INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

IJIE

Vol. 17 No. 1 (2025) 81-94
https://publisher.uthm.edu.my/ojs/index.php/ijie

Reinforcement Learning Algorithms for Probability Models Based on Mobile Robot Agent Path Planning in Unpredictable Environment

Vengatesan Arumugam[1]*, Vasudevan Alagumalai[1]

[1] *Department of Mechanical Engineering, Saveetha School of Engineering, SIMATS, Chennai, Tamil Nadu, Pin code:602105, INDIA*

*Corresponding Author: venkatesana9006.sse@saveetha.com
DOI: https://doi.org/10.30880/ijie.2025.17.01.007

Article Info

Received: 15 September 2024
Accepted: 17 February 2025
Available online: 30 April 2025

Keywords

Reinforced learning algorithm, Q-learning algorithm, Markov decision process, PCTL, unpredictable environment, mobile robot agent

Abstract

In recent years, the development of reinforcement learning algorithms (RLAs) significantly impacted various fields, including robotics. Mobile robots, which must navigate through unpredictable environments, present a complex challenge that traditional probability model-checking methods often struggle to address under dynamic and uncertain conditions. This research work focuses on modifying the Q-learning algorithm, a type of RLA, which is sequentially applied to establish a probability matrix under uncertain conditions. Subsequently, a probability model is developed with the assistance of the robot agent, selecting positions based on the maximum Q-table value of a matrix size 6x6 as per the assumed environment. The learned behaviour of the mobile robot agent, derived from the Q-learning method, is represented as a Markov Decision Process (MDP) model. To specify the dependability criteria of the mobile agent control system, Probabilistic Computation Tree Logic (PCTL) is employed. Furthermore, the MDP model, along with its designated attributes, is input into the Probabilistic Model Checker (PRISM) to facilitate automated verification. This approach proves effective in determining the goal position and selecting the optimal control model for evaluating performance, feasibility, reliability, and attaining the target point most efficiently. From the PRISM model for the episode 2500, the average reward and average steps obtained was 182.8 and 187.6 respectively. The mobile agent learned from the Q-learning algorithm for PRISM performance achieved a maximum reward of 84.99 and a minimum reward of 61.41 during the simulation.

1. Introduction

The mobile robot agent finds applications in various fields, including automobiles, construction, medicine, and agriculture. This research investigates the industrial applications of the mobile robot agent, specifically focusing on efficiently transporting materials between different locations and detecting obstacles [1]. The agent rapidly adapts to complex environments, swiftly reaching its target points. Currently and in the future, RLA is incorporated into various real-life scenarios, encompassing self-driving cars, industrial automation of vehicles, trading and finance, healthcare, news recommendation, gaming, and marketing, as well as the implementation of RLA in robotics manipulation and industrial mobile robots [2]. The research highlights the use of RLA in machine learning, showcasing its ability to enhance the achievement of goal points by the agent through a selected learning approach [3]. The emphasis extends to agents learning decision-making in unfamiliar settings and optimizing action, state, and reward variables to attain optimal values [4]. A mobile robot, functioning as an agent with

qualities like initiative, sociality, persistence, and responsiveness, operates within this framework [5]. This agent, operating on a mobile robot (MR), moves to desired locations or alters its state in a specific environment. Artificial intelligence research heavily relies on these MR agents [6], with the agent accomplishing objectives most effectively based on the movement options provided by the control system [7].

Throughout this research, various path-planning algorithms are explored. Three noteworthy algorithms emerge based on their effectiveness the Genetic Algorithm (GA) [8], the Ant Colony Optimization Algorithm [9], and reinforcement learning approaches applied in mobile robot path planning detection techniques [10]. The current research places a major emphasis on RL, with a focus on online learning approaches to achieve a balance between discovery and utilization in reaching the target point [11-12]. However, certain optimal paths may deviate or encounter unforeseen events in a stochastic, uncertain environment [13]. The term "stochastic uncertainty" describes the likelihood that abnormalities or events will occur in specific environmental locations due to various circumstances, such as software and hardware malfunctions, human error, and natural disasters [14]. The primary focus of this paper lies in a probability model based on a mobile robot agent reaching a goal point within an uncertain environment [15].

According to the above literature survey, the significance of mobile robot path planning approaches was determined by the algorithms employed. Notably, probability model checking was not utilized in the context of RLA, highlighting a significant gap in the existing survey. The author discusses present success rates for various path planning environments in the regular G2RL at 99.7%, random G2RL at 99.7%, and free G2RL at the highest value of 99.8%. Conversely, the success rate for global re-planning using discrete ORCA was found to be the minimum. Regarding computing time, G2RL recorded the maximum value at 6.367, while discrete ORCA and PRIMAL demonstrated minimum values at 2.064 and 5.892, respectively [16]. The research analyzes path planning for mobile devices using reinforcement learning techniques applied to mobile robots in an uncertain environment [17,18]. The study also employs PRISM 4.0 for probability-based real-time structural verification, incorporating this update into the PRISM probabilistic framework for verification [19,20].

The main contributions of the research gap identification points are: In this research, the RLA was facilitated using the proposed Q-learning algorithm. This algorithm generated a Q table in a matrix of size 6x6, enabling the mobile robot agent to learn in an unpredictable environment. An integrated framework for learning and verification was established, achieving a goal point with a maximum reward accuracy of 100% and a probability set at 97%. The research presented an anticipated task for the effectiveness of the unpredictable environment using the Q-learning algorithm. In this context, the agent was permitted to explore its surroundings and select the path planning for the mobile robot with the highest expected reward. The determination of the ideal path by the Q-learning algorithm, the robot agent control system was formalized as a MDP model. The properties of the system model were confirmed using PRISM. The proposed Q-learning algorithm in the research considered the optimization process of Q-table values for both numerical data (current state 2 mentions, therefore, 2-3-4-5, 2-3-1-5) and simulation data. The most efficient paths (1,5) (2,3,4,5) and (4,5) were determined, showcasing the best route for the mobile robot to reach the goal point. The training episodes considered are between 2500 and 3000, aiming to determine the maximum reward and total agent values.

