Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An anti-collision algorithm for robotic search-and-rescue tasks in unknown dynamic environments*

Yang CHEN†1, Dianxi SHI†‡2,3, Huanhuan YANG4, Tongyue LI3, Zhen WANG3

[1]*School of Computer Science, Peking University, Beijing 100871, China*
[2]*Tianjin Artificial Intelligence Innovation Center, Tianjin 300457, China*
[3]*Intelligent Game and Decision Lab, Beijing 100071, China*
[4]*College of Computer, National University of Defense Technology, Changsha 410073, China*
†E-mail: chenyang20@stu.pku.edu.cn; dxshi@nudt.edu.cn

Received Mar. 3, 2023; Revision accepted Aug. 25, 2023; Crosschecked Mar. 6, 2024

Abstract: This paper deals with the search-and-rescue tasks of a mobile robot with multiple interesting targets in an unknown dynamic environment. The problem is challenging because the mobile robot needs to search for multiple targets while avoiding obstacles simultaneously. To ensure that the mobile robot avoids obstacles properly, we propose a mixed-strategy Nash equilibrium based Dyna-Q (MNDQ) algorithm. First, a multi-objective layered structure is introduced to simplify the representation of multiple objectives and reduce computational complexity. This structure divides the overall task into subtasks, including searching for targets and avoiding obstacles. Second, a risk-monitoring mechanism is proposed based on the relative positions of dynamic risks. This mechanism helps the robot avoid potential collisions and unnecessary detours. Then, to improve sampling efficiency, MNDQ is presented, which combines Dyna-Q and mixed-strategy Nash equilibrium. By using mixed-strategy Nash equilibrium, the agent makes decisions in the form of probabilities, maximizing the expected rewards and improving the overall performance of the Dyna-Q algorithm. Furthermore, a series of simulations are conducted to verify the effectiveness of the proposed method. The results show that MNDQ performs well and exhibits robustness, providing a competitive solution for future autonomous robot navigation tasks.

Key words: Search and rescue; Reinforcement learning; Game theory; Collision avoidance; Decision-making
https://doi.org/10.1631/FITEE.2300151 **CLC number:** TP183

1 Introduction

With the rapid advancement of artificial intelligence technology, the application of a robot is progressively shifting toward intelligence-oriented directions. Researchers study various applications of robot systems to accomplish difficult tasks, such as search and rescue and logistics (Geng et al., 2019), as shown in Fig. 1. The utilization of a mobile robot is becoming increasingly widespread in various applications such as search and rescue, exploring underwater resources (Zheng Z et al., 2016; Geng et al., 2019), and the Mars exploration project (Wakayama and Ahmed, 2020). In addition, there are some specific types of tasks (Li CH et al., 2019), such as patrolling (Martins-Filho and Macau, 2007; Hwang et al., 2011; Banerjee et al., 2015), surveillance (Curiac et al., 2018), and demining (Prado and Marques, 2014), which require not only finding all targets but also avoiding patrolmen and other dynamic objects.

The above applications all involve the key issue of collecting targets while avoiding obstacles. The problem is complex due to the constraints of limited response time, an unstable environment, and

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 91948303)
ORCID: Yang CHEN, https://orcid.org/0000-0003-3414-3328; Dianxi SHI, https://orcid.org/0000-0002-8112-371X

the need for multiple candidate search modes (Hong et al., 2021). In recent years, many search planning methods have been proposed. Numerous experiments have been conducted to validate their effectiveness (Brito et al., 2019; Gaertner et al., 2021; Hubert et al., 2021; Liu Z et al., 2023; Luo et al., 2023). However, some works show a low exploration rate, which should be further improved for dynamic real-world applications (Lu YM and Kamgarpour, 2020; Gaertner et al., 2021; Nasar et al., 2023). In addition, several methods aim to investigate motion using the greedy algorithm. The techniques include maximizing instant information acquisition or navigating toward the closest unknown region (Wu YX et al., 2019; Liu YY et al., 2022; Pei et al., 2022). Although the greedy strategy may seem efficient in the short term, it often fails to achieve global optimality (Zou et al., 2020; Hayamizu et al., 2021). Moreover, numerous methods exhibit significant computational overhead and necessitate prompt and frequent adaptation to environmental fluctuations (Brito et al., 2021; Padakandla, 2021).

