with the discount factor), current and new information (connected to the learning rate), and exploration of unknown areas (regulated by epsilon). Given the small state size (16, limited states to explore), epsilon sensitivity is relatively lower compared to other parameters.

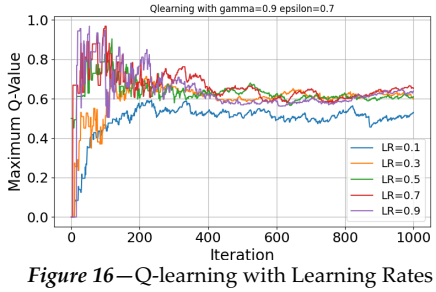


Figure 16—Q-learning with Learning Rates

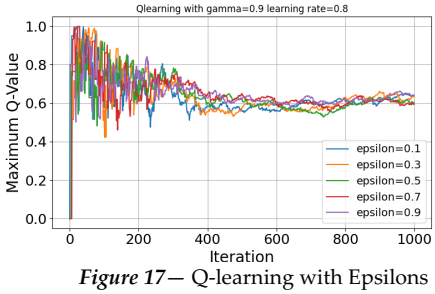


Figure 17— Q-learning with Epsilons

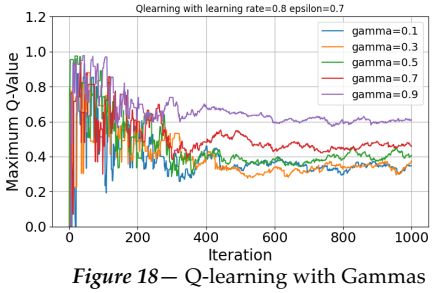


Figure 18— Q-learning with Gammas

Fig. 16-18 and tuning experiments recommend large values for learning rate (LR, 0.8), epsilon (0.7), and gamma (0.9). The emphasis on exploration and new information at the initial stage, along with nearly equal importance on immediate and long-term rewards, aligns with the nature of the Frozen Lake problem.

Regarding the small state size of this Frozen Lake problem, a relatively high LR (0.8) enables the agent to rapidly adapt to new information, and a decaying schedule of 0.5 allows LR to transition from initial exploration to gradual exploitation. Besides, a high epsilon (0.7) encourages the agent to explore at the initial stage. As the epsilon decays, it also allows the agent to shift to exploitation. Setting a higher (0.9) or lower (0.1) learning rate or epsilon value yields suboptimal results (average success rate < 72%), while LR=0.8 with epsilon=0.7 yields the optimal outcome (average success rate > 75%), implying the algorithms achieve the **balanced exploration-exploitation trade-offs**.

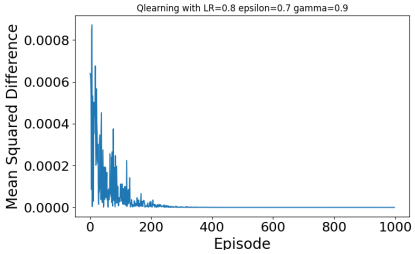


Figure 19— QL Convergence

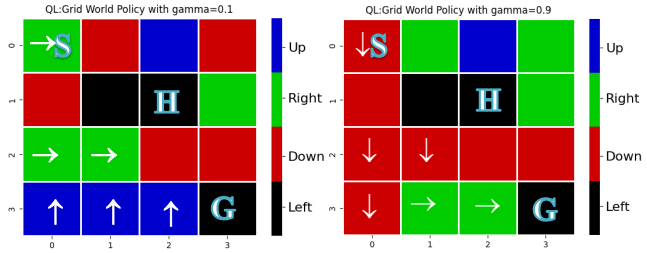


Figure 20— QL Policy Map (left: gamma=0.1, right: gamma=0.9)

With LR=0.8 (decaying ratio of 0.5), epsilon=0.7 (decaying ratio of 0.8), and gamma=0.9 (selected based on the analysis of Fig. 16-18 and grid search), QL converges around 200 episodes in a run of 10^3 episodes (Fig. 19) with a runtime of 0.25 s. Fig. 20 (left) is the policy with gamma=0.1, when we increase gamma to 0.9, QL converges to the same policy (Fig. 20 right) found by VI/PI (Fig. 15 right). Therefore, testing the policy with 100 random seeds yields the same results: the average success rate is 75.4%. A larger gamma improves the agent's behavior by taking actions towards the goal state (white arrows in Fig. 20).