2. Related to Work

In the literature survey, extensive research has been dedicated to the agent's movement in the path planning approach, emphasizing the learning algorithms employed for reaching the target point. However, there has been insufficient attention given to statistically validating the learned models, particularly in terms of considering correlations in agent-state transitions [21]. This section provides a summary of recent developments in path planning, agent mobility, and correlated efforts on model checking for probabilistic model verification in unpredictable environments.

2.1 RLA

In Fig. 1. as illustrated, the basic architecture of RLA. The reinforcement learning problems are formalized using the Markov Decision Process, or MDP. A Markov procedure can be used to describe the dynamics of the environment if it is fully observable. In MDP, the agent engages in continuous interaction with the environment, taking actions, and the environment reacts by creating a new state after each interaction.

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) p. 81-94

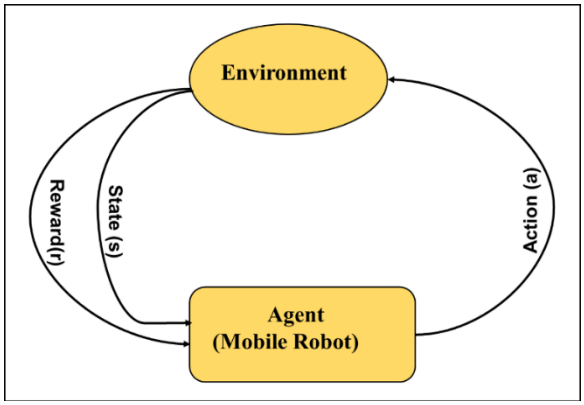

Fig. 1 *Basic Architecture of RLA*

2.2 Probability Model Checking

The PRISM and PRISM-games model checkers are instrumental in supporting the models discussed in our paper, which researched the realm of probabilistic model checking. This encompassed turn-based and concurrent deterministic activities, as well as fully and partially observable MDP, along with the related temporal logics with probabilities. To show the framework's usefulness, representative examples from autonomous robot systems were presented [22]. In this research work, we present the foundational principles of statistical model verification, with a particular emphasis on leveraging concepts and quantitative features represented in probabilistic temporal logic for finite-state Markov decisions. We highlight Markov decision-making methods as a core operating paradigm, especially suited for dealing with unexpected probabilistic systems. This paradigm aims to validate the accuracy of probabilistic scheme models against measurable probabilistic conditions, particularly those capable of describing data in probabilistic model checking [23]. We have explored the advancements and applications of probabilistic model verification, emphasizing the importance of an effective method for explicitly validating the quantitative characteristics of systems that exhibit stochastic behaviour, termed probabilistic model checking [24]. The investigated route planning for moving robotic vehicles using probabilistic model checking under uncertainty. A probabilistic model verification technique is introduced to address the mobile robots' path planning difficulty. Considering that the behaviour of mobile robots is continually influenced by their surroundings, we examine four primary environmental aspects as influencing factors. We employ a random sampling-based approach to construct the map, enabling the representation of unpredictable movement or behaviour as a result of an MDP. Concurrently, we demonstrate that rich mission specifications can be articulated using PCTL, accurately describing the system's features [25].

In this research paper, we are constructing an MDP model of the control system functioning along that route. We utilize a learning approach to select the route of a mobile robot agent working in a stochastic, unpredictable situation. The PRISM tool is employed to test the model.

3. Preliminary

This section delivers essential background information necessary for the subsequent discussion.

3.1 Q Learning

RLA employed by agents to acquire proficient action strategies for goal achievement is known as Q-learning [26]. In Q-learning, the algorithm calculates and organizes the highest anticipated future rewards associated with actions within each state, presenting this information in a table referred to as the Q-table. Initially, the Q-table values are set to zero, and a reward matrix is introduced in the Q-learning process. The learning process involves three fundamental components such as states, actions, and rewards, with each state being represented by a location. The Q-table values are updated as the agent explores its environment [26]. Furthermore, the stabilization of the Q-table occurs during the learning process, as indicated by the representation below Equation (1).

$$Q\ new\ (St, At) = Q\ (St, At) + \beta\ [\ R + \delta \max Q\ (St+1, At) - Q\ (St, At)] \quad (1)$$

Where St is denoted the state at a time t, St+1 is to mention the next state at a time t+1, At is to mention the action at a time t, β is to mention the learning rate, therefore $0< \beta<1$. δ is to mention the discount factor indicated to the future state to the current state, and R is to mention the reward value.

3.2 MDP

A stochastic control process with discrete time is called an MDP. It enables the modelling of decision-making in scenarios using a mathematical approach when the decision-maker has some control over the outcome and some probability.

Description 1: A MDP is a 4- tuple (S, A, Pa, Ra) [27] were,
Syntax:

- S is a state known as state space.
- A set of actions is known as action space.
- Pa (S, S') = Pr (S t+1= S' | St = S, at = a) in order is the probability chance an action taken at time t, which is state S, will result in state S' at time t+1.
- Ra (S, S') is the instant benefit obtained following the change from state S to state S' as a result of action a, Equations (2) and (3) are presented subsequently.

Transition Probability (2)

$$P_{ss'} = P\,[\,State_{t+1} = s' \,|\, State_t = s, Action_t = a]$$

Where, state s and success of state's'

$$R_s = E\,[R_{t+1} |\, State_t = s, Action_t = a] \quad (3)$$

$$q_\pi\,(s, a) = E_\pi \sum_{k=0}^{\infty} \beta^k R_{t+k+1} \,|State_t = s, Action_t = a]. \quad (4)$$

Where, the depiction above corresponds to Equation (4), R_{t+k+1} denotes the receiving reward value after k 1 time-steps, qπ (s, a) denotes the function that links state to action value policy (or) Q function policy.