To address the above issues, we propose a mixed-strategy Nash equilibrium (MNE) based Dyna-Q (MNDQ) algorithm to solve the search-and-rescue problem in unknown dynamic environments. Our approach involves decomposing the problem to reduce computational complexity. Additionally, we incorporate the game theory into the improved algorithm, enabling the selection of optimal samples that maximize expected rewards. The effectiveness of our proposed algorithm is supported by theoretical analysis and simulation results. The main contributions are as follows:

1. We develop a multi-objective layered structure with the aim of simplifying the complexity of multi-objective problems. The design is based on a layered approach, where each subtask focuses solely on its own states and actions. This approach allows us to reduce the computational complexity of the overall problem.

2. We propose a risk-monitoring mechanism that relies on the relative position of dynamic risks to aid in decision-making. The state of these dynamic risks is expressed using the Manhattan distance and relative orientation. This mechanism assists in creating an intuitive environment model, which helps prevent potential collisions and unnecessary detours.

3. To maximize rewards, we incorporate the principles of game theory into our approach. We introduce a method called MNDQ, which combines reinforcement learning (RL) with Dyna-Q and MNE. The MNE policy is proposed to enable the robot to choose better state–action pairs for learning. The simulation results show that the proposed method performs satisfactorily in terms of convergence property and learning efficiency.

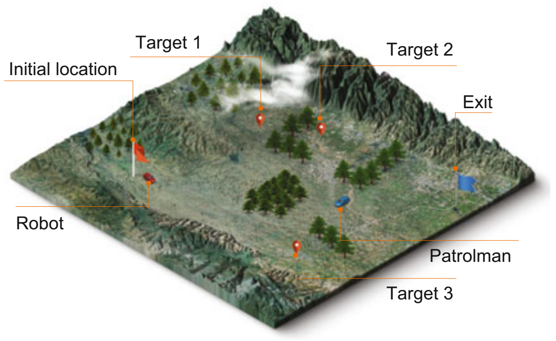

Fig. 1 Robot search-and-rescue task

2 Related works

2.1 Autonomous exploration algorithms

Online autonomous exploration in unknown dynamic environments attracts significant interest, particularly in the context of search-and-rescue planning. Typically, autonomous exploration methods are broadly classified into two types: classic methods and learning-based methods. Classic methods encompass optimization techniques, rule-based approaches, and evolutionary algorithms (Ng and Bräunl, 2007; McGuire et al., 2019; Patle et al., 2019; Aggarwal and Kumar, 2020; Wyrąbkiewicz et al., 2020; Hong et al., 2021). Classic methods cannot deal with emergencies or extract information from perception and analysis of the environment, especially in unknown dynamic environments. Learning-based methods use machine learning techniques, such as deep learning, transfer learning, and RL (Chiu et al., 2021; Li ZR et al., 2022; Li HQ et al., 2023).

When confronted with a complex and dynamic environment, RL-based methods help the robot obtain feedback and control through real-time interactive measurement of the environment without any prior knowledge. RL offers numerous advantages over traditional algorithms (Faust et al., 2018;

Jaderberg et al., 2019; Ohnishi et al., 2019; Li HR et al., 2020; Chiu et al., 2021). Zheng KY et al. (2021) solved the search problem by using a multi-resolution planning algorithm based on an online Monte Carlo tree search. Lu YL and Yan (2020) considered safe task planning in uncertain dynamic environments and devised a Monte Carlo method to obtain a safe path efficiently. Gregor et al. (2018) proposed a visual attention operator based on RL and controlled the Pac-Man to collect all pac-dots and avoid getting eaten by ghosts. Furthermore, when the number of targets to be collected in the environment increases, the difficulty of the search also increases (Niroui et al., 2017; Wu JF et al., 2021).

