Comparison of Q-learning and Sarsa Algorithm for Automated Guided Vehicle Path Planning

J. O. Jeffrey Oon[1], S. N. L. K. Nor Azmi[1], N.I. Anwar Apandi[1*], N. Z. Abd Rahman[2], N. A. Muhammad[3]

[1]Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

[2]Faculty of Engineering & Technology, Multimedia University, 74540 Melaka, Malaysia

[3]Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), 81310 UTM Johor Bahru, Johor, Malaysia

Corresponding author's email: ilyana@utem.edu.my

Abstract – *An Automated Guided Vehicle (AGV) system is a type of material handling equipment that navigates through a facility using a combination of sensors and computer control. However, traditional path planning methods for AGVs often face challenges in determining efficient routes while ensuring obstacle avoidance and minimizing computational overhead. These limitations hinder the continuity and stabilization of production processes, particularly in complex and dynamic environment. This work explores path planning for AGVs based on reinforcement learning, specifically the Sarsa algorithm, where the AGV functions as an agent, influencing the continuity and stabilization of the production process. The problem is framed as a Markov Decision Process (MDP), allowing the AGV to model its environment and make sequential decisions to optimize its path. As the agent undergoes training, the emphasis gradually shifts towards exploitation rather than exploration. Problems involving obstacle avoidance strategies for static environments are also addressed, considering various learning rates, discount factors, and steps. Simulation results demonstrate that the AGV can avoid obstacles in a grid-mapped environment and reach its destination. Therefore, the Sarsa algorithm converges faster and requires fewer steps compared to Q-learning implementation.*

Keywords: *Automated Guided Vehicle (AGV), Discount Factor, Learning Rates, Markov Decision Process (MDP), Q-Learning, Sarsa Algorithm.*

Article History
Received 2 September 2024
Received in revised form 2 December 2024
Accepted 8 January 2025

I. Introduction

Automated Guided Vehicle (AGV) play an important role in the industrial environment, especially in intralogistics and material handling processes [1]. As smart manufacturing environments become widespread, the introduction of the Factory of the Future (FoF) system employs the Internet of Things (IoT) and multi-access or mobile edge computing systems to control and manage AGV fleets. In the past decades, there have been research works on localization, scheduling, and path-planning regarding AGV. Path planning refers to the ability of AGV to search for the optimal path from the start point to the target point with minimum time, in the meantime, AGV considers the robotic constraints (obstacle avoidance) and inter-robotic constraints (collision avoidance). Path planning for the AGVs is one of the core challenges in the field of autonomous manufacturing [2],[3] . For example, the material handling process in the shop floor production line is related to the continuity and stabilization of the production process. Research directions have focused on localization, scheduling, and path planning for AGVs [4], while further exploring the optimization of AGV path planning by considering robotic and inter-robotic constraints such as obstacle avoidance and collision avoidance [5]-[8],

Recently, reinforcement learning (RL) has served as a solution to material handling challenges, particularly in navigating obstacles and averting collisions to ensure smooth distribution processes [9] - [12]. Machine learning is a process where a device program increases its performance by learning from experience. Machine learning algorithms are divided into three categories basically, which are supervised learning, unsupervised learning and RL. Supervised learning refers to training a device is trained using labeled data in the performance classification or regression based on inductive inference. Unsupervised learning trains a device using unlabeled data

This is an Open Access article distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License, permitting copy and redistribution of the material and adaptation for commercial and uncommercial use.

by density estimation or clustering. RL trains a device by agents and the interaction with the environment. Actions taken at every state affect the reward received, and the successive state, and future rewards [13]. The performance of AGVs can be evaluated based on factors such as learning rate (α), discount factor (γ), and steps [14]. RL emerges as a powerful tool to empower AGVs to make autonomous decisions based on their experiences [2], [15]-[17].

By training AGVs through interaction with the environment, RL enables actions to impact received rewards and future states. Actions taken in one state lead to transitions to the next state with associated probabilities, and rewards guide subsequent actions [13]. Markov Decision Process (MDP) is a conventional framework widely recognized for its efficiency in decision-making problems based on RL. MDP is defined by a set of $< S, A, T, R >$ which stands for "state", "action", "transition" and, "reward" function, which describe the function the agent's interactions with the environment. It allows the agent to evaluate different actions in each state and determine an optimal policy to maximize cumulative rewards over time. This structured approach is particularly useful for AGV path planning, where the goal is to find the best route while accounting for obstacles and maximizing operational efficiency. Understanding the relationships among these components is crucial for informed decision-making. Update targets are established based on received and expected future rewards, employing a one-step look-ahead method by achieving a balance of exploration and exploitation is important for effective decision-making. Exploration allows the agent to discover new states and rewards, while exploitation is applying known information to maximize immediate rewards. Striking this balance ensures that the agent does not get stuck in suboptimal actions and continuously improves its policy.

