International Journal of Autonomous Robots and Intelligent Systems
Vol. 1, No. 2, December 2025 [143-156]

A Comparative Study of On-Policy and Off-Policy Tabular RL in the Taxi-v3 Path-Planning Task

Muqri Muazam Zainal[1], Kamarulzaman Kamarudin[1*], Nasr Abdalmanan[1], Wan Rahiman[2], M Aizat Abu Bakar[1] and M Rizal Manan[1]

[1]Mechatronic Department, Faculty of Electrical Engineering & Technology, Universiti Malaysia Perlis, 02600 Arau, Perlis
[2]School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Pulau Pinang, Malaysia

Received 27 October 2025, Revised 19 November 2025, Accepted 3 December 2025

ABSTRACT

Mobile robots are increasingly relied upon for navigation and exploration in unknown environments, where path planning is crucial. Reinforcement Learning (RL) algorithms, particularly Q-Learning (off-policy) and SARSA (on-policy), have proven effective for autonomous decision-making during path planning. This study presents a comparative analysis of both algorithms using the Taxi-v3 environment in OpenAI Gym, focusing on differences in policy behaviour and learning dynamics. Experiments were conducted using the Taxi environment in OpenAI Gym with learning rates from 0.1 to 0.5, across 10,000 to 50,000 episodes. Performance was evaluated based on convergence rate, cumulative reward, and path efficiency. Both algorithms converged within the first 10-20% of training episodes, with Q-learning converging 5-10% faster and accumulating fewer penalties due to its greedy strategy. Post-convergence analysis showed Q-learning required on average 9 steps using a direct, shortest-path approach, while SARSA required 13 steps following an exploration-type path. Q-learning is suitable for tasks requiring fast, shortest paths, while SARSA is preferred for exploration tasks with precautionary conditions.

Keywords: Reinforcement Learning**,** On-Policy, Off-Policy, Path Planning

1. INTRODUCTION

Reinforcement Learning (RL) has become one of the most powerful frameworks in artificial intelligence for solving sequential decision-making problems where an autonomous agent learns to interact with an environment to achieve long-term goals. In recent years, RL has been widely adopted in industrial and research applications, particularly in autonomous and mobile robot systems, to enhance productivity, minimize human intervention and reduce operation time for improved efficiency and performance [1]. By using a system of rewards and penalties, RL enables agents to learn optimal strategies for navigating complex and uncertain environments without explicit supervision. Among the many applications of RL, path planning for mobile robots remains a core domain, where an agent must determine the most efficient route to reach a target while minimizing costs such as time, distance or energy [2]. In such problems, learning an optimal policy that can balance exploration and exploitation is crucial for achieving high performance and adaptability.

RL algorithms can generally be categorized into two major classes, including on-policy and off-policy learning methods [3]. The difference between these two lies in estimating action-value. On-policy methods, such as State-Action-Reward-State-Action (SARSA), learn the value of the policy that the agent is currently following. This means that both learning and acting are based on the same behaviour policy, making the agent's updates dependent on the actual actions taken during

*kamarulzaman@unimap.edu.my

exploration [4][5]. In contrast, off-policy methods, such as Q-learning, learn the value of the optimal policy independently of the agent's current behaviour. Off-policy learning enables the agent to explore with one policy (behaviour policy) while learning the value of another (target policy), allowing it to converge to an optimal solution more aggressively [6][7].

Both approaches have unique advantages and limitations. Q-learning, being off-policy, is often more sample-efficient and can converge faster to an optimal policy in deterministic or static environments. However, its tendency to learn based on the best possible action, regardless of whether that action was actually taken, can lead to instability or unsafe exploration in random or dynamic settings [8]. SARSA, on the other hand, tends to produce safer and more stable learning behaviour, as its updates reflect the consequences of actual actions taken under the current policy [9]. This difference becomes particularly relevant in navigation and path-planning tasks, where risky or suboptimal decisions can lead to repeated failures or inefficient routes.

This paper performs a comparative analysis between Q-learning and SARSA to evaluate each algorithm capabilities in solving path-planning problems using the Taxi-v3 environment, a well-known benchmark from OpenAI Gym. Taxi-v3 environment simulates a discrete grid world where an autonomous taxi agent must navigate between locations to pick up and drop off passengers efficiently. This simulated environment reflects mobile robot navigation in unknown environments, where obstacle positions and free spaces are initially unknown to the agent and discovered only through exploration and exploitation. The environment's discrete states, well-defined rewards and manageable complexity, make it ideal for examining learning behaviours, convergence rates and policy quality of different RL algorithms, providing a simplified yet insightful testbed for analysing RL performance in constrained navigation scenarios.

The primary objective of this research is to investigate how the learning type between on-policy and off-policy affects the efficiency, stability and effectiveness of policy learning in a structured environment. Despite the widespread application of both Q-learning and SARSA in path planning tasks, there remains a gap in understanding their comparative performance under identical experimental conditions. Specifically, this work addresses the following research gaps:

1. Limited empirical comparison of convergence rate and stability between off-policy and on-policy when applied in structured environments like Taxi-v3.
2. Insufficient analysis of policy optimality differences between on-policy and off-policy methods in terms of average cumulative reward and total steps required to reach the goal.

To address these gaps, both algorithms will be implemented under identical experimental conditions with the same hyperparameters, including learning rate, discount factor and exploration policy. The comparison will focus on key performance indicators including convergence speed, reward variance and final policy optimality. Through quantitative analysis and performance visualization, this study aims to provide a clear understanding of how off-policy and on-policy learning differ in their approach to acquire efficient navigation strategies.

