Mobile agent path planning under uncertain environment using reinforcement learning and probabilistic model checking

Mobile Agent Path Planning under Uncertain Environment Using Reinforcement Learning and Probabilistic Model Checking

Xia Wang[a,b], Jun Liu[b], Chris Nugent[b], Ian Cleland[b], Yang Xu*[c]

[a]*School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 610031, China*
[b]*School of Computing, Ulster University, Northern Ireland BT15 1ED, UK*
[c]*School of Mathematics, Southwest Jiaotong University, Chengdu 610031, China*

Abstract

The major challenge in mobile agent path planning, within an uncertain environment, is effectively determining an optimal control model to discover the target location as quickly as possible and evaluating the control systems reliability. To address this challenge, we introduce a learning-verification integrated mobile agent path planning method to achieve both the effectiveness and the reliability. More specifically, we first propose a modified Q-learning algorithm (a popular reinforcement learning algorithm), called $Q^{EA}-learning$ algorithm, to find the best Q-table in the environment. We then determine the location transition probability matrix, and establish a probability model using the assumption that the agent selects a location with a higher Q-value. Secondly, the learnt behaviour of the mobile agent based on $Q^{EA}-learning$ algorithm, is formalized as a Discrete-time Markov Chain (DTMC) model. Thirdly, the required reliability requirements of the mobile agent control system are specified using Probabilistic Computation Tree Logic (PCTL). In addition, the DTMC model and the specified properties are taken as the input of the Probabilistic Model Checker PRISM for automatic verification. This is preformed to evaluate and verify the control system's reliability. Finally, a case study of a mobile agent walking in a grids map is used to illustrate the proposed learning algorithm. Here we have a special focus on the modelling approach demonstrating how PRISM can be used to analyse and evaluate the reliability of the mobile agent control system learnt via the proposed algorithm. The results show that the path identified using the proposed integrated method yields the largest expected reward.

Keywords: Expected reward, mobile agent, uncertain environment, probabilistic model checking, $Q-learning$

1. Introduction

An agent is an entity that can think independently and interact with its environment [1]. It has the characteristics of persistence, reactivity, sociality, and initiative. A mobile agent is an entity that can reach a target location by location movement or state transfer in a certain environment. Artificial intelligence research has made extensive use of mobile agents. Most of the researches, particularly in the development of robotics, has been centred on mobile agents [2, 3, 4]. The mobile agent control system is able to provide the corresponding strategies for the agent movement so that it can complete the task in a better way, such as the shortest time and the safest way. Strategies can be generated by collecting data, integrating information, etc [5, 6].

Existing agent path planning algorithms, e.g., genetic algorithm [7], reinforcement learning [8], and ant colony algorithm [9], are capable of locating the ideal path in a static environment. Reinforcement Learning (RL) is one of the paradigms and methodologies of machine learning, which is used to describe and address the issue of how an agent learns strategies to maximize rewards or achieve specific goals while interacting with the environment. RL focuses on online learning and attempts to maintain a balance between exploration-exploitation [10]. RL, in contrast to supervised learning and unsupervised learning, does not require a large amount of data to be collected beforehand; instead, it acquires learning information and modifies model parameters by receiving rewards (feedback) from the environment in exchange for actions. RL can successfully identify the quickest path to obtain the maximum reward value in a broad context. Nevertheless, in a stochastic uncertain environment, some optimal paths might change or encounter some unexpected events. Stochastic uncertainty refers to the potentials of anomalies or emergencies to occur at specific locations in the environment as a result of a variety of factors, including hardware and software failures, human interferences, and natural disasters. Path planning for mobile agents in uncertain environments will thus become more challenging.

On the other hand, it is crucial to research the safety development of mobile agent control systems. The usage of the formal methods to evaluate the credibility of intelligent control systems is growing as a result of the shortcomings of testing and simulation techniques [11]. The formal methods are techniques for mathematically exploring important computer science problems and offering a framework for creating and mathematically verifying systems. Through reasoning and verification, formal system development uses a predetermined language to describe a system and make sure it satisfies all the requirements. For the creation of the mobile agent control system to ensure accurate behaviour, interoperability, and safety, formal methods are therefore required. Model checking is one of the primary and popular formal verification technique, which verifies system specifications on finite-state concurrent systems. In most cases, temporal logic is utilised to specify system requirements. Symbolic-based model checking algorithms are then employed to determine whether those requirements are satisfied by the system model. In addition, quantitative verification approaches such as probabilistic model checking have lately gained popularity due to their ability to manage the uncertainty. Designers can use probabilistic model checking to assess whether a system's underlying probabilistic behaviour is satisfied by a set of quantitative properties. PRISM [12] is a popular and powerful symbolic model checker intended for the formal modelling and analysis of systems that exhibit stochastic or probability behaviour. This present work benefits from the PRISM model checker to complete the verification of Mobile Agent Path Planning (MAPP). DTMC is available to represent a decentralized Markov model suitable for representing the state transfer of an agent. Furthermore, DTMC models can be transformed into a customized input format to accomplish probabilistic model checking in PRISM.

For the reasons stated above, the contributions of this paper are summarized below.

- A learning and verification integrated framework for mobile agent path planning and control under stochastic uncertain environments is proposed with the aim of achieving both the effectiveness and the reliability.
- The Expected Assignment Q-learning (Q^{EA}-learning) algorithm is proposed for effective MAPP under uncertain environment, where the agent is allowed to explore the environment and find the path to the target location with the maximum expected reward.
- After the optimal path is obtained by Q^{EA}-learning algorithm, the mobile agent control

system is formalized as a DTMC model. The required properties are specified by PCTL, and the system model is verified by PRISM in terms of properties.

- The applicability and effectiveness of the integrated method is demonstrated via the case studies and comparative experimental results show the advantage of the proposed method over the traditional methods.

The remainder of this paper is organized as follows. The background and related works are provided in Section 2. Preliminaries are presented in Section 3. Section 4 describes the learning-verification integrated framework for mobile agent path planning. A case study is detailed to illustrate the proposed framework in Section 5. Section 6 discusses the effect of the proposed method by comparing it with two other methods. Finally, Section 7 brings the paper to a close while outlining potential future research.

2. Related works

In the literature, there has been a lot of work conducted on the agent's movement. This includes the use of learning algorithms to help the agent reach the target state rapidly [13]. Nevertheless, insufficient attention has been paid to statistically verifying the learned models to account for agent state transition correlations. This section reviews current research on agent movement and path planning and related research on model checking for verifying probabilistic models in an uncertain environment.

