IRSTI 28.23.29 https://doi.org/10.26577/jpcsit2023v1i4a8

Ye.A. Amanbay, N. Azatbekuly*, A.A. Mukhanbet

Al-Farabi Kazakh National University, Almaty, Kazakhstan
*e-mail: nurtugang17@gmail.com

IMPLEMENTATION AND COMPARISON OF REINFORCEMENT LEARNING ALGORITHMS FOR SOLVING THE PROBLEM OF FINDING A PATH IN A 2D MATRIX

Abstract. The topic of this paper is the study of two reinforcement learning algorithms, SARSA and Q-Learning. Reinforcement learning is generating significant interest due to its potential applications in various domains such as robotics, gaming, optimization, etc. In addition, reinforcement learning is an interesting object of research from the perspective of theory and practice, as it is related to concepts such as exploration and use, learning and planning, consistency, and stabilization, etc. SARSA and Q-Learning are two of the most well-known and widely used reinforcement learning algorithms, which are based on the evaluation of the value function of states and actions. The aim of this paper is to study the learning characteristics of these algorithms in different scenarios of agent's interaction with the environment. To this end, experiments were conducted in which the agent had to find an optimal path in a 2D matrix containing walls to reach the final position safely. The results showed that SARSA was on average 28.3% faster than Q-Learning.

Key words: SARSA, Q-Leaning, Reinforcement Learning.

1 Introduction

Reinforcement learning (RL) [1] is a task faced by an agent that must learn behavior through trial-and-error interactions with the environment to achieve a goal. There are three main classes of methods for solving the OP problem: dynamic programming, Monte Carlo methods, and temporal difference (TD) learning. Reinforcement learning, which is a field of machine learning, is becoming one of the main tools of computational intelligence as a technique in which computers make their own choices in each environment without historical or labeled data [2]. By interacting with the environment, the agent predicts the optimal decision and continuously develops and learns the optimal policy based on the value function [3]. In [4], TD methods update estimates partly based on other estimates. They are trained to build hypotheses on hypotheses. Since TD-learning is a combination of Monte Carlo and dynamic programming ideas, much attention has recently been paid to the study of TD-learning [5,6]. It is precisely TD-learning that formed the basis of this research to implement SARSA and Q-Learning algorithms. It is stated in [4] that one of the early breakthroughs in reinforcement learning was the development of a TD algorithm for split strategy control, known as Q-Learning. And a classic on-policy TD algorithm called SARSA (State Action Reward State Action) [7] is considered critical to the success of OD [8]. In [9], it is found that SARSA and Q-learning have different performance depending on the learning environment. From this, it can be hypothesized that in the task of path finding in 2D matrix, SARSA may be a better algorithm option.

The paper primarily seeks for an agent to maximize total rewards within a virtual environment through sequential actions. Specifically focusing on SARSA and Q-Learning algorithms in a 2D matrix, the objective is for the agent to navigate to the target while navigating obstacles. This scenario is an illustrative example of the application of reinforcement learning in solving the problem of optimal behavior under resource constraints and possible complications. Illustrating the practical use of reinforcement learning in addressing decision-making challenges, the paper analyzes algorithm performance, aiming to highlight their distinct strengths in solving this problem.

2 Materials and methods of research

2.1 Time-Difference Forecasting

In time-difference (TD) based forecasting, as in the Monte Carlo method, the value of states is sought. However, in the Monte Carlo method, the value function is estimated by simple mean

reversion, whereas in TD-learning, the value of the current state is updated by the current state. In TD-learning, the so-called time-difference based update rule is used to update the state value, it is represented in formula (1):

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)), \quad (1)$$

where, *V(s)* – value of the previous state, *α* – learning rate, *r* – reward, *γ* – correction factor, *V(s')* – value of the current state.