3.3 Checking Models with Probability Formula

Probabilistic model checking aims to verify whether the probability model satisfies the correct qualitative and quantitative features. This process involves determining the existence of specific conditions. Qualitative requirements were confirmed, as well as the probability that the quantitative requirements of other attributes were satisfied. PRISM, a symbolic model analyzer [19], is employed as a valuable tool for understanding systems with unpredictable or random behaviour and for formal modelling. It finds application in systems analysis across various domains, including e-business [28]. PCTL is employed to verify safety and reliability attributes.

Description 2: PCTL syntax is the state and path formula given to the following [29],

- – ψ:: = true | action | ψ ∧ ψ | ¬ψ | P~p [φ]
- – φ:: = ψ | φ ∧ φ | ¬ φ | X φ | φ U φ
- Where, action ∈ AP finite set of atomic propositions.
- Ψ and φ are the state and path formulas interconnected over the state and path of M.
- ¬ denote the boolean connective in the usual way.
- p ∈ [0,1] probability bounded and ~ ∈ {, ≤, ≥} probability operated functions.

4. Mobile Robot Agent Path Planning Based on Unpredictable Environment

In the conducted research, a path planning agent in an uncertain environment operated on the mobile robot, learning through RLA. Actions were taken in response to achieving the maximum possible reward upon reaching the goal point.

4.1 Modification of Q Learning Algorithm

As a modification to the desired location reward using the Q-learning algorithm, designers to its highest value. The agent then continues to explore its surroundings in search of the target spot, with the objective of quickly identifying the location with the maximum payout. This motivation drives the agent to locate the target as soon as

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) p. 81-94

possible. A reward is feedback the agent receives based on its activities and the resulting state at each stage of its path. It measures the immediate benefit of the agent's action. The "payout" refers to the total compensation an agent earns for a series of tasks or an episode. The payout, which is the sum of all incentives earned over time, represents the long-term gain of the agent's actions. A larger payout is the maximum rewarding result produced by the robot activity, such as a quicker arrival at the goal, the avoidance of obstacles, or the use of less energy. In essence, a higher reward suggests that the robot made a better or more successful choice.

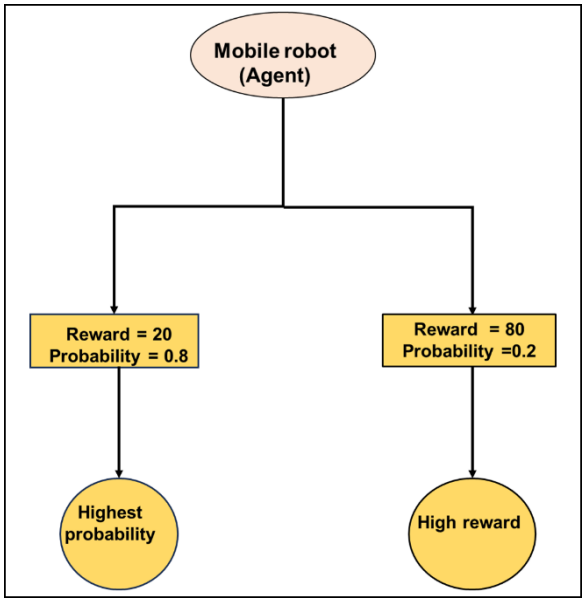

Fig. 2 *Mobile robot (agent) path planning finds for an unpredictable environment*

Illustrated in Fig. 2, the agent faces a choice between two paths, each offering different rewards and probabilities. The mobile robot (agent) represents the decision-making entity in an RLA. The agent's goal is to maximize its total reward over time by choosing the best actions based on the given probabilities and rewards. The path choices are as follows: the left path offers a reward of 20 with a high probability of 0.8. The label "Highest probability" indicates that this path is more likely to yield the expected reward. The right path offers a higher reward of 80 but with a lower probability of 0.2. The label "High reward" indicates that although the reward is significant, the likelihood of obtaining it is much lower compared to the left path. The agent must decide which path to take based on the trade-off between reward size and probability. The left path is safer and more reliable, providing consistent but smaller rewards. The right path is riskier, with a higher potential reward but a much lower chance of success. Consequently, the decision impacts learning performance and adaptability in dynamic environments. The primary contribution to the predicted gain is multiplied by the probability of achievement and integrated into the reward [30]. The Q-learning algorithm updates the reward matrix based on expected rewards, and the overall predicted gain value is calculated in the path planning formula, as expressed in Equation (5).

$$\text{Expected [Reward]} = \text{sum of probability success} * \text{reward} \tag{5}$$

Algorithm 1 pseudo-code for the Q-learning algorithm
Given: Mobile robot agent, state, action and reward
Determined: Q -table value and probability values
Step 1 Initial Q (S_t, A_t) arbitrarily for all state-action pairs
Step 2 Predictable reward matrix R (S_t, A_t)
Step 3 Continue (for each episode)
Step 4 Initialize (A_0, S_0)
Step 5 Continue (for each step by step of episode)
Step 6 Select A from S using the policy expression of Q
Step 7 Take Action A, State S, Reward R, and S_{t+1}
Step 8 Q_{new} (S_t, A_t)= Q (S_t, A_t) + β [R + δ max Q (S_{t+1}, A_t) − Q (S_t, A_t)]
Step 9 $S' \rightarrow S$
Step 10 Return to Q- table matrix.