Q-learning is one of the remarkable classical RL algorithms. Q-Learning was introduced by Watkins in the year of 1989 [18]. Q-Learning is an off-policy algorithm, which means target and behavior policies use different policies. Target policy follows greedy policy in action selection while behavior policy follows ε -greedy policy to select the actual action [19]. For instance, in aircraft component assembly lines, where diverse AGV types manage 13 stations, Q-Learning can aid in navigating AGV states, including collisions, and facilitating conflict resolution and task completion[14], [20]-[22].

However, Q-learning may face challenges in multi-agent environments due to its large memory requirements, leading to complex problems. State–Action–Reward–State–Action, commonly referred to as Sarsa (λ) algorithm, represents an improved version of Q-Learning, employing the same policy for both target and behavior and utilizing an ε-greedy strategy for action selection [18]. Moreover, research has shown that Q-Learning performs sub-optimally compared to the Sarsa algorithm in solving scheduling problems [20], [23]. The Sarsa algorithm addresses mobile robot path planning, demonstrating efficacy in resolving challenges such as obstacle avoidance and path planning in complex environments through two- and three-dimensional simulations [24].

The contributions of this paper are as follows. First, this paper addresses the path planning challenges for AGVs in the context of material handling in a shop floor area, specifically in an indoor environment. Unlike [17], this study includes investigating the path planning for AGV by utilizing the RL and principles of the Sarsa algorithm. On top of that, this study also presents a comprehensive framework that utilizes MDP to model AGV path planning task.

By implementing both Q-learning and the Sarsa algorithm, the study demonstrates improved path planning efficiency, reducing collision rates and optimizing routes in real-time. The analysis of key performance metrics, such as learning rate (α), discount factor (γ), and steps provides valuable insights into the effectiveness of RL in AGV applications. Moreover, the research addresses the computational challenges associated with RL, proposing strategies to achieve faster learning speeds and shorter convergence times, thereby making RL more practical for industrial applications.

II. System Model

In this section, the system model for AGV path planning using Q-learning is presented which defines the framework within which the AGVs operate, outlining the key model in RL-based decision-making, $< S, A, T, R >$.

In this model, each state and action pair correspond to a Q-value, in which the sum of the existing Q-value is updated with the new Q-value for the action for the current state, S to determine the optimal action in the current state. Q-learning continuously updates the Q-values for each state until stabilized, meaning no further changes occur, or until a predefined stopping criterion is met. For every action taken by the agent, the Q-value is updated only once. In environments with a large state-action space, significant storage is required to accommodate the extensive Q-table and its associated rewards.

Table I shows the table of Q-value, where the rows is state, s and the columns represent action, a. Each state and action pair corresponds to a Q-value. Action selection by an agent is determined by Q-value, $Q(s, a)$, which is updated as the sum of the received reward, r, and expected future value $\gamma \max Q(s', a')$.

TABLE I
Q-VALUE TABLE

State Action	a_1	a_{n+1}
s_1	$q(s_1,a_1)$	$q(s_1,a_{n+1})$
s_{n+1}	$q(s_{n+1},a_1)$	$q(s_{n+1},a_{n+1})$

The agent can move in four directions: north, south, east, and west, which constitute action space. This movement corresponds to the ability of AGV to transition between the adjacent cells. Fig. 1 illustrates the agent's in circle indicates the detectable directions within the Grid World environment.

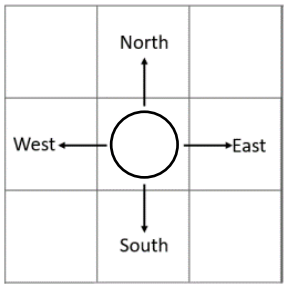

Fig. 1. Detectable direction of the agent in Grid World Environment

The agent has four possible actions, and each exploration updates the corresponding Q-value in the Q-table. Given the state-action pair design, with two 4×4 states, thus, the total number of Q-values to be learned is two 4×4×4. Therefore, there are 256 state-action pairs of Q values to be learned. The obtained Q value matrix is shown in (1).

$$Q_{64\times4} = \begin{bmatrix} Q_{s0a0} & Q_{s0a+1} \\ Q_{s+1a0} & Q_{s+1a+1} \end{bmatrix} \quad (1)$$

$$r = \begin{cases} +10, & if\ state = [4,4] \\ -1, & otherwise \end{cases} \quad (2)$$

Every movement of the agent results in encountering different states, the agent obtains the reward based on the state. regardless of the action taken, according to prior knowledge. The agent receives a reward, r =+10 when it reach the terminal state and penalty $r = -1$ for every other action, as shown in equation (2).