2.2 Optimization-based collision avoidance

As one of the core contents of mobile robot research, the anti-collision algorithm aims to effectively avoid obstacles and carry out tasks in the process of movement (Jaderberg et al., 2019; Li HR et al., 2020). Specifically, the robot faces challenges and risks. It requires overcoming the environment's unknown factors and efficiently reaching its given goals while avoiding potential conflicts with other robots or dynamic obstacles. To avoid conflicts with static and dynamic obstacles, Pei et al. (2022) presented the notion of safe transition probability to model the evolution of the hazard process. Lei et al. (2018) used lidar to detect the environment dot matrix information and achieved local path planning with double deep Q network (DDQN). Wang et al. (2020) introduced a local RL-based planner, which generates actions by exploiting surrounding environmental information.

Furthermore, when RL is used to prevent collisions, the effectiveness and efficiency of motion planning heavily rely on the design of the sample selection policy. Exploration strategy with the rasterized sampling is learned, and the action space is composed of these sampling points (Li HR et al., 2020). A random minibatch of experience tuples from replay memory is sampled based on an extended deep deterministic policy gradient (DDPG) algorithm (Yu et al., 2021). The Dyna-Q learning algorithm improves the policy through samples of actual experience (Zhang et al., 2021). However, the relationships between the robot and obstacles need to be taken seriously in the process of sample selection (Fudenberg and Tirole, 1991; Osborne and Rubinstein, 1994; Hu and Wellman, 2003), which lacks better state–action pairs.

3 Preliminaries

The algorithm proposed in this study is based mainly on RL and the game theory. A brief overview of relevant knowledge is provided in this section.

3.1 Markov decision process (MDP)

The process of motion planning is regarded as a problem of a series of sequential decisions, and the MDP is a classic formalization of sequential decision-making (Sutton and Barto, 2018). MDP offers a straightforward approach to learning and achieving a goal through interaction. In this context, the robot serves as the learner and decision-maker, referred to as the agent. The environment encompasses everything outside the robot with which it interacts. The environment gives rise to rewards. The robot aims to maximize the cumulative rewards over time through its decisions of actions.

In MDP, the actions of a robot influence not only immediate rewards but also subsequent states and future rewards. In other words, MDP involves delayed rewards and the need to trade immediate and delayed rewards. As a mathematically idealized form of optimization problems by combining the Markov decision theory with dynamic programming, MDP is widely used in many types of tasks, including automated control, economics, robotics, and manufacturing (Puterman, 1990; Shi et al., 2018). MDP is regarded as a discrete-time stochastic control process. The robot and environment interact at each of a sequence of discrete-time steps, $t = 0, 1, 2, ...$. At each discrete-time step t, the robot receives the current state of the environment S_t and, based on this, the robot selects a possible action A_t, which follows the next state S_{t+1}. Subsequently, the robot interacts with the environment and gets a reward R_t.

In the task of robot search and rescue in a dynamic environment, the sets of states, actions, and rewards consist of a finite number of elements. In this situation, the random variables R_t and S_t have discrete probability distributions dependent only on the preceding state and action. For particular values of these random variables, s' and r, there is a probability of those values occurring at time t with particular values of the preceding state and action:

$$\begin{aligned} & p(s', r|s, a) \\ & \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}. \end{aligned} \quad (1)$$

The goal of the robot is to learn a policy π to plan an anti-collision path.

The additional concept is discounting. According to this approach of discounting, the robot attempts to select actions in order that the sum of the discounted rewards, in the long run, is maximized. The robot chooses A_t to maximize the expected discounted return G_t, and G_t is defined by the following expression:

$$\begin{aligned} G_t & \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \\ & = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \end{aligned} \quad (2)$$

where γ ($0 \le \gamma \le 1$) is a parameter that expresses the discount rate; γ determines the present value of future rewards. A policy is a mapping from states to the probabilities of selecting each possible action.

The value function $Q_\pi(S_t, A_t)$ of taking action A_t in state S_t under policy π is defined as follows:

$$Q_\pi(S_t, A_t) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t, A_t\right], \quad (3)$$

where $E_\pi[\cdot]$ denotes the expected value of a random variable with policy π, and t is any time step. Function Q_π is regarded as the action–value function for policy π.

