Memorial University

REINFORCEMENT LEARNING:

ASSIGNMENT 3

(Spring 2025)

by

Hoang Phi Yen Duong - 202396817

Tinh Thanh Bui - 202398612

(PhD students)

St. John's, August 03, 2025

Outline

Code is available on GitHub: Click here

SIMULATION RESULTS and ANALYSIS:

In this assignment, we consider a 5 x 5 gridworld problem shown in Fig. 1. The gridworld environment consists of 25 cells (possible states). An agent can take a step up, down, left, or right. The agent gets a reward of -20 and goes back to the start position at (4, 0) if the agent moves to any red state. Meanwhile, the agent get a reward of -1 if the agent moves between any two other states or tries to step off the grid. The black squares are terminal states. Two algorithms including Sarsa and Q-learning algorithms with epsilon-greedy action selection are used for comparison.

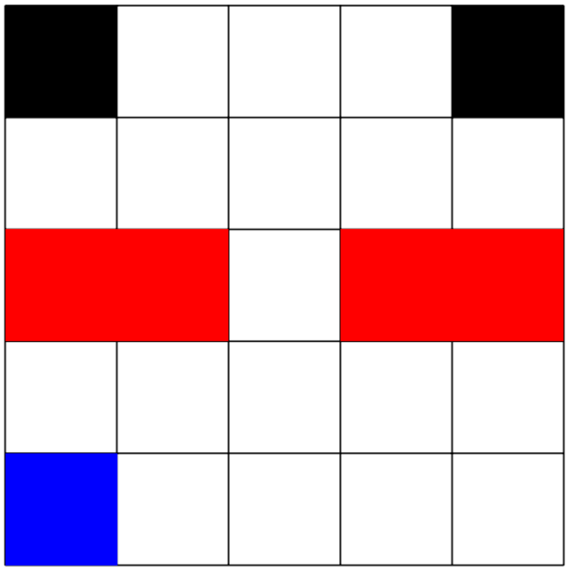

Fig. 1. *A simple 5 x 5 gridworld problem*

Set-up parameters: In the simulation, the values of learning rate, discount factor, and epsilon applied for both Sarsa and Q-learning algorithms are as follows

- Learning rate (α) is **0.4**. The given environment is relatively simple (small 5x5 grid). A moderate learning rate (i.e. 0.4) works best because the agent quickly learns without overshooting.
- Discount factor (γ) is **0.95**. This relatively high discount factor encourages the agent to value future rewards strongly, ensuring it learns to avoid the red states (highly negative) to reach terminal black states.
- Greedy action selection (ε) is **0.2**. This exploration rate ensures the agent explores sufficiently and avoid non-stop episodes if the step-off action is operated following the current policy.