A. Policy

This section provides the highlights of the policy employed in this study, focusing on implementing ε-greedy strategy. This strategy is a method used to balance exploration and exploitation in RL. It ensures that the agent explores the environment sufficiently while also exploiting the knowledge it has gained to maximize rewards.

Initially, when the agent has limited knowledge about the problem environment, it tends to explore more. However, as the agent undergoes training, the emphasis gradually shifts towards exploitation rather than exploration. Developing effective exploration strategies for RL agents remains an active area of research.

To improve the decision-making process, the ε-greedy strategy is implemented as shown in equation (3), the agent chooses an action randomly with a probability of $0 < \varepsilon < 1$, allowing for exploration of new actions.

$$\begin{cases} argmax_a Q(s', a), & 1 - \varepsilon \\ Uniform, & \varepsilon \end{cases} \quad (3)$$

Conversely, with a probability of 1- ε, the agent selects the action with the highest Q-value for the current state, thereby exploiting its existing knowledge to achieve the best possible outcome. The rules used by the ε-greedy policy ensure that the action with the maximum Q-value in a specific state is selected with a probability of 1- ε, while one of all possible actions in the state is chosen randomly with a probability of ε.

A stochastic policy, π is a mapping from states to probabilities, where $\pi(a|s)$ represents the probability of acting a in state s. This means that for any given state, the policy provides a probability distribution over possible action. The aim of RL is to find the optimal policy, $\pi*$, which maximizes the expected sum of discounted rewards over time. The optimal policy is determined by evaluating different policies and selecting the one that yields the highest cumulative reward. The equation for the optimal policy is given by (4).

$$\pi = \begin{cases} argmax_a Q(s', a), & \mathbb{E}_\pi\{\sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s\} \\ \pi, & \end{cases} \quad (4)$$

For states in the set S, where $s \in S$, $rk = R(sk, ak)$ represents the reward at time k. The value function $V\pi(s)$ at state s, following policy π, denotes the expected reward when starting at state s and adhering to the policy π thereafter. Table II shows the key components of the optimal policy equation in RL.

TABLE II
KEY COMPONENTS OF THE OPTIMAL POLICY EQUATION IN RL

Components	Descriptions
$argmax_\pi$	The operation of finding the policy that yields the highest expected value
$\mathbb{E}_\pi$	The expected value under policy, π considering all possible outcomes
$\left\{\sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s\right\}$	The total of discounted rewards over a time horizon, H
k	Time step index, from 0 to H-1
γ^k	Discount factor raised to the power of k
r_{k+1}	The reward at time step $k+1$
$(s_0 = s)$	Expectation is taken over all trajectories starting the initial state $s_0 = s$

B. Discount Factor

The discount factor $\gamma \in [0,1]$ represents future reward controlled by a learning agent, aiming to maximize cumulative rewards over time since state and action in classical RL are discrete data where the action-value function is tabulated [18].

A low discount factor initiates myopic behavior, emphasizing the maximization of short-term rewards that must be achieved by the agents. Conversely, a high discount factor initiates rewards maximization of longer frame rewards, making agents become more forward looking. Action value function (Q-function) is defined as in equation (5).

$$Q_\pi(s,a) = \mathbb{E}_\pi\left(\sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s\,, a_0 = a\right) \quad (5)$$

The Bellman optimality equation for the action-value function is introduced in equation (6). The Bellman equation provides a recursive decomposition of the value function, which will illustrate better on how decision-making improves over time.

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s | A_{t=a}\right] \quad (6)$$

In this study, the implementation of Bellman equation, one can update the value of Q-values iteratively, ensuring that the learning agent converges with the optimal policy that maximizes the cumulative rewards.

III. Design the Algorithms

This section demonstrates the design of the algorithms used in this study for path planning of AGV using the RL application. The objective of this study is to determine the most efficient route for AGVs while avoiding obstacles and optimizing their performance within a material handling environment. To solve this, reinforcement learning methods specifically Q-learning and Sarsa are implemented and compared to evaluate their effectiveness in AGV navigation performance.

A. Problem Definition

The main problem addressed in this study is the development of an intelligent path planning strategy for AGVs operating in an indoor manufacturing environment. The AGV must learn to navigate from a designated start point to a target location while optimizing travel time and avoiding collisions with obstacles.