3.2 Dyna architecture

The Dyna architecture integrates learning and planning. The general Dyna architecture is shown in Fig. 2. Real experience passes back and forth between the environment and the policy. Meanwhile, real experience affects policy and value functions. As a typical algorithm with the Dyna architecture, Dyna-Q includes all processes (planning, acting, model learning, and direct RL based on Q-learning), which is effective in solving the motion planning problem (Hwang et al., 2015; Zhang et al., 2021).

3.3 Nash equilibrium

Game theory is one of the most fundamental ways to describe the relationships in mobile robot systems (Lu YL and Yan, 2020). Among the concepts in game theory, a mixed strategy is a probability distribution over the player's pure strategies (Roughgarden, 2010).

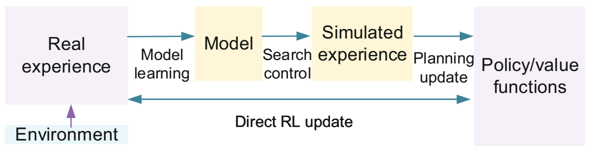

Fig. 2 The general Dyna architecture: real experience passes back and forth between the environment and the policy (RL: reinforcement learning)

It is a general notion of a steady state to allow the player's choices to vary or deviate on each occasion while playing the game. This notion means that players might choose probability distributions over a set of actions available to them. Such a steady state is called stochastic (involving probability) and modeled by an MNE. An MNE is a mixed-strategy action profile, in which a single player cannot obtain a higher expected payoff according to the player's preference over all such lotteries. In other words, every player simultaneously chooses a mixed strategy to maximize his/her expected rewards. Their relationship is often represented in a "bimatrix" form. One player chooses a row. The other player chooses a column. The rewards for the row and column players are represented by the numbers in the corresponding matrix (Roughgarden, 2016).

A pair of probability distributions with the property indicates an MNE. Every bimatrix game has a Nash equilibrium. In an MNE, the players randomize independently, and unilateral deviations increase only a player's expected cost. Unilateral deviation means that one player changes his/her own strategy. Expected cost represents the weighted sum of costs in a probability distribution of strategies. Each player's mixed strategy is optimal given the equilibrium mixed strategies of the other players. In other words, no player has the incentive to deviate from his/her MNE strategy to another pure or mixed strategy, conditional on the other players choosing their equilibrium strategies.

A cost-minimization game has the following main ingredients: a finite number of players h, a finite strategy set G_i for each player i, and a cost function $C_i(g_i, g_{-i})$ for each player i (where g_i denotes the strategy profile of player i and g_{-i} denotes the strategy profile of the other players). Distributions

$\sigma_1, \sigma_2, ..., \sigma_h$ over strategy sets $G_1, G_2, ..., G_h$ of a cost-minimization game constitute an MNE if for every player $i \in \{1, 2, ..., h\}$ and every unilateral deviation $g_i' \in G_i$, there is

$$E_{g\sim\sigma}\left[C_i\left(g_i^*, g_{-i}^*\right)\right] \leq E_{g\sim\sigma}\left[C_i\left(g_i', g_{-i}^*\right)\right], \quad (4)$$

where $\sigma = \sigma_1 \times \sigma_2 \times \cdots \times \sigma_h$ denotes the product distribution.

4 Problem definition

We define the model of the robot and the unknown dynamic environment in this section. According to the model, the motion planning problem is presented.

4.1 Model of the robot

The robot performs tasks in a discrete two-dimensional (2D) grid space, where each grid represents a cell. The robot selects actions and moves to adjacent grids, as shown in Fig. 3. Black grids represent static obstacles or walls. Targets that need to be collected are indicated by green hollow circles. The robot itself is represented by a solid red dot, and the blue dot represents the patrolman. The possible actions are selected within the collection $\mathcal{A} = \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}, \mathrm{O}\}$, representing the four directions or the person staying still. The robot takes an action and moves to the adjacent position, except when the movement is blocked by an obstacle or the edge of the map. The robot obtains information about the environment within a sensing range $d_{\max}$ not blocked by obstacles.