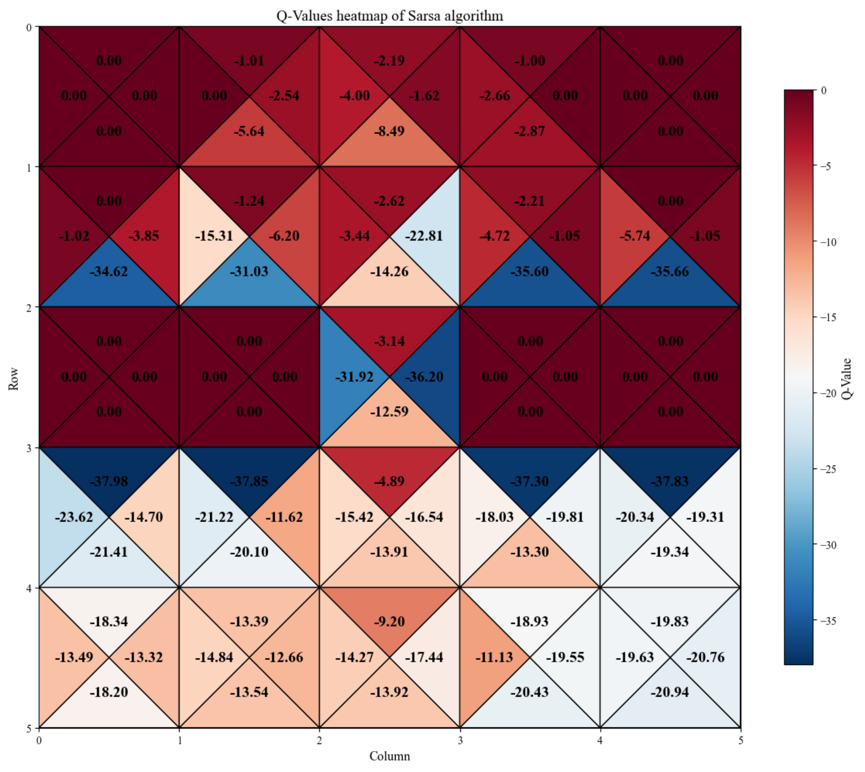

Fig. 2. *Q-value heatmap of Sarsa algorithm*

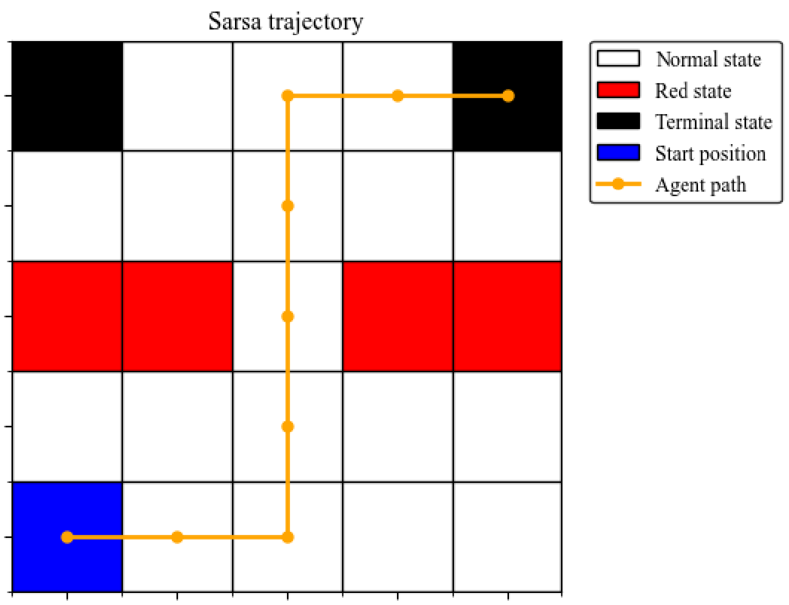

Fig. 3. *Agent's trajectory using the policy learned by Sarsa algorithm*

Sarsa algorithm: The Q-value heatmap and the trajectory of the agent using the policy learned by Sarsa algorithm are shown in Fig. 2 and Fig. 3.