Ultimately, the contribution of this work lies in offering empirical insights into the comparative strengths and weaknesses of SARSA and Q-learning in a path-planning context. By leveraging the Taxi-v3 environment as a controlled benchmark, this paper aims to highlight the practical trade-offs between exploration stability and learning efficiency that characterize on-policy and off-policy methods. These findings can serve as a foundation for future work on hybrid or adaptive RL strategies that combine the stability of on-policy learning with the efficiency of off-policy methods, particularly in robotic navigation and autonomous decision-making systems.

1.1 Research Background

RL enables an autonomous agent to learn an optimal strategy for interacting with an environment by maximizing cumulative rewards through trial and error. In this framework, the policy function serves as the core decision rule that determines the agent's actions based on the current state of the environment. As illustrated in Figure 1, the policy continuously improves through a feedback loop of actions, state transitions, and rewards, enabling the agent to refine its behaviour over time [10]. Two fundamental RL algorithms which are SARSA and Q-learning, represent the core distinction between on-policy and off-policy learning. The on-policy SARSA algorithm updates its action-value estimates based on the actual actions executed by the agent under the current policy, leading to safer but slower convergence. In contrast, the off-policy Q-learning algorithm updates its estimates toward the optimal greedy policy, which often accelerates learning but can result in unstable exploration or suboptimal paths in stochastic environments [11].

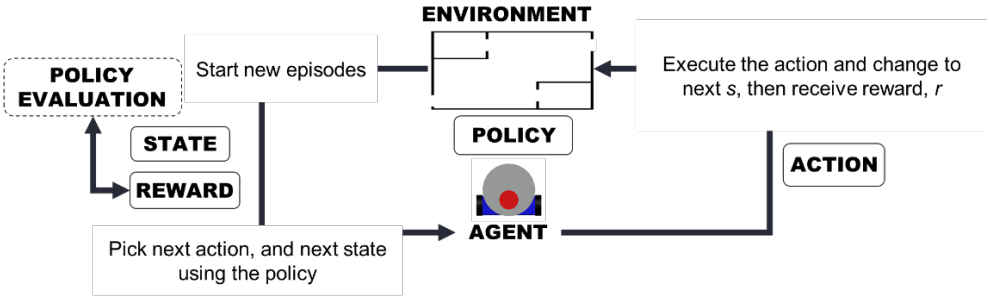


Figure 1. RL framework [19]

Several studies have compared these two approaches to understand differences in learning behaviour and policy outcomes. Zhong et al. found that SARSA produced conservative routes with stable convergence in the Cliff Walking environment, while Q-learning converged faster but exhibited riskier behaviour [12]. Imtiaz et al. reported that Q-learning achieved higher success rates (93%) than SARSA (80%) in a robotic pick-and-place task due to greater exploitation of high-reward actions [13]. In energy management, Ramesh et al. found SARSA maintained more conservative operational behaviour due to its on-policy nature [14]. However, most comparisons focus on other tasks like highlighted by these literatures, rather than handling path planning. These studies confirm that Q-learning tends to learn faster and more efficiently in deterministic environments, while SARSA provides smoother and safer learning dynamics in stochastic or dynamic contexts.

Another recent study by Abhinav Sharma et al. explored the effect of the discount factor on policy behaviour by extending both SARSA and Q-learning using a Transition-Dependent Discount (TDD) mechanism [15]. The study's simulations conducted in three benchmark environments, namely the Taxi Domain, Mountain Car and Cliff Walking demonstrated that Q-learning generally required fewer episodes to converge, whereas SARSA achieved higher cumulative rewards due to its on-policy learning updates [16][17][18]. In the Taxi environment which reflect path planning application, both algorithms were designed to minimize the total cost of reaching and dropping off passengers, with Q-learning converging faster but SARSA exhibiting steadier performance.

Although previous study provides valuable insights into the influence of discount factors on policy evolution, the investigation primarily emphasizes the mathematical effects of transition-dependent parameters rather than a direct comparison of the practical capabilities of on-policy and off-policy learning in path-planning applications. Consequently, this paper addresses this gap by experimentally analysing the performance of Q-learning and SARSA within the Taxi-v3 environment, focusing specifically on how each policy framework affects learning efficiency,

convergence stability, and planned path optimality. This comparative analysis aims to enhance the understanding of policy-driven performance variations between on-policy and off-policy reinforcement learning algorithms in path planning task.

2. MATERIAL AND METHODS

This section outlines the methodology used to compare SARSA (on-policy) and Q-learning (off-policy) algorithms in path-planning tasks within the Taxi-v3 environment. It covers the conceptual framework, hardware and software setup, and implementation of both algorithms under identical conditions. Performance was evaluated using convergence rate, cumulative reward and policy stability to understand the behavioural differences between the two learning strategies.

2.1 Concept on Comparing RL Policies

The comparison process examines reinforcement learning algorithms representing on-policy and off-policy methods. The work starts with hardware and software setup, followed by fundamental components of SARSA and Q-learning, including each algorithm policy frameworks and algorithmic equations, and experimental setup to ensure fair comparison. The second stage collects experimental data by simulating both algorithms under identical conditions. Key metrics include cumulative rewards and episodes required for convergence, serving as primary indicators of learning efficiency and policy performance. The final stage analyses the collected data through graphical and tabular presentations, followed by detailed evaluation of each algorithm's behaviour and effectiveness. The flowchart of this work is as shown in Figure 2.