As mentioned earlier, the RL framework is defined in terms of its essential variables and parameters, which serve as the foundation for the agent's learning process. These include the environment state, possible actions, learning rate, discount factor, exploration-exploitation strategy, and reward function. The outputs of this learning process are the Q-values for each state-action pair and the optimal policy derived from them. Table III shows the input and output variables used in the design and implementation of the algorithms.

TABLE III
VARIABLES AND PARAMETERS OF ALGORITHMS

Type of Variables	System Parameters	Notation
Input	Environment State	s
Input	Action	a
Input	Learning Rate	α
Input	Discount Factor	γ
Input	Exploration-Exploitation Policy	ϵ
Input	Reward Function	r
Output	Q-Value	Q
Output	Optimal Policy	π

B. Q-Learning Algorithm

The equation of Q-learning, as described in (7), is applied to the Q-values in Table I. This table shows how each state s and action a pair corresponds to a specific Q-value ($Q(s,a)$). For instance, in state s_j, the agent evaluates the potential Q-values for actions a_i and a_{i-1} to determine the most favorable course of action. The update process involves recalculating $Q(s,a)$ using the immediate reward (r) and the discounted maximum expected future value $\gamma \max_a Q(s',a')$. Equation (7) is used to update the Q-value in Table I through the learning process, as the agent optimizes its decision-making policy based on the cumulative reward received over time.

$$Q'(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s'a') - Q(s,a)] \quad (7)$$

Algorithm 1 : Q-Learning Algorithm

Input : s, a, γ, α
Initialize $Q(s,a)$:
$\forall s, \forall a, \pi(a|s) = \frac{1}{|A|}$;
repeat
  Initialization state S
  repeat
    Using (3), select action a under state s;
    Obtain reward r and the nest state s';
    Using (7), update $Q(s,a)$;
    $s \leftarrow s'$;
    until s is terminated;
until $\forall s, \forall a, Q(a|s)$ optimize;
Output : policy $\pi(s) = \underset{\pi \in A}{argmax}\ Q(s,a)$

Initially, the rewards are present in the Q-table. An agent chooses an action through a policy in the starting state and moves to the next state. This process is repeated until the overall Q-value converges to a specific value where the Q-table is used to solve a given problem.

C. Sarsa Algorithm

Sarsa is also based on the Q-table. The difference between Q-Learning and Sarsa is the value of the actual future action a_{t+1} used instead of the maximum future value as shown in (7). At each step of each episode, the next action to take is determined rather than dynamically determining the step at the beginning of the next step. The target policy and behavior policy of SARSA follow ε-greedy policies which depend on Q-value. Sarsa eventually converges on the near-optimal policy and the actual optimal policy cannot be obtained. Sarsa algorithm is as equation (8).

$$Q'(s_t, a_t) \leftarrow Q(s_t, a_t) - Q(s_t, a_t) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1})] \quad (8)$$

Algorithm 2 : Sarsa Algorithm

Input : s, a, γ, α
Initialize $Q(s, a)$:
$\forall s, \forall a, \pi(a|s) = \frac{1}{|A|}$;
repeat
  Initialization state S
  repeat
    Using (3), select action a under state s;
    Obtain reward r and the nest state s';
    Using (8), update $Q(s, a)$;
    $s \leftarrow s'$; $a \leftarrow a'$
    until s is terminated;
until $\forall s, \forall a, Q(a|s)$ optimize;
Output : policy $\pi(s) = \underset{\pi \in A}{argmax}\ Q(s, a)$

D. Implementation of the AGV Path Planning

In this study, the AGV utilizes Sarsa algorithm which can dynamically interact with its external environment exploring paths through trial and error, and selecting the optimal route based on the accumulated learning experiences and an action selection strategy. During the process of continuous interaction with the environment, the AGV calculates the state-action value function $Q(s, a)$ and stores it in the Q-table. As the AGV continues to explore and learn, $Q(s, a)$ gradually converges to stable values. Once the Q-table converges, the AGV selects the action with the highest Q-value at each state to determine and execute the optimal path [17].

IV. Results and Discussion

The simulation has been conducted in MATLAB. The outcomes are systematically tabulated and presented in graphical form to illustrate the agent's performance under the designated value of parameters. This section discusses the findings from the simulations, comparing the performance of the Q-learning and Sarsa algorithm in terms of learning rate and discount factor, denoted by α and γ, respectively, and step against episode.

