Model-Free Reinforcement Learning: Comparison of Monte Carlo, SARSA, and Q-learning

Sungho Yoon
AE 598 RL - Department of Aerospace Engineering
University of Illinois Urbana–Champaign
Email: sunghoy2@illinois.edu

***Abstract*—This report investigates three model-free reinforcement learning algorithms—First-Visit Monte Carlo (MC), SARSA, and Q-learning—applied to the Frozen Lake environment. The environment is modeled as a Markov Decision Process (MDP) with absorbing terminal states and either deterministic (non-slippery) or stochastic (slippery) transitions. Each method is implemented with ε-greedy exploration and evaluated periodically via greedy rollouts. Results compare learning curves, policies, and value functions across both environment settings.**

I. Introduction

The Frozen Lake problem is a benchmark for reinforcement learning, where an agent must navigate a grid world from a start state to a goal while avoiding holes. The first homework assignment introduced the best policy through direct programming (DP) with two approaches: value iteration (VI) and policy iteration (PI), which require complete knowledge of the transition and reward models. In contrast, model-free methods learn directly from sampled trajectories. This report applies Monte Carlo control, SARSA (on-policy TD), and Q-learning (off-policy TD) to analyze their behavior and compare their performance in solving the Frozen Lake problem.

II. Methods

A. Problem Model

We use the OpenAI Gymnasium implementation of FrozenLake-v1 with a 4×4 grid:

- State space S: 16 discrete states with the map of $[SFFF, FHFH, FFFH, HFFG]$
- Action space A: {UP, DOWN, LEFT, RIGHT}.
- Transition model: deterministic (non-slippery) or stochastic with $1/3$ probability of deviating to perpendicular actions (slippery).
- Reward model: $+1$ for reaching the goal, 0 otherwise.
- Discount factor: $\gamma = 0.95$.

B. Algorithms

Each algorithm follows the standardized evaluation procedure described in Chapter 2.7 of Albrecht et al. [2]. Training is conducted over 8000 episodes using ε-greedy exploration ($\varepsilon=, 0.10$) and a learning rate $\alpha=, 0.10$ for TD methods. Every 500 episodes, the current policy is evaluated through 200 greedy rollouts (ε = 0) to estimate its expected return. The reported learning curves show the mean and standard deviation over five random seeds, as recommended for statistically robust comparison.

1) Monte Carlo Control: On-Policy First-Visit Monte Carlo requires the full episode to update Q. The ε-greedy with respect to the current Q is employed as a behavior policy to ensure the agent's exploration during training. For a given episode, the ε-greedy estimates $Q(s, a)$ using returns observed at the first occurrence of (s, a):

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{N(s, a)} \left(g - Q(s, a)\right),$$

where $N(s, a)$ counts first visits and g is the return from the first visit onward. The target policy is the greedy policy with respect to Q:

$$\pi^*(s) = \arg\max_a Q(s, a)$$

Algorithm 1 On-policy First-Visit MC Control

```
Initialize Q(s,a) = 0, N(s,a) = 0
for each episode do
  Generate episode using ε-greedy(Q)
  for each timestep in episode do
    Compute return G backward
    if (s,a) is first-visited then
      Update Q(s,a) by sample mean
      Increment N(s,a)
    end if
    Update G
  end for
end for
return Q
```

C. SARSA (On-policy TD Control)

SARSA (State-Action-Reward-State-Action) is an on-policy temporal-difference control method that updates $Q(s, a)$ at every step, rather than waiting for full episodes. The behavior policy is the ε-greedy policy with respect to the current

Q, ensuring exploration. At each transition (s, a, r, s', a'), SARSA performs the update using the on-policy TD target:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r + \gamma Q(s',a') - Q(s,a)\Big],$$

where the next action a' is chosen ε-greedily from Q. This makes SARSA both sample efficient (bootstrapping at each step) and consistent with the behavior policy, ensuring on-policy convergence.

