Reinforcement Learning Assignment 2

Li Tao, Yuxiang Cao, Huitong Gao

May 19, 2023

1 Describe the algorithms

1.1 Q-learning

1. Initialize Q table where $Q(s, a)$ represents the long-time cumulative rewards after taking an action a in state s.
2. In each episode, initialize state s_0
3. In each step, choose an action a by behavior policy, perform it and observe a new state s'. Improve Q table according to target policy, then update state s to s' until terminal state.

Behavior policy: the algorithm chooses action by epsilon greedy (With 5% probability to choose action randomly, otherwise choose the action with the maximum value in Q table under state s).

Target policy: improve $Q(s, a)$ by greedy (Using the maximum value under a determined state as target).

1.2 Double Q-learning

1. Initialize table Q_1 and Q_2 where $Q_n(s, a)$ represents the long-time cumulative rewards after taking an action a in state s.
2. In each episode, initialize state s_0
3. In each step, choose an action a by behavior policy, perform it and observe a new state s'. Improve the Q table according to target policy, then update state s to s' until terminal state.

Behavior policy: the algorithm chooses action by epsilon greedy considering both Q_1 and Q_2 (With 5% probability to choose action randomly, otherwise choose the action with the maximum value in both Q_1 and Q_2 under state s).

Target policy: there is 50% probability to improve either $Q_1(s, a)$ or $Q_2(s, a)$ by greedy (Using another Q table to find the maximum value under a determined state as target).

1.3 SARSA

1. Initialize table Q where $Q(s, a)$ represents the long-time cumulative rewards after taking an action a in state s.
2. In each episode, initialize state s_0 and choose an action a by behavior policy.
3. In each step, take action a and observe a new state s', choose a new action a' from s' by behavior policy and improve the Q table according to target policy, then update state s to $s\prime$, action a to a' until terminal state.

Behavior policy: the algorithm chooses action by epsilon greedy (With 5% probability to choose action randomly, otherwise choose the action with the maximum value in Q table under state s).

Target policy: improve $Q(s, a)$ by the action $a\prime$ that behavior policy chose previously (Using $Q(s\prime, a\prime)$ as target).

1.4 Expected SARSA

1. Initialize table Q where $Q(s, a)$ represents the long-time cumulative rewards after taking an action a in state s.
2. In each episode, initialize state s_0 and choose an action a by behavior policy.
3. In each step, take action a and observe a new state s', choose a new action a' from s' by behavior policy and improve the Q table according to target policy, then update state s to $s\prime$, action a to a' until terminal state.

Behavior policy: the algorithm chooses action by epsilon greedy (With 5% probability to choose action randomly, otherwise choose the action with the maximum value in Q table under state s).

Target policy: improve $Q(s, a)$ using the expected value, taking into account how likely each action is under the current policy. The probability of each action could be calculated as following:

For non-optimal action a:

$$\pi(a|S_{t+1}) = \frac{1-\epsilon}{total\ action} \tag{1}$$

For optimal action a:

$$\pi(a|S_{t+1}) = \frac{\epsilon}{max\ action} + \frac{1-\epsilon}{total\ action} \tag{2}$$

(Using $\sum_a \pi(a|S_{t+1})Q(S_{t+1}, a)$as target)

2 Run and plot rewards

If we set $is\_slippery = True$, then the result were showed as several pictures in different environment and different agents. And the following parts are all based on a slippery Frozen Lake and River Swim environment.

2.1 Q-learning

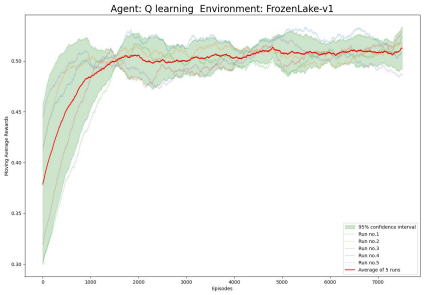

Figure 1: Q-learning in Frozen Lake.

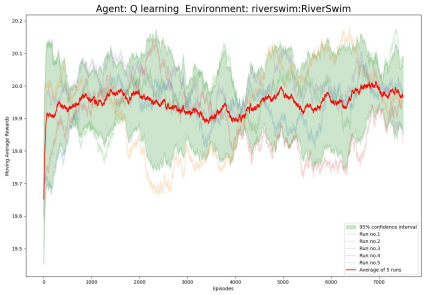

Figure 2: Q-learning in River Swim.

2.2 Double Q-learning

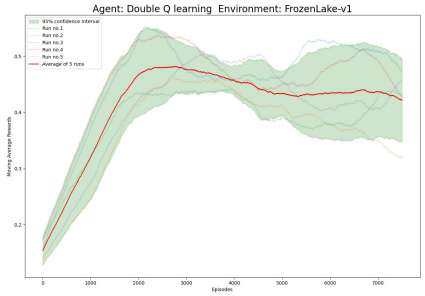

Figure 3: Double Q-learning in Frozen Lake.

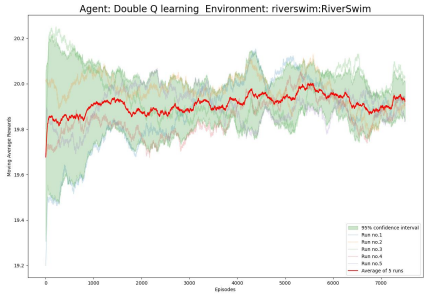

Figure 4: Double Q-learning in River Swim.

2.3 SARSA

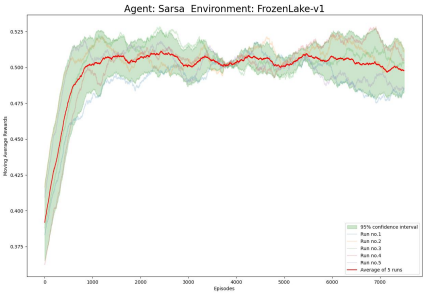

Figure 5: SARSA in Frozen Lake.

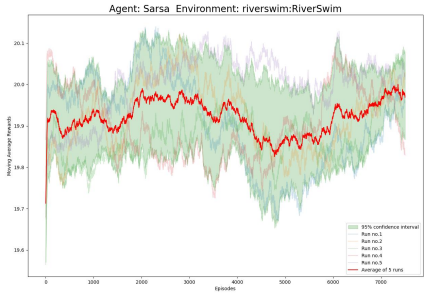

Figure 6: SARSA in River Swim.