4.2 Search-and-rescue tasks in an unknown dynamic environment

Each cell on the map is occupied by obstacles or free space. The robot moves and collects the targets in the free space. The location of the robot is $(x_{\text{robot}}, y_{\text{robot}})$. The robot performs tasks without any prior information. The environment is dynamic with a patrolman that can move and patrol. The patrolman can be represented as a dynamic obstacle. It patrols the targets in turn, with a certain possibility to track the robot when the robot is within its observation range. The patrol path of the patrolman is trained according to RL, and the patrolman adjusts the path based on its observation information

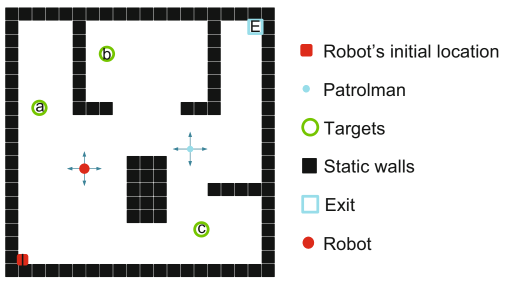

Fig. 3 A search-and-rescue task in a grid space. References to color refer to the online version of this figure

to maximize its rewards. The patrolman can judge the current situation and make the most favorable behavior decision to continue patrolling or tracking the robot. Such behavior choice brings uncertainty to the robot.

Considering real problems (Jaderberg et al., 2019; Li HR et al., 2020) that arise in search-and-rescue scenarios, when the rescuer enters the maze to rescue the hostages, the robot needs to avoid pursuing the patrolman, according to observation information. In Fig. 3, the robot starts from the initial location to collect targets a, b, and c, and departs from the environment at the exit. Meanwhile, the patrolman starts at the exit and patrols the targets one by one, over and over again.

5 Methodology

In this section, we present MNDQ, a new algorithm for motion planning in an unknown dynamic environment using the game theory. The general framework of MNDQ is shown in Fig. 4. First, search-and-rescue tasks in dynamic environments are simplified through a multi-task hierarchical structure. In the process of simplification, a risk-monitoring mechanism is proposed to more clearly express the impact of patrols. Then, the basic interaction between the agent and the environment generates the trajectory of real experience. The arrow of direct RL update in the figure represents direct RL based on real experience, which improves the value function and strategy. The model learning section is a model-based process, and the model is learned from real experience and gives rise to simulated experience. Search control is used to refer to the process of selecting starting states and actions for the simulated experience generated by the model. In search

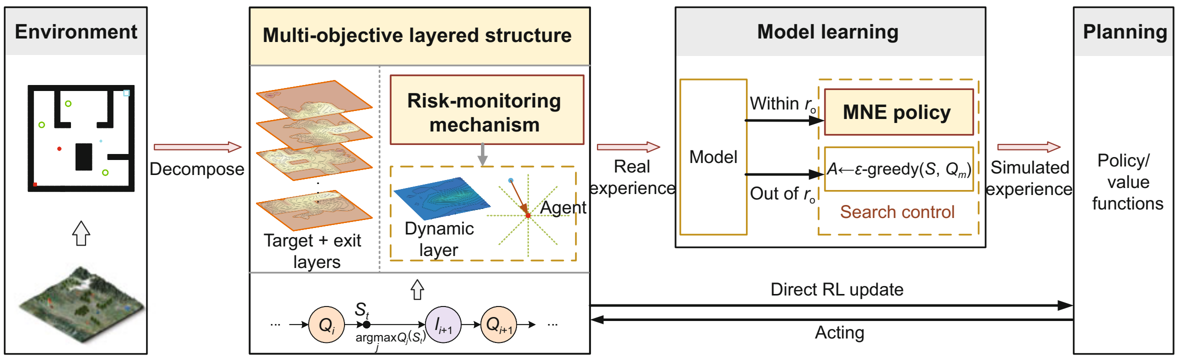

Fig. 4 The general framework of the mixed-strategy Nash equilibrium based Dyna-Q (MNDQ) algorithm. The framework integrates the Dyna architecture with three key technologies: multi-objective layered structure, risk-monitoring mechanism, and mixed-strategy Nash equilibrium (MNE) policy (RL: reinforcement learning)

control, if the patrolman is within the observation range r_o of our robot, actions are selected according to the probability distribution given by the MNE policy, as shown in Fig. 5. If not, ε-greedy action is chosen (Sutton and Barto, 1999). The real experience generated by the environment and the simulation experience observed from a model are used to update the policy/value functions. Actions are selected based on the policy function. Acting causes transitions from state to state.