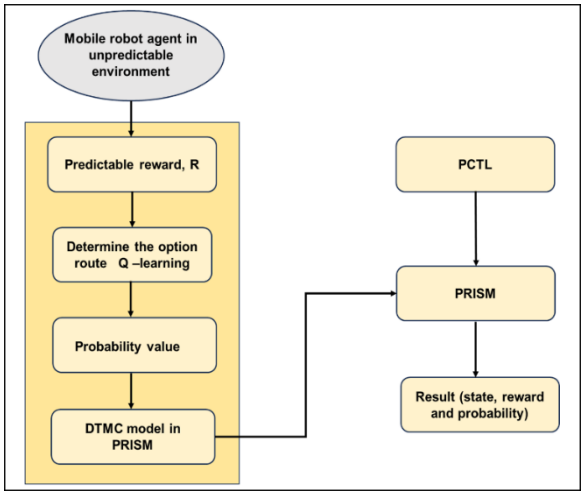

Fig. 3 *Framework in mobile robot agent path planning (MRAPP)*

Mobile agents in unpredictable environments of path planning are the subject of this work. The suggested integrated method's framework is displayed in Fig. 3. Using the modified Q-learning method, path planning is first accomplished. More precisely, random initial values are assigned to the environment's relevant characteristics, including the chances of successful passage, the reward for each state, and the possible actions the mobile agent can perform. Subsequently, the anticipated payout for every condition is computed to provide the expected R matrix, which is then fed into the Q algorithm. The Q-learning algorithm is a combination of expectation assignment and Q-learning. Every episode results in the creation of a fresh Q-table. The agent discovers the optimal route to reach the desired state after the Q-table converges. In other words, the agent can examine the converged Q-table directly, select the course of action with the highest predicted reward, go to the next stage, and then steadily advance until it reaches the desired state. Secondly, the chosen path (i.e., the MRAPP behaviour) and probability statistics are entered into a DTMC perfect as the input of the model checker PRISM to confirm the dependability qualities of this MAPP. Properties with PCTL are used to describe the requirements for MRAPP system reliability. They are employed in the computation of the probability of failed states occurring while the system is being executed. Next, by determining if the system can meet reliability requirements, PRISM is a tool that we can use to analyze the reliability of MRAPP. The result and discussion described in Section 5 provide additional examples of the framework.

5. Result and Discussion

In this section, the Q-learning algorithm is introduced, and the agents traverse a grid map where each grid corresponds to a room. The assumption is made that there are six rooms, with room 0 serving as the starting point and room 5 as the goal. An incentive value is granted to the mobile robot (agent) upon completing each room. Where ri mentions the rewards that can be obtained from the room, i is the number of rooms, i = {0, 1, 2..., 5}. Pi mentions the probability of the agent passing through the rooms.

5.1 Path Planning for Mobile Robots Using Q Learning Algorithm

The path planning for the identified room grid follows the sequence: room0, room1, room2, room3, room4, and room5. The primary objectives are to navigate from the starting point to the goal point via the shortest path and to identify the most efficient route, as illustrated in Fig. 4. depicting various grid mappings. The state and action values, namely 0, 1, 2, 3, 4, and 5, are represented in box number 1. The identification of -1 signifies null values, while 100 denotes direct links. Rewards of (100) are assigned to transitions (1,5), (4,5), and (5,5), and a value of 0 indicates a direct link with no rewards for transitions like (0,4), (1,3), (2,3), (3,1), (3,2), (3,4), (4,0), (4,3), (5,1), and (5,4). The remaining values are set to -1.

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) p. 81-94

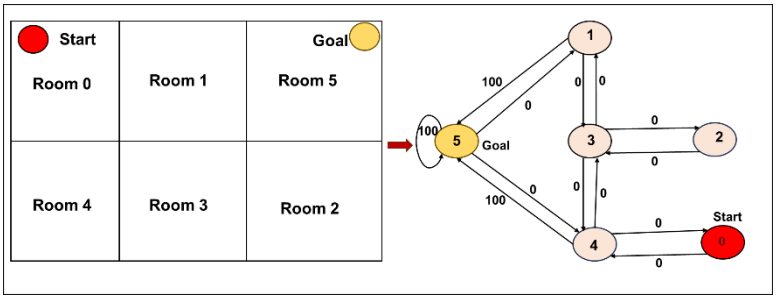

Fig. 4 *Different grids mapping*

Prism code for the Q learning algorithm

Module: 1 (M)

Step: 1

S:0 [0,5] initial 0;
[ ] S = 0 → (S′ = 4);
[ ] S = 4 → (S′ = 0) + 100 (S′ = 5);
[ ] S = 5 → (S′ = 4);

Step: 2

S: [0,5] initial 0;
[ ] S = 0 → (S′ = 4);
[ ] S = 4 → (S′ = 0) + (S′ = 3);
[ ] S = 3 → (S′ = 4) + (S′ = 1);
[ ] S = 1 → (S′ = 3) + 100 (S′ = 5);
[ ] S 5 → (S′ = 5);

Step: 3

S: [0,5] initial 0;

[ ] S = 0 → (S′ = 4);
[ ] S = 4 → (S′ = 0) + (S′ = 3);
[ ] S = 3 → (S′ = 4) + (S′ = 2) + (S′ = 1);
[ ] S = 2 → (S′ = 3);
[ ] S = 1 → (S′ = 3) + 100 (S′ = 5);
[ ] S = 5 → (S′ = 5);
end module.

The model was at a starting state 0 through 5, where states 1 through 5 denote the mobile agent moving from in step:1 was room1 and room5. S=1 (room1) to S= 5 (room5) to the 100 reward points are mentioned. The current room is represented on the left side of the symbol "→," and the space and probability which could be entered in the next phase are represented on the right side.

5.2 Mapping Modulation for Q Learning

The agents need to reach room 5 as quickly as possible. The agent is initially placed in room 0 (start). In the subsequent state, it can enter room 1 and room 5 (indicated by the blue line arrow). The agent has a 97.36% chance of successfully entering room 1, and if it does, it can receive 7 prizes. Agents that complete room 4 will receive a reward of 27, and their chances of passing are 89.36%. After exiting room 4 (indicated by the red line arrow), it can enter room 0, room 4, and room 3, depending on the state's locations. The agent has a 92% chance of entering room 3 and will receive 20. Fig. 5. illustrates how agents' grids map an unidentified environment. The likelihood of successfully passing through room 2 is 94.66%, and the reward is 13.