2.4 Expected SARSA

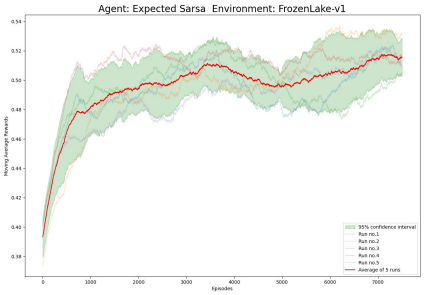

Figure 7: Expected SARSA in Frozen Lake.

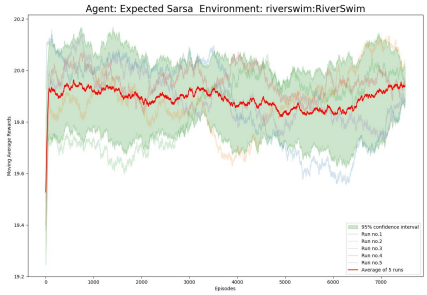

Figure 8: Expected SARSA in River Swim.

3 Visualize Q-values and greedy policy

3.1 Q-learning

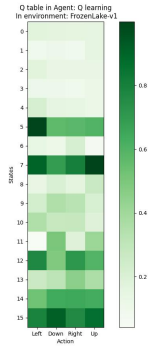

Figure 9: Q-values for Q-learning in Frozen Lake.

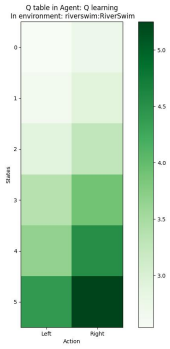

Figure 10: Q-values for Q-learning in River Swim.

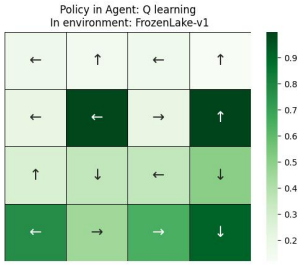

Figure 11: Policy for Q-learning in Frozen Lake.

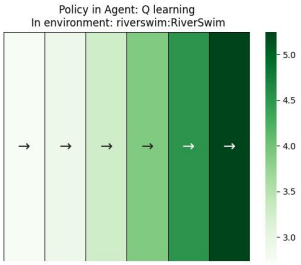

Figure 12: Policy for Q-learning in River Swim.

3.2 Double Q-learning

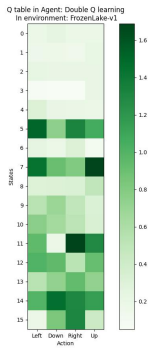

Figure 13: Q-values for Double Q-learning in Frozen Lake.

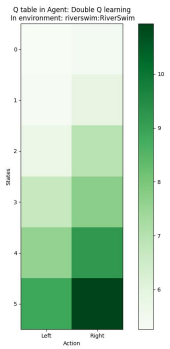

Figure 14: Q-values for Double Q-learning in River Swim.

Figure 15: Policy for Double Q-learning in Frozen Lake.

Figure 16: Policy for Double Q-learning in River Swim.

3.3 SARSA

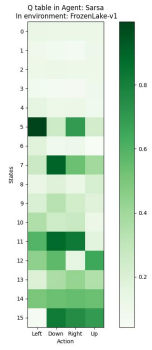

Figure 17: Q-values for SARSA in Frozen Lake.

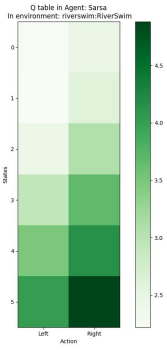

Figure 18: Q-values for SARSA in River Swim.

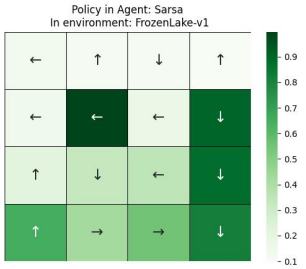

Figure 19: Policy for SARSA in Frozen Lake.

Figure 20: Policy for SARSA in River Swim.

3.4 Expected SARSA

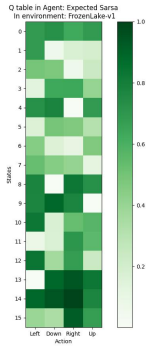

Figure 21: Q-values for Expected SARSA in Frozen Lake.

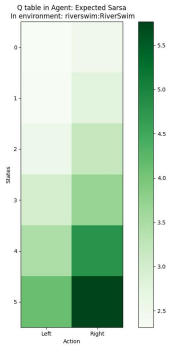

Figure 22: Q-values for Expected SARSA in River Swim.

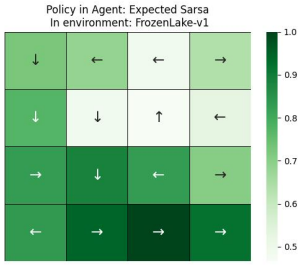

Figure 23: Policy for Expected SARSA in Frozen Lake.

Figure 24: Policy for Expected SARSA in River Swim.

3.5 Conclusion of policy

We can see from the figures above that all agents performed in River Swim have obtained the optimal policy, the actions taken at each state are all went right, where is direction to achieve goal of environment. Meanwhile, since Frozen Lake are slippery, the agent moved in intended direction with probability of $\frac{1}{3}$ or moved in either perpendicular direction with equal probability of $\frac{1}{3}$ in both directions. It is hard to see whether the policies displayed are optimal or not.

4 Initialization of Q-values

Compare randomly initialize Q-values with initialize them with 1

4.1 Affect rewards

4.1.1 In Frozen Lake

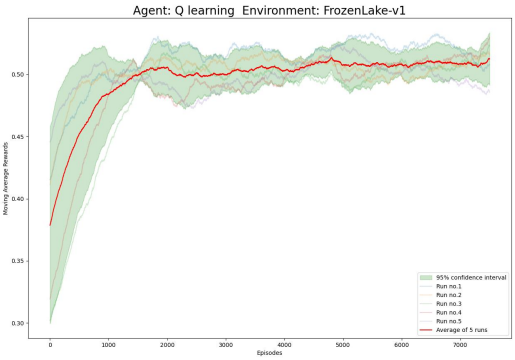

Figure 25: Q-learning with random Q-table.