2.1. Mobile agent path planning

There has been a lot of research on path planning for agent movement. Zhang *et al*. conducted the modeling, optimization criteria, and solution algorithms for the path planning of mobile agents [14]. Nestinger described a mobile agent-based architecture for dynamic control algorithm and task placement in an automation system [15]. Nazarahari *et al*. provided a hybrid solution for multiple mobile robot path planning in continuous situations [7]. Lamini *et al*. presented an improved crossover operator for employing genetic algorithms to tackle path planning problems [16]. Li *et al*. presented a unified model for robots navigating in confined workspace that automatically synthesises local communication and decision-making policies [17]. He and colleagues also proposed the Message-Aware Graph Attention neTwork (MAGAT) that is comparable to that of a coupled centralized expert algorithm [18]. Wang *et al*. proposed the globally Guided Reinforcement Learning approach (G2RL) where the agent is able to exploit environmental spatio-temporal information [19]. The method can be used for path planning of multiple agents, and the control of agents is completely distributed. Dai *et al*. presented an improved ant colony algorithm for mobile robots to obtain efficient path planning capabilities in intricate maps [20]. The A* algorithm and the MAX-MIN Ant system are used in the improved ant colony algorithm. Chang *et al*. proposed an improved Dynamic Window Approach (DWA) based on Q-learning to enhance the performance of the robot for global navigation [21]. Song *et al*. suggested a real-time obstacle avoidance decision model based on Machine Learning (ML) algorithms, an improved Smooth Rapidly exploring Random Tree (S-RRT) algorithm, and an improved Hybrid Genetic Algorithm-Ant Colony Optimization (HGA-ACO) to improve the accuracy of real-time obstacle avoidance prediction of local path planning [22]. Research on path planning for agents in stochastic environments is still inadequate, and it is also critical to assure the agent's safety and temporal property during movement [23].

2.2. Probabilistic model checking

As an automatic verification technique, model checking has been used to verify various desirable properties, such as deadlock freedom, safety, and reachability. It is worth noting that verification of quantitative properties of control systems in uncertain environments is as important as verification of qualitative properties. Probabilistic model checking [24] extends model checking, which enables the verification of quantitative assessment of the system, such as probabilistic behaviour, constraint behaviour, and adaptiveness. Probabilistic model checking has been effectively used in several fields. In the research area of robotics, Zhao *et al*. developed a framework for probabilistic model checking on a layered Markov model to verifying the safety and reliability requirements of robotics, both at permission stage and during runtime [25]. Depending on the type of knowledge about the uncertainties and imperfections in the operator autonomy interactions, Feng *et al*. used abstractions based on Markov decision processes and augmented these model to stochastic two-player games [26]. Wang *et al*. proposed to use the method of probabilistic model checking to verify and quantitatively analyze the obstacle avoidance strategy of mobile robots in non-deterministic environments [27]. Dimitrios *et al*. demonstrated how time-bounded path property model checking can be preformed as a Bayesian inference [28]. Compared with statistical model checking, the method has high accuracy and significant computational power. Nathalie *et al*. provided an algorithm for performing Fault Maintenance Trees (FMT) abstraction to reduce the size of its equivalent, as well as a framework for modeling FMT using probability model checking [29].

For industrial applications, Zhu *et al*. formalized the fetch process of express delivery system between business and customer in the form of DTMC and introduced the PCTL to the temporal behaviour checking [30]. Franco *et al*. used a DTMC to create a probabilistic model that accounted for various parameters linked to cloud-infrastructure execution traces and possible tactics. For example, if there are at least two active servers and one of them fails, the system will shut down the less trustworthy ones. After that, the model is sent into the PRISM model checker, which verifies the infrastructure's stability before recommending the optimum configuration plan [31]. For online transactions, Gao and colleagues used a probabilistic model to verify the uncertainty of user behaviours by computing probabilities for financial production risk evaluation and control, and they formalised investor behaviour into a DTMC model that can accurately describe the probability profiles of investors' purchases and redemptions. They employed the PCTL to predict whether users would engage in purchasing or redemption activity [32]. They also took the method into the medical insurance. Gao *et al*. presented a probabilistic model checking-based strategy for predicting Alzheimer's patient actions and detecting anomalous behaviour in accordance with mild cognitive impairment in order to assist patients in regaining confidence [33].

Several research apply probabilistic model checking in an indirect manner. Ouchani *et al*. proposed a probabilistic abstraction framework to efficiently verifying SysML activity diagrams [34]. Wan *et al*. analyzed quantitative and uncertain properties and presented a new probabilistic model checking of epistemic multi-agent systems specified by Probabilistic Computation Tree Logic and Knowledge (PCTLK) [35]. They transformed the PCTLK model checking problem into the PCTL one. This work is performed in PRISM, and compared with MCMAS and MCK, the results show the time efficient and scalability of the PRISM. Sultan *et al*. verified Multi-agent Systems (MASs) where agents have knowledge and communicate through manipulating uncertain social commitments via probabilistic model checking, especially when the scope of commitments extends beyond the common agent-to-agent scheme [36].

In this research, we use a learning method to pick the path of an agent operating in a stochastic uncertain environment, then construct a DTMC model for the control system operating under that path, and verify the model using PRISM.

3. Preliminaries

Some of the relevant backgrounds that we need to know are given in this section.

3.1. Q_learning

Q_learning is a typical algorithm of RLs for agents to learn how to act the correct actions to achieve a goal [37]. Q-table in the Q-learning is a table where we calculate the maximum expected future rewards for actions at each state. In the Q-learning, a reward matrix and a Q-table with all zeros are presented first. The three fundamental components of learning are states, actions, and rewards. Each state represents a location. As the agent explores with (1) [37], the value of the Q-table is updated until the results converge, and the value of the Q-table no longer varies.

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma max_a Q(s_{t+1}, a_t) - Q(s, a_t)] \tag{1}$$

Where s_t denotes state at the time t. s_{t+1} denotes a new state at the time $t+1$. a_t indicates the optional actions at the time t. α is the learning rate, while $0 < \alpha < 1$. γ is the discount factor, indicating the influence of future states on the present state. The bigger γ indicates that the future state has a greater influence on the current choice, and vice versa.

3.2. Markov model

When a stochastic process possesses the Markov property, the conditional probability distribution of its future state depends solely on the current state, given the current state and all previous states.

Definition 1. *(Markov process)[38]. Suppose the* $\{X(t), t \in T\}$ *is a stochastic process,* $\mathbf{E}$ *is the state space, if* $\{t_1 < t_2 ... < t_n < t\}$*, any* $x_1, x_2, ..., x_n, x \in \mathbf{E}$*, the conditional distribution function of the random variable* $X(t)$ *under the known variable* $X(t_1) = x_1, ..., X(t_n) = x_n$ *is only related to* $X(t_n) = x_n$*, but not related to* $X(t_1) = x_1, ..., X(t_{n-1}) = x_{n-1}$*. That is, the conditional distribution function satisfies the equation (2). If* $X(t)$ *is a discrete random variable, then the Markov property satisfies the equation (3).*

$$F(x, t|x_n, x_{n-1}, ..., x_2, x_1, t_n, t_{n-1}, ..., t_2, t_1) = F(x, t|x_n, t_n) \tag{2}$$

$$P\{X(t) = x|X(t_n) = x_n, ..., X(t_1) = x_1\} = P\{X(t) = x|X(t_n)\} \tag{3}$$

DTMC is a special form of Markov process, which have discrete values in the general definition by implementing the time parameters and the state space.