In the case of 4 × 4 Frozen Lake, VI/PI takes 0.004/0.012 s to converge to an optimal policy, while QL needs 0.25 s to converge to a similar policy. Besides, VI and PI require less tunning effort. Therefore, **VI and PI demonstrate superiority over QL in solving the 4 × 4 Frozen Lake**.

Furthermore, **a larger state size results in higher computational costs** – QL on Blackjack (70 s) demands longer runtime than on 4 × 4 Frozen Lake (0.25 s) to converge to the optimal policy.

2.2.3 *16 × 16 Map Size with Model-based Algorithms (VI &PI)*

With gammas ranging from 0.01 to 1, VI and PI exhibit similar sensitivity to large gamma values (Fig. 21). This behavior can be attributed to the increased reliance of the larger Frozen Lake on future rewards. Additional states add complexity, and its reward structure (mentioned in section 1.1.2) is sparse in this large state size.

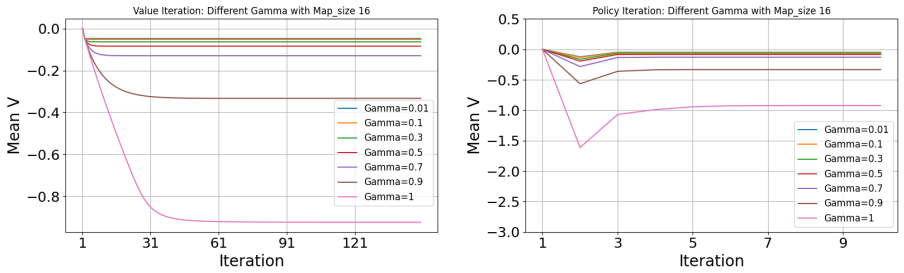


Figure 21—VI (left) and PI (right) with Different Gammas

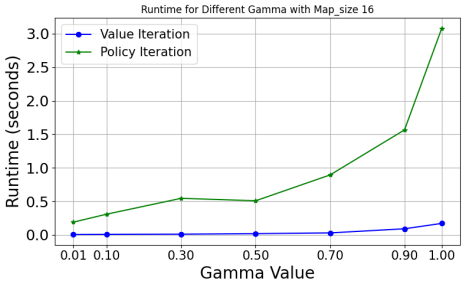


Figure 22— Wall Clock Time for VI/PI

Based on the domain knowledge of the Frozen Lake, a higher gamma should yield a better result. With a high gamma (0.9), VI and PI converge within 92 and 5 iterations, respectively (Fig. 23 and 24). However, decreasing

rewards implies that the agent is not effectively learning to reach the goal (Fig. 23). Both algorithms converge to the same policy shown in Fig. 25, with VI taking 0.09 s and PI taking 3.1 s. In comparison, the 4×4 Frozen Lake (where VI takes 0.004 s and PI takes 0.012 s to reach the optimal policy) highlights **the increased computational cost associated with a large state size**.

Similar outcomes are observed in the 16×16 Frozen Lake: **VI and PI produce identical policies** (Fig. 25); **PI converges faster than VI** (Fig. 23, 24) **but demands higher computational costs** (Fig. 22). The explanations are detailed in section 2.1.1.