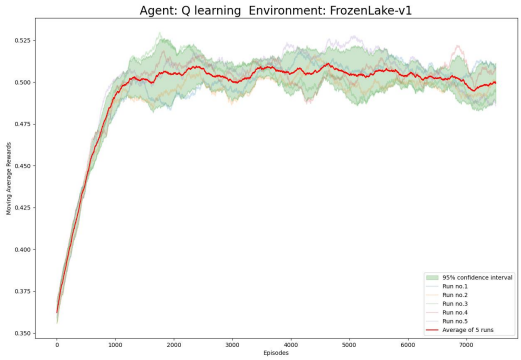

Figure 26: Q-learning with ones Q-table.

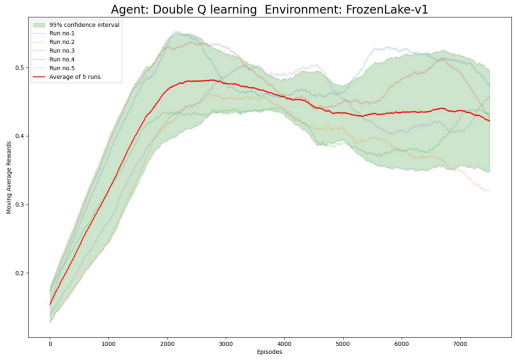

Figure 27: Double Q-learning with random Q-table.

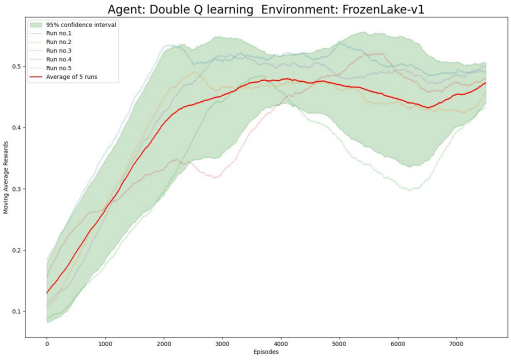

Figure 28: Double Q-learning with ones Q-table.

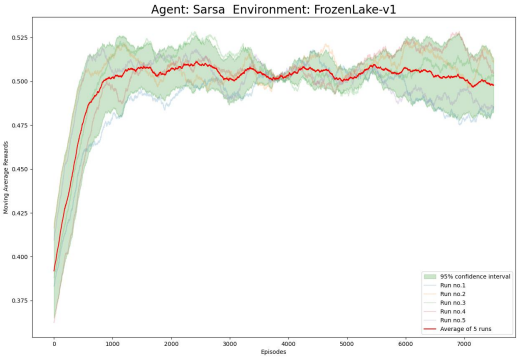

Figure 29: SARSA with random Q-table.

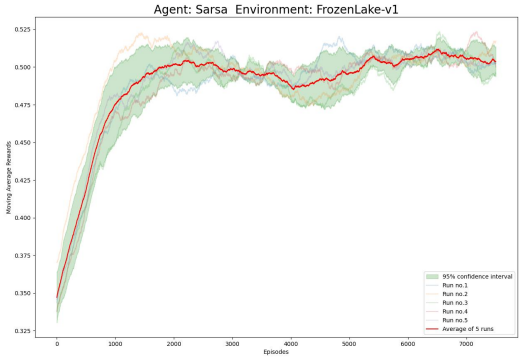

Figure 30: SARSA with ones Q-table.

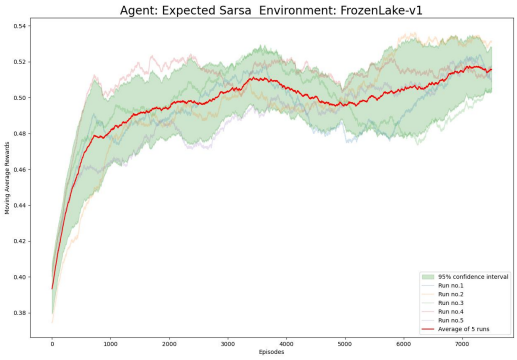

Figure 31: Expected SARSA with random Q- table.

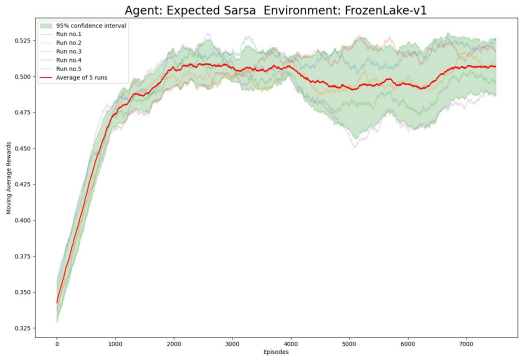

Figure 32: Expected SARSA with ones Q-table.

It's visible that the 95% confidence interval in the beginning stage becomes smaller after initialized all Q-values to 1 in all agents except Double Q-learning, the reason is that the agent starts with a biased assumption that all actions are equally valuable and have the same expected reward. This bias can persist throughout the learning process and hinder the agent from accurately estimating the true values of different actions in different states. But there's no significant impact on the rewards achieved by agents, moving average rewards for randomly table and ones table are pretty much the same.

4.1.2 In River Swim

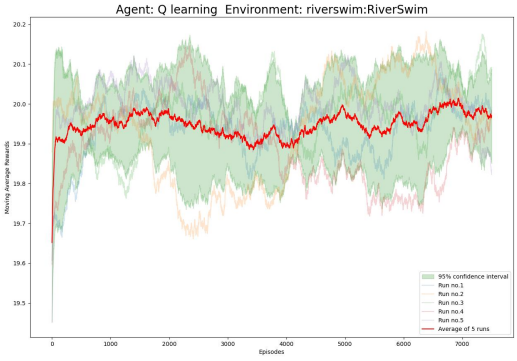

Figure 33: Q-learning with random Q-table.