Algorithm 2 On-policy SARSA(0) Control

```
Initialize Q(s,a) = 0 for all (s,a)
for each episode do
  Initialize state s
  Choose action a using ε-greedy w.r.t. Q
  for each timestep in the episode do
    Take action a, observe reward r and next state s'
    Choose next action a' using ε-greedy with respect to Q
    Update Q(s,a) toward r + γQ(s',a')
    Set s ← s', a ← a'
    if s' is terminal then
      break
    end if
  end for
end for
return Q
```

D. Q-learning (Off-policy TD Control)

Q-learning is an off-policy temporal-difference control method that also updates $Q(s,a)$ at each step, but uses the greedy target rather than the behavior policy. The behavior policy is still ε-greedy to ensure exploration, while the target policy is greedy with respect to Q. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big].$$

By bootstrapping with the greedy action at s', Q-learning learns the optimal policy $\pi^*(s) = \arg\max_a Q(s,a)$ even while following an exploratory behavior policy. This off-policy nature often makes Q-learning more aggressive and sometimes faster than SARSA, but also more sensitive to variance in stochastic environments.

E. Implementation Details

For the simplicity of the code, ε is fixed instead of decaying like GLIE. The number of episodes for training and the frequency of evaluation are determined arbitrarily as long as meaningful convergence is achieved. As Albrech et al. [2] suggested, multiple seeds are used to compute the mean standard deviation to plot the return over timesteps.

- ε fixed at 0.10
- $\alpha = 0.10$ for SARSA and Q-learning.
- Training: 8000 episodes; evaluation every 500 episodes with 200 greedy rollouts.
- Multiple seeds (0–4) are averaged to produce mean $\pm$ standard deviation learning curves.

Algorithm 3 Off-policy Q-learning Control

```
Initialize Q(s,a) = 0 for all (s,a)
for each episode do
  Initialize state s
  for each timestep in the episode do
    Choose action a using ε-greedy with respect to Q
    Take action a, observe reward r and next state s'
    Update Q(s,a) toward r + γ max_a' Q(s',a')
    Set s ← s'
    if s' is terminal then
      break
    end if
  end for
end for
return Q
```

III. Results

A. Slippery Dynamics

Figure 1 shows the evaluation returns as a function of environment timesteps. Both SARSA and Q-learning converge more quickly and reach higher success rates (≈ 0.6–0.7) than Monte Carlo control, which stabilizes at a lower level (≈ 0.3). This difference arises because MC requires full episodes before updating Q, leading to higher-variance estimates and fewer overall timesteps, whereas TD methods update incrementally at each step and propagate value information more efficiently. Among the TD methods, Q-learning achieves the highest returns since it optimizes directly toward the greedy target, while SARSA is slightly more conservative due to its on-policy nature.

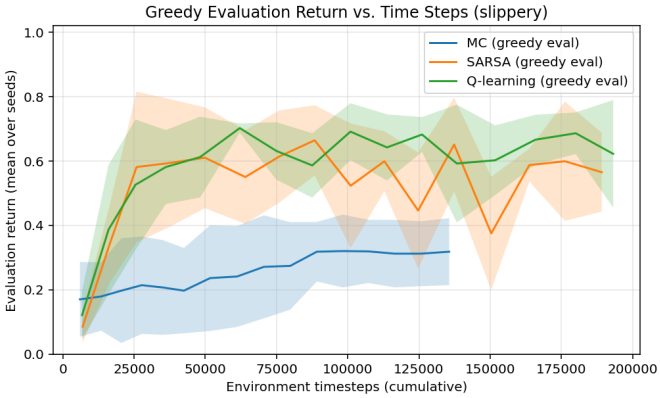
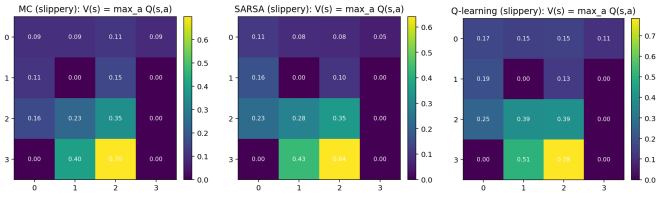


Fig. 1: Evaluation return vs. timesteps (slippery).