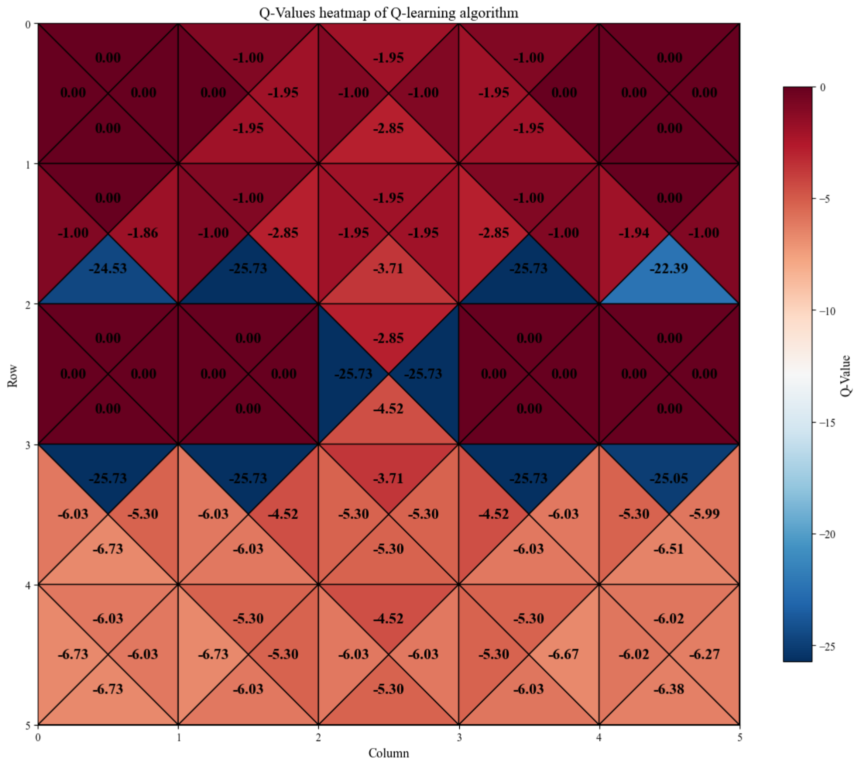

Fig. 4. *Q-value heatmap of Q-learning algorithm*

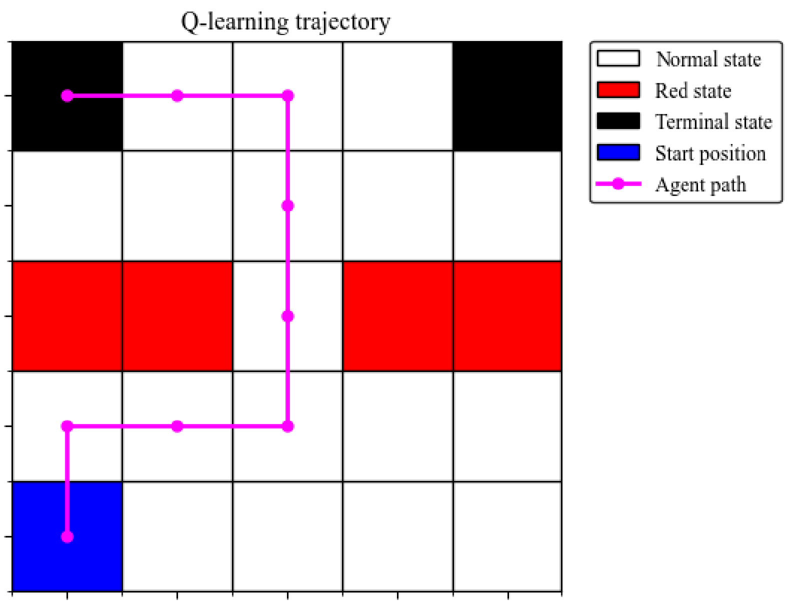

Fig. 5. *Agent's trajectory using the policy learned by Q-learning algorithm*

Q-learning algorithm: The Q-value heatmap and the trajectory of the agent using the policy learned by Q-learning algorithm are shown in Fig. 4 and Fig. 5. Additionally, the total rewards of these two algorithms within 20000 episodes is described in Fig. 6.