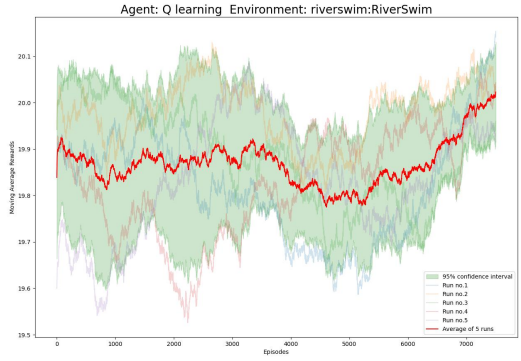

Figure 34: Q-learning with ones Q-table.

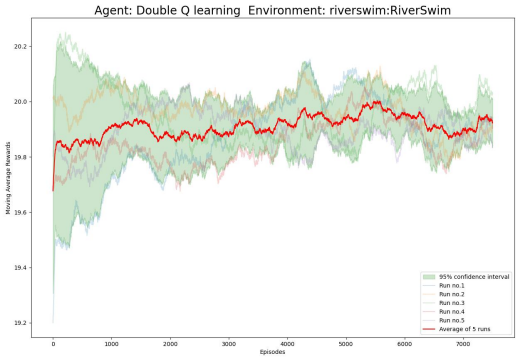

Figure 35: Double Q-learning with random Q-table.

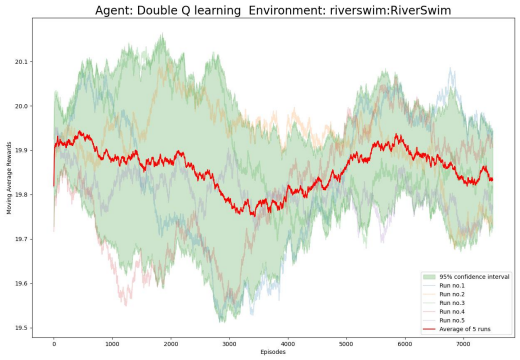

Figure 36: Double Q-learning with ones Q-table.

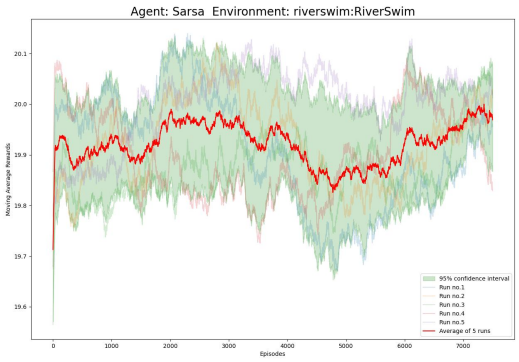

Figure 37: SARSA with random Q-table.

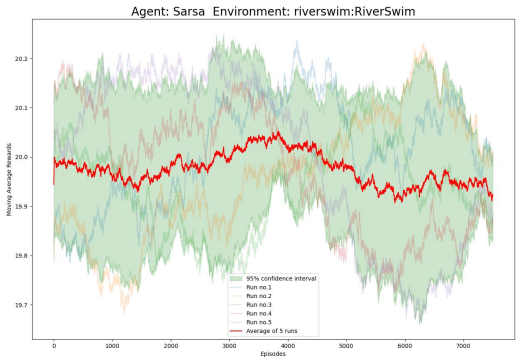

Figure 38: SARSA with ones Q-table.

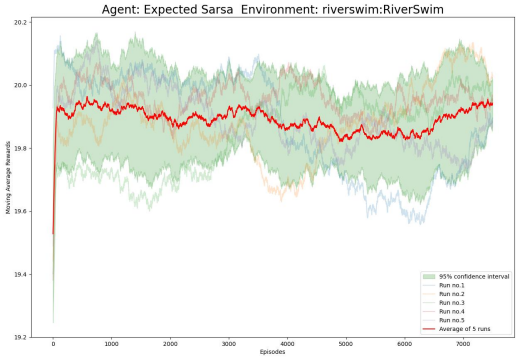

Figure 39: Expected SARSA with random Q-table.

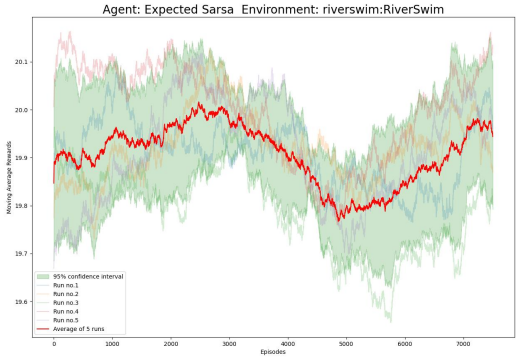

Figure 40: Expected SARSA with ones Q-table.

Initializing Q-tables optimistically (larger than the true table to be) in River Swim is to assume Q-table to be 5 at the beginning, but it needs enormous time since we have to set max step in each episode as large as possible, and we didn't take it into consideration into our report.

4.2 Affect policies

Initializing Q-tables optimistically delays the convergence process of greedy policies. Because the agent increases the exploration process due to larger Q-tables, even though these actions may not be the optimal ones, so the exploration time is increased, but as the agent continuously updates the Q-tables and collects experience, it will slowly converge to the optimal policy.

4.2.1 In Frozen Lake

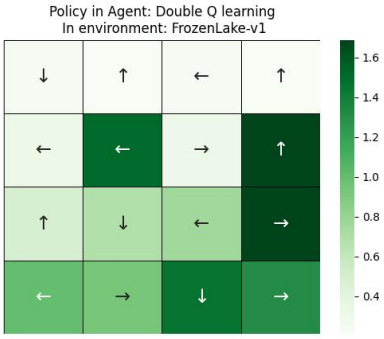

Figure 41: Q-learning with random Q-table.

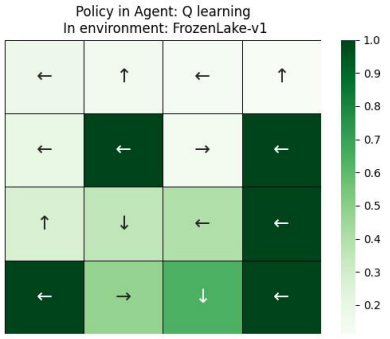

Figure 42: Q-learning with ones Q-table.