A. Performance of learning rate (α)

In the analysis of α performance for both Q-learning and Sarsa, the constant parameters are tabulated as in Table IV with discount factor, γ = 0.9. The value range of α=[0.1,0.9] being analyzed were categorized into three categories which are lower α=[0.1,0.9], and higher α=[0.7,0.9]. Fig. 2 and Fig. 3 shows the performance of average rewards with lower and higher α respectively.

TABLE IV
CONSTANT PARAMETERS TO ANALYSIS PERFORMANCE

Notation	System Parameters	Values
M	Number of maximum states per episode	500
N_i	Number of iterations	100
ε	Epsilon	0.9

Fig. 2 shows the impact of different learning rates on average rewards in two reinforcement learning (RL) algorithms. For lower learning values, α=0.1, there are significant changes in average reward, with early rewards gradually increasing until reaching optimal levels. Increasing α to 0.3 leads to higher average rewards for both algorithms. Both algorithms at the lower learning rates converge around the 40th episode and converge faster compared to higher rates. Q-Learning shows smaller deviations in average rewards than Sarsa, implying that Q-Learning may offer more consistent and reliable performance in achieving optimal rewards under the given parameters. However, Sarsa consistently displayed smaller reward deviations than Q-Learning, suggesting Sarsa's potential for more stable performance across various learning rates and its advantage in achieving optimal rewards. Q-Learning displays a wider range of average rewards and converges more slowly compared to Sarsa, which typically exhibits a narrower range of rewards and converges faster.

In Fig. 3, the values of α are set to be α=[0.7,0.9] for higher learning rates. When α=0.7, the rewards increase at first and then decrease. Then, α=0.8 decreases inconsistently at first and then increases, and α=0.9 gradually decreases and reaches the lower limit of the average reward. Q-Learning with α=0.7 converges at the 44th episode, while the other values of α continue to fluctuate over the episodes without reaching a stable state. In the case of Sarsa, performance with α=0.7 remains considerably stable compared to the other two values of α. The latter exhibited dramatic oscillations around the near-optimal value without achieving a stable state. Based on the observations of both algorithms, higher α leads to instabilities and hinders convergence to an optimal policy. Large learning rate values can result in erratic performance, making it challenging to achieve stable and reliable learning outcomes. Striking a balance between exploration and exploitation is crucial to ensure convergence and stability in RL algorithms.

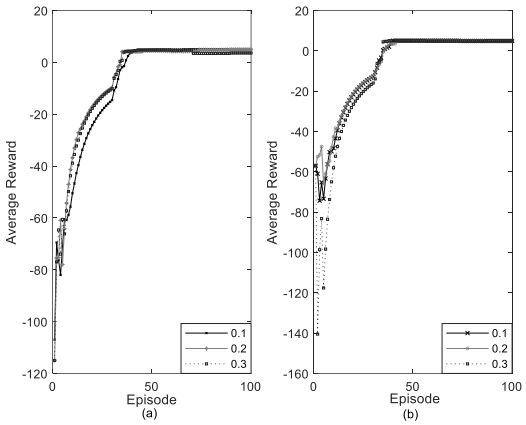

Fig. 2. Performance of average reward with lower α for (a) Q-Learning; (b) Sarsa

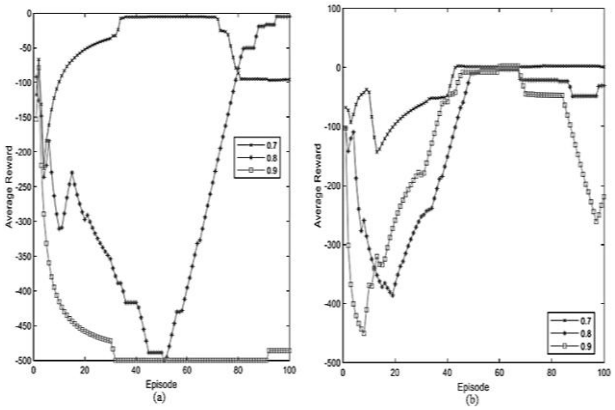

Fig. 3. Performance of average reward with higher α for (a) Q-Learning; (b) Sarsa

B. *Performance of discount factor (γ)*

In the performance analysis of the discount factor of Q-learning and Sarsa, constant parameters have been used for both. The parameters include the number of maximum states per episode, the number of episodes, ε and α= [7, 19]. The parameter values are tabulated in Table IV with α=0.7. The value range of γ=[0.1,0.9] being analyzed was categorized into three categories which are lower γ =[0.1,0.3] and higher γ =[0.7,0.9].