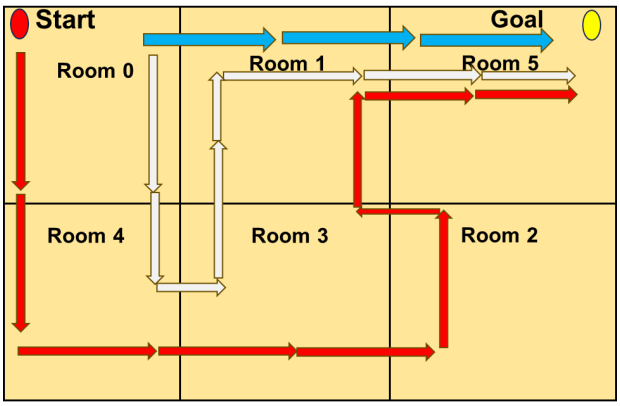

Fig. 5 *An agent grid mapping for probability and reward*

The rooms indicated by the white line arrow room0, room4, room3, room1, and room5 reached the goal point. When the agent reaches room 5, it has achieved its goal and completed one task. The Q-learning algorithm states that before the agent begins its investigation, an R-matrix with the reward value is required. In a broad setting, the designer needs to allow the agent to explore independently to calculate the quickest route to the highest reward. Since there are essential risks in every room in a stochastic and unpredictable situation, it is critical to weigh the likelihood of passing a particular room as well as the potential reward. To provide an R-matrix, the expected assignments approach is used, referred to as the R matrix and displayed in (6a) and (6b), which was previously suggested in Fig. 6.

$$R = \begin{array}{c} \text{Action} \\ \text{State} \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \quad \text{(6a)} \quad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array} \quad \text{(6b)}$$

Fig. 6 *Matrix size 6x6*

The mobile robot agent was R0, R1, R2, R3, R4 and R5 are room0, room1, room2, room3, room4 and room5 are mentioned in Fig. 7. The Q learning algorithm is found in the best route, and the total reward values are 1+7+33+27+20+13=101 and expected reward for total is 565.57. Most efficient path room1 to room5, reward values are 40 and the expected reward in equation (5) is 73.584.

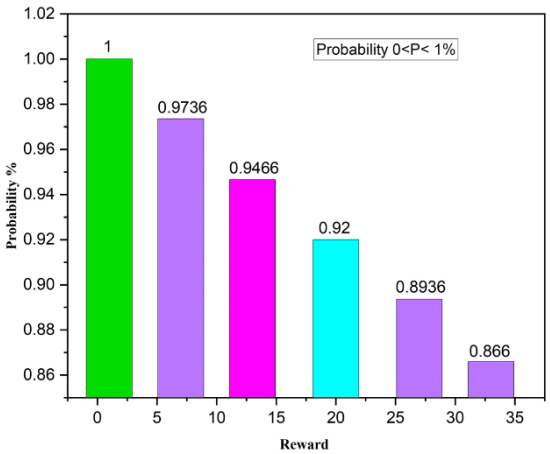

Fig. 7 *Mapping grids for probability and reward*

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) p. 81-94

Prism code for module: 2

Agent grids mapping mobile robot, Fig. 7.

S: [0,5] initial 0;

[ ] S = 0 → 0.92: (S′ = 3) + 0.866: (S′ = 5);

[ ] S = 1 → 0.9736:(S′ = 4) + 0.866: (S′ = 5);

[ ] S = 2 → 0.8936:(S′ = 4) + 0.9736:(S′ = 1) +0.866: (S′ = 5);

[ ] S = 3 → (S′ = 0) +0.866: (S′ = 5);

[ ] S = 4 → 0.9736: (S′ = 1) + 0.866: (S′ = 5) + 0.92 (S′ = 3) + 0.866: (S′ = 5);

[ ] S = 5 → 0.866 (S′ = 5);

end module.

The prism module grids mapped the mobile robots in the states from the first stage 0 to the final stage 5 in the framework of the prism module process.

5.3 Training and Simulation Results

Table 1 *Hyperparameters for the Q learning algorithm*

S No	Parameters	Value
1	Learning rate	0.01
2	Discount factor	0.99
3	Epsilon delay	0.005
4	Minimum epsilon	0.01
5	Batch size	64.0
6	Experience buffer length	1x104
7	Optimizer	Adam
8	Gradient decay	0.9
9	Target update frequency	1.0
10	Initial epsilon	1.0

The Q-learning algorithm's parameters are listed in Table 1. These settings determine the step size at each iteration while moving toward a minimum of the loss function. The learning rate (α) is set to 0.01. While a larger learning rate may expedite training at the expense of overshooting the minimum rate. With a value of 0.99, the discount factor (γ) signifies the significance of potential future rewards. Long-term benefits are prioritized by a value near 1, while immediate benefits are prioritized by a value closer to 0. The epsilon decay, which is set to 0.005, determines the pace of exploration in the initial phases of training, which is crucial for identifying the best policies. By setting the minimum epsilon to 0.01, the agent is guaranteed to continue exploring the environment to some degree even in the later stages of training. The batch size parameter, set to 64, defines the number of experiences sampled from the replay buffer to update the network weights. The experience buffer length parameter, which is set to 1x10^4, determines the size of the replay buffer that holds the agent's experience for training. The network weights are modified using the Adam optimizer, which is selected for its solid performance in various scenarios and flexible learning rate properties. The gradient decay, related to the momentum term in gradient-based optimization algorithms, is set to 0.9. This speeds up convergence and smooths out the optimization path. The parameter that controls the frequency at which the weights of the target and main networks are adjusted to match is called the target update frequency, set to 1.0. The initial epsilon value is 1.0, representing the initial chance of selecting a random action (exploration) over the most well-known action (exploitation). Beginning with a high level of exploration enables the agent to gather diverse experiences.