Specifically, the real experience is composed of a group of subtasks (collecting targets, avoiding the patrolman, and exiting). The subtask of avoiding the patrolman is presented by the risk-monitoring mechanism. The simulated experience is generated by the model with the MNE policy. Three key technologies are presented in the following subsections. To simplify notation, we write $\{S_t, A_t, R_t, S_{t+1}\}$ simply as $\{S, A, R, S'\}$.

5.1 Multi-objective layered structure

When multiple objectives need to be collected in a dynamic environment, the execution results would be poor. To solve this problem, we design a multi-objective layered structure to simplify these multi-stage tasks. The task model is layered to improve learning accuracy and efficiency. The overall task is decomposed into a group of discrete subtasks associated with each other through the multi-objective layered structure.

The objective is to maximize the total rewards obtained in the long run. R_i denotes the reward received after the i^th selection of the action, and Q_n denotes the estimate of the action's value after it has

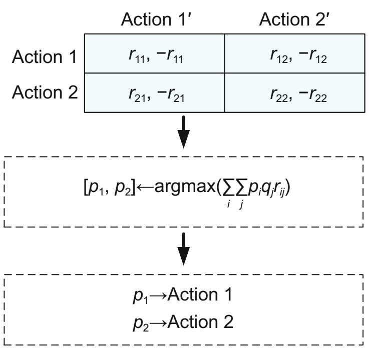
Action 1′
Action 2′
Action 1
Action 2
$r_{11}, -r_{11}$
$r_{12}, -r_{12}$
$r_{21}, -r_{21}$
$r_{22}, -r_{22}$
$[p_1, p_2] \leftarrow \text{argmax}(\sum_i \sum_j p_i q_j r_{ij})$
$p_1 \rightarrow$Action 1
$p_2 \rightarrow$Action 2

Fig. 5 Mixed-strategy Nash equilibrium policy

been selected $n-1$ times. There are multiple targets to be collected. When the targets are collected, or the exit is reached, the robot reaps the reward. The action–value function Q directly approximates Q_*, the optimal action–value function, independent of the policy followed. In the proposed structure, the whole original learning task is decomposed into smaller learning subtasks: target collection, exit, and avoidance of dynamic risks. The number of smaller learning subtasks is defined as k. The action–value function of the i^th learning subtask Q_i is defined by the following expression:

$$Q_i(S,A) \leftarrow$$
$$Q_i(S,A) + \alpha \left[R_i + \gamma \max_a Q_i(S', a) - Q_i(S,A) \right], \tag{5}$$

where α denotes the step-size parameter and $1 \leq i \leq k$. The order and orchestration of tasks depend on the value of Q_i in the current state S. After the learning subtask i is learned, the state at the time S is regarded as the terminal state of learning subtask i. The layer of the i^th learning subtask is defined as l_i.

The order of learning tasks is determined according to

$$l_{i+1}(S) = \arg\max_j Q_j(S), \tag{6}$$

where $j = K \setminus \{\{l_1\} \cup \{l_2\} \cup \cdots \cup \{l_i\}\}$ denotes all the learning tasks excluding the layers that have been learned $(l_1, l_2, ..., l_i)$ with $K = \{l_1, l_2, ..., l_k\}$.

The decision mode of the task sequence is expressed as shown in Fig. 6. The action–value function corresponding to each layer of learning subtasks under different states with the multi-objective layered structure is shown in Fig. 7. Decomposition allows each subtask to focus solely on the relevant states and actions that are specific to that particular subtask. This approach effectively reduces the computational complexity of the overall problem.