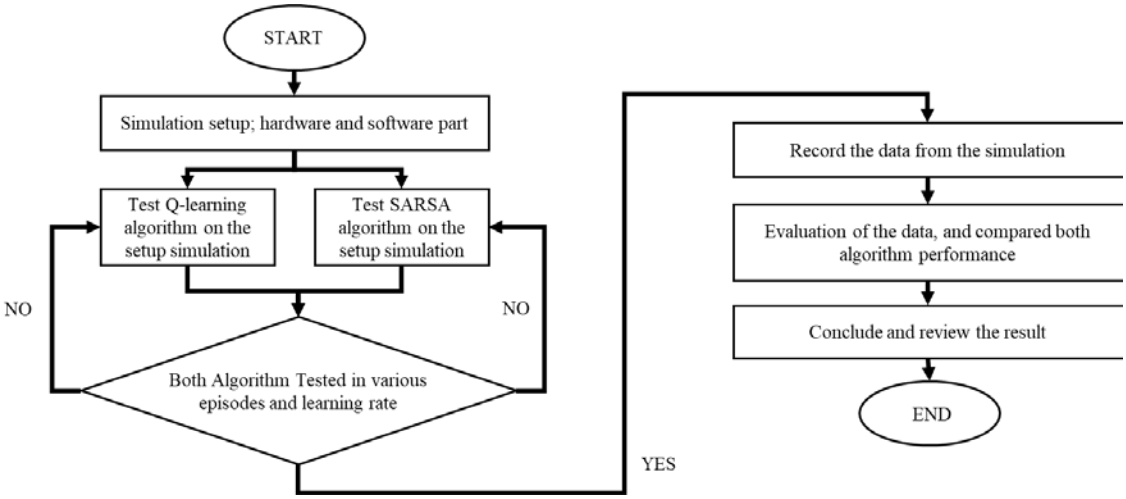


Figure 2. Flowchart of comparing RL policies

2.2 Hardware and Software Setup

The hardware components play a crucial role in this experiment by running simulations and recording data for both Q-learning and SARSA algorithms. To minimize hardware influence on simulation results, the selection criteria emphasized computational performance, processor type and graphics card capabilities.

The experiments were conducted using an Asus Vivobook Pro 15 laptop with an AMD Ryzen 9 5900H processor (6 cores, 12 threads), Nvidia GeForce RTX 3050 Studio GPU (6GB GDDR6), 16GB RAM and 500GB SSD. This configuration ensures fast simulation rendering and reduced

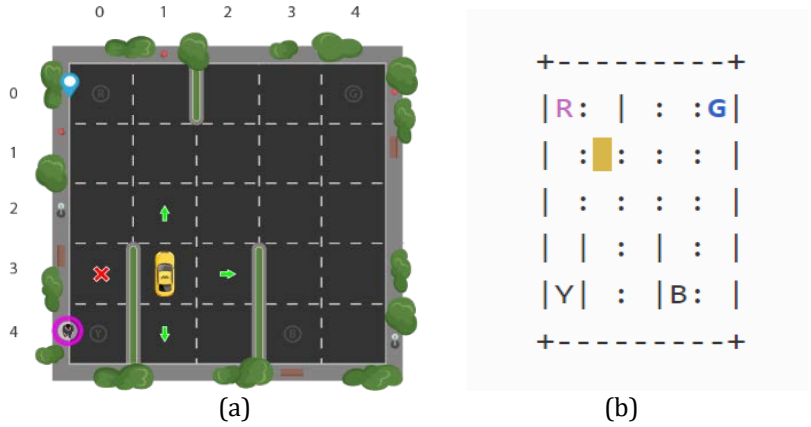
computation time for reinforcement learning training. The software integrates NumPy and OpenAI Gym libraries to simulate and compare Q-learning and SARSA policies. The Taxi Domain environment was selected as the simulation platform, featuring a taxi agent on a 5×5 grid with four designated locations (R, G, Y, B) and six possible actions.

OpenAI Gym provides a structured interface that supports reinforcement learning experiments by defining states, actions and reward functions for each environment. The built-in Taxi Domain environment offers all necessary components for implementing and testing RL algorithms, including state representations, valid action sets, and reward mechanisms that reflect agent performance [20], [21]. The taxi can perform up to six possible actions as in Equation (1):

$$action = (north, south, east, west, pickup, dropoff) \quad (1)$$

The task assigned to the taxi agent is to navigate to a target location, pick up a passenger and transport them to another designated location for drop-off. The state space and environment illustration used in this experiment are shown in Figure 3(a) [22]. The actual output is displayed in Figure 3(b), printed on the testing terminal. In the Taxi environment, the setup from the agent to the possible actions remains fixed, with only the algorithm used to solve the path planning task being modified.

Within the environment, obstacles are generated and the arrangement remains consistent across tests to evaluate the RL policies' capability to solve path planning tasks. The environment and generated data do not display the possible paths recorded by the agent; only the rewards and the taxi's capability to reach the target are shown. All rewards are stored in the Q-table, which displays the number of episodes set before starting the experiment. The number of episodes set for this experiment is 10,000.



(a) (b)

Figure 3. Illustration of: (a) Taxi environment, and (b) actual output, used in this experiment

2.3 Implementation of RL Algorithm

The algorithms tested in this experiment are the basic versions. Since the purpose is solely to compare policy performance in solving path-planning tasks, the algorithms serve only as platforms to deliver the policies. Any additional modifications or derivations in the algorithms would affect the results and compromise the validity of the comparison. This section is divided into two subsections, each representing the explanation for one algorithm. The explanation covers the equations used in the coding implementation, and the algorithm in pseudo-code form.