2.2 SARSA algorithm

Rummery and Niranjan [10] proposed a modified Q-learning algorithm called SARSA. Unlike the traditional Q-learning algorithm, SARSA is a policy enabled TD algorithm whose updates are policy dependent.

SARSA is one of the most popular reinforcement learning algorithms used to train agents to interact with the environment. It gets its name from the sequence of actions and states in a task: State-Action-Reward-State-Action.

SARSA updates Q-value estimates for state-action pairs based on the rewards received and the agent's experience. The updates of Q-values are calculated using formula (2):

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - \\ -Q(s,a)), \quad (2)$$

where, α is the learning rate, a'- the action chosen by the epsilon greedy strategy (ε> 0), γ – correction factor.

The process of updating Q-values in SARSA is as follows:

1. The agent starts in state $'s'$ and chooses action $'a'$ according to its strategy;

2. The agent performs action $'a'$ and moves to a new state $'s'$ while receiving reward $'r'$;

3. The agent chooses a new action $'a'$, again according to its strategy.

A block diagram of the SARSA algorithm is shown in (Fig.1):

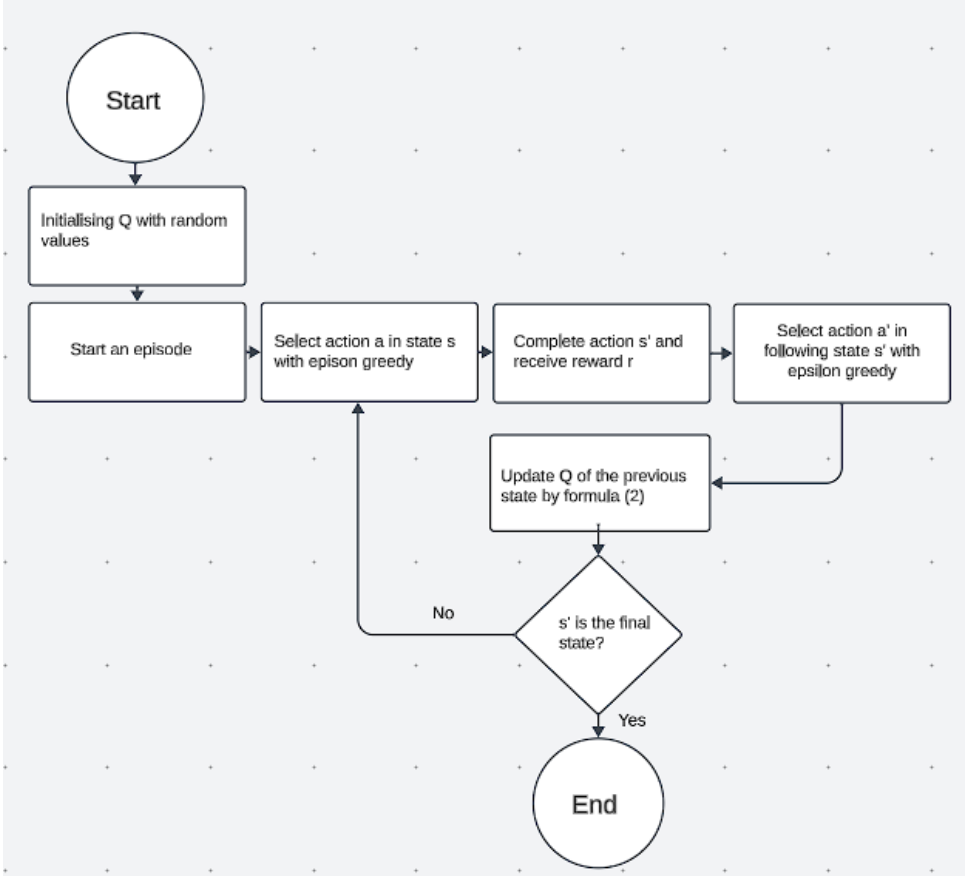

Figure 1 – Block diagram of the SARSA algorithm

1.3 Q-learning algorithm

Q-learning is another important reinforcement learning algorithm that is also used to find the optimal strategy in a path planning task. Unlike SARSA, Q-learning estimates the maximum expected Q-value for each state-action pair.