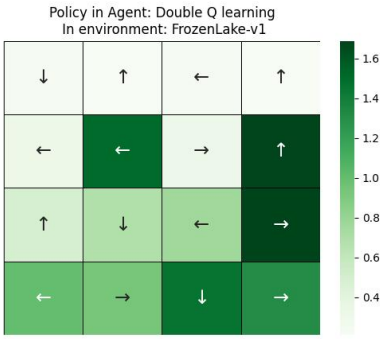

Figure 43: Double Q-learning with random Q-table.

Figure 44: Double Q-learning with ones Q-table.

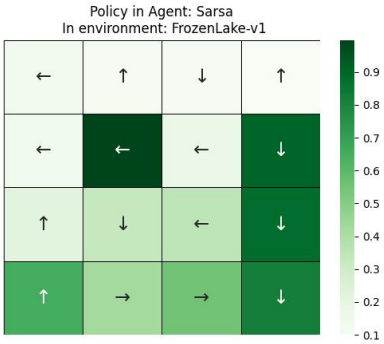

Figure 45: SARSA with random Q-table.

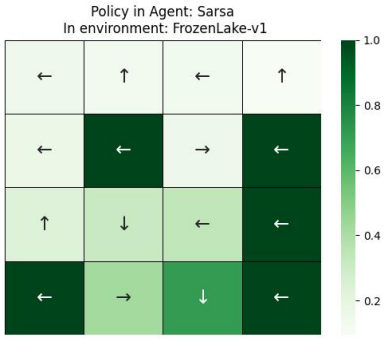

Figure 46: SARSA with ones Q-table.

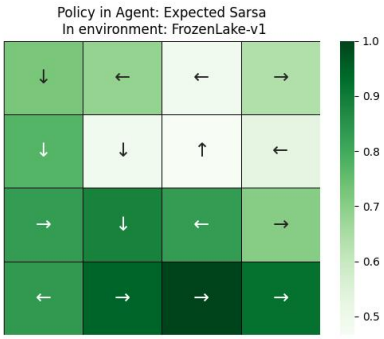

Figure 47: Expected SARSA with random Q-table.

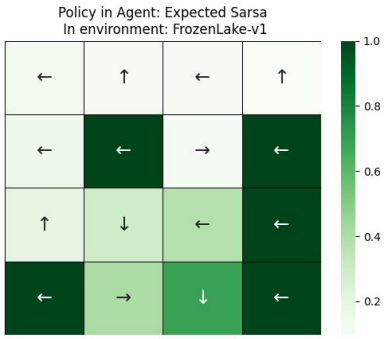

Figure 48: Expected SARSA with ones Q-table.

There are some similarities between policies from 4 agents when we randomly initialized Q-tables, but after setting Q-tables to 1, the policy obtained by 4 agents are the same.

4.2.2 In River Swim

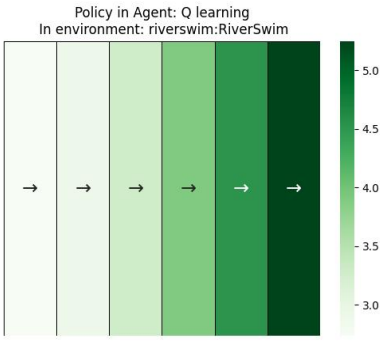

Figure 49: Q-learning with random Q-table.

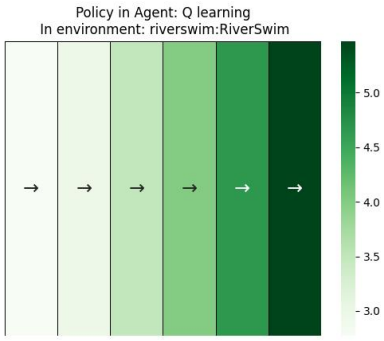

Figure 50: Q-learning with ones Q-table.

Figure 51: Double Q-learning with random Q-table.

Figure 52: Double Q-learning with ones Q-table.

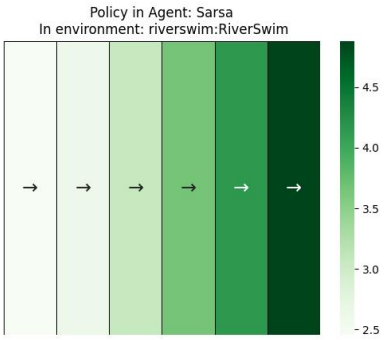

Figure 53: SARSA with random Q-table.

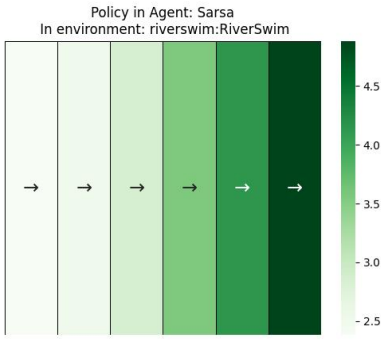

Figure 54: SARSA with ones Q-table.

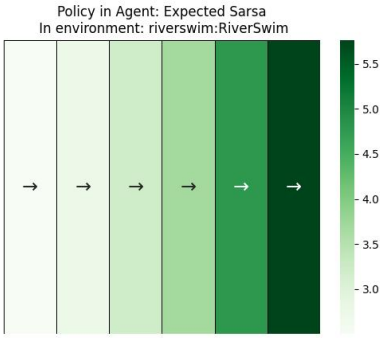

Figure 55: Expected SARSA with random Q-table.

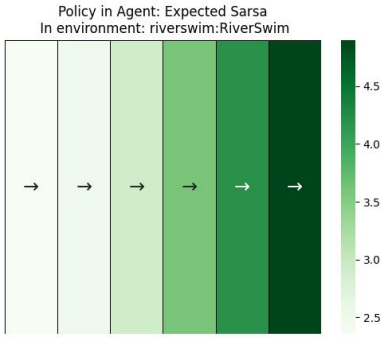

Figure 56: Expected SARSA with ones Q-table.

There is no significant differences for River Swim between policies in 4 agents whether how initialization of Q-tables, they are all the same, and optimal.

4.3 Affect exploration

Initializing Q-tables optimistically has two effects on exploration; on the one hand, initially high Q-tables allow agents to explore different actions and reach different states, motivating them to obtain potentially higher future rewards. However, on the other hand, this may also discourage the agent from exploring actions beyond the initial high value, i.e., the agent learns from the reward and updates the Q table, it will focus more on the learned information and may explore less over time.