2.3.1 Q-learning Algorithm with Off-Policy

Q-learning uses an off-policy approach as the basis for decision-making. The off-policy method allows the algorithm to find solutions to the task greedily. With this greedy approach, the expected result demonstrates whether the solution is the shortest or fastest one. Equation (2) shows the Q-learning formula based on the Bellman equation, which is the fundamental equation for Q-learning [23]. Figure 4 presents Q-learning in pseudocode form.

$$New\ Q\ (S_t,\ A_t)\ =\ Q\ (S_t,\ A_t)\ +\ a\ [R_{t+1} +\ y\ Max\ Q\ (S_{t+1},\ A_{t+1})\ -\ Q\ (S_t,\ A_t)\ ] \quad (2)$$

$S_t = Current\ State$
$A_t = Current\ Action$
$a = Learning\ Rate$
$R_{t+1} = Reward\ based\ on\ Action$
$S_{t+1} = Next\ State$
$A_{t+1} = Next\ Action$
$y = Discount\ Factor$
$\max Q = Highest\ Q\ value\ in\ Q\ table\ (Greedy\ Policy)$

Q – learning pseudocode form	
1	**Initialize** each state i ∈ S and action u∈A
2	**for** each episode **do**
3	Initialize state **S**
4	repeat:
5	Choose action at current state using greedy policy
6	Take action, observe reward and next state
7	$Q(s,a)\ =\ Q\ (s,a)\ +\ a\ [r +\ y\ Max\ Q'\ (s', a')\ -\ Q\ (s,a)]$
8	s ← s'
9	until s is done

Figure 4. Q–learning pseudocode

2.3.2 SARSA Algorithm with On-Policy

SARSA uses an on-policy approach as the base reference for its decision-making process. The on-policy method allows the user to declare a specific policy to be followed by the algorithm. When the algorithm runs, the declared policy is continuously evaluated and improved. The expected result from this algorithm is a solution based on the requirements specified by the user. Equation (3) shows the basic SARSA equation [24], while Figure 5 presents SARSA in pseudocode form.

$$New\ Q\ (S_t,\ A_t)\ =\ Q\ (S_t,\ A_t)\ +\ a\ [R_{t+1} +\ y\ Q\ (S_{t+1},\ A_{t+1})\ -\ Q\ (S_t,\ A_t)\ ] \quad (3)$$

$S_t = Current\ State$
$A_t = Current\ Action$
$a = Learning\ Rate$
$R_{t+1} = Reward\ based\ on\ Action$
$S_{t+1} = Next\ State$
$A_{t+1} = Next\ Action$
$y = Discount\ Factor$
$y\ Q\ (S_{t+1},\ A_{t+1}) = Behaviour\ Policy$

SARSA pseudocode form

Initialize each state i ∈ S and action u∈A
for each episode **do**
    Initialize state **S**
    repeat:
        Choose action at current state using setup policy
        Take action, observe reward and next state
        $Q(s,a) = Q(s,a) + a[r + y\,Q'(s',a') - Q(s,a)]$
        s ← s' and a ← a'
    until s is done

Figure 5. SARSA pseudocode

2.4 Data Collection, Evaluation and Comparison

The simulation and testing in the Taxi Domain using OpenAI Gym generated data containing the agent's rewards, episodes and actions. These values were stored in a Q-table, where policies were evaluated and rewards were compared across episodes. The data was then retrieved from the PyCharm terminal and plotted in reward versus episode graphs for analysis. The evaluation compared Q-learning and SARSA. The primary focus was on the convergence speed of rewards, specifically examining the number of episodes required at different learning rates. The objective was to determine which algorithm, whether Q-learning or SARSA, will solve the task more efficiently. The hypothesis was that Q-learning would require fewer episodes to achieve reward convergence compared to SARSA, indicating faster task completion.

3. RESULTS AND DISCUSSION

This section presents the results of the comparative study between SARSA and Q-learning algorithms and analysis of each algorithm performance in solving the path-planning task within the Taxi-v3 environment. The section begins with the interpretation of RL testing data, focusing on cumulative reward patterns and the number of episodes required for reward convergence. These metrics indicate the learning efficiency and performance of both algorithms. The data is presented through graphs and analysis to clearly show the performance trends under identical experimental conditions. Following the data interpretation, a detailed review of both policies performance is conducted, examining the path planning outcomes and the time efficiency in solving the task. This review evaluates how each policy type which are on-policy for SARSA and off-policy for Q-learning influences decision-making and solution quality.

3.1 RL Testing Data Interpretation

The experiment compared two parameters between SARSA (on-policy) and Q-learning (off-policy). First, it presented cumulative rewards across five different sets of episodes and learning rates, displayed as graphs. Second, it showed a table summarizing the number of episodes each algorithm required to converge at different episode counts and learning rates. Figures 6 to 8 show line graphs of cumulative rewards versus the number of episodes from single test runs for each increment, ranging from 10,000 to 50,000 episodes with learning rates (α) from 0.1 to 0.5.