5.2 Risk-monitoring mechanism

A risk-monitoring mechanism is proposed to provide a clearer representation of the impact of a patrolman. This mechanism is based on assessment of the relative positions of the dynamic risks. The state of a dynamic risk is expressed using the Manhattan distance and relative orientation. The value of the Manhattan distance is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes, as shown in Fig. 8a. If the current position of the patrolman is $(x_{\text{patrol}}, y_{\text{patrol}})$ and the current position of the robot is $(x_{\text{robot}}, y_{\text{robot}})$, the Manhattan distance $d_{\text{p-r}}$ between the robot and the patrolman is determined according to

$$d_{\text{p-r}} = |x_{\text{patrol}} - x_{\text{robot}}| + |y_{\text{patrol}} - y_{\text{robot}}|. \tag{7}$$

The relative orientation $\theta_{\text{p-r}}$ is used to describe the location of the patrolman from the robot

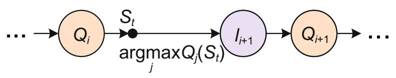

Fig. 6 Decision mode of the task sequence. The order and orchestration of tasks depend on the value of Q_i in the current state S

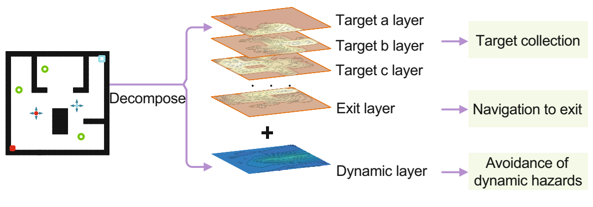

Fig. 7 Multi-objective layered structure

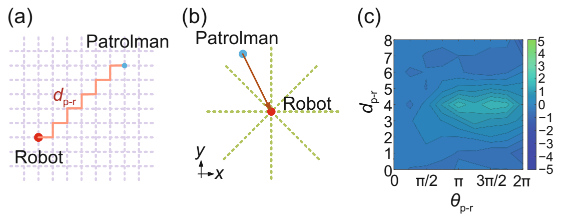

Fig. 8 Manhattan distance $d_{\text{p-r}}$ (a), relative orientation $\theta_{\text{p-r}}$ (b), and the contour plot of $\max_A Q_e(S_e)$ (c)

(Fig. 8b). $\theta_{\text{p-r}}$ is defined with the positions of the robot and the obstacle:

$$\theta_{\text{p-r}} = \begin{cases} 0, & \Delta y = 0, \Delta x \geq 0, \\ \arctan(\Delta y/\Delta x), & \Delta y > 0, \Delta x > 0, \\ \pi/2, & \Delta y > 0, \Delta x = 0, \\ \pi + \arctan(\Delta y/\Delta x), & \Delta y > 0, \Delta x < 0, \\ \pi, & \Delta y = 0, \Delta x < 0, \\ \pi + \arctan(\Delta y/\Delta x), & \Delta y < 0, \Delta x < 0, \\ 3\pi/2, & \Delta y < 0, \Delta x = 0, \\ 2\pi + \arctan(\Delta y/\Delta x), & \Delta y < 0, \Delta x > 0, \end{cases} \tag{8}$$

where $\Delta x = x_{\text{patrol}} - x_{\text{robot}}$ and $\Delta y = y_{\text{patrol}} - y_{\text{robot}}$. According to Eqs. (5), (7), and (8), we combine the multi-objective layered structure and risk-monitoring mechanism. Avoiding dynamic risks is regarded as a smaller learning subtask in the task level that runs through the learning process entirely. The state in this task level is defined as follows:

$$S_{\text{e}} = (\theta_{\text{p-r}}, d_{\text{p-r}}), \tag{9}$$

where the value of $\theta_{\text{p-r}}$ is based on Eq. (8) and the value of $d_{\text{p-r}}$ is within the sensing range of the robot. Combining Eqs. (5) and (10), the action–value function Q_{e} of avoiding dynamic risks in the task level is defined by the following expression:

$$Q_{\text{e}}(S_{\text{e}}, A) \leftarrow Q_{\text{e}}(S_{\text{e}}, A) + \alpha \left[R + \gamma \max_a Q_{\text{e}}(S'_{\text{e}}, a) - Q_{\text{e}}(S_{\text{e}}, A)\right]. \tag{10}$$