Definition 2. *(DTMC)[39]. A DTMC is a tuple* $\mathbb{D} = (S, \overline{s}, P, L, AP)$*. Where,*

- S is a finite set of states.

- $\overline{s}$ denotes the initial state.
- P is a probabilistic transition function, such that for every state $s \in S$, and $\Sigma_{s' \in S} P(s, s') = 1$.
- $L : S \rightarrow 2^{AP}$ is a labelling function that assigns a set of atomic propositions to each state $s \in S$.
- AP is a finite set of atomic propositions.

3.3. Probabilistic model checking

The purpose of probabilistic model checking is to confirm that the probabilistic model satisfies the appropriate qualitative and quantitative properties. It entails determining if several certain properties are qualitatively satisfied as well as the probability that other properties are quantitatively satisfied. The probability symbolic model checker PRISM [12] was created at the University of Birmingham and improved at the University of Oxford. It is a useful tool for formal modeling and the study of systems that behave in a stochastic or probabilistic way. It has been used to examine systems in a variety of application domains, including e-business [40], communication protocols [41], security protocols [42], etc [43]. The model in PRISM subsumes some well-known temporal logics by including PCTL, Continuous Stochastic Logic (CSL) [44], and Probabilistic Logic (PL) [45]. PCTL is a well-known temporal logic for probabilistic verification of DTMC models. The safety and reliability properties to be checked can be specified in PCTL.

Definition 3. *(PCTL syntax). Given a set of atomic propositions AP, the PCTL formulae are defined by the following BNF grammar [46]. Where,*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbb{P}_{\bowtie m}(\psi)$$

$$\psi ::= X\varphi \mid F\varphi \mid \varphi U \psi \mid \varphi U^{\leq k} \psi$$

- $p \in AP$ is a finite set of atomic propositions.
- φ and ψ are the state formulae and path formulae interpreted over the states and paths of M, respectively.
- $\neg$ and $\vee$ are the boolean connectives that are defined in the usual way.
- $\mathbb{P}_{\bowtie m}(\psi)$ is a probabilistic operator where $\bowtie \in \{<, \leq, \geq, >\}$ and $m \in [0, 1]$ is a probability bound or threshold.
- $k \in \mathbb{N}^+$ is a positive integer number reflecting the maximum number of transitions needed to reach a certain system.
- X, F, U are defined as 'next', 'finally', and 'until', respectively.

A state s in a DTMC model meets an atomic proposition p if $p \in AP$. s satisfies a state formula $\mathbb{P}_{\bowtie m}(\psi)$, denotes $s \models \mathbb{P}_{\bowtie m}(\psi)$, if the probability of taking a path starting from s and satisfying ψ satisfies the bound $\bowtie m$. The path formula $\varphi U^{\leq k} \psi$ is true on a path if ψ holds in the state at several time steps $i \leq k$ and at all preceding states φ holds.

4. Mobile Agent Path Planning Under Uncertainty

In the present work, we are interested in the MAPP system within uncertain environments and the verification of the system's reliability, i.e., how reliable it controls the behaviour of an agent in an uncertain environment.

4.1. MAPP based on the modified Q-learning algorithm

In the traditional Q-learning algorithm, the designers set the reward of the target location to the maximum, and the agent just keeps exploring the environment to find the target location that can obtain the maximum reward as soon as possible. This encourages the agent to find the target location as quickly as possible. However, agent path planning based solely on reward values is no longer appropriate in stochastic uncertain environments. Fig. 1 illustrates a problem scenario. An agent has two choices in the current scenario: If the agent chooses to go up, it gets 50 rewards. If the agent goes down, it can get 25 rewards. However, at the beginning, it is not known how many rewards can be obtained, and the agent can only gain experience by attempting. In most cases, the agent will choose the path with the largest reward. It is worth noting that this is also where the probability value reflecting the ability to pass the path is determined. For example, a probability of 0.8 for the above path indicates that the agent has an 0.8 chance of successfully completing the path and receiving the reward. The probability of successfully passing the bottom path is only 0.6. Obviously, taking the above path has a higher probability of getting a reward. However, in doing so will decrease the chances of getting a bigger reward. As a result, we must make choice of which path to select, this will affect the learning performance.

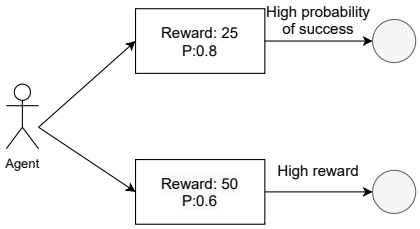

Figure 1: An agent finds a path in an uncertain environment.

To address this selection issue, we propose to use the concept of the expected reward, which is the reward multiplied by the probability of success. The reward matrix in the Q-learning algorithm is then updated using the expected rewards. The assignment method of expected reward is inspired by the Expected Monetary Value (EMV) [47]. The total expected reward value of a path can then be expressed as Eq.(4).

$$E[R] = \Sigma p_i r_i \tag{4}$$

$E[R]$ indicates the total expected reward of the planning path. p_i represents the probability of successfully passing the state i. r_i signifies the reward that can be obtained by passing the state i. This idea is based on expectation assignment, which leads to a modified Q-learning algorithm based on the expected reward, called $Q^{EA} - learning$ algorithm. The agent's exploration in the environment will follow the steps given in the $Q^{EA} - learning$ algorithm, which are detailed below, and its pseudo-code is given in Algorithm 1.

Algorithm 1 Expectation Assignment $Q^{EA}-learning$ algorithm

Require: Mobile agent, state, reward.
Ensure: $Q^{EA}-table$ and probability value
Initial $Q^{EA}(s_t, a_t)$ arbitrarily;
Expected reward matrix $R^{EA}(s_t, a_t)$;
Repeat (for each episode):
  Initialize s_0;
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q;
    Take action a, observe r, s_{t+1};
    $Q^{new}(s_t, a_t) \leftarrow Q^{EA}(s_t, a_t) + \alpha[r + \gamma max_a Q^{EA}(s_{t+1}, a_t) - Q^{EA}(s, a_t)]$
    $s \leftarrow s'$;
  until s is terminal;
Return $Q^{EA}-table$.

Step 1 : Given an initialized Q-table matrix with all zeros;

Step 2 : Each room's expected reward value is computed, and $R^{EA}(s_t, a_t)$ is listed based on the successful probability and available rewards. $R^{EA}(s_t, a_t)$ represents the expected reward matrix.

Step 3 : The agent stays at the initial state and chooses an action to move to the next state. Observe the next state and the optional action of the next state. Use Eq. (1) to update the Q-table once. If the target state is not found, then repeat this step until the target state is found, end the episode once.

