# Exploring the Frozen Lake Problem

**To Ly (CS 4033) Thanh Hai Nguyen (CS 4033)**

## 1. Abstract

This investigation explores the application of reinforcement learning to the Frozen Lake problem. We are focusing on two main strategies: Q-learning and SARSA. Our goal is to determine which strategy works best for earning the highest reward by training models. We are also experimenting with different ways to adjust epsilon decay strategies, policy implementations, and reward shaping techniques to assess their impact on the algorithms' performance. Early results indicate that both Q-learning and SARSA can be effectively adjusted with certain tweaks, such as the application of exponential $\epsilon$-decay and adaptive reward functions, to optimize reward acquisition in the Frozen Lake setting

### 1.1 History

The Frozen Lake problem involves an agent on a frozen surface dotted with many holes. The goal is for the agent to reach a designated spot on the opposite side of the lake. However, the ice is slippery, so the agent must carefully navigate the treacherous ice to avoid falling into the holes

### 1.2 OpenAI Gym

OpenAI's Gym is a tool that makes it simple to work with different learning scenarios. It lets an agent do actions and get points for results (1). We will use it to teach models with the "FrozenLake-v1" setup.

### 1.3 Environment

The observation in the Frozen Lake environment is a value representing the player's current position calculated as $current\_row \times nrows + current\_col$, where both the row and col start at 0. Our goal position in a $4 \times 4$ map can be calculated as $3 \times 4 + 3 = 15$. It means that the player starts at position 0 and the goal is at position 15.

## 2. Hypotheses

Hypotheses are the following:

**2.1** Two of algorithms implemented such as SARSA, and Q-Learning will over n episodes. Q-Learning will be getting the greatest average reward.

**2.2** The speed at which epsilon-decay-rate ($\epsilon$-decay) directly affects how a learning agent improves. A good decay rate finds the right balance between trying new things (exploration) and sticking with what works (exploitation).

**2.3** The discount-factor $\gamma$ helps the agent perform better because it knows how to weigh future rewards against immediate ones in a smart way.

**2.4** The initial value of epsilon ($\epsilon$) in epsilon-greedy strategies plays a crucial role in reinforcement learning, balancing exploration and exploitation from the very beginning of the learning process. It will learn faster if the $\epsilon$ is set to an appropriate value.

## 3. Experiments

### 3.1 State Aggregation

Frozen Lake emphasizes its 4x4 grid layout = 16, resulting in 16 states with the player starting in state [0] at the top-left corner (location [0, 0]) and ending at state 15, the bottom-right corner which result $3 \times 4 + 3$. Mention the four possible actions (up, down, left, and right).

### 3.2 Hypothesis 1

Two models were taught using Q-Learning and SARSA, all with set values for important settings: epsilon($\epsilon$) was 1, epsilon-decay-rate($\epsilon$-decay) was 0.0001, $learning\_rate\_a$ ($\alpha$) was 0.9, and $discount\_factor\_g$ ($\gamma$) was 0.9. We used an $\epsilon$-greedy policy, which helps with balancing between exploration and exploitation. These algorithms were chosen because they don't require prior knowledge of the Markov Decision Process. Each model underwent training for 15,000 episodes, and rewards were recorded and averaged every 100 episodes to enhance clarity and minimize random fluctuations in the results.

Table1: Mean and standard deviation

| Algorithm | Mean | Std. Dev. |
|---|---|---|
| Q-learning | 28.7 | 33.3 |
| Sarsa | 23.5 | 26.9 |

Referring to Table 1 and Figure 1, we can see that Q-Learning achieved a higher mean reward (28.7) compared to SARSA (23.5) over 15000 episodes. However, both algorithms exhibited considerable variability in their per-

formance, as indicated by their relatively high standard deviations. Based on several other tests, SARSA also could achieve a higher mean reward compared to Q-learning.