The process of updating Q-values in Q-learning is as follows:

1. The agent starts in state 's' and selects action 'a' according to its strategy.

2. The agent performs action 'a' and moves to a new state 's″ while receiving reward 'r'.

The value of Q is updated according to formula (3):

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma maxQ(s',a') - Q(s,a)), \quad (3)$$

where, α is the learning rate, a'- the action chosen by the epsilon greedy strategy (ε> 0), γ – correction factor.

Q-Learning evaluates the optimal strategy by maximizing Q-values for each state and action. This allows the agent to converge to the optimal strategy faster but may require more computational power and may be more sensitive to noise in the data.

A block diagram of the Q-Learning algorithm is shown in (Figure 2).

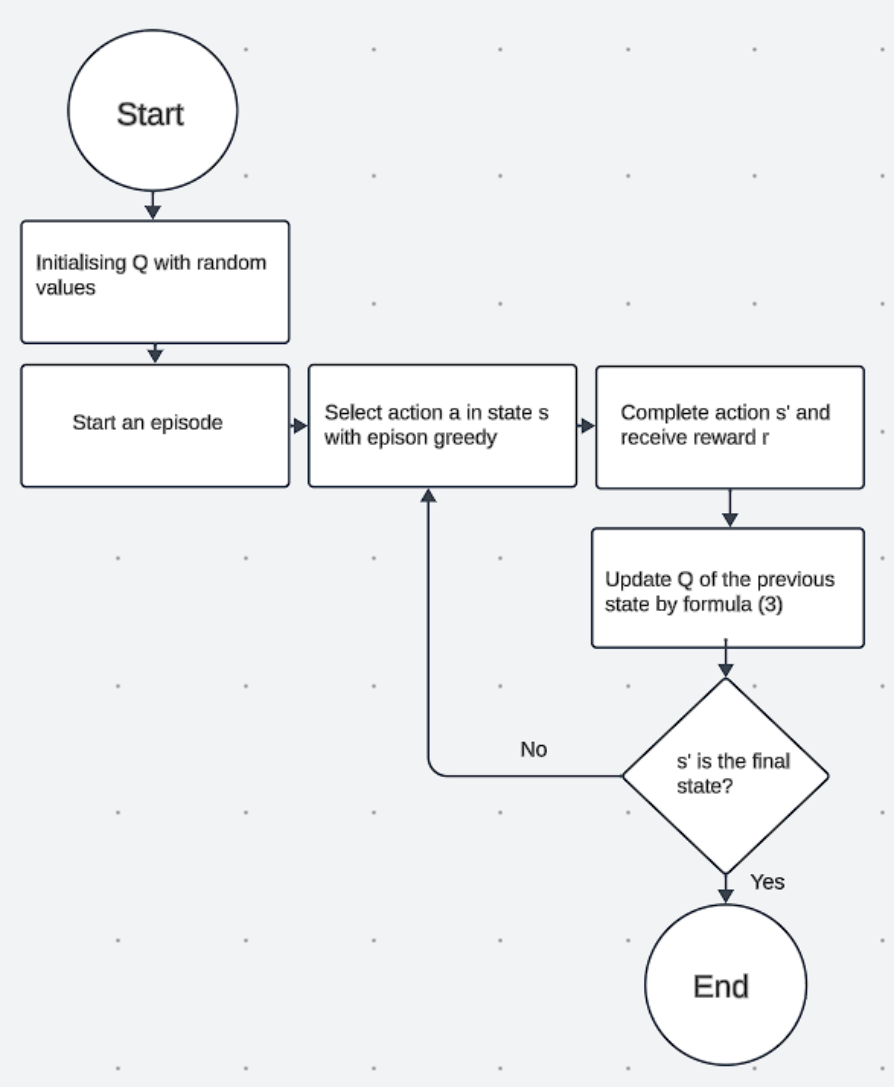

Figure 2 – Block diagram of the Q-Learning algorithm

3 Results of research

We run SARSA and Q-Learning in a 2D matrix with dimensions 10 by 10 and 20 by 20 (Figure 3). A small negative reward, -1, is given for each move, and a large reward, 100, is given for reaching the end point.