5 Discussions

5.1 SARSA vs Q-learning

The Sarsa algorithm is similar to Q-learning in that both Sarsa and Q-learning have a Q table. And both algorithms use epsilon-greedy to select actions. The difference is that SARSA selects the action that follows the current policy and updates its Q value, while Q-learning selects the more greedy action that provides the maximum Q value for the state. We can see from the images of the moving average reward of the two agents in Figure 57 and 58 that the 95% confidence interval of SARSA is smaller than the 95% confidence interval of Q-learning in the FrozenLake environment in preliminary period, which is because Sarsa has chosen a conservative strategy. SARSA has planned actions for the future when updating Q values, and is more sensitive to errors and deaths, so the difference in moving average reward obtained from 5 runs is not significant. Q-learning is a reckless, bold and greedy algorithm that

does not care about death and mistakes, so the results of 5 runs are more random and the difference may be large.

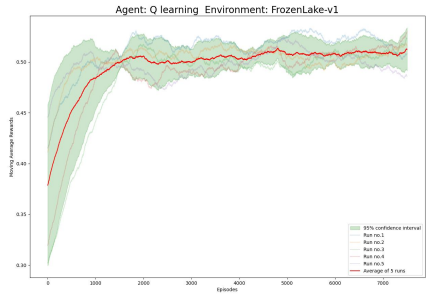

Figure 57: Q-learning in Frozen Lake.

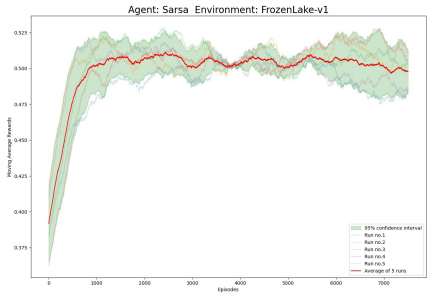

Figure 58: SARSA in Frozen Lake.

5.2 Q-learning vs Double Q-learning

The key idea in Double Q-learning is to mitigate the overestimation bias present in Q-learning. By decoupling the action selection and value estimation, it provides a more accurate estimation of the Q-values and can lead to improved policy convergence. However, in environment Frozen Lake, this reduction in overestimation may result in underestimation of the true values. As a result, in Double q learning, the learned Q-values may be more conservative, leading to lower rewards during training and potentially slower convergence. What' more, because of the random switching between the two Q tables, this randomness leads to greater fluctuations in the rewards obtained during the training period, so there is some variation in the curves of our 5 runs in Double q learning algorithm.

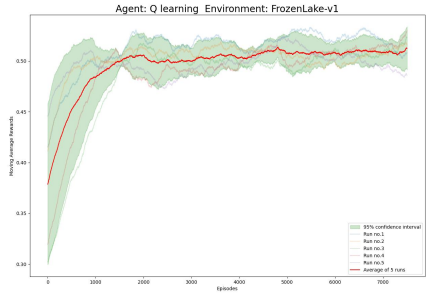

Figure 59: Q-learning in Frozen Lake.

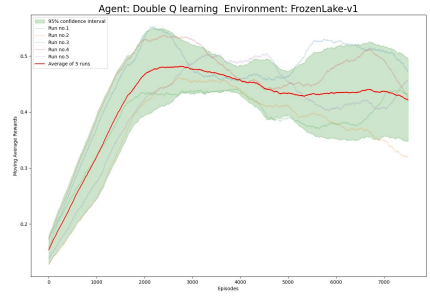

Figure 60: Double Q-learning in Frozen Lake.

5.3 SARSA vs Expected SARSA

The main difference between SARSA and Expected SARSA is how they estimate the value of the next state-action pair. SARSA uses the actual next action taken based on the epsilon-greedy strategy, while Expected SARSA considers the expected values of all possible actions.

Since we had set $is\_filppery = True$ in the Frozen Lake environment, this increases the randomness of the action. It is visible from Figure 61 and Figure 62 that both SARSA and Expected SARSA show much more noise for their rewards over episodes, indicated by the 95% confidence interval.

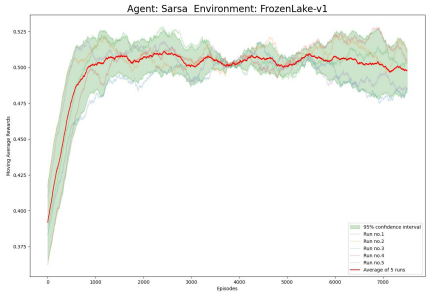

Figure 61: SARSA in Frozen Lake.

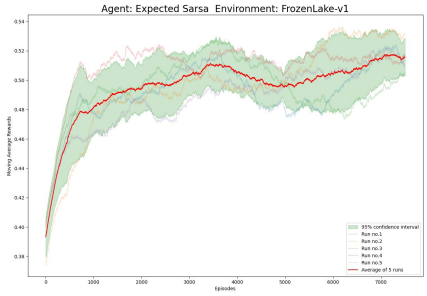

Figure 62: Expected SARSA in Frozen Lake.

Let's see how they performed when it is a deterministic Frozen Lake environment with setting $is\_filppery = False$:

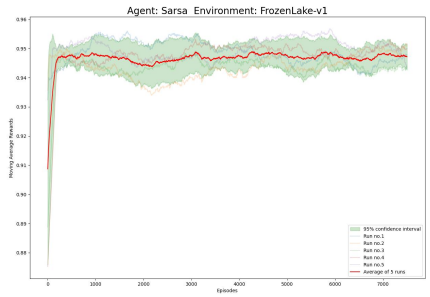

Figure 63: SARSA in Frozen Lake. Non-slippery

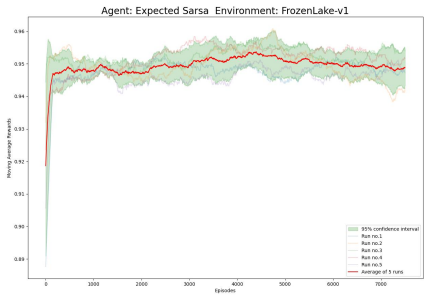

Figure 64: Expected SARSA in Frozen Lake. Non-slippery

SARSA more often samples noisy actions to update its Q tables, while Expected SARSA eliminates variance and causes much more stable results since it always consider all possible actions when updating Q tables.