Fig. 4 shows the cumulative reward for each episode with various γ. Based on the analysis of learning factor, α, the best α value that produces the highest generation value of average rewards for Q-Learning is 0.5 while that for Sarsa is 0.6. For comparison purposes, 0.5 of α was taken in the performance analysis of the discount factor. This value is used for the performance analysis of γ as the fixed parameter setting.

The study assessed the influence of the discount factor, γ on the Q-Learning and Sarsa algorithms by analyzing its performance in lower and higher ranges, based on the results presented in Fig. 5. Within the range γ = [0.1, 0.3], Q-Learning achieved a cumulative reward of 3.33 %, which was higher than Sarsa's 0.33 %. Although both algorithms produced negative cumulative rewards in each of the 100 episodes, Q-Learning performed better, demonstrating a larger positive reward percentage and a more focused distribution of rewards within a smaller range.

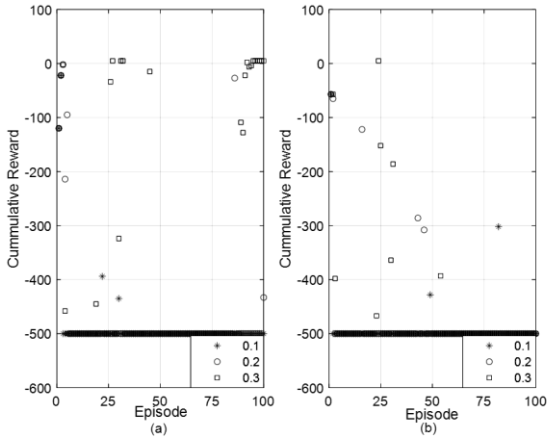

Fig. 4. Performance of discount factor with various γ for (a) Q-Learning; (b) Sarsa

Transitioning to γ = [0.4, 0.6], Sarsa was the best with a significantly larger positive cumulative reward than Q-Learning. In this range, Sarsa outperformed Q-Learning by exhibiting a more converging range of average rewards. As γ values increased to γ = [0.7, 0.9], Q-Learning maintained its dominance, reaching a solid positive cumulative reward of 78% compared to Sarsa's significantly larger positive cumulative reward over the episodes. Both algorithms achieved their highest positive cumulative rewards at γ = 0.9, indicating a preference for long-term benefits over immediate gains at higher γ values.

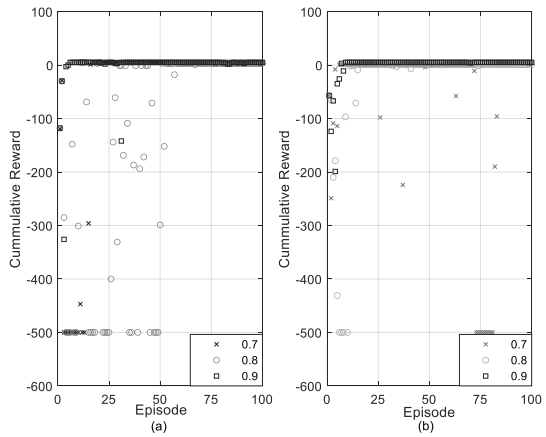

Fig. 5. Performance of discount factor with higher γ for (a) Q-Learning; (b) Sarsa

Fig. 5 shows that Q-Learning outperformed Sarsa for cumulative reward at discount factor levels γ =[0.7,0.9]. Q-Learning received a positive cumulative reward of 78%, whereas Sarsa received an even higher positive

cumulative reward of 82.66%. Notably, Q-Learning had a higher percentage by 86% of cumulative rewards falling between -20 and 100, compared to Sarsa's by 84% within the broader range of -60 to 50.

This demonstrates ability Q-Learning's capacity to accumulate rewards across several episodes, outperforming Sarsa's performance. The higher cumulative reward gained by Q-Learning demonstrates its capacity to successfully learn and utilise rewards within the limitations imposed. Q-Learning converged earlier than Sarsa, around the, respectively, 6th and 9th episodes, but had higher fluctuations. Thus, analysing discount factor performance demonstrates its remarkable influence on cumulative rewards and convergence speed in Q-Learning and Sarsa algorithms, with consequences for learning efficiency and reward outcomes.