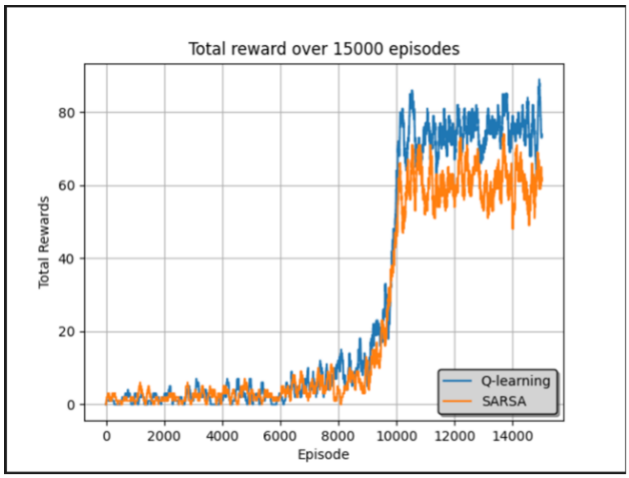


*Figure 1.* Average total reward per episode using Q-Learning and SARSA, $\alpha = 0.9$, $\gamma = 0.9$, $\epsilon = 1$, and $\epsilon$-decay = 0.0001

### 3.3 Hypothesis 2

The initial epsilon_decay_rate is 0.0001. Then, we will increase the $\epsilon$-decay to 0.0005 to observe the agent's speed. This adjustment implies that we intend for the agent to spend less time exploring and to focus more on exploitation. A smaller $\epsilon$-decay number is primarily beneficial for complex environments. However, in the case of the Frozen Lake 4x4 environment, which is not overly complicated, this adjustment is appropriate.

Table2: Mean and standard deviation after change $\epsilon$- decay

| Algorithm | Mean | Std. Dev. |
|---|---|---|
| Q-learning | 45.0 | 16.5 |
| Sarsa | 33.2 | 12.2 |

Referring to Table 2 and Figure 2, we observe that Q-Learning achieved a higher mean reward (45.0) compared to SARSA (33.2) over 15,000 episodes. Both algorithms exhibited considerable variability in their performance, and it is evident that the reward speed skyrocketed after 2000 episodes. This demonstrates that increasing $\epsilon$-decay in uncomplicated environments to achieve greater efficiency is indeed correct.

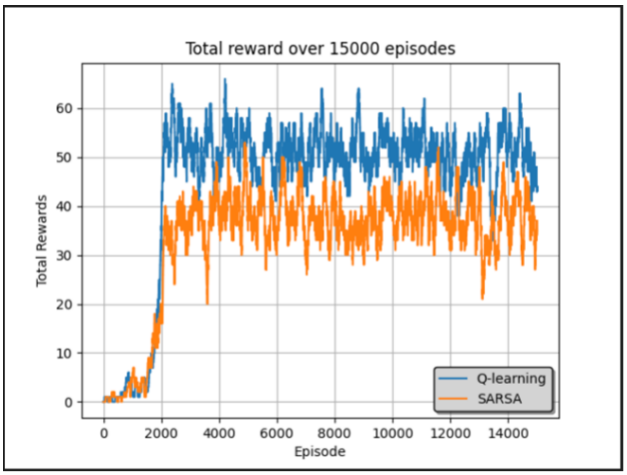


*Figure 2.* Average total reward per episode using Q-Learning and SARSA with increasing $\epsilon$-decay = 0.0005, $\alpha = 0.9$, $\gamma = 0.9$, and $\epsilon = 1$.

### 3.4 Hypothesis 3

Instead of each discount_factor_g being 0.9, we will reduce it to 0.5 to observe the impact it will have. This adjustment is made because the discount factor optimizes short-term rewards when low, whereas higher discount factors optimize long-term rewards.

Table 3: Mean and standard deviation after change $\gamma$

| Algorithm | Mean | Std. Dev. |
|---|---|---|
| Q-learning | 12.4 | 12.8 |
| Sarsa | 9.2 | 8.2 |

Referring to Table 3 and Figure 3, we can observe that the total reward has been significantly reduced after reducing the discount factor (discount_factor_g), which suggests that the agent is not effectively optimizing its performance over the long run during the execution process.