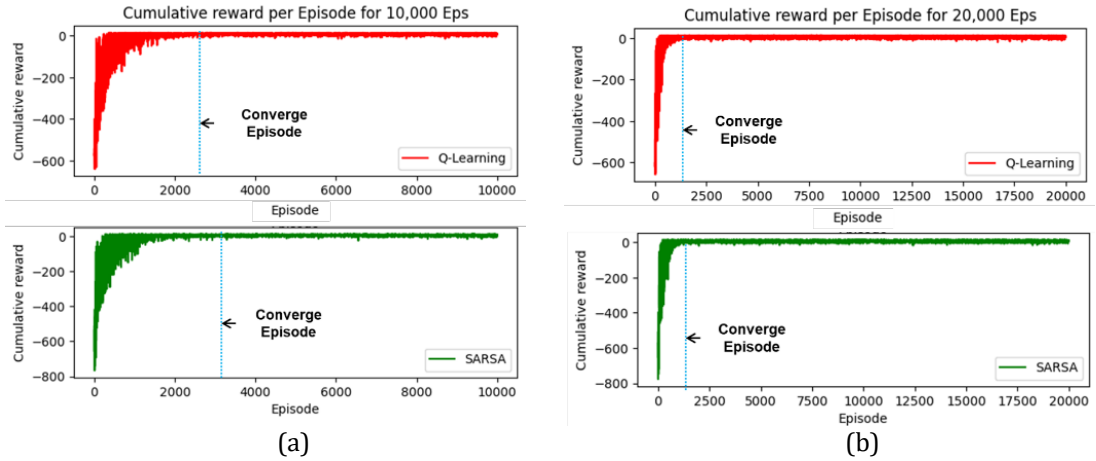


Figure 6. Cumulative reward data from SARSA and Q-Learning for: (a) 10, 000 episodes with 0.1 learning rate, α, and (b) 20,000 episodes with 0.2 learning rate, α

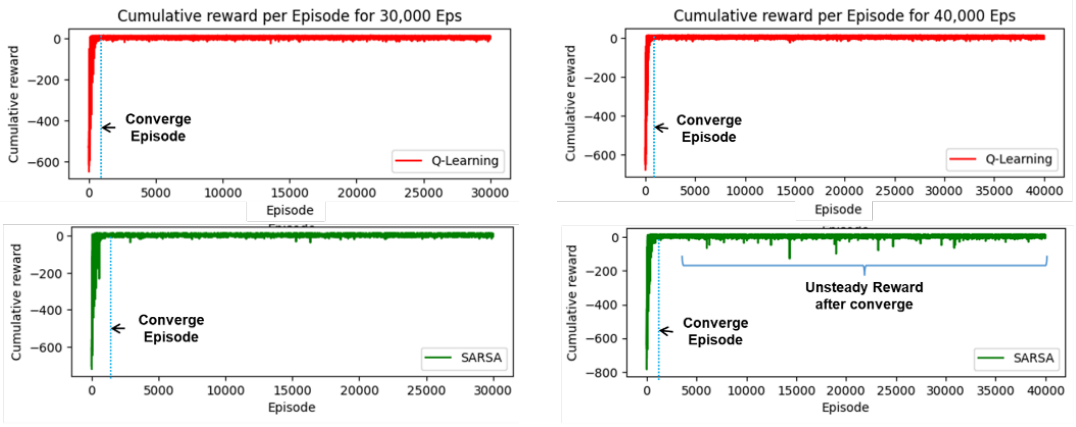


Figure 7. Cumulative reward data from SARSA and Q-Learning for: (a) 30,000 episodes with 0.3 learning rate, α, and (b) 40,000 episodes with 0.4 learning rate, α

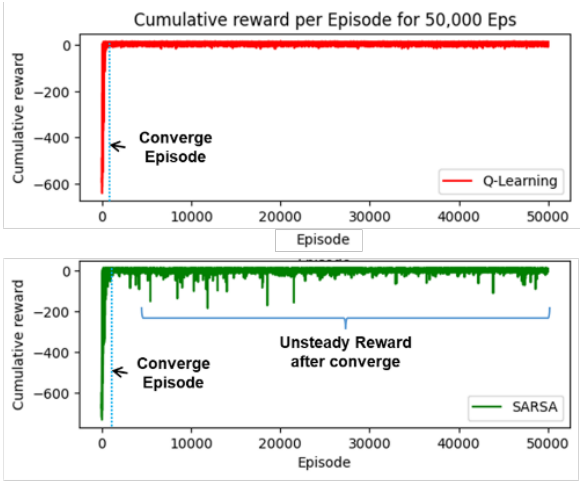


Figure 8. Cumulative reward data from SARSA and Q-Learning for 50,000 episodes with 0.5 learning rate

From these figures, it can be seen that the cumulative rewards for both algorithms converged within the first 10-20% of total training episodes, with Q-learning converging slightly faster (approximately 5-10% fewer episodes), regardless of the total episodes or learning rate, α. When

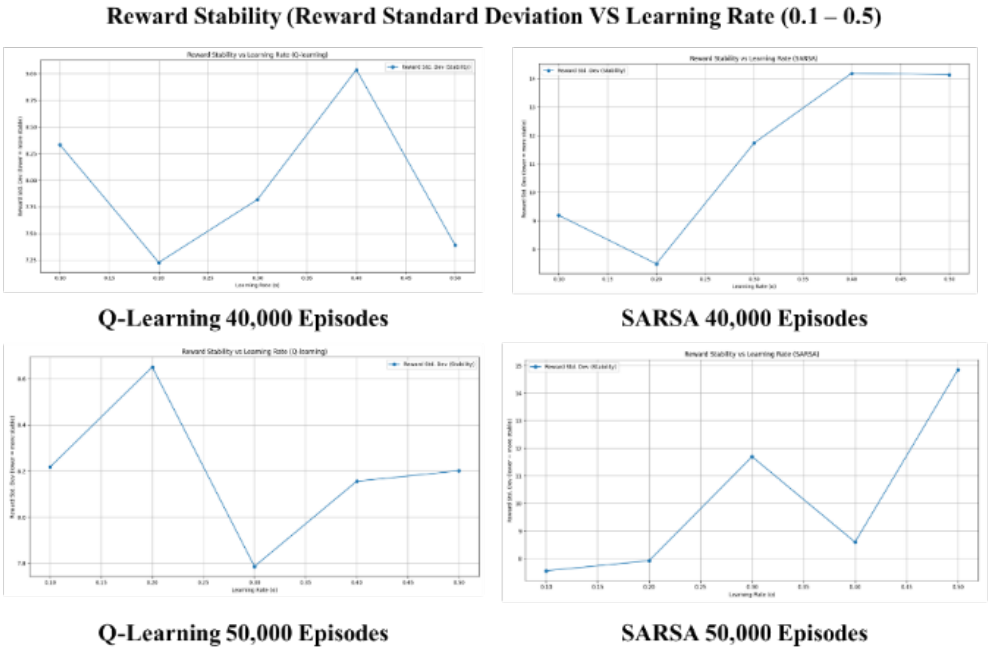
comparing both algorithms in each figure, SARSA accumulates slightly more penalties even after convergence compared to Q-learning. Despite these differences in penalties, both agents successfully reached the goal within the given episodes.