6 Appendix - for non-slippery Frozen Lake

Policies can be easily detected as an optimal policy by the visualized matrix follows.

6.1 Q-learning

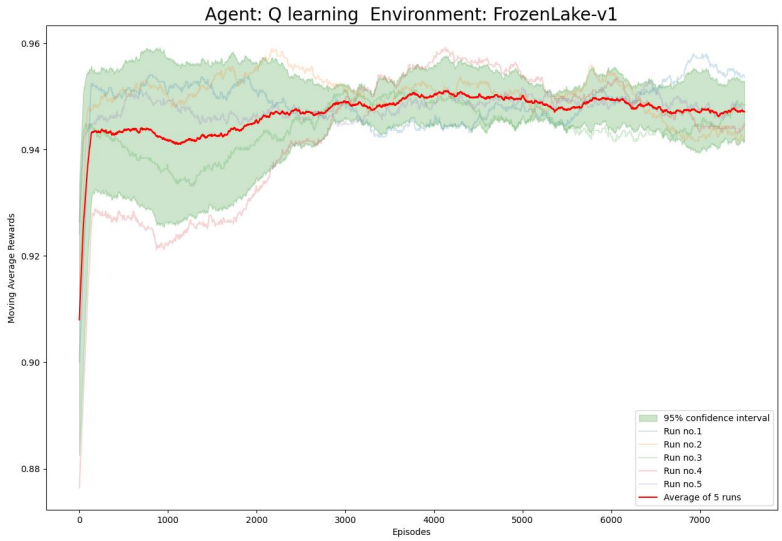

Figure 65: Q-learning in Frozen Lake.

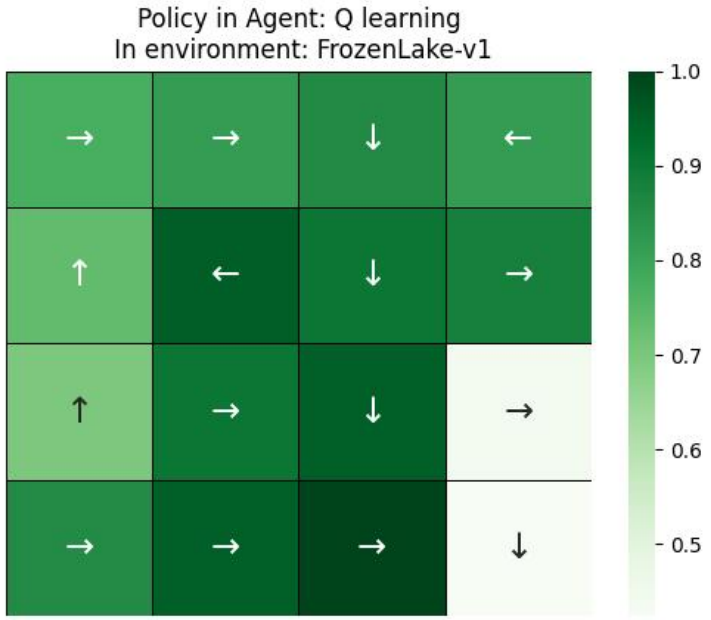

Figure 66: Q-learning in Frozen Lake.

6.2 Double Q-learning

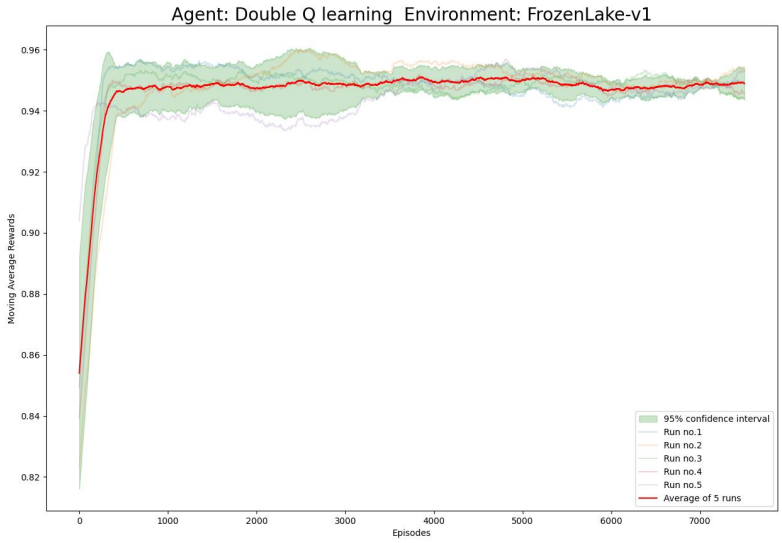

Figure 67: Double Q-learning in Frozen Lake.

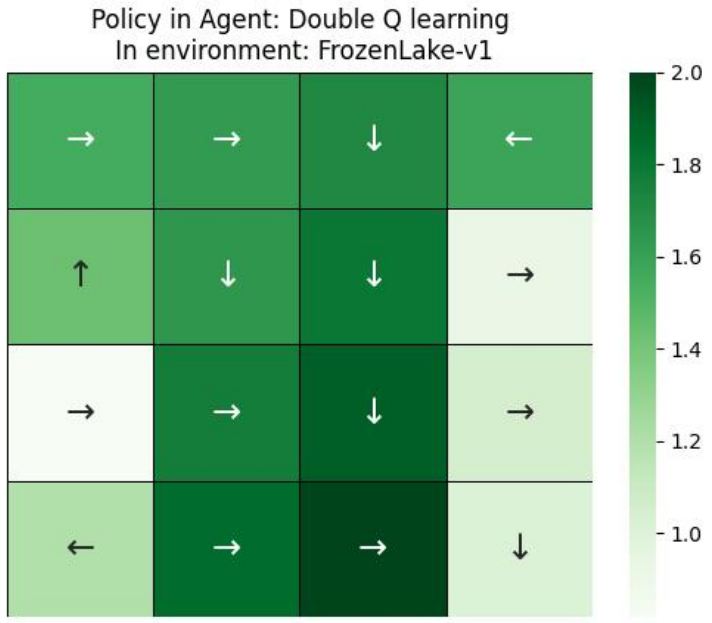

Figure 68: Double Q-learning in Frozen Lake.