Step 4 : Each time after reaching the target state, it represents the completion of one exploration. The agent returns to its initial position and starts a new round of exploration. The $Q^{EA}-table$ is updated once for each exploration until converges and stops.

Step 5 : Print the last $Q^{EA}-table$.

An DTMC model can be created to be utilised as an input model to the model checker PRISM. The module defines the state transfer statements. First, all of the variables and comments in this module are defined. The format of the comments is: $[action]guard- > rate_1 : updated_1 + ... + rate_n : updated_n$ [12]. The module synchronization is done using the action. If the action synchronizes and the guard is fulfilled, one update based on the value of the rate will occur. Each update demonstrates a change that occurred. To avoid code duplication, the PRISM model defines formulas with names and expressions. The reliability related properties are specified into logical formulas using PCTL, they are used to calculate the likelihood of the occurrence of failed states. Then, we can use PRISM to perform reliability analysis for MAPP by checking whether the system can satisfy reliability requirements.

4.2. The integrated learning and verification framework

Path planning for mobile agents in uncertain environments is the subject of this work. Fig. 2 shows the framework of the proposed integrated method.

Path planning is firstly achieved through the modified Q-leaning algorithm. More specifically, the initial values for the environment's relevant parameters, such as the reward for each state, the probability of successful passage, and the actions that the mobile agent can take, are allocated randomly. Then, the expected reward for each state is calculated to derive the expected R matrix and used as input to the Q algorithm. The algorithm which combines

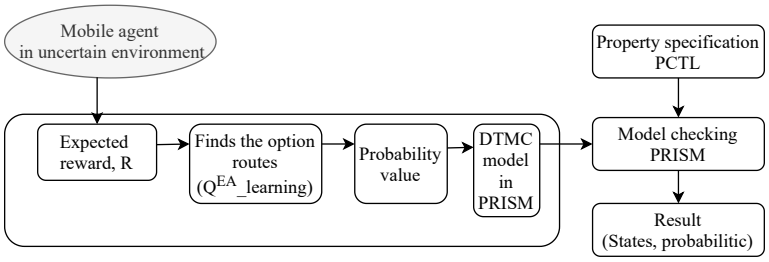

Figure 2: The integrated learning and verification framework for MAPP.

expectation assignment and Q-learning is called $Q^{EA} - learning$ algorithm. A new $Q^{EA} - table$ will be generated after each episode. Once the $Q^{EA} - table$ converges, the agent learns the best path to find the target state. That is, the agent can look directly at the converged Q-table, choose the action with best expected reward, enter the next state, and move forward gradually until it reaches the target state.

Secondly, to verify the reliability properties of this MAPP, the selected path (i.e., the MAPP behaviour) and probability statistics are specified into a DTMC model as the input of the model checker PRISM. MAPP system reliability requirements are described as properties with PCTL. They are used to calculate the likelihood of the occurrence of failed states during the execution of the system. Then, we can perform reliability analysis for MAPP using PRISM by checking whether the system can satisfy reliability requirements. The framework is further illustrated in the implementation and experimental studies detailed in Section 5.

5. Implementation and Experimental Results

In this section, we demonstrate and evaluate the proposed framework using an example of an agent walking in a grids map, as shown in Fig. 3. Each grid represents a room. Assume that there are 16 rooms altogether, with $Room_0$ being the starting point and $Room_{15}$ being the ending room. When the agent passes through each room, it will receive a reward value, which varies depending on the room. r_i indicates the rewards that can be obtained through $Room_i$, i is the number of the room and $i \in \{0, 1, 2, ..., 15\}$. The likelihood that the agent will be able to pass each room successfully varies because the rooms carry a random risk. p_i denotes the probability of an agent passing through $Room_i$.

5.1. Path planning using the $Q^{EA} - learning$ algorithm

The agent's objective is to reach $Room_{15}$ as soon as possible. It cannot switch to a non-adjacent room; it can only select the room next to the one it is currently in at a time. The agent is first assigned the location S_0 and is situated in $Room_0$. It can enter $Room_1$ or $Room_4$ in the next state. The probability of successfully passing to $Room_1$ is 0.504, and the agent can get 15 awards for doing so. The award that agent can get for successfully passing $Room_4$ is 32 and the probability of successfully passing is 0.138. If it enters $Room_4$ from $Room_0$, it can then enters $Room_0$, $Room_5$, or $Room_8$ in the next location. Passing through $Room_0$ will get reward 1, pass through $Room_5$ to get reward 12, and pass through $Room_8$ will earn reward 90. Assuming that it enters $Room_8$ in the third step, it can enter $Room_4$, $Room_9$, $Room_{12}$ in the next location. Just keep exploring until the agent reaches $Room_{15}$. One task is completed when the agent enters $Room_{15}$.

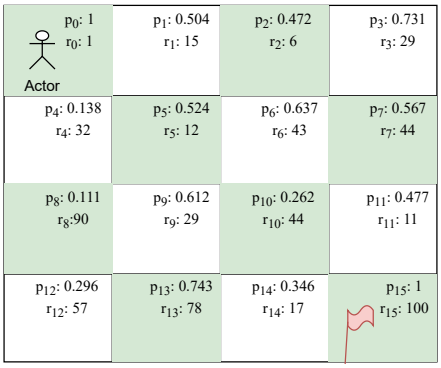

Figure 3: An agent exploring an uncertain grids map.

As mentioned in the $Q-$ *learning* algorithm, we need to give an R-matrix that contains the reward value before the agent starts the exploration. To determine the shortest path to the greatest reward in a general environment, the designer only needs to let the agent explore the environment on its own. Due to the inherent risks in each room in a stochastic uncertain environment, it is vital to consider both the potential reward and the probability of successfully passing a given room. Therefore, the R-matrix is given based on the expectation assignment method proposed earlier in Section 4, denoted as R^{EA}-matrix, and shown in (5).

$$
\begin{array}{c}
 \\
0 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
8 \\
9 \\
10 \\
11 \\
12 \\
13 \\
14 \\
15
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15
\end{array} \\
\left[\begin{array}{cccccccccccccccc}
-1 & 8 & -1 & -1 & 4 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 3 & -1 & -1 & 6 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 8 & -1 & 21 & -1 & -1 & 27 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 3 & -1 & -1 & -1 & -1 & 25 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & 6 & -1 & -1 & 10 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 8 & -1 & -1 & 4 & -1 & 27 & -1 & -1 & 18 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 3 & -1 & -1 & 6 & -1 & 25 & -1 & -1 & 12 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 21 & -1 & -1 & 27 & -1 & -1 & -1 & -1 & 5 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 4 & -1 & -1 & -1 & -1 & 18 & -1 & -1 & 17 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 6 & -1 & -1 & 10 & -1 & 12 & -1 & -1 & 58 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 27 & -1 & -1 & 18 & -1 & 5 & -1 & -1 & 6 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 12 & -1 & -1 & -1 & -1 & 100 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 10 & -1 & -1 & -1 & -1 & 58 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 18 & -1 & -1 & 17 & -1 & 6 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 12 & -1 & -1 & 58 & -1 & 100 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 100
\end{array}\right]
\end{array}
\tag{5}
$$

The R^{EA}-matrix (5) is clarified in detail as below: row i represents the current room, and column j represents the room to be entered next. $R_{i\to j}$ represents the reward value that will be obtained from the room i to j. For example, $R_{1\to 2} = 3$, which means that the agent can get 3 rewards value when moving from $Room_1$ to $Room_2$. A negative value of $R_{i\to j}$ indicates that there is no door between the two rooms and cannot be reached directly.