The comparison of cumulative rewards reveals the different principles guiding each agent toward the goal. Q-learning uses an off-policy approach, which allows the agent to greedily find solutions to the task. The results in Figures 6 to 8 support this fundamental off-policy concept. The rewards between episodes increase steadily, demonstrating that the agent consistently selects actions with the highest rewards while avoiding actions with low rewards or large penalties. The greedy strategy offered by off-policy ensures the action selection process is not tied to any predefined concept or plan.

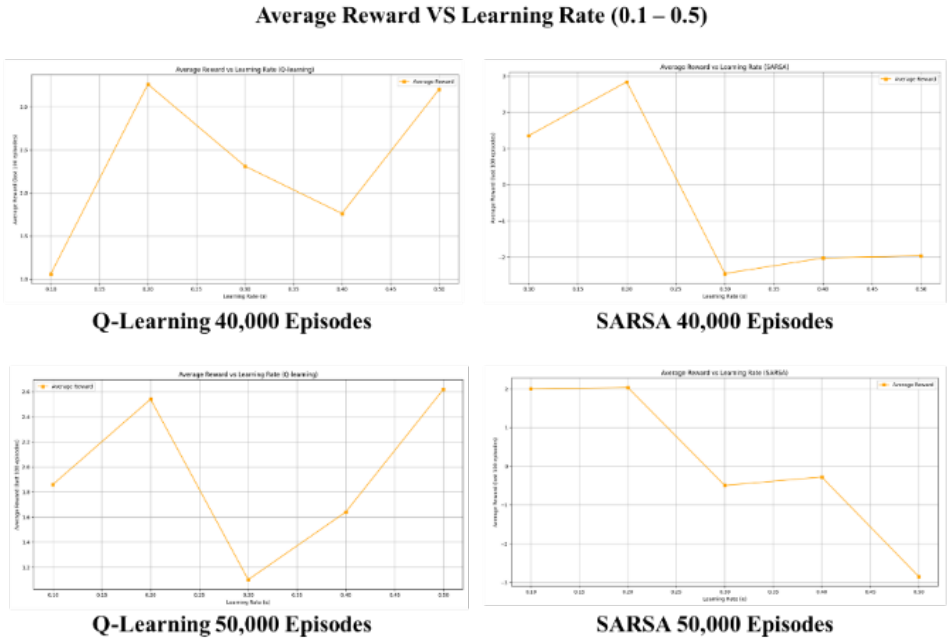
In contrast, SARSA implements an on-policy approach that requires the agent to follow a specific plan and strategy to find the path. The results in Figures 6 to 8 demonstrate that on-policy continuously explores to find the desired path. Even when actions result in penalties or low rewards, the agent still takes those actions. The cumulative reward in each episode increases slowly, indicating that on-policy does not provide the fastest path to the goal. However, the reward still converges at a low number of episodes and allows the algorithm to learn path planning based on the desired strategy.

Comparing across all figures, Q-learning consistently converges as the learning rate increases. However, SARSA becomes unstable after the learning rate increases, as clearly shown in Figure 7(b) and Figure 8. To show clearly the relationship between increment of learning rate and its stability, another test was done specifically for episode 40,000 and 50,000 with learning rate increment from 0.1 to 0.5 due to SARSA become unstable in this range of total episodes. The result of the stability shown in Figure 9.

Figure 9(a) illustrates the relationship between reward stability (reward standard deviation) and learning rate. Q-learning shows an exponential decrease in reward standard deviation as the learning rate increases from 0.3 to 0.5, indicating improved stability. In contrast, SARSA exhibits a linear increase in reward standard deviation with higher learning rates, reflecting growing instability. This occurs because Q-learning, being off-policy, updates toward the optimal action independently of exploration, while SARSA's on-policy update is influenced by exploratory actions. As the learning rate increases, SARSA's updates become too aggressive, causing larger fluctuations in Q-values. Supporting this, Figure 9(b) shows Q-learning maintains higher average rewards from 0.3 to 0.5, while SARSA's performance drops sharply. These findings confirm that on-policy algorithms like SARSA are unstable at high learning rates, whereas off-policy methods such as Q-learning remain more stable and efficient.



(a)



(b)

Figure 9. Relationship between: (a) reward standard deviation and learning rate, and (b) average reward and learning rate

Table 3 presents the number of episodes required by each algorithm to converge during training. The results indicate that Q-learning converged slightly faster than SARSA, requiring fewer episodes to achieve stable rewards. After convergence, the number of steps taken to reach all targets was also analysed. On average, the trained Q-learning agent required approximately 10 steps to complete the task, while the SARSA agent required 13 steps. This demonstrates that Q-learning not only converges faster but also learns a more efficient path.