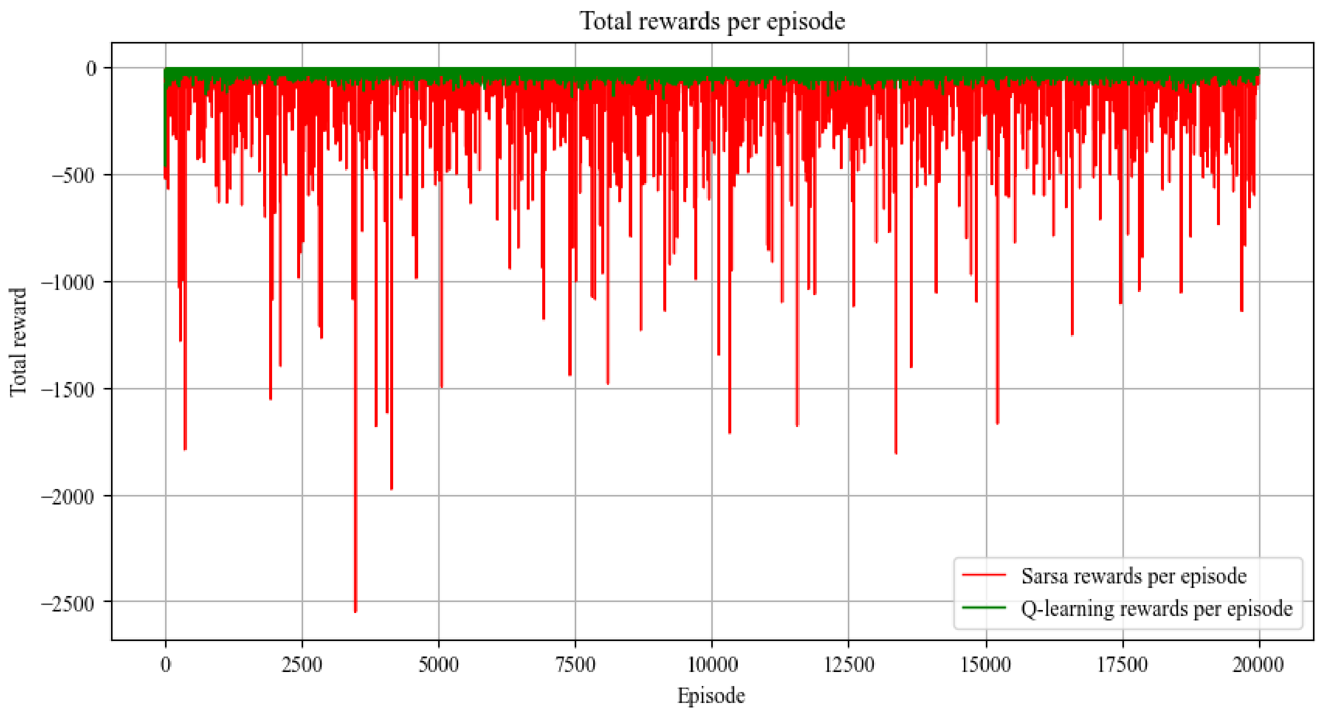

Fig. 6. *The comparison on total rewards of Sarsa and Q-learning algorithms per episode*

Are they different or similar? Why or why not?

Regarding the **Sarsa algorithm**, the trajectory of the agent in Fig. 3 proves the efficiency of the algorithm in finding the optimal path from the starting blue square to the terminal black squares at the top of the grid. Due to the on-policy characteristic, Sarsa tends to explore a route using the current policy and update this policy. The Sarsa algorithm considers the agent's actual next action at the next state. In detail, to pass through the red squares and reach the terminal black squares, the only option is to go through square (2, 2). From starting blue square (4, 0), the agent can go to adjacent squares (3, 0) and (4, 1). However, the policy learned by Sarsa recommends going to square (4, 1) since square (3, 0) has a higher chance of moving to red squares. At square (4, 0), action *up* has the value of -18.34 while action *right* has the value of -13.32, as shown in Fig. 2. In addition, significant fluctuations in the total rewards of the Sarsa algorithm depicted in Fig. 6 are caused by frequent exploratory phases that encounter high penalty red squares of -20. However, over a high number of learning episodes, the sum of rewards is improved as the policy becomes more refined to avoid red squares. This gradual improvement, despite ongoing fluctuations, highlights Sarsa's exploratory nature and its learning strategy of continuously adjusting the policy according to the specific actions experienced in each episode.

In terms of the **Q-learning** algorithm, the optimal trajectory shown in Fig. 5 is more direct and more stable since there is less exploration in the algorithm by taking the maximum of the Q-value of the next state into account. For example, at square (4, 0) in Fig. 4, actions *up* and *right* have the same Q-value of -6.03 since the algorithm only considers the maximum value of next states (3, 0) and (4, 1) (both are -5.3). The outcome is expected given the off-policy nature of the Q-learning algorithm, which prioritizes maximizing future rewards regardless of the current policy's chosen actions. As a result, the agent quickly finds ways to bypass the penalty red states and has an optimal trajectory. The total rewards per episode of Q-learning algorithm shown in Fig. 6 denoted by green line is more statble compared to the one of Sarsa algorithm. The rewards quickly converged to less negative values, demonstrating the algorithm's efficiency in learning the optimal policy.

The trajectories learned by Sarsa and Q-learning algorithms in the given gridworld problem exhibit both similarities and differences, primarily caused by their on-policy and off-policy natures, respectively. With the similarities, both algorithms employ ε-greedy action selection to have a balance in exploration and exploitation to navigate the agent to terminal states and avoid penalty states. Using both optimal policies, the agent needs 8 steps to successfully reach the terminal states, as shown in Fig. 3 and Fig. 5. However, the Sarsa algorithm's policy tends to explore more paths and has greater initial variability in the total rewards since it updates Q-values based on the actual actions following the current policy. In contrast, with the off-policy Q-learning algorithm, the optimal policy navigates the agent to a more direct path to the goal and has quicker convergence in the value of total rewards since it updates Q-values using the maximum future Q-values regardless of the next action. Thus, the Sarsa algorithm tends to avoid near-risky states while the Q-learning algorithm tends to identify optimal paths quicker if exploration is sufficient.

How does the sum of rewards over an episode behaves for each of these two methods?

Additionally, the total rewards per episode for Sarsa fluctuate significantly at the beginning due to high exploration, then gradually improve, but the variability still persists due to the ongoing balance of exploration and exploitation. Meanwhile, the total rewards of the Q-learning algorithm are more stable, indicating a faster and more efficient learning process by focusing on maximizing the future rewards and having a quicker policy convergence.

REFERENCES

[1] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", vol. 1, no. 1. Cambridge: MIT press, 1998.

[2] https://github.com/BY571/Medium_Code_Examples. Accessed on 15th of July 2025.