The Q^{EA} $-$ *learning* algorithm (Algorithm 1) yields the best route for $Room_0 \to Room_1 \to Room_5 \to Room_9 \to Room_{13} \to Room_{14} \to Room_{15}$. The reward obtained is (1+15+12+29+78+17+100=)252. The expected reward is calculated by Eq. (4) is 197.

5.2. Formal modelling of MAPP

Based on the proposed Q^{EA} – *learning* algorithm, the path of the agent has been determined as $Room_0 \rightarrow Room_1 \rightarrow Room_5 \rightarrow Room_9 \rightarrow Room_{13} \rightarrow Room_{14} \rightarrow Room_{15}$. In this section, we demonstrate the way to construct the DTMC model for the MAPP.

Fig. 4 illustrates the location transfer process, which is explained in detail below.

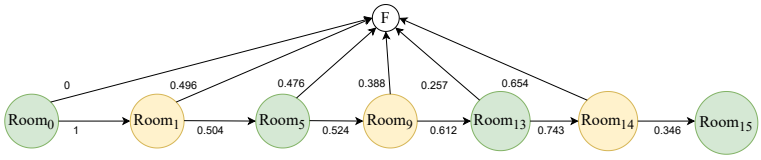

Figure 4: DTMC diagram of a robot navigating in the grids map.

First, the agent arrives at $Room_1$ from $Room_0$. The probability that the agent passes through $Room_1$ is 0.504, and there is a 0.496 probability that it encounters a hazard and falls into a fault state. Point F denotes an abnormality has occurred and the system is in failure. If a failure occurs, the task fails, and the agent starts again from the beginning.

If the agent passes $Room_1$ successfully, it will go to $Room_5$ (in Fig. 4). The probability that the agent passes through $Room_5$ is 0.524, and there is a 0.476 probability that it encounters a hazard and falls into a fault state. After passing through $Room_5$, the agent can enter to $Room_9$. The agent has a 0.612 chance of passing through $Room_9$, and it has a 0.388 chance that it may run into a hazard and enter a fault state. After passing through $Room_9$, the agent can enter into $Room_{13}$. After entering room 13, the agent has a 0.743 chance of successfully passing $Room_{13}$, and a 0.257 chance of having an anomaly and falling into a fault state. After passing through $Room_{13}$, the agent will enter into $Room_{14}$. The probability that the agent passes through $Room_{14}$ is 0.346, and there is a 0.654 probability that it encounters a hazard and falls into a fault state.

After passing through $Room_{14}$, the agent can enter to $Room_{15}$. Upon entering $Room_{15}$, a task is completed, and the maximum reward is available. Based on the entire process analysis, we develop the DTMC model as the input to PRISM, which is detailed below.

Model 1 :

dtmc

module Agent walking in the gripmap

s : [0..6] *init* 0;

d : [0..2] *init* 0;

$[]s = 0 \rightarrow 0.504 : (s' = 1) + 0.496 : (s' = 6)\&(d' = 1)$;

$[]s = 1 \rightarrow 0.524 : (s' = 2) + 0.476 : (s' = 6)\&(d' = 1)$;

$[]s = 2 \rightarrow 0.612 : (s' = 3) + 0.388 : (s' = 6)\&(d' = 1)$;

$[]s = 3 \rightarrow 0.743 : (s' = 4) + 0.257 : (s' = 6)\&(d' = 1)$;

$[]s = 4 \rightarrow 0.346 : (s' = 5) + 0.654 : (s' = 6)\&(d' = 1)$;

$[]s = 5 \rightarrow 1 : (s' = 6)\&(d' = 2)$;

$[]s = 6 \rightarrow (s' = 6)$;

end module

Explanation: this model has an initial state 0 and 6 states, states 1 to 5 indicate the agent is in $Room_1$, $Room_5$, $Room_9$, $Room_{13}$, and $Room_{14}$, respectively. State 6 represents the termination state, which may be a fault state or the arrival in $Room_{15}$. d : [0..2] denotes the goal value is 0, 1, 2. 0 is the initial value, 1 for failure, and 2 denotes it has entered $Room_{15}$. There are 7 sentences

here. The left side of the "→" represents the current room, and the right side represents the room and probability that may be entered in the next step. For example, $[]s = 2 \rightarrow 0.612 : (s' = 3) + 0.388 : (s' = 6)\&(d' = 1)$ means that the agent is now in $Room_5$, The probability of successfully passing $Room_5$ and getting to $Room_9$ is 0.612, and the failure probability is 0.388.

5.3. Required properties specification and verification

The purpose of MAPP is to allow the agent to explore the uncertain environment and reach the target location. Given a DTMC model created based on MAPP, it then needs to describe its required properties for quantitative verification. First, we explain how to specify the required properties with PCTL including a probability formula and a reward formula. For example, the following property can be specified as a PCTL formula.

const int InitState
$\mathbf{P}$ =?[$\mathbf{F\ s = 6 \wedge d = x}$]

The first formula expresses a constant given the initial state. The second formula means "what is the probability, from the InitState of the model, of terminal enter the state 6 and obtain a d value". For example, suppose x=1 is considered. The P =?$[F\ s = 6 \wedge d = 1]$ means "What is the probability that from the initial state to the final state of 6 and d takes a value of 1". As we mentioned previously, d equals 1 to indicate that the agent fails. Therefore, this formula can also be interpreted as the probability that the final state of the agent is fail.

We demonstrate and evaluate our framework by using the PRISM simulation module. Table 1 shows the probabilities for different sample volumes. The row (1) denotes the probability of the agent ultimately failing, which contains the verification results and simulation results under different samples. The verification results show that in the learned model, the probability of the agent finally failing is 0.958, and the probability of entering $Room_{15}$ is 0.042. The simulation method is adopted to explain that the verification result is consistent with real-life conditions. The simulated samples represent distinct states, and the flows of several transitional paths represent the choices made by the agent. In this experiment, the number of samples used in the simulation procedure ranged from 50 to 10000. We can see from the findings that as the number of simulation samples grows, the experimental result approaches the verification result. Fig. 5 presents this concept.