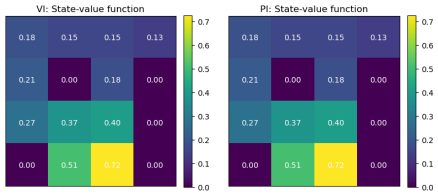
1) State-value Heatmaps: The learned state-value heatmaps (Figures 2a–2c) show that all model-free methods identify value concentration along safe paths toward the goal, with Q-learning generally assigning the strongest values. First-Visit MC produces weaker and noisier estimates, while SARSA is more balanced but slightly conservative due to its on-policy nature.

For comparison, Figures 3a–3b show the state-value functions obtained by Value Iteration (VI) and Policy Iteration (PI). These model-based methods produce smooth, consistent value gradients toward the goal, which can be considered the ground-truth baseline. Among the model-free methods, Q-learning most closely resembles the structure of VI and PI, while SARSA captures the safe path more cautiously, and MC underestimates most of the values due to its reliance on full-episode returns.



(a) MC (b) SARSA (c) Q-learning

Fig. 2: State-value functions under slippery dynamics.



(a) Value Iteration (VI) (b) Policy Iteration (PI)

Fig. 3: State-value functions obtained by Dynamic Programming.

2) Greedy Policies (Slippery): The greedy policies in Figure 4 illustrate key differences among the three methods. First-Visit MC produces a noisy and inconsistent policy, reflecting the high variance of return estimates and the fact that updates only occur after complete episodes. SARSA yields a more cautious policy, avoiding risky actions that may lead to holes; this is a direct consequence of its on-policy nature, where value updates incorporate the ε-greedy behavior actually followed during training. In contrast, Q-learning produces a sharply goal-directed policy, aggressively steering the agent toward the goal. This reflects its off-policy character: updates use the greedy target $\max_{a'} Q(s', a')$ regardless of whether exploratory moves are made. As a result, Q-learning generally achieves higher state values, but may also adopt riskier paths compared to SARSA.

→ ↑ ↓ ↑	↓ ↑ ← ↑	← ↑ ← ↑
← H → H	← H ← H	← H ← H
↑ → ← H	↑ ↓ ↓ H	↑ ↓ ← H
H → → G	H → ↓ G	H → ↓ G

(a) MC (b) SARSA (c) Q-learning

Fig. 4: Greedy policies under slippery dynamics.

B. Non-slippery Dynamics

Under deterministic transitions, all methods converge to near-optimal policies with evaluation returns close to 1.0 (Figure 5).

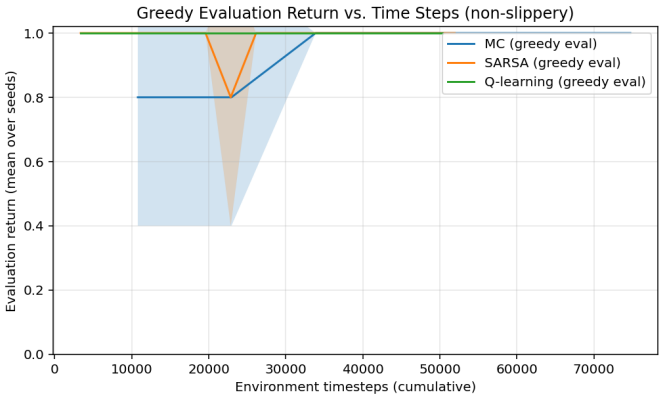
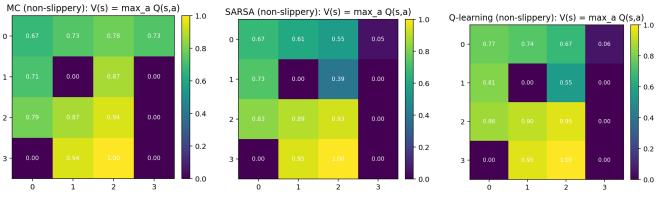


Fig. 5: Evaluation return vs. timesteps (non-slippery).