The planned paths are also shown in Figure 10, tested with a learning rate of 0.3 and 30,000 episodes, as this setup provides stable performance for both algorithms. The figure illustrates

that Q-learning eagerly reaches all targets using the shortest path, requiring only 12 steps, compared to SARSA which required 14 steps with an exploration-type path.

Table 3 Number of episodes to converge

Total Episode	Learning Rate, *a*	Reward Converge Episodes		Number of Steps to reach all target after training (Steps)	
		Q – Learning	SARSA	Q – Learning	SARSA
10,000	0.1	2650 ± 100	3100 ± 100	11	12
20,000	0.2	1600 ± 100	1600 ± 100	13	15
30,000	0.3	1300 ± 100	1400 ± 100	12	14
40,000	0.4	1100 ± 100	1200 ± 100	8	10
50,000	0.5	1000 ± 100	1000 ± 100	8	14

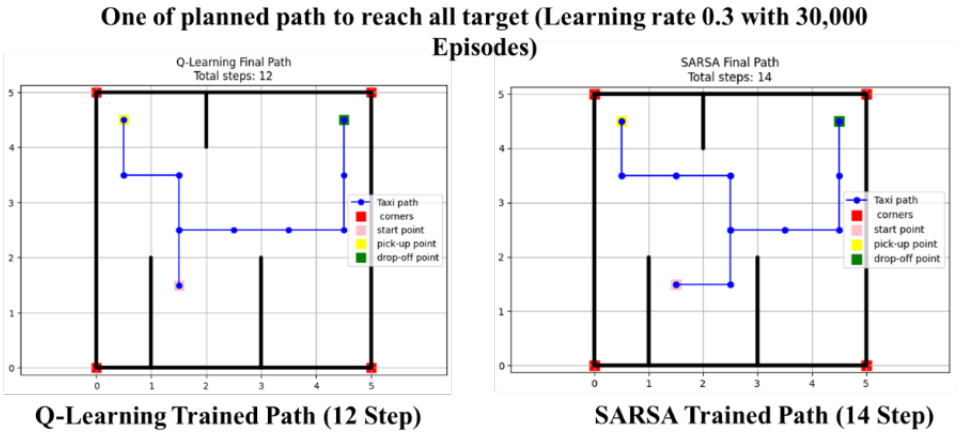


Figure 10. Planned path to reach all target

These results align with the fundamental differences between off-policy and on-policy learning approaches. Q-learning, as an off-policy method, uses a greedy principle that selects the maximum reward value from the Q-table to estimate and decide the next action. This allows the algorithm to learn from the optimal greedy policy regardless of the agent's actual behaviour, enabling faster exploitation of the optimal route and making Q-learning more suitable for tasks requiring rapid and efficient path completion.

In contrast, SARSA, being an on-policy method, learns from its current exploratory actions based on the policy set by the user. This policy uses parameters defined by the user to determine the next action, allowing the algorithm to move the agent according to the desired plan. This results in safer but slightly longer paths, making SARSA more advantageous for exploration or environment surveying applications where smooth policy transitions and safer navigation are prioritized.

Based on the experimental results, Q-learning shows that higher learning rates result in fewer episodes required for convergence. This indicates that the number of convergence episodes is inversely proportional to the learning rate value, α. However, SARSA shows the opposite pattern: higher learning rates result in more episodes required for convergence, demonstrating a directly proportional relationship.

3.2 Review on Both Policies Performance

The comparison of both policies is evaluated in two categories that can affect performance interpretation:

- Desired planned path
- Desired duration to solve the task

The first category is the desired planned path for path planning purposes. Both policies provide different strategies reflected in their path-planning behaviour and efficiency. Off-policy uses a greedy strategy that selects actions based on maximum reward, resulting in the shortest and most direct route with fewer steps. Therefore, if the desired path prioritizes speed and efficiency while disregarding safety factors, this policy is suitable. On-policy uses a strategy based on user-defined parameters, following an exploration-type approach that requires more steps but adheres to specified conditions. Therefore, if the desired path has specific requirements, safety factors, or precautionary conditions, this policy should be considered.

The second category is the desired duration for solving the path planning task. Hyperparameter tuning is crucial in achieving the desired duration. Off-policy requires a high number of episodes and high learning rate to converge quickly. In contrast, on-policy requires a low learning rate and high number of episodes, as it is sensitive to drastic changes that cause penalties at high learning rates. These findings highlight the importance of selecting a suitable policy based on task requirements which Q-learning is more efficient for rapid convergence with higher learning rates, while SARSA provides smoother but slower learning with conservative settings. These findings highlight the importance of selecting a suitable policy based on task requirements. Q-learning is more efficient for rapid convergence with higher learning rates, while SARSA provides smoother but slower learning with conservative settings.

Overall, based on the review of these two categories, the performance of both policies aligns with each algorithm theoretical working principles. Two key considerations for improving policy performance which are hyperparameter tuning must be adjusted for different tasks to achieve optimal results users must determine whether the agent needs time for exploration or must reach the target quickly. Then, adequate processor specifications are important to process the algorithm accurately and minimize errors and delays.

4. CONCLUSION

In conclusion, this study successfully compared reinforcement learning policies for path planning tasks through experimental validation in the Taxi-v3 environment. Results show that Q-learning (off-policy) converged approximately 5-10% faster than SARSA (on-policy) and required 31% fewer steps on average, demonstrating a more direct path-planning approach. These findings confirm that algorithm performance depends on application requirements which Q-learning is suitable for tasks requiring fast, efficient paths, while SARSA is preferred for exploration tasks requiring precautionary conditions and safer navigation. Both policies execute tasks according to each algorithm theoretical foundations, offering distinct advantages based on specific use-case requirements. While these findings are derived from the simplified Taxi-v3 environment, generalization to real-world mobile robot navigation may face additional challenges such as continuous state spaces, sensor noise and dynamic obstacles.