The number of episodes is equal to 1000, learning rate $\alpha = 0.1$, $\gamma = 0.9$, $\varepsilon = 0.01$

In a comparison of plots of the number of steps per episode on a 10 by 10 matrix, SARSA is faster and finds the optimal path in the least number of episodes than Q – Learning (Fig.4).

When increasing the size of the 2D matrix to 20 by 20, both algorithms needed more steps to reach the target point, however, the SARSA algorithm was able to reach the target point faster by minimizing the number of steps with each episode (Figure 5).

As can be seen in (Fig.6), the SARSA algorithm minimizes the number of collisions to zero with each episode. This is due to the fact that SARSA is an on-policy algorithm, i.e., the agent updates Q-values given its own actions, according to the current strategy. This makes the agent more cautious by choosing actions that are already known to be safe. While Q-Learning is an off-policy algorithm, based on this the agent updates Q-values based on an optimal strategy, which may be more exploratory and lead to the selection of less known but potentially dangerous actions.

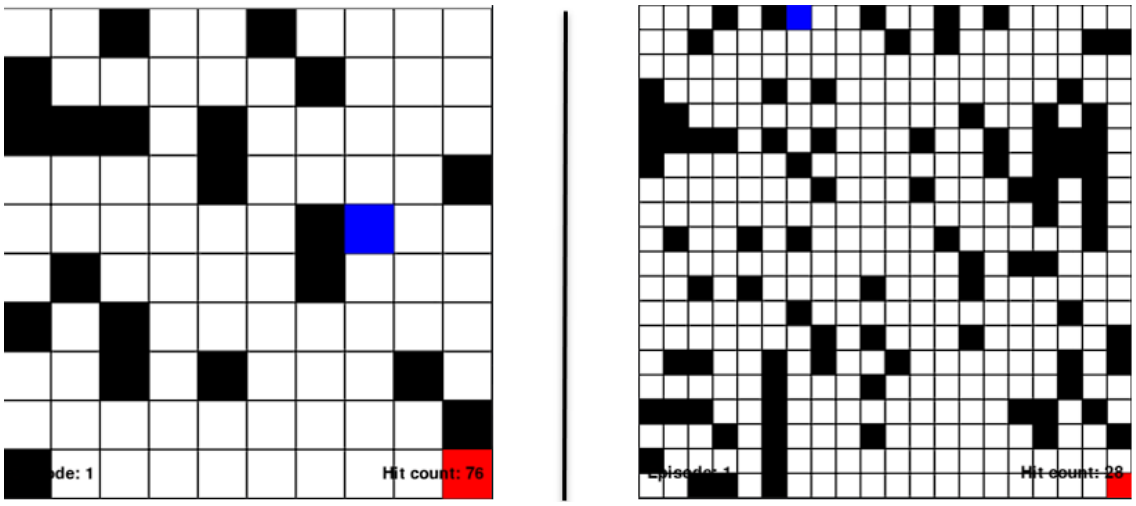

Figure 3 – 10 by 10 and 20 by 20 matrices, where (Blue square is agent, red square is target position)