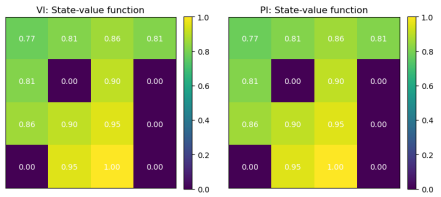
1) State-value Heatmaps: Value heatmaps (Figures 6a–6c) show consistently high values along direct paths to the goal under deterministic dynamics. A notable difference across methods appears at state s=3 (top-right corner), where the agent must navigate around two holes to reach the goal. Monte Carlo assigns this state a value comparable to other frozen tiles, while both TD methods (SARSA and Q-learning) correctly assign it a near-zero value. More broadly, the upper-right region is generally rated lower by the TD methods, reflecting their ability to learn from incremental feedback and avoid risky states during training. This highlights a key advantage of TD learning over MC: TD methods update value estimates online as experience unfolds, allowing them to down-weight "bad" states before full episodes are completed, whereas MC can only adjust after the entire episode is observed.



(a) MC (b) SARSA (c) Q-learning

Fig. 6: State-value functions under non-slippery dynamics.

Figures 7a–7b show the DP baselines. VI and PI produce nearly identical, smooth value surfaces with peak values of approximately 1.0 along the shortest safe path. Among the model-free methods, Q-learning generally maintains the highest overall magnitudes, followed by SARSA and then MC. However, near the top-right region, including state s=3, the value distribution of MC more closely resembles that of VI and PI than the TD-based methods. This occurs because, under deterministic transitions, Monte Carlo updates based on full-episode returns can capture accurate value estimates



(a) VI (b) PI

Fig. 7: State-value functions obtained by Dynamic Programming.

without the bootstrapping bias inherent in TD learning. In contrast, TD methods—though advantageous in stochastic environments—tend to underestimate such regions due to their incremental updates and reliance on current policy behavior. This is opposite to the slippery case, where TD methods more closely aligned with DP results owing to their ability to average over stochastic transitions through continual bootstrapping.

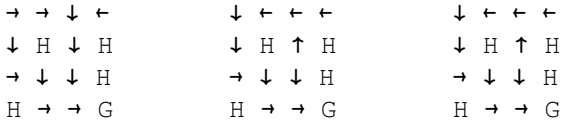
2) Greedy Policies (Non-slippery): Figure 8 shows the corresponding greedy policies under deterministic dynamics. Several key differences are observed among the three methods.

First, Monte Carlo (MC) begins by moving to the right from the start state, whereas both SARSA and Q-learning initially move downward. This difference reflects MC's reliance on complete-episode returns—without intermediate bootstrapping—which allows it to occasionally favor trajectories that were successful in earlier episodes, even if they are suboptimal in terms of step efficiency. In contrast, the TD-based methods refine their estimates continuously during training, leading to more consistent early downward movement along the optimal path.

Second, at state s=6 (second row, third column), both TD methods choose to move upward rather than downward toward the goal. This seemingly counterintuitive behavior arises because returning upward offers a safer trajectory in earlier training stages, avoiding nearby holes that would terminate the episode. The TD updates therefore propagate more conservative value estimates in risky regions, resulting in behavior that prioritizes safety over shortest distance.

Third, SARSA and Q-learning produce identical policies in this deterministic setting. The lack of difference between the on-policy and off-policy variants can be attributed to the absence of stochasticity in transitions. In deterministic environments, ε-greedy exploration has little effect once both methods converge, and their updates produce the same greedy limit policy.