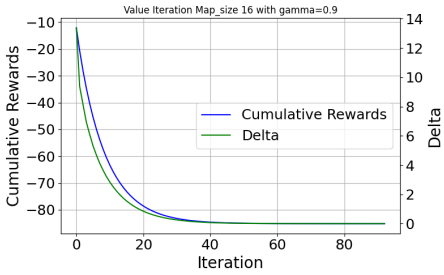


Figure 23— Value Iteration Convergence

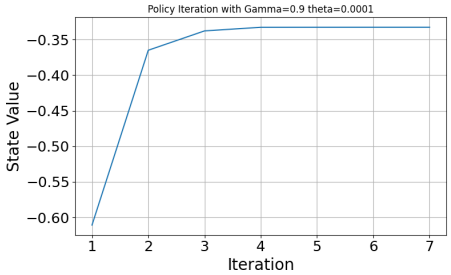


Figure 24— Policy Iteration Convergence

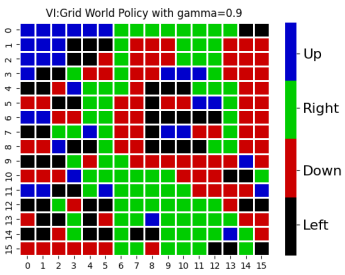


Figure 25— VI Policy Map

The negative state value arises from the sparse reward structure, indicating that the agent consistently fails to reach the goal. This suggests that the algorithms converge to a suboptimal policy. It is further proved by 1000 rounds of tests with 100 random seeds: the agent never reaches the goal state.

The complexity of the large size environment and the sparse rewards pose challenges for the algorithms to explore efficiently to converge to globally optimal policies. To enhance algorithm performance, reward reshaping (such as adjusting reward values based on their distance to the state goal) and comprehensive parameter tuning experiments are recommended.

2.2.4 *16 × 16 Map Size with Model-free Algorithms (Q-learning)*

Fig. 26-28 demonstrate the sensitivity of Frozen Lake to parameter tuning in Q-learning (QL). Tuning experiments recommend large values for learning rate (LR=0.9), epsilon (0.7), and gamma (0.9). To solve the 16 × 16 Frozen Lake, the agent needs to emphasize long-term rewards, new information, and a thorough exploration of unknown areas. The emphasis on exploration and new information at the initial stage, along with nearly equal importance on immediate and long-term rewards, aligns with the nature of the Frozen Lake problem.

Regarding the large state size, a high epsilon (0.7) encourages the agent to explore at the initial stage. As the epsilon decays slowly (decaying ratio=0.8), it allows the agent to gradually shift to exploitation. This **exploration-exploitation trade-off strategy** matches the domain intuition and is recommended by parameter tuning (Fig. 27).

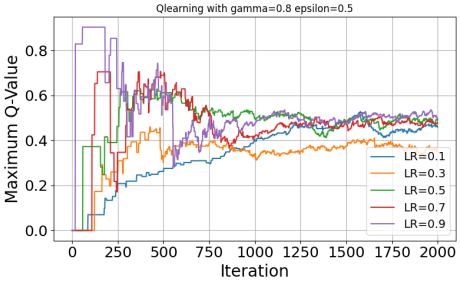


Figure 26—Q-learning with Learning Rates

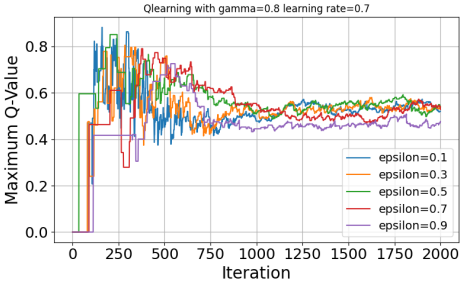


Figure 27— Q-learning with Epsilons

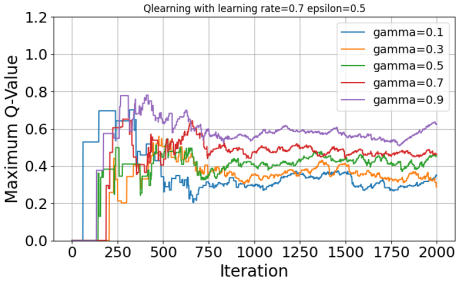


Figure 28— Q-learning with Gammas

With LR=0.9 (decaying ratio of 0.5), epsilon=0.7 (decaying ratio of 0.8), and gamma=0.9 (chosen through analysis of Fig. 26-28 and grid search), QL converges in approximately 20,000 episodes during a run of 10^5 episodes (Fig. 29) with a runtime of 125 s. QL yields a policy similar to that of VI and PI (Fig. 30 right and Fig. 25). Fig. 30 illustrates

the impact of parameter tuning on the agent's behavior: increasing the discount factor from 0.1 to 0.9 updates the center upper area, changing the 'left' actions (in black) to 'right' and 'down' actions (in green and red), thereby guiding the agent towards the goal state (the lower right corner).