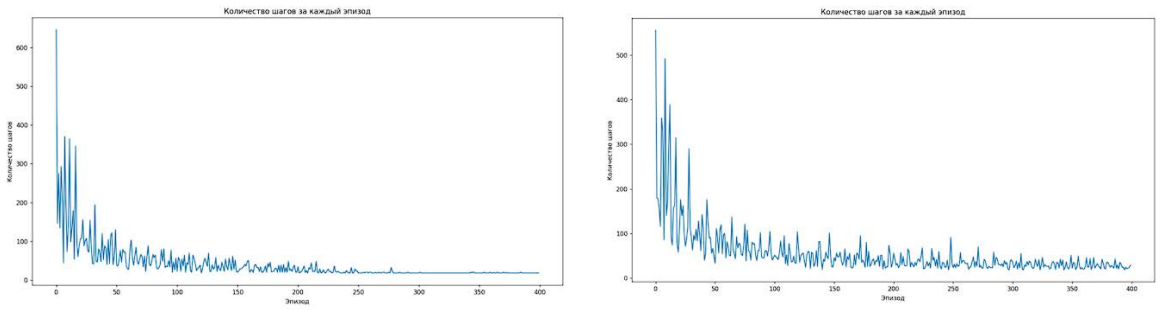

Figure 4 – Graph of the number of steps per episode of the SARSA (left) and Q-Learning (right) algorithms

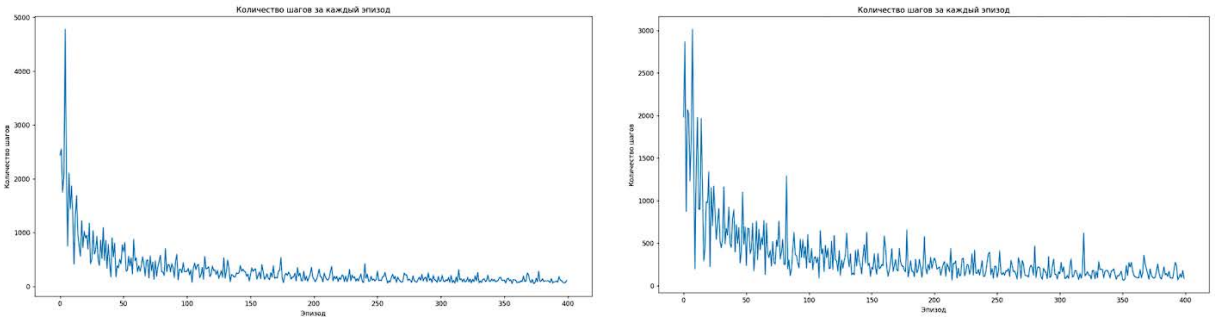

Figure 5 – Graph of the number of steps per episode of the SARSA (left) and Q-Learning (right) algorithms

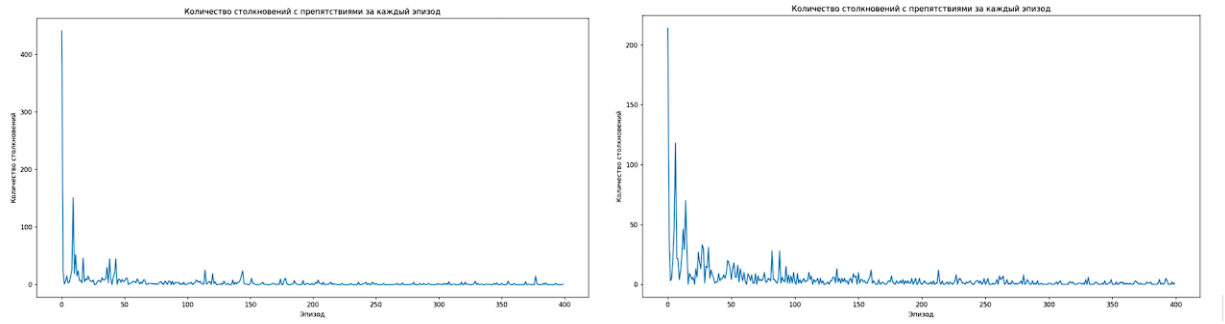

Figure 6 – Graph of the number of obstacle collisions per episode of the SARSA (left) and Q-Learning (right) algorithms, in a 20 by 20 environment

From the results of these experiments, it is evident that the SARSA algorithm demonstrated better performance in several key aspects compared to the Q-learning algorithm. Specifically, SARSA showed significant advantages in the following parameters:

1. Number of collisions: SARSA also demonstrated a lower number of collisions with obstacles. This is due to the fact that SARSA considers recent actions and avoids repeated errors, which makes it more robust in collision avoidance.