Table 1: Simulation results for the robot system

Room	Number of Simulation Samples						Verify Prob
	50	100	500	1000	5000	10000	
(1)	0.980	0.940	0.942	0.949	0.962	**0.959**	0.958
(2)	0.020	0.060	0.058	0.051	0.038	**0.041**	0.042

6. Discussions

To demonstrate the applicability and effectiveness of our method, we compare it with two traditional methods. The first method is the probabilistic model, which means that the agent explores the environment without learning, while from one room, the probability of entering an adjacent room is equal. The second method uses the traditional Q-learning algorithm to

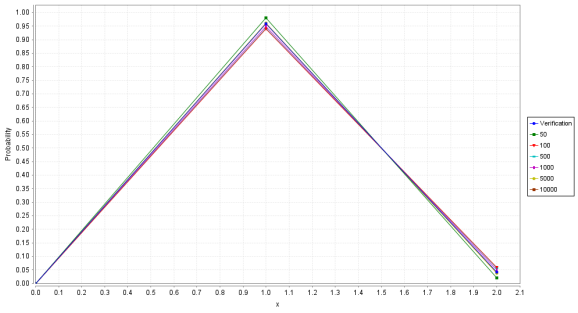

Figure 5: Simulations with different samples.

determine the path, which implies that the agent considers only the reward value earned by passing the room rather than the probability of successfully traversing the room. These two methods are briefly presented in the following sections.

6.1. MAPP based on the probability model

When the agent has not engaged in any learning or exploration, it has the same probability of moving from its current location to any adjacent location.

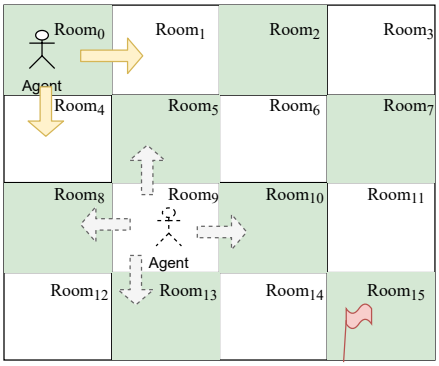

Figure 6: An agent random exploring in the grids map.

For example, as shown in Fig. 6, when the agent is in $Room_5$, the probability of entering $Room_1$, $Room_4$, $Room_6$, and $Room_9$ is 1/4. Assuming that the agent chooses to enter $Room_6$, the probability of successfully passing through $Room_6$ is $1/4 \times 0.637 \approx 0.159$. The situation in other rooms is similar.

Model 2 : The state transition probability matrix of an agent that has not engaged learning is presented in Table 2. The DTMC model can be established according to Table 2 and PRISM is used to complete the verification. The verification result shows the probability of $P =?[F\ s = 15 \wedge d = 1]$ is 0.9993 and the probability of $P =?[F\ s = 15 \wedge d = 2]$ is 0.0007. It is clear that the probability of the agent failing is higher than the result in *Model* 1.

Table 2: The probability of robot transition

CR	NR	Prob	CR	NR	Prob	CR	NR	Prob	CR	NR	Prob
0	1	0.252	4	8	0.151	8	4	0.046	11	15	0.333
	4	0.069		F	0.341		9	0.204		F	0.390
	F	0.679	5	1	0.126		12	0.099	12	8	0.227
1	2	0.157		6	0.159		F	0.651		13	0.372
	1	0.333		9	0.153	9	5	0.131		F	0.401
	5	0.175		4	0.035		10	0.066	13	9	0.204
	F	0.335		F	0.527		13	0.186		14	0.115
2	3	0.244	6	2	0.118		8	0.114		12	0.099
	6	0.212		7	0.142		F	0.504		F	0.582
	1	0.168		10	0.066	10	6	0.159		10	0.087
	F	0.376		5	0.131		11	0.119	14	13	0.248
3	2	0.236		F	0.544		14	0.087		15	0.333
	7	0.284	7	3	0.244		9	0.153		F	0.332
	F	0.320		6	0.212		F	0.482	15	15	1
4	1	0.333		11	0.159	11	7	0.189			
	5	0.175		F	0.385		10	0.087			

6.2. MAPP based on the Q − learning

The Q^{EA} − *learning* algorithm suggested in this paper adds the concept of expected assignment to the general Q-learning. For comparison, here we analyze the case of the maximum reward value and use basic Q-learning to find the shortest path to the maximum reward. The input R-matrix is given as:

$$
\begin{array}{c}
 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\
\left[\begin{array}{cccccccccccccccc}
-1 & 15 & -1 & -1 & 32 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 6 & -1 & -1 & 12 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 15 & -1 & 29 & -1 & -1 & 43 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 6 & -1 & -1 & -1 & -1 & 44 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & 12 & -1 & -1 & 90 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 15 & -1 & -1 & 32 & -1 & 43 & -1 & -1 & 29 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 6 & -1 & -1 & 12 & -1 & 44 & -1 & -1 & 44 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 29 & -1 & -1 & 43 & -1 & -1 & -1 & -1 & 11 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 32 & -1 & -1 & -1 & -1 & 29 & -1 & -1 & 57 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 12 & -1 & -1 & 90 & -1 & 44 & -1 & -1 & 78 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 43 & -1 & -1 & 29 & -1 & 11 & -1 & -1 & 17 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 44 & -1 & -1 & -1 & -1 & 100 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 90 & -1 & -1 & -1 & -1 & 78 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 29 & -1 & -1 & 57 & -1 & 17 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 44 & -1 & -1 & 78 & -1 & 100 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 100
\end{array}\right]
\end{array}
\tag{6}
$$

Accordingly, the Q − *learning* algorithm yields the best route for $Room_0 \rightarrow Room_4 \rightarrow Room_8 \rightarrow Room_{12} \rightarrow Room_{13} \rightarrow Room_{14} \rightarrow Room_{15}$. The reward obtained is

(1+32+90+57+78+17+100)=375. According to the chosen location transfer path, we develop the following DTMC model.

Model 3 :

dtmc

module Agent walking in the gripmap

s : [0..6] *init* 0;

d : [0..2] *init* 0;

$[]s = 0 \rightarrow 0.138 : (s' = 1) + 0.862 : (s' = 6)\&(d' = 1)$;

$[]s = 1 \rightarrow 0.111 : (s' = 2) + 0.889 : (s' = 6)\&(d' = 1)$;

$[]s = 2 \rightarrow 0.296 : (s' = 3) + 0.704 : (s' = 6)\&(d' = 1)$;

$[]s = 3 \rightarrow 0.743 : (s' = 4) + 0.257 : (s' = 6)\&(d' = 1)$;

$[]s = 4 \rightarrow 0.346 : (s' = 5) + 0.654 : (s' = 6)\&(d' = 1)$;

$[]s = 5 \rightarrow 1 : (s' = 6)\&(d' = 2)$;

$[]s = 6 \rightarrow (s' = 6)$;

end module

The verification result shows the value of $P =?[Fs = 6 \wedge d = 1]$ is 0.9988 and the value of $P =?[Fs = 6 \wedge d = 2]$ is 0.0012. It is clear that the probability of the agent fails is higher than the result in *Model* 1. The maximum possible reward for *Model* 1 is 252, and the maximum possible reward for *Model* 3 is 375.