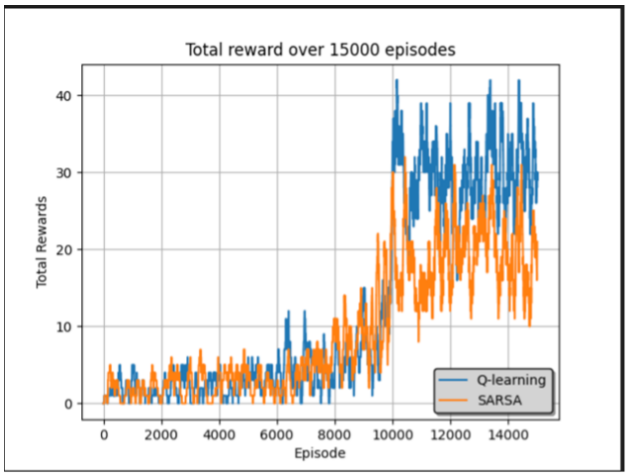


*Figure 3.* Average total reward per episode using Q-Learning and SARSA with decrease $\gamma$ =0.5, $\alpha$ = 0.9, $\epsilon$ = 1 ,and $\epsilon$-decay = 0.0001.

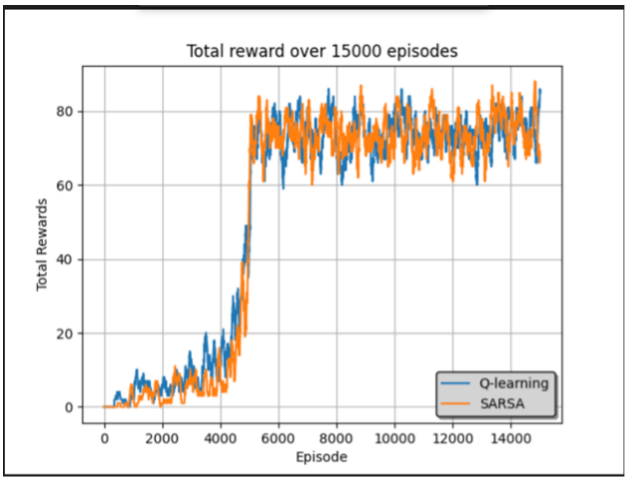


*Figure 4.* Average total reward per episode using Q-Learning and SARSA with decrease $\epsilon$ = 0.5, $\epsilon$-decay = 0.0001, $\alpha$ = 0.9, and $\gamma$ = 0.9.

### 3.5 Hypothesis 4

How will the initial epsilon affect the agent's performance, as we know higher epsilon values encourage strong exploration, low values will focus on further exploration based on previous discoveries, and epsilon ($\epsilon$) Initially will decrease from 1 to 0.5 to see what the effect will be.

Table 4: Mean and standard deviation after change the $\epsilon$

| Algorithm | Mean | Std. Dev. |
|---|---|---|
| Q-learning | 52.4 | 31.0 |
| Sarsa | 51.4 | 32.4 |

Referring to Table 4 and Figure 4, we observe that Q-Learning achieved a higher mean reward (52.9) compared to SARSA (45.1) over 15,000 episodes. Both algorithms exhibited considerable variability in their performance, and it is evident that the reward speed is considered to be the most effective in the proposed hypotheses. If we combine the initial decrease in epsilon with the increase in epsilon decay, the effect will be even more amazing. However, SARSA has a slightly higher standard deviation of 32.4 , this might be because SARSA, being an on-policy algorithm, evaluates and improves the same policy that it uses to make decisions, which could lead to increased sensitivity to changes in epsilon.

## 4. Learning Methods

Q-learning is a model-free reinforcement learning algorithm that learns the optimal action-selection policy for a given environment by iteratively updating Q-values based on observed rewards. It was selected for its simplicity and effectiveness in learning from discrete state-action pairs, making it well-suited for the discrete state space of the Frozen Lake environment.

On the other hand, SARSA (State-Action-Reward-State-Action) is another model-free algorithm that learns action-selection policies by directly updating Q-values based on observed transitions from state-action pairs to next-state-action pairs. SARSA was chosen for its on-policy nature, which allows it to evaluate and improve the same policy it uses for decision-making, making it particularly suitable for environments where exploration and exploitation need to be balanced dynamically.