C. Performance of episode steps

The performance analysis of Q-learning and Sarsa algorithms, conducted under constant parameters including γ=0.9, selected based on the highest cumulative rewards observed. Fig. 6.(a) shows the performance in terms of episode steps, indicating notable differences in their exploration and convergence behaviors Q-Learning has a greater maximum route length of 337, indicating a more extensive exploration process and possibly slower convergence to optimum policy, whereas Sarsa has a shorter maximum path length of 210, implying faster convergence. Moreover, Q-Learning's longer total execution time implies greater computational demands or slower convergence relative to Sarsa. This could be attributed to Q-Learning's broader exploration, resulting in a more comprehensive search for optimal policy and longer convergence time, while Sarsa's focused exploration enables relatively quicker convergence.

Fig. 6.(b) shows the performance of total agent steps, with Q-Learning demonstrating phased increases and Sarsa displaying a more gradual rise with an early spike. The higher overall steps required by Q-Learning indicate its exploration of a larger action-state space or prolonged convergence, highlighting the trade-off between exploration and exploitation.

Q-Learning's phased increase suggests a balance between intensive exploration and subsequent exploitation, potentially leading to the discovery of optimal policy, whereas Sarsa's more gradual exploration may prioritize efficiency but risks missing optimal solutions. Sarsa focuses on the agent's performance during the learning process by considering the exploration-exploitation trade-off and incorporating an epsilon-greedy policy and it can learn to avoid dangerous actions more quickly than Q-learning. It updates its Q-values based on the current state-action pair and the immediate reward, considering the next action chosen according to the policy being followed.

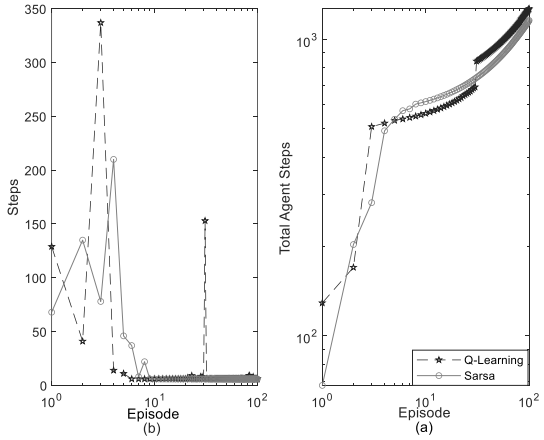

Fig. 6. Performance of (a) episode steps (b) total agent steps.

Sarsa is recommended for situations where the agent's performance during the learning process matters, and the agent's actions can directly impact the learning process itself, where the agent prefers a safer path and minimizes the risks during the learning phase.

V. Conclusion

In this paper, the AGV path planning was addressed utilizing the RL, Q-learning and Sarsa. AGV performance was analyzed by comparison between Q-learning and Sarsa is analyzed based on learning rate (α), discount factor (γ), and step against episode. The results demonstrated that the optimal rate (α) for Q-Learning and Sarsa is 0.5 and 0.6 respectively, while both algorithms perform better high discount factors (γ *close to* 1). The convergence time of Sarsa was greater than that of Q-learning, indicating that Q-learning fewer steps to stabilize.

The dissimilar update rules of Q-learning and Sarsa render them suitable for different scenarios. Q-learning focuses on identifying the optimal policy by updating Q-values based on the maximum expected future reward, making it suitable for environments where exploration is important. It can be effectively used in a practice phase where the agent explores using an epsilon-greedy policy (ε), followed by an optimal greedy policy during an important deployment. On the other hand, Sarsa's on policy approach makes it beneficial in scenarios requiring more stable learning during the training phase. For future work, visualization of AGV performance can be further developed to apply in AGV system for solving material handling problem, contributing to the future of Industry 4.0 in the Asian region.

Acknowledgements

The authors wish to acknowledge the Ministry of Higher Education (MOHE) of Malaysia and Universiti Teknikal Malaysia Melaka (UTeM), for supporting this research financially through the Fundamental Research Grant Scheme, No.: FRGS/1/2022/TK07/UTEM/02/33.

Conflict of Interest

The authors declare no conflict of interest in the publication process of the research article.

Author Contributions

Author 1: Data collection, analysis, writing – original draft preparation; Author 2: Draft review and editing; Author 3: Conceptualization, review system model; Funding acquisition; Author 4: project administration; Author 5: review analysis.

References

[1] E. A. Oyekanlu *et al.*, "A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5G-based smart manufacturing applications," 2020, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2020.3035729.

[2] E. Turki and H. Al-Rawi, "Multi-Robot Path-Planning Problem for a Heavy Traffic Control Application: A Survey," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 6, 2016, doi: 10.14569/ijacsa.2016.070623.