6.3 SARSA

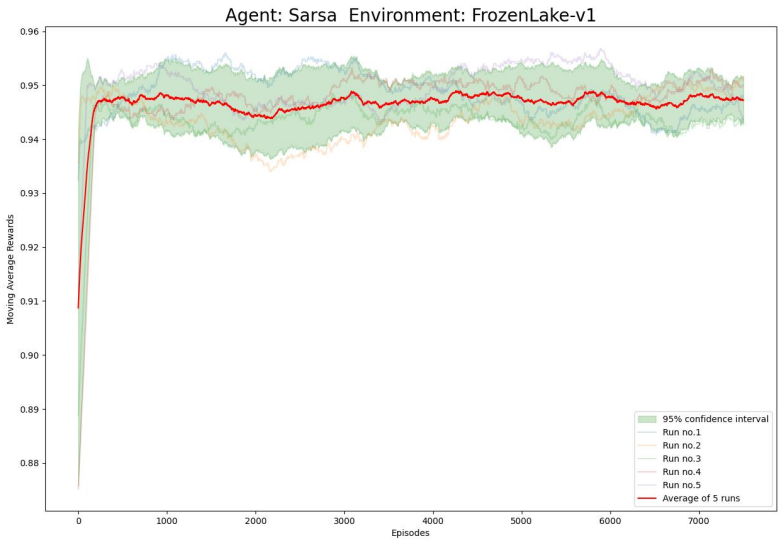

Figure 69: SARSA in Frozen Lake.

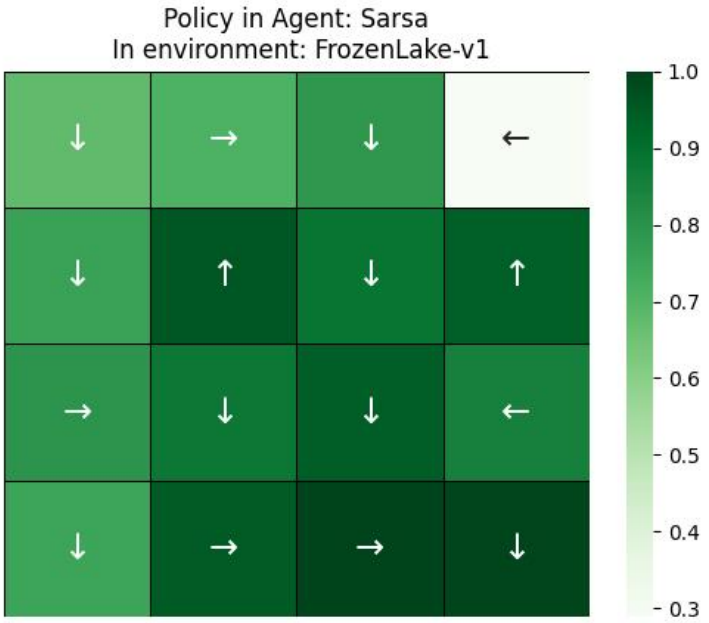

Figure 70: SARSA in Frozen Lake.

6.4 Expected SARSA

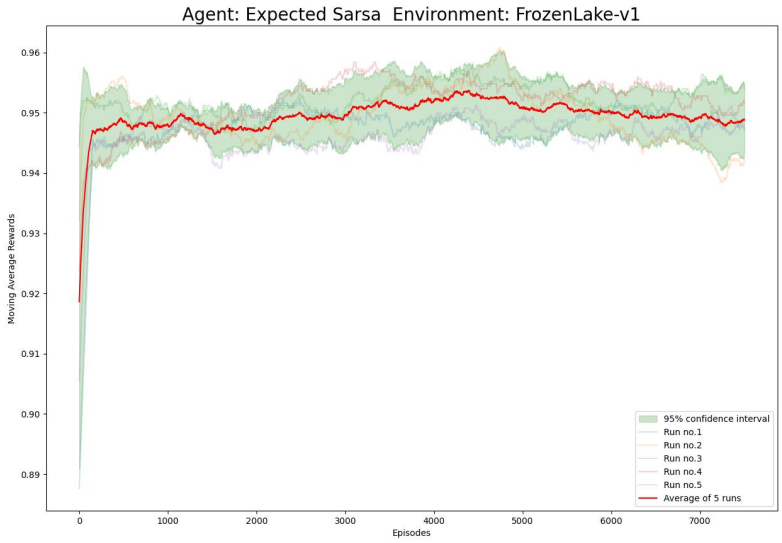

Figure 71: Expected SARSA in Frozen Lake.

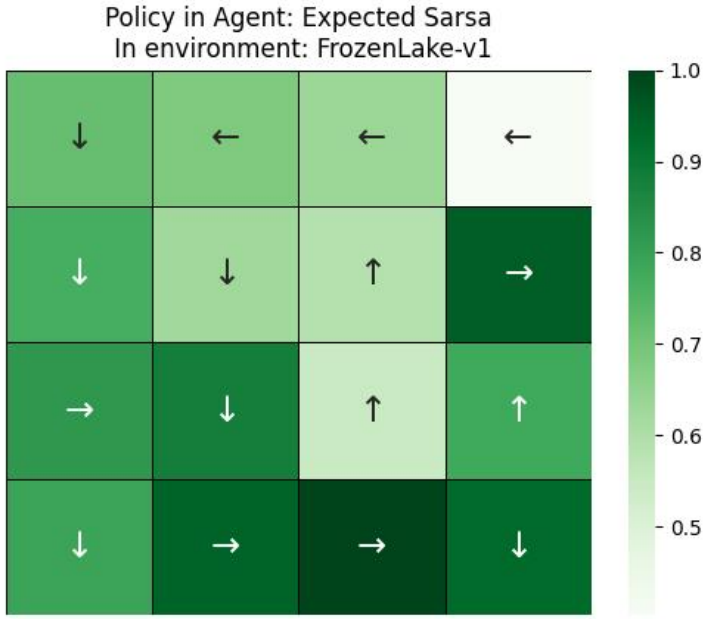

Figure 72: Expected SARSA in Frozen Lake.