ACKNOWLEDGEMENTS

The author would like to acknowledge the support from the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2022/TK08/UNIMAP/03/13 from the Ministry of Higher Education Malaysia.

REFERENCES

[1] A. Y. Majid, S. Saaybi, V. Francois-Lavet, R. V. Prasad, C. Verhoeven, 2023. Deep Reinforcement Learning Versus Evolution Strategies: A Comparative Survey, IEEE Trans Neural Netw Learn Syst, doi: 10.1109/TNNLS.2023.3264540.

[2] L. Yang et al., 2023. Path Planning Technique for Mobile Robots: A Review. Multidisciplinary Digital Publishing Institute (MDPI), doi: 10.3390/machines11100980.

[3] "What Is a Policy in Reinforcement Learning?", Gabriele De Luca, Accessed: Jun. 14, 2023. [Online]. Available: https://www.baeldung.com/cs/ml-policy-reinforcement-learning

[4] "What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study", M. Andrychowicz et al., Jun. 2020. [Online]. Available: http://arxiv.org/abs/2006.05990

[5] J. García, F. Fernández, 2015. A Comprehensive Survey on Safe Reinforcement Learning.

[6] "Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning", C. Voloshin, H. M. Le, N. Jiang, Y. Yue, Nov. 2019. [Online]. Available: http://arxiv.org/abs/1911.06854

[7] R. Munos, G. Deepmind, T. Stepleton, A. Harutyunyan, M. G. Bellemare. Safe and efficient off-policy reinforcement learning.

[8] C. J. C. H. Watkins, P. Dayan, 1992. Q-Learning.

[9] H. W. K. S., Y. Z. Dongbin Zhao, 2016. Deep Reinforcement Learning with Experience Replay Based on SARSA.

[10] F. Emmert-Streib, M. Dehmer, 2022. Taxonomy of machine learning paradigms: A data-centric perspective. John Wiley and Sons Inc. doi: 10.1002/widm.1470.

[11] M. Hausknecht, P. Stone, On-Policy vs. Off-Policy Updates for Deep Reinforcement Learning.

[12] L. Zhong, 2024. Comparison of Q-learning and SARSA Reinforcement Learning Models on Cliff Walking Problem. pp. 207–213. doi: 10.2991/978-94-6463-370-2_23.

[13] M. B. Imtiaz, Y. Qiao, B. Lee, 2021. A Comparison of Two Reinforcement Learning Algorithms for Robotic Pick and Place with Non-Visual Sensing. International Journal of Mechanical Engineering and Robotics Research, vol. 10, no. 10, pp. 526–535, doi: 10.18178/ijmerr.10.10.526-535.

[14] S. Ramesh, S. B. N, S. J. Sathyavarapu, V. Sharma, N. K. Nippun, M. Khanna, 2025. Comparative analysis of Q-learning, SARSA, and deep Q-network for microgrid energy management. Sci Rep, vol. 15, no. 1, doi: 10.1038/s41598-024-83625-8.

[15] A. Sharma, R. Gupta, K. Lakshmanan, A. Gupta, 2021. Transition based discount factor for model free algorithms in reinforcement learning. Symmetry (Basel), vol. 13, no. 7, doi: 10.3390/sym13071197.

[16] "Taxi – Gym Documentation," Gymnasium (OpenAI), Accessed: Oct. 18, 2025. [Online]. Available: www.gymlibrary.dev/environments/toy_text/taxi/

[17] "MountainCar – Gym Documentation," Gymnasium (OpenAI), Accessed: Oct. 18, 2025. [Online]. Available: www.gymlibrary.dev/environments/classic_control/mountain_car/

[18] "CliffWalking – Gym Documentation," Gymnasium (OpenAI), Accessed: Oct. 18, 2025. [Online]. Available: https://www.gymlibrary.dev/environments/toy_text/cliff_walking/

[19] H. Wang et al., 2017. Integrating reinforcement learning with multi-agent techniques for adaptive service composition, ACM Transactions on Autonomous and Adaptive Systems, vol. 12, no. 2, doi: 10.1145/3058592.

[20] J. Arroyo, C. Manna, F. Spiessens, L. Helsen, D. Saelens, J. Laverge, et al., 2022. An OpenAI-gym environment for the building optimization testing (BOPTEST) framework, Proc. Building Simulation 2021: 17th Conf. IBPSA, vol. 17, pp. 175–182.

[21] "COOL-MC: A Comprehensive Tool for Reinforcement Learning and Model Checking," D. Gross, N. Jansen, S. Junges, G. A. Perez, Sep. 2022 [Online]. Available: http://arxiv.org/abs/2209.07133.
[22] "A Hierarchical Region-Based Approach for Efficient Multi-Robot Exploration," D. Meng, T. Zhao, C. Xue, J. Wu, Q. Zhu, Mar. 2025 [Online]. Available: http://arxiv.org/abs/2503.12876.
[23] Y. Hu, L. Yang, Y. Lou, 2021. Path Planning with Q-Learning. Journal of Physics: Conference Series, doi: 10.1088/1742-6596/1948/1/012038.
[24] J. Song, 2024. A Modified U-Tree Algorithm Based on Sarsa(λ). 2024 IEEE 7th International Conference on Automation, Electronics and Electrical Engineering, AUTEEE 2024, pp. 973–978. doi: 10.1109/AUTEEE62881.2024.10869793.