[3] "TV2 Galeri Mandarin - DF Automation & Robotics Sdn Bhd - YouTube." Accessed: Jan. 26, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=NrUliUkX

[4] R. H. Mohammed, M. E. Aboelmorsy, and B. E. Elnaghi, "Path tracking control of differential drive mobile robot based on chaotic-billiards optimization algorithm," *Int. J. Electr. Comput. Eng*, vol. 13, pp. 1449–1462, 2023.

[5] E. T. S. Alotaibi and H. Al-Rawi, "A complete multi-robot path-planning algorithm," *Auton Agent Multi Agent Syst*, vol. 32, pp. 693–740, 2018.

[6] M. Javaid, A. Haleem, S. Rab, R. P. Singh, R. Suman, and S. Mohan, "Progressive schema of 5G for Industry 4.0: features, enablers, and services," *Industrial Robot: the international journal of robotics research and application*, vol. 49, no. 3, pp. 527–543, Jan. 2022, doi: 10.1108/IR-10-2021-0226.

[7] W. S. WAN, N. I. A. Apandi, and N. A. Muhammad, "Task Scheduling Based on Genetic Algorithm for Robotic System in Manufacturing Industry," *International Journal of Electrical Engineering and Applied Sciences (IJEEAS)*, vol. 5, no. 1, 2022.

[8] Z. Ma and D. Wang, "A CNN Based Q-learning Algorithm for Path Planning of Automated Guided Vehicle," in *2021 IEEE International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT)*, IEEE, Jul. 2021, pp. 704–708. doi: 10.1109/ICEEMT52412.2021.9601907.

[9] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, p. 1278, 2021.

[10] N. P. Farazi, B. Zou, T. Ahamed, and L. Barua, "Deep reinforcement learning in transportation research: A review," *Transp Res Interdiscip Perspect*, vol. 11, p. 100425, 2021.

[11] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE access*, vol. 8, pp. 29064–29074, 2020.

[12] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[13] B.R. Kiran *et al.*, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[14] F. Martinez, H. Montiel, and L. Wanumen, "A deep reinforcement learning strategy for autonomous robot flocking," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 5, p. 5707, Oct. 2023, doi: 10.11591/ijece.v13i5.pp5707-5716.

[15] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021, doi: 10.1109/ACCESS.2021.3070054.

[16] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Trans Intell Technol*, vol. 5, no. 3, pp. 177–183, Sep. 2020, doi: 10.1049/trit.2020.0024.

[17] X. Liao, Y. Wang, Y. Xuan, and D. Wu, "AGV Path Planning Model based on Reinforcement Learning," in *2020 Chinese Automation Congress (CAC)*, IEEE, Nov. 2020, pp. 6722–6726. doi: 10.1109/CAC51589.2020.9326742.

[18] M. Rothmann and M. Porrmann, "A Survey of Domain-Specific Architectures for Reinforcement Learning," *IEEE Access*, vol. 10, pp. 13753–13767, Jan. 2022, doi: 10.1109/ACCESS.2022.3146518.

[19] H. Jiang, R. Gui, Z. Chen, J. Dang, J. Zhou, and S. Member, "An Improved Sarsa(λ) Reinforcement Learning Algorithm for Wireless Communication Systems," *Special Section On Artificial Intelligence For Physical-Layer Wireless Communications*, vol. 7, pp. 115418–115427, Jun. 2019, doi: 10.1109/ACCESS.2019.2935255.

[20] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[21] H. Hu, X. Jia, K. Liu, and B. Sun, "Self-adaptive traffic control model with Behavior Trees and Reinforcement Learning for AGV in Industry 4.0," *IEEE Trans Industr Inform*, 2021, doi: 10.1109/TII.2021.3059676.

[22] E. M. Raouhi, M. Lachgar, H. Hrimech, and A. Kartit, "Optimizing olive disease classification through transfer learning with unmanned aerial vehicle imagery," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 1, p. 891, Feb. 2024, doi: 10.11591/ijece.v14i1.pp891-903.

[23] A. Momenikorbekandi and M. Abbod, "Intelligent Scheduling Based on Reinforcement Learning Approaches: Applying Advanced Q-Learning and State–Action–Reward–State–Action Reinforcement Learning Models for the Optimisation of Job Shop Scheduling Problems," *Electronics (Basel)*, vol. 12, no. 23, p. 4752, Nov. 2023, doi: 10.3390/electronics12234752.

[24] D. Xu, Y. Fang, Z. Zhang, and Y. Meng, "Path Planning Method Combining Depth Learning and Sarsa Algorithm," in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, IEEE, Dec. 2017, pp. 77–82. doi: 10.1109/ISCID.2017.145.