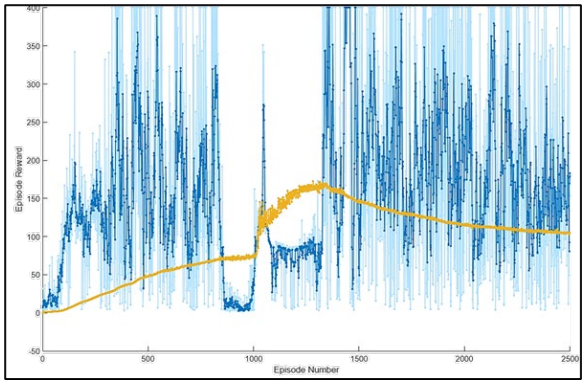

Fig. 8 *Q learning algorithm for episodes 2500*

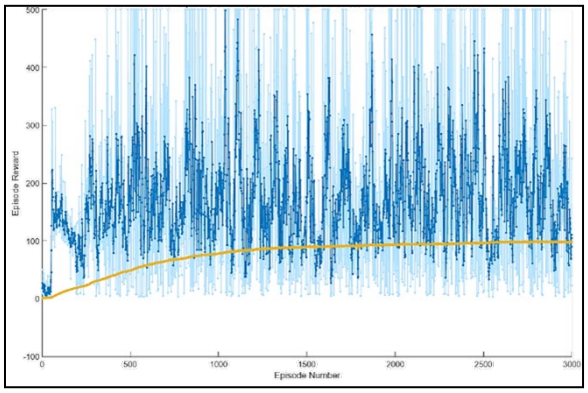

Fig. 9 *Q learning algorithm for episodes 3000*

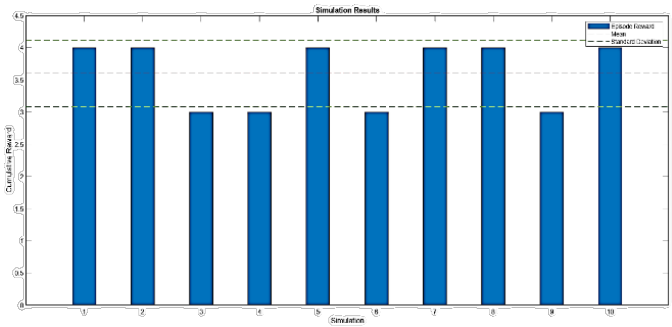

Fig. 10 *Simulation results*

Table 2 *Number of episode values for the Q learning algorithm*

S No	Parameters	Episodes values	
		2500	3000
1	Episode reward	153	23
2	Episode steps	159	29
3	Total agent steps	394199	527691
4	Average reward	182.8	89.8
5	Average steps	187.6	95.8
6	Episode initial Q value	105.1808	97.6509

Fig. 8. shows the Q-learning algorithm's performance across a sequence of training events (0–2500 episodes). The x-axis represents the episode number, while the y-axis shows the episode reward. Initially, during the first training phase (episodes 0-500), episode rewards are highly variable and relatively low. This is common in the initial

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) p. 81-94

phases of training when the agent's primary task is to explore the environment. The high variance suggests that the agent is experimenting to find the best paths of action. As training progresses (episodes 500-1500), there is a discernible rising trend in the episode rewards. This implies that the agent is gaining experience and increasingly using the learned policies to achieve better rewards. The graph illustrates the balance between exploration and exploitation with multiple spikes and troughs in rewards. This stage is critical as it allows the agent to continuously learn from both new and old experiences to improve its policies. Towards the end of the training period (episodes 1500-2500), the episode rewards begin to stabilize, suggesting that the agent has improved the consistency of its policy for maximizing rewards. While there are still sporadic swings, the variance in rewards decreases. These variations may result from ongoing exploration or changing environmental conditions. Around episode 1250, there is a noticeable drop in average rewards, followed by a recovery and further stabilization. This might result from a change in the agent's learning rate or strategy, requiring some time for relearning and adaptation. The yellow line in the plot represents the moving average of the episode rewards over a set window. This line smooths out short-term variations and highlights the general trend of the agent's performance.

In Fig. 9, which extends the training observation to 3000 episodes, the training performance (episodes 0-3000) remains variable; however, the moving average stabilizes. Despite continued exploration leading to inconsistent rewards, long-range patterns indicate that the agent appears to be approaching an optimal strategy, as the moving average line shows steady improvement, albeit at a slower rate. The agent's methodical routine of alternating between exploitation and exploration is indicated by the points and drops. The plots illustrate the typical learning behaviour of a reinforcement learning algorithm: an initial phase of intense exploration followed by a period of learning and policy refinement. The moving average lines provide a smoother perspective of the agent's performance, emphasizing the overall trend of improvement despite episodic fluctuations. The Q-learning simulation results are illustrated in Fig. 10, showing an average reward of 182.8 and average steps of 187.6 for episode 2500. Similarly, for episode 3000, the results are presented in Table 2.

Table 3 *Simulation results for mobile robot systems*

Room No	Sample					Probability
	200	400	600	800	1000	
1	0.98	0.987	0.985	0.975	0.976	0.98
2	0.024	0.034	0.025	0.027	0.026	0.0267
4	0.045	0.055	0.046	0.052	0.055	0.0545

Table 3 represents the simulation results of mobile robot systems. The likelihood that the agent would have finally failed was indicated by row (1), which also included simulation and verification results for various sample sizes. According to the verification results, there was a 0.98 chance of the agent failing in the learned model, which was 0.0267 and 0.0545. The chance of the agent entering room 5 was examined. The consistency of the verification results with actual conditions was explained by using the simulation approach. The flows of many transitional pathways depicted the decisions the agent had made, while the simulated samples represented discrete states. The range of samples employed in the simulation technique in this experiment was 200 to 1000.