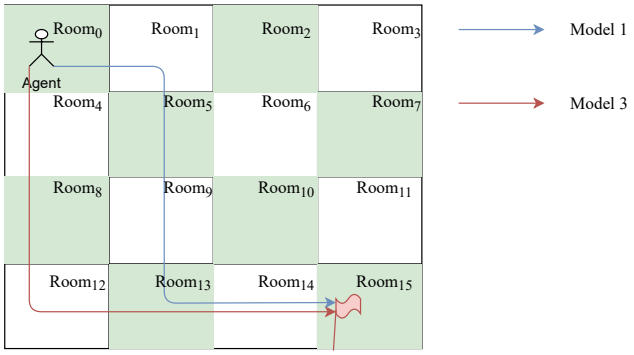

Figure 7: Two different paths for two models.

Table 3 shows the experimental comparison of the results of the three DTMC models created based on the three different methods respectively. We can find that *Model* 2 has the lowest probability of successfully reaching $Room_{15}$ because it has not been learnt and the agent shifts its location randomly until it reaches $Room_{15}$. Since it is not possible to determine the route it took, it is not possible to calculate the reward it could obtain. *Model* 3 specifically has a lower probability compared to *Model* 1, but may gain a larger reward. Therefore, if the probability of success is more significant, choose path $Room_0 \rightarrow Room_1 \rightarrow Room_5 \rightarrow Room_9 \rightarrow Room_{13} \rightarrow Room_{14} \rightarrow Room_{15}$. On the other hand, if the focus is likely to yield the greatest reward, choose path $Room_0 \rightarrow Room_4 \rightarrow Room_8 \rightarrow Room_{12} \rightarrow Room_{13} \rightarrow Room_{14} \rightarrow Room_{15}$.

The experiments were completed on a Intel(R) Core(TM) i5-7200U CPU with 2.50GHz and 2.71 GHz. The Algorithm were completed in Python 3.10, and the PRISM that we used for verification was PRISM 4.7 (GUI mode).

Table 3: Experimental comparisons for the three models

NO.	$Room_{15}$	Failure	Reward	Expected Reward
Model 1	0.0420	0.9580	252	197
Model 2	0.0007	0.9993	*	*
Model 3	0.0012	0.9988	375	196

7. Conclusions and Future Work

MAPP and control system reliability must be confirmed in order for agent technology to improve and intelligence to rise. Because the real environment is unpredictable and complex, an agent may come across a number of anomalies. The randomness and uncertainty of the environment must be taken into account in the MAPP. Using RL methods, such as Q-learning algorithm, as a key intelligent technique in the field of agent and multi-agent research, can significantly improve an agent's ability to finish a task. However, this has not yet worked well in the uncertain environment. The modified Q-learning algorithm, called $Q^{EA} - learning$ algorithm suggested in this paper enabled the agent to efficiently explore an uncertain environment and generate an optimal path. Nevertheless, for the learnt model, reliability analysis cannot be ignored. Model checking is a typical formal verification technique that plays a key role in assuring the control system's safety and has also been increasingly applied in industrial contexts. Compared with traditional model checking, probabilistic model checking adds the verification function of quantitative properties. The application of probabilistic model checking allows for the verification of system models in uncertain environments, with the results being used to evaluate whether the required properties are met, as well as the extent to which they are fulfilled.

This paper presented an easy-to-understand path planning and control system modelling framework of the mobile agent within uncertain environments. First, randomly assign success probability and reward values for all locations aside from the beginning and target locations. According to the probability value and the reward of each location, the expected reward is calculated, and the expected R-matrix is generated. The expected R-matrix will be used as input to the $Q^{EA} - learning$ algorithm to select the fastest and highest expected reward path. Moreover, a DTMC model was created for the control system under the selected path, and the model was verified with PRISM. Finally, in the discussion section, two DTMC models are constructed for the paths selected by $Q - learning$ and general probability, respectively, and the same properties are verified to demonstrate the effectiveness of the path planning method proposed in this paper under stochastic uncertainty environments. The results show that our method has a higher probability of obtaining the maximum expected reward.

This paper primarily considers scenarios involving only a single agent's movement. Most industrial application environments, such as automated driving and smart warehousing, have multiple agents moving simultaneously. More complex issues, such as collision-free control, need to be considered in multiple agent applications, which is also more application-worthy. Therefore, multi-agent movement in uncertain environments is the predominant direction for our future research.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (No. 61976130, 62206227), the Chengdu International Science Cooperation Project under Grant

2020-GH02-00064-HZ, and China Scholarship Council.

References

[1] N. R. Jennings, M. Wooldridge, Applying agent technology, Applied Artificial Intelligence an International Journal 9 (4) (1995) 357–369.

[2] M. Rath, B. K. Pattanayak, Security protocol with ids framework using mobile agent in robotic manet, International Journal of Information Security and Privacy (IJISP) 13 (1) (2019) 46–58.

[3] I. Carlucho, M. De Paula, G. G. Acosta, Double q-pid algorithm for mobile robot control, Expert Systems with Applications 137 (2019) 292–307.

[4] M. Jana, L. Vachhani, A. Sinha, A deep reinforcement learning approach for multi-agent mobile robot patrolling, International Journal of Intelligent Robotics and Applications (2022) 1–22.

[5] P. Zhang, T. Li, Z. Yuan, L. Chuan, G. Wang, J. Liu, S. Du, A data-level fusion model for unsupervised attribute selection in multi-source homogeneous data, Information Fusion 80 (2022) 87–103.

[6] P. Zhang, T. Li, Z. Yuan, C. Luo, K. Liu, X. Yang, Heterogeneous feature selection based on neighborhood combination entropy, IEEE Transactions on Neural Networks and Learning Systems (2022) 1–14.

[7] M. Nazarahari, E. Khanmirza, S. Doostie, Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm, Expert Systems with Applications 115 (2019) 106–120.

[8] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, Journal of artificial intelligence research 4 (1996) 237–285.

[9] Q. Luo, H. Wang, Y. Zheng, J. He, Research on path planning of mobile robot based on improved ant colony algorithm, Neural Computing and Applications 32 (6) (2020) 1555–1566.

[10] Y. Cui, W. Hu, A. Rahmani, A reinforcement learning based artificial bee colony algorithm with application in robot path planning, Expert Systems with Applications (2022) 117389.

[11] M. H. ter Beek, K. G. Larsen, D. Ničković, T. A. Willemse, Formal methods and tools for industrial critical systems (2022).

[12] M. Kwiatkowska, G. Norman, D. Parker, Prism 4.0: Verification of probabilistic real-time systems, in: International conference on computer aided verification, Springer, 2011, pp. 585–591.