2. Number of steps per episode: SARSA often required fewer steps to reach a goal compared to Q-learning. This indicates its ability to more efficiently select actions that lead to successful task completion.

In case the environment has a high noise level or is unstable, SARSA may be preferable, as it updates its strategy based on the actual actions taken by the agent. In tasks with large state and action space, Q-learning may be more effective as it can learn based on the value of the best action. However, when emphasizing the criteria of optimal path length, number of collisions, and number of steps per episode, SARSA appears to be a more appropriate algorithm.

4 Conclusion

In this paper, two popular reinforcement learning algorithms, SARSA and Q-Learning, were implemented in the context of the 2D pathfinding task. Experimental results showed that SARSA was more efficient in reaching the optimal path, converging to it 28.3% faster compared to Q-Learning.

This improvement in the convergence speed of SARSA represents a significant finding that emphasizes the advantages of this algorithm in solving a particular problem. The success of SARSA can be attributed to its ability to learn more efficiently in environments where obstacles are present.

The SARSA algorithm showed significant advantages over Q-Learning in several important characteristics such as the length of the optimal path, the number of collisions, and the number of steps per episode. SARSA proved to be more

efficient in finding optimal strategies for moving the agent, which is of practical importance in robotics and autonomous systems. The results suggest that SARSA is the preferred choice for the 2D matrix pathfinding problem, collision avoidance and step count optimization play an important role.

This analysis allowed us to gain a deep understanding of how different algorithms affect the robot's navigation performance.

This study emphasizes the importance of selecting the most appropriate algorithm for a particular task. In the experiments, SARSA showed advantages in environments where minimizing the length of the optimal path and avoiding collisions are critical factors. Despite the advantages of SARSA, Q-Learning remains an important tool, especially in partial observability environments where it can adapt to unexpected situations and find optimal solutions.

References

1. Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998.

2. Jordan, M. I., and T. M. Mitchell. 2015. “Machine Learning: Trends, Perspectives, and Prospects.” Science. American Association for the Advancement of Science (AAAS). https://doi.org/10.1126/science.aaa8415.

3. Tesauro, Gerald. 1995. “Temporal Difference Learning and TD-Gammon.” Communications of the ACM. Association for Computing Machinery (ACM). https://doi.org/10.1145/203330.203343.

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

5. Sutton, Richard S. "Temporal Credit Assignment in Reinforcement Learning." Ph.D. thesis, University of Massachusetts, 1984.

6. Sutton, Richard S. 1988. “Learning to Predict by the Methods of Temporal Differences.” Machine Learning. Springer Science and Business Media LLC. https://doi.org/10.1007/bf00115009.

7. Sutton, Richard. “Generalization in reinforcement learning: Successful examples using sparse coarse coding Advances in Neural Information Processing Systems, edited by David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, pp. 1038-1044. The MIT Press, 1996.

8.Wang, Yin-Hao, Tzuu-Hseng S. Li, and Chih-Jui Lin. 2013. “Backward Q-Learning: The Combination of Sarsa Algorithm and Q-Learning.” Engineering Applications of Artificial Intelligence. Elsevier BV. https://doi.org/10.1016/j.engappai.2013.06.016.

9. Sutton, Richard. “Open theoretical questions in reinforcement learning”. Proceedings of EuroCOLT’99, 1999

10. Rummery, G.A., and Niranjan, M. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Cambridge: Cambridge University Engineering Department, 1994.