Overall, MC demonstrates more exploratory and occasionally inefficient choices, while SARSA and Q-learning converge to nearly optimal and identical safe paths toward the goal. The alignment between SARSA and Q-learning also indicates that, under deterministic conditions, the effect of the behavior–target policy mismatch in off-policy learning becomes negligible.



(a) MC (b) SARSA (c) Q-learning

Fig. 8: Greedy policies under non-slippery dynamics.

IV. Conclusions

This report compared three model-free control algorithms—First-Visit Monte Carlo (MC), SARSA, and Q-learning— on the 4×4 FrozenLake MDP with both slippery (stochastic) and non-slippery (deterministic) dynamics, and contrasted their learned value functions with model-based Dynamic Programming (DP) baselines (VI/PI). All methods were trained with fixed ε-greedy exploration and periodically evaluated with greedy rollouts over multiple random seeds, reporting mean $\pm$ standard deviation performance versus environment timesteps.

1) Learning speed and asymptotic performance.: Under slippery dynamics, temporal-difference (TD) methods (SARSA and Q-learning) converged faster and to higher success rates (≈ 0.6–0.7) than MC (≈ 0.3). This is consistent with TD's stepwise bootstrapping, which propagates value information after every transition and reduces the variance of target estimates relative to MC's full-episode returns. Between the TD methods, Q-learning typically achieved the highest returns due to its off-policy greedy target, while SARSA was slightly more conservative because it learns the value of the ε-greedy behavior it actually follows. In the non-slippery case, all methods achieved near-optimal performance (returns close to 1.0) once sufficient experience was gathered.

2) Policies and risk posture.: Greedy policy visualizations revealed systematic behavioral differences. In the slippery setting, Q-learning produced the most aggressively goal-directed policies, sometimes preferring riskier routes; SARSA favored safer actions near risky tiles; and MC exhibited more inconsistency due to high-variance returns and delayed (end-of-episode) updates. In the non-slippery setting, SARSA and Q-learning converged to essentially identical policies, reflecting that, with deterministic transitions, on- and off-policy TD targets coincide at convergence; MC occasionally preferred historically successful but longer routes early on, before aligning with the direct path.

3) Value-function structure and comparison to DP.: Heatmaps showed that all model-free methods learned value concentration along safe paths to the goal. In the slippery case, Q-learning's value maps most closely matched the smooth VI/PI baselines, with SARSA slightly lower near risky regions and MC noticeably underestimating due to episodic target variance. In the non-slippery case, all methods approached the DP value surface; interestingly, in the upper-right region (e.g., around state s=3), MC sometimes resembled VI/PI more closely than TD. This reversal highlights that MC's full-episode targets can be highly accurate when transitions are

deterministic, whereas TD's incremental updates may transiently underweight seldom-visited or locally riskier regions under greedy behavior.

4) Methodological takeaways.: (1) TD control is preferable in stochastic environments due to efficient bootstrapping and lower target variance; (2) Q-learning often attains the strongest performance but can favor riskier policies, while SARSA trades a small amount of performance for robustness; (3) MC can be competitive in deterministic settings but typically learns more slowly in terms of timesteps and is sensitive to episode length and return variance.

5) Limitations and future work.: We fixed ε and α for clarity; adopting GLIE-style decay or more adaptive schedules may further improve asymptotic performance and reduce exploration-induced bias. Extending to larger maps, alternative reward structures (e.g., step penalties), or continuing tasks would stress-test the methods' scalability. Finally, replacing tabular Q with function approximation (e.g., linear or neural) and evaluating with target networks or double Q-learning could mitigate overestimation and improve stability in stochastic settings.

Overall, Q-learning achieved the most consistent and optimistic estimates, converging faster to strong policies. SARSA, while slightly more conservative, performed comparably in both environments. MC was clearly disadvantaged in the stochastic setting, illustrating the benefit of temporal-difference bootstrapping over pure return-based estimation.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018. Available: http://incompleteideas.net/book/the-book-2nd.html

[2] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, MIT Press, 2024. Available: https://www.marl-book.com