[13] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, A. K. Nagar, A deterministic improved q-learning for path planning of a mobile robot, IEEE Transactions on Systems, Man, and Cybernetics: Systems 43 (5) (2013) 1141–1153.

[14] H.-y. Zhang, W.-m. Lin, A.-x. Chen, Path planning for the mobile robot: A review, Symmetry 10 (10) (2018) 450.

[15] S. S. Nestinger, B. Chen, H. H. Cheng, A mobile agent-based framework for flexible automation systems, IEEE/Asme Transactions on Mechatronics 15 (6) (2009) 942–951.

[16] C. Lamini, S. Benhlima, A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, Procedia Computer Science 127 (2018) 180–189.

[17] Q. Li, F. Gama, A. Ribeiro, A. Prorok, Graph neural networks for decentralized multi-robot path planning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 11785–11792.

[18] Q. Li, W. Lin, Z. Liu, A. Prorok, Message-aware graph attention networks for large-scale multi-robot path planning, IEEE Robotics and Automation Letters 6 (3) (2021) 5533–5540.

[19] B. Wang, Z. Liu, Q. Li, A. Prorok, Mobile robot path planning in dynamic environments through globally guided reinforcement learning, IEEE Robotics and Automation Letters 5 (4) (2020) 6932–6939.

[20] X. Dai, S. Long, Z. Zhang, D. Gong, Mobile robot path planning based on ant colony algorithm with a* heuristic method, Frontiers in neurorobotics 13 (2019) 15.

[21] L. Chang, L. Shan, C. Jiang, Y. Dai, Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment, Autonomous Robots 45 (1) (2021) 51–76.

[22] Q. Song, S. Li, J. Yang, Q. Bai, J. Hu, X. Zhang, A. Zhang, Intelligent optimization algorithm-based path planning for a mobile robot, Computational Intelligence and Neuroscience 2021 (2021) 1–17.

[23] C. Zhu, W. Bu, J. Chen, C. Liu, Multi-agents based safe path planning of autonomous mobile robots in smart factories, in: 2021 China Automation Congress (CAC), IEEE, 2021, pp. 2206–2211.

[24] M. Kwiatkowska, G. Norman, D. Parker, Advances and challenges of probabilistic model checking, in: 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2010, pp. 1691–1698.

[25] X. Zhao, V. Robu, D. Flynn, F. Dinmohammadi, M. Fisher, M. Webster, Probabilistic model checking of robots deployed in extreme environments, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 8066–8074.

[26] L. Feng, C. Wiltsche, L. Humphrey, U. Topcu, Controller synthesis for autonomous systems interacting with human operators, in: Proceedings of the acm/ieee sixth international conference on cyber-physical systems, 2015, pp. 70–79.

[27] M. Wang, R. Wang, X. Li, Y. Guan, J. Zhang, H. Wei, Verification of obstacle avoidance strategies for mobile robots in nondeterministic environmentsłbased on formal modeling and probabilistic analysis, Computer Engineering and Applications 52 (10) (2016) 31–38.
[28] D. Milios, G. Sanguinetti, D. Schnoerr, Probabilistic model checking for continuous-time markov chains via sequential bayesian inference, in: International Conference on Quantitative Evaluation of Systems, Springer, 2018, pp. 289–305.
[29] N. Cauchi, K. A. Hoque, A. Abate, M. Stoelinga, Efficient probabilistic model checking of smart building maintenance using fault maintenance trees, in: Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments, 2017, pp. 1–10.
[30] Y. Zhu, X. Xue, K. Zhang, S. Mao, H. Gao, Applying probabilistic model checking to express delivery location selection and optimization, in: 2016 IEEE 13th International Conference on e-Business Engineering (ICEBE), IEEE, 2016, pp. 32–39.
[31] J. M. Franco, F. Correia, R. Barbosa, M. Zenha-Rela, B. Schmerl, D. Garlan, Improving self-adaptation planning through software architecture-based stochastic modeling, Journal of Systems and software 115 (2016) 42–60.
[32] H. Gao, S. Mao, W. Huang, X. Yang, Applying probabilistic model checking to financial production risk evaluation and control: A case study of alibabas yue bao, IEEE Transactions on Computational Social Systems 5 (3) (2018) 785–795.
[33] H. Gao, L. Zhou, J. Y. Kim, Y. Li, W. Huang, The behavior guidance and abnormality detection for a-mci patients under wireless sensor network, ACM Transactions on Sensor Networks (2022).
[34] S. Ouchani, O. A. Mohamed, M. Debbabi, A property-based abstraction framework for sysml activity diagrams, Knowledge-Based Systems 56 (2014) 328–343.
[35] W. Wan, J. Bentahar, A. B. Hamza, Model checking epistemic–probabilistic logic using probabilistic interpreted systems, Knowledge-Based Systems 50 (2013) 279–295.
[36] K. Sultan, J. Bentahar, H. Yahyaoui, R. Mizouni, Model checking agent-based communities against uncertain group commitments and knowledge, Expert Systems with Applications 177 (2021) 114792.
[37] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3) (1992) 279–292.
[38] S. N. Ethier, T. G. Kurtz, Markov processes: characterization and convergence, John Wiley & Sons, 2009.
[39] K. Sultan, J. Bentahar, M. El-Menshawy, Model checking probabilistic social commitments for intelligent agent communication, Applied Soft Computing 22 (2014) 397–409.
[40] T. Mizuno, S.-Y. Nishizaki, Model checking and analysis of systems dependent on cpu speed, E-Commerce, E-Business and E-Service 1 (2014) 177.
[41] M. Duflot, M. Kwiatkowska, G. Norman, D. Parker, S. Peyronnet, C. Picaronny, J. Sproston, Practical applications of probabilistic model checking to communication protocols (2012).
[42] O. Siedlecka-Lamch, M. Kurkowski, J. Piatkowski, Probabilistic model checking of security protocols without perfect cryptography assumption, in: International Conference on Computer Networks, Springer, 2016, pp. 107–117.
[43] M. Kwiatkowska, G. Norman, D. Parker, Prism: Probabilistic symbolic model checker, in: International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Springer, 2002, pp. 200–204.
[44] J. Desharnais, P. Panangaden, Continuous stochastic logic characterizes bisimulation of continuous-time markov processes, The Journal of Logic and Algebraic Programming 56 (1-2) (2003) 99–115.
[45] A. Kimmig, B. Demoen, L. De Raedt, V. S. Costa, R. Rocha, On the implementation of the probabilistic logic programming language problog, Theory and Practice of Logic Programming 11 (2-3) (2011) 235–262.
[46] F. Ciesinski, M. Größer, On probabilistic computation tree logic, in: Validation of Stochastic Systems, Springer, 2004, pp. 147–188.
[47] R. T. Clemen, T. Reilly, Making hard decisions with DecisionTools, Cengage Learning, 2013.