5.4 Discussion

The simulation results were based on Python 3.12.0 coding, utilizing the Q learning algorithm and numpy library as np. The matrix size was 6x6, with actions starting from 0, 1, 2, 3, 4, and 5, and states ranging from 0 to 5. The initial gamma value was mentioned as 0.85, and the learning parameters at the initial state were set to 0. At the current state of 1, the mobile agent learned from the Q learning algorithm, achieving a maximum reward of 84.99 and a minimum reward of 61.41. The most efficient path was reached by the mobile agent at (1,5). In the second step, the gamma value remained at 0.85, and the current state was 2. The most efficient path planning for the mobile robot agent involved learning from the sequence (2,3,4,5). The third step indicated that the gamma value remained the same, and the current state was 4. The most efficient path reached by the mobile agent was (4,5), as depicted in Fig. 11. The simulation and experiment were completed in Python 3.12.0, 64bit operating system (AMD64) on win32, AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz, 8.00 GB Windows 11. Home Insider Preview Single Language and PRISM tool used for PRISM 10.0.

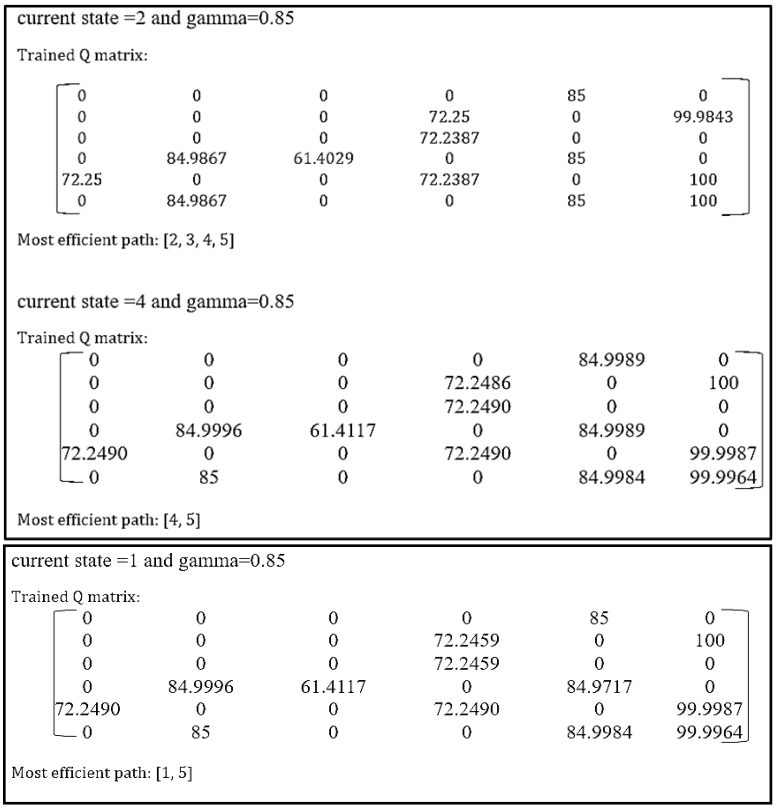

Fig. 11 *Trained in Q learning algorithm matrix size 6x6*

6. Conclusion and Future Scope

In this research work, we explored the application of RLAs for probability model checking in the context of mobile robot agent path planning in uncertain environments. The main objective was to modify the Q-learning algorithm to establish a robust probability matrix and develop a probability model for path selection. The robot agent chose locations based on the maximum Q-table values in the probability model created using the modified Q-learning technique. An MDP model was used to depict the mobile robot agent's learned behaviour obtained through Q-learning. This representation allowed the decision-making process of the agent to be modelled in an organized and mathematically correct manner. PCTL was used to define the dependability standards of the mobile robot agent control system. PCTL provided a formal framework for defining and confirming the robot agent's dependability and performance in various scenarios.

To enable automated verification, the MDP model and its assigned attributes were entered into the PRISM. The method worked well for choosing the best control model and goal position to assess dependability, performance, and feasibility. From the PRISM model training, for episodes between 2,500 and 3,000, the average reward and average steps obtained were 182.8 187.6, and 89.8 and 95.8, respectively. The mobile agent proved that the updated Q-learning algorithm was successful in producing the intended results. During the simulation, the PRISM performance achieved a maximum reward of 84.99 and a minimum reward of 61.41. Future research could explore the use of multi-agent robots and various algorithms such as enhanced learning, hierarchical reinforcement learning, deep recurrent Q-network algorithms, real-time adaptation and learning, robust policy learning, transfer learning, and human-robot interaction.

Acknowledgement

The authors would like to thank the Department of Mechanical Engineering, Saveetha School of Engineering supporting this project.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm their contribution to the paper as follows: ***study conception and design:*** *Vengatesan Arumugam.* ***Data collection****: Vasudevan Alagumalai;* ***analysis and interpretation of results****: Vengatesan Arumugam* ***draft manuscript preparation****: Vengatesan Arumugam, Vasudevan Alagumalai. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Utami, N. S., Jazidie, A., & Kadier, R. E. A. (2019). Path Planning for Differential Drive Mobile Robot to Avoid Static Obstacles Collision using Modified Crossover Genetic Algorithm. 2019 International Seminar on Intelligent Technology and Its Applications (ISITIA). https://doi.org/10.1109/isitia.2019.8937184

[2] Ryou, G., Sim, Y., Yeon, S. H., & Seok, S. (2018). Applying Asynchronous Deep Classification Networks and Gaming Reinforcement Learning-Based Motion Planners to Mobile Robots. 2018 IEEE International Conference on Robotics and Automation (ICRA). https://doi.org/10.1109/icra.2018.8460798

[3] Mohanty, P.K. (2024). Path Planning of Mobile Robots Under Uncertain Navigation Environments Using FCM Clustering ANFIS. *Wireless Personal Communication 137*, 1251–1276. https://doi.org/10.1007/s11277-024-11463-y

[4] Li, Z., Davis, J., & Jarvis, S. A. (2018). Optimizing Machine Learning on Apache Spark in HPC Environments. 2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC). https://doi.org/10.1109/mlhpc.2018.8638643

[5] Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). The Jason Agent Programming Language. (n.d.). *Programming Multi-Agent Systems in Agent Speak Using Jason*, (pp. 31–66). John Wiley & Sons. https://doi.org/10.1002/9780470061848.ch3