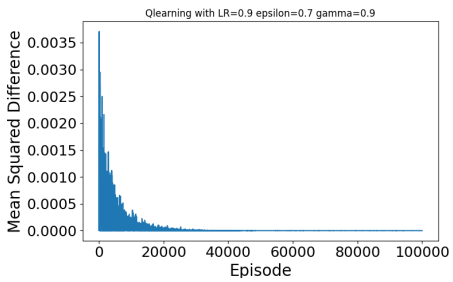


Figure 29— QL Convergence

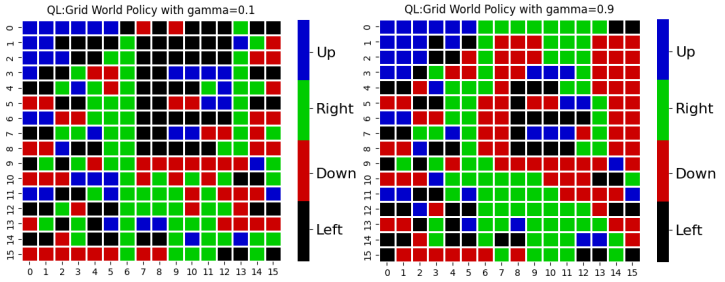


Figure 30— QL Policy Map (left: gamma=0.1, right: gamma=0.9)

16 × 16 Frozen Lake presents a significantly larger state space compared to 4 × 4 Frozen Lake. This large-size MDP problem demands higher computational costs, requiring more episodes for VI, PI, and QL to converge to similar policies. A well-defined reward structure is essential for guiding the agent toward the goal state, and coarse-defined rewards may lead to suboptimal solutions. The increased number of states adds complexity, posing challenges in balancing exploration-exploitation trade-offs during the thorough search of the extended state space. Besides, the larger state space requires careful adjustment of parameters in algorithms to achieve optimal performance, and it is more sensitive to parameter tuning, especially for model-free algorithms like QL, which incorporates more parameters than model-based VI and PI.

Furthermore, in the 16 × 16 Frozen Lake, QL requires 125 s to converge, whereas VI/PI take 0.09/3.1 s to converge to a similar policy. **The larger state size exacerbates QL's underperformance.**

3 CONCLUSIONS

In Blackjack, the focus on winning each short round places a premium on immediate rewards and new information. Hence, a small discount factor (gamma), a high learning rate, and a moderate level of exploration-exploitation (epsilon) are preferred, aligning with the goal of maximizing short-term gains. In contrast, Frozen Lake emphasizes the exploration of the environment to reach the goal state. A large gamma is advantageous for long-term rewards, a relatively high learning rate is sufficient to balance current knowledge and new information, and a high level of exploration at the initial stage supports a thorough exploration of the state space.

Larger state size demands more computational resources, increased episodes for convergence, additional efforts on parameter tuning, and the necessity of a well-defined reward structure to guide the algorithms effectively toward an optimal policy. In this case, a model-free algorithm such as Q-learning (QL) faces more challenges to converge to the same optimal policies than model-based algorithms because larger state sizes exacerbate the exploration-exploitation trade-off. The increased complexity makes it harder for QL to efficiently explore the state space and learn optimal policies, requiring more computational resources and careful tuning to achieve satisfactory convergence within a reasonable timeframe.

VI and PI are model-dependent algorithms, which guarantee convergences to the optimal policy if given enough iterations. In this study, PI converges faster but demands higher computational costs than VI, and both algorithms yield the same optimal policy while working on the same problem. On the other hand, QL is a model-free algorithm, which is applicable to environments with unknown dynamics, but optimal convergence is not guaranteed. Efficient exploration in large state spaces can be challenging for QL, leading to slower convergence. According to the observation in this study, VI and PI outperform QL while working in environments with known dynamics by faster convergence, shorter runtime, less effort on parameter tuning, and guaranteed convergence to the optimal policy.