## 5. Literature Review

### 5.1 Q-Learning vs SARSA-Learning

The comparison between Q-Learning and SARSA is a common theme in RL literature. Q-Learning is an off-policy RL algorithm that learns the optimal action-value function, whereas SARSA is an on-policy algorithm that learns the value of the current policy. The results presented in the report align with previous studies demonstrating that Q-Learning often achieves higher mean rewards compared to SARSA in certain environments, as observed in (4). However, it's important to note that SARSA may outperform Q-Learning in specific scenarios, as indicated by the authors'

mention of SARSA achieving higher rewards in other tests. This variability in performance highlights the importance of understanding the characteristics of the environment and the algorithms' properties.

### 5.2 Effect of Epsilon Decay Rate

The epsilon-greedy policy is a fundamental exploration-exploitation strategy in RL, where epsilon ($\epsilon$) determines the probability of selecting a random action versus exploiting the current knowledge. Adjusting the epsilon decay rate can significantly impact the agent's learning dynamics. The report demonstrates that increasing the epsilon decay rate can accelerate learning in relatively simple environments like Frozen Lake. This finding is consistent with previous research indicating that a higher epsilon decay rate can lead to faster convergence and improved efficiency in learning tasks (3).

### 5.3 Impact of Discount Factor ($\gamma$)

The discount factor ($\gamma$) plays a crucial role in balancing immediate rewards versus long-term goals in RL. Lowering the discount factor prioritizes short-term rewards, while higher values encourage the agent to focus on maximizing cumulative rewards over time. The results from the report suggest that reducing the discount factor negatively affects the agent's performance in Frozen Lake. This aligns with the theoretical understanding that higher discount factors are beneficial for optimizing long-term rewards, particularly in environments with sparse or delayed rewards (5).

### 5.4 Initial Epsilon Exploration

The initial value of epsilon ($\epsilon$) influences the agent's exploration behavior at the beginning of training. Higher epsilon values promote more extensive exploration, while lower values prioritize exploitation of known strategies. The report demonstrates that decreasing the initial epsilon from 1 to 0.5 leads to improved performance for both Q-Learning and SARSA in the Frozen Lake environment. This finding is consistent with previous studies suggesting that gradually decreasing epsilon during training can facilitate efficient exploration while maintaining a balance with exploitation (2).

## 6. Conclusion

The literature review highlights the significance of algorithm selection and parameter tuning in RL, particularly in environments like Frozen Lake. Q-Learning and SARSA are foundational algorithms with distinct characteristics that make them suitable for different scenarios. Additionally, adjusting parameters such as epsilon decay rate and discount factor can significantly impact learning dynamics and ultimately affect agent performance. Understanding these factors and their interactions is essential for designing effective RL systems in real-world applications.

## 7. Future Work

Future work for Frozen Lake could involve several avenues for research and experimentation. Firstly, exploring deep reinforcement learning techniques, such as employing neural networks for function approximation, could enhance the agent's ability to learn complex policies and generalize across states. Additionally, investigating variations of the Frozen Lake environment with continuous action spaces presents an intriguing challenge that could lead to novel algorithmic solutions. Furthermore, incorporating hierarchical reinforcement learning methods or introducing multi-agent settings could offer opportunities to tackle more intricate decision-making scenarios and foster collaborative or competitive behaviors. Moreover, studying transfer learning techniques to leverage knowledge gained in Frozen Lake for other tasks or domains could accelerate learning in new environments. Finally, assessing the robustness and generalization capabilities of RL agents trained in Frozen Lake to varying conditions and unseen scenarios could provide valuable insights into the effectiveness and limitations of reinforcement learning algorithms in real-world applications.

## 8. Contribution

Thanh Hai Nguyen conducted experiments based on Hypotheses 1 and 2, while To Ly conducted experiments corresponding to Hypotheses 3 and 4.

## References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, and David Silver. Deep q-learning from demonstrations. In *Advances in Neural Information Processing Systems*, pages 2095–2105, 2018.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[4] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[5] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.