[6] Rath, M., & Pattanayak, B. K. (2019). Security Protocol with IDS Framework Using Mobile Agent in Robotic MANET. *International Journal of Information Security and Privacy*, 13(1), 46–58. https://doi.org/10.4018/ijisp.2019010104

[7] Carlucho, I., De Paula, M., & Acosta, G. G. (2019). Double Q-PID algorithm for mobile robot control. *Expert Systems with Applications*, *137*, 292–307. https://doi.org/10.1016/j.eswa.2019.06.066

[8] Nazarahari, M., Khanmirza, E., & Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, *115*, 106–120. https://doi.org/10.1016/j.eswa.2018.08.008

[9] Zhang, Y., & Pang, D. (2022). Research on Path Planning of Mobile Robot Based on Improved Ant Colony Algorithm. 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC). https://doi.org/10.1109/itoec53115.2022.9734356

[10] Haney, B. S. (2020). Deep Reinforcement Learning Patents: An Empirical Survey. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3570254

[11] Cui, Y., Hu, W., & Rahmani, A. (2022). A reinforcement learning based artificial bee colony algorithm with application in robot path planning. *Expert Systems with Applications*, *203*, 117389. https://doi.org/10.1016/j.eswa.2022.117389

[12] ter Beek, M.H., Larsen, K.G., Ničković, D., & Willemse, T. A. C., (2022). Formal methods and tools for industrial critical systems. *International Journal on Software Tools for Technology Transfer*, *24*, 325–330. https://doi.org/10.1007/s10009-022-00660-4

[13] Lamini, C., Benhlima, S., & Elbekri, A. (2018). Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Computer Science*, *127*, 180–189. https://doi.org/10.1016/j.procs.2018.01.113

[14] Papakostas, N., Mourtzis, D., Bechrakis, K., Chryssolouris, G., Doukas, D., & Doyle, R. (2023). A flexible agent-based framework for manufacturing decision-making. *Proceeding of Flexible Automation and Integrated Manufacturing 1999*. https://doi.org/10.1615/faim1999.670

[15] Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2020). Graph Neural Networks for Decentralized Multi-Robot Path Planning. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). https://doi.org/10.1109/iros45743.2020.9341668

[16] Wang, B., Liu, Z., Li, Q., & Prorok, A. (2020). Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning. *IEEE Robotics and Automation Letters*, *5*(*4*), 6932–6939. https://doi.org/10.1109/lra.2020.3026638

[17] Chang, L., Shan, L., Jiang, C., & Dai, Y. (2020). Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Autonomous Robots*, *45*(*1*), 51–76. https://doi.org/10.1007/s10514-020-09947-4

[18] Franco, J. M., Correia, F., Barbosa, R., Zenha-Rela, M., Schmerl, B., & Garlan, D. (2016). Improving self-adaptation planning through software architecture-based stochastic modeling. *Journal of Systems and Software*, *115*, 42–60. https://doi.org/10.1016/j.jss.2016.01.026

[19] Kwiatkowska, M., Norman, G., Parker, D. (2011). PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds) Computer Aided Verification. CAV 2011. Lecture Notes in Computer Science, vol 6806. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22110-1_47
[20] Hahn, E. M., Hermanns, H., & Zhang, L. (2010). Probabilistic reachability for parametric Markov models. *International Journal on Software Tools for Technology Transfer*, *13*(*1*), 3–19. https://doi.org/10.1007/s10009-010-0146-x
[21] Votion, J., Feng, T., & Cao, Y. (2021). Risk-Aware Multi-Agent Path Planning for Target Detection: A Multi-Agent Reinforcement Learning Approach. AIAA Scitech Forum 2021-0267, Session: Autonomy II - Multi-Agent Systems. https://doi.org/10.2514/6.2021-0267
[22] Kwiatkowska, M., Norman, G., & Parker, D. (2022). Probabilistic Model Checking and Autonomy. *Annual Review of Control, Robotics, and Autonomous Systems*, *5*(*1*), 385–410. https://doi.org/10.1146/annurev-control-042820-010947
[23] Baier, C., de Alfaro, L., Forejt, V., & Kwiatkowska, M. (2018). Model Checking Probabilistic Systems. In: Clarke, E., Henzinger, T., Veith, H., Bloem, R. (eds) *Handbook of Model Checking*. 963–999. Springer, Cham. https://doi.org/10.1007/978-3-319-10575-8_28
[24] Kwiatkowska, M., Norman, G., & Parker, D. (2018). Probabilistic Model Checking: Advances and Applications. In: Drechsler, R. (eds) *Formal System Verification*. 73–121. Springer, Cham. https://doi.org/10.1007/978-3-319-57685-5_3
[25] Lou, W., & Xia, C. (2015). Mobile Robot Path Planning based on Probabilistic Model Checking under Uncertainties. Proceedings of the 2015 3rd International Conference on Machinery, Materials and Information Technology Applications. In: *Advances in Computer Science Research*. https://doi.org/10.2991/icmmita-15.2015.265
[26] Watkins, C.J.C.H., Dayan, P. (1992). Q-learning. *Machine Learning 8*, 279–292. https://doi.org/10.1007/BF00992698
[27] Ethier, S.N. & Kurtz, T.G. (1986). *Markov processes: characterization and convergence*. John Wiley & Sons. https://doi.org/10.1002/9780470316658
[28] Mizuno, T., & Nishizaki, S.-Y. (2014). Model checking and analysis of systems dependent on CPU speed. In: *E-Commerce, E-Business and E-Service*. 189–194. CRC Press. https://doi.org/10.1201/b17084-37
[29] Ciesinski, F., & Größer, M. (2004). On Probabilistic Computation Tree Logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, JP., Siegle, M. (eds) Validation of Stochastic Systems. *Lecture Notes in Computer Science*. vol 2925, 147–188. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24611-4_5
[30] Kasser, J. E. (2018). Decisions and Decision-Making. In: *Systems Thinker's Toolbox*. 113–169. CRC Press. https://doi.org/10.1201/9780429466786-4