Risks are quantified and expressed intuitively based on the above risk-monitoring mechanism. The contour plot of $\max_A Q_{\text{e}}(S_{\text{e}})$ is shown in Fig. 8c, and the position of the patrolman affects the action–value function. The mechanism exploits observation information within a local area to avoid potential collisions and unnecessary detours.

5.3 The MNE policy

Simulated transitions in Dyna-Q are started in state–action pairs selected uniformly and randomly from all previously experienced pairs. The random selection policy may slow down the convergence rate. Experience and samples should be focused on specific state–action pairs. The game theory helps the robot choose better specific state–action pairs. Continued exploration may cause failure to avoid obstacles in time, and excessive avoidance of obstacles may reduce exploration efficiency. The decision between continuing to explore targets and avoiding the patrolman is necessary.

For better decisions, combined with the idea of operational research, the Nash equilibrium strategy is used to select actions to obtain higher returns. Pure strategic Nash equilibrium adopts only the action with the highest return (Rosenthal, 1973), limiting exploration to other potential solutions. In the case of different returns from search-and-rescue tasks, pure strategy Nash equilibrium does not necessarily exist. However, there must be mixed-strategy Nash equilibrium (Greenwald and Hall, 2003). Mixed strategies are probability distributions over the pure strategies, and the payoff functions are the expectations of the players, thus becoming multilinear forms in the probabilities with which the players play their various pure strategies (Nash, 1950).

To prevent the patrolman from predicting our robot's strategy, it is advisable not to reveal it definitively. If the robot's action is certain, the patrolman may use this certainty to cause harm to the robot. Therefore, it is considered optimal to adopt a randomized approach. To prevent the only action from being predicted by the patrolman and to encourage the robot to explore potential solutions, we propose an MNE policy to select an action in the form of probability distribution.

In motion planning with an unknown dynamic environment, the relationship between the patrolman and the robot is considered a zero-sum game, meaning that the rewards of the patrolman and the robot sum to zero. Such a game is specified by a single matrix, with the two strategy sets corresponding to row i and column j, as shown in Fig. 9. The matrix describes the rewards in the motion planning game. The entry r_{ij}^{A} specifies the reward of the row player (robot) in the outcome (i,j) and r_{ij}^{B} represents the reward of the column player (patrolman) in this outcome, satisfying $r_{ij}^{\mathrm{A}}+r_{ij}^{\mathrm{B}}=0$. Row and column players prefer larger and smaller values of r_{ij}^{A}, respectively.

The notations $[p_1,p_2]$ and $[q_1,q_2]$ are used to denote the mixed strategies (probability distributions) over the rows and columns, respectively. With mixed strategies, each player is randomized independently. According to the definition of the problem mentioned above, the enemy decides whether to continue patrolling or tracking the robot according to the current state. Meanwhile, the robot should choose to avoid the enemy immediately or continue exploring targets based on the observation information. The robot aims to maximize its expected rewards based on the MNE. Thus, the robot's behavior is determined according to

$$[p_1,p_2]\leftarrow \arg\max\left(\sum_i\sum_j p_i q_j r_{ij}^{\mathrm{A}}\right). \tag{11}$$

In Eq. (11), $\sum p_i=1$ and $\sum q_j=1$. The probability of the robot's behavior is expressed as follows:

$$\begin{cases} p_1 r_{11}^{\mathrm{A}}+p_2 r_{21}^{\mathrm{A}}=p_1 r_{12}^{\mathrm{A}}+p_2 r_{22}^{\mathrm{A}},\\ p_1+p_2=1, \end{cases} \tag{12}$$

which is converted into the following form:

$$\begin{bmatrix} r_{11}^{\mathrm{A}}-r_{12}^{\mathrm{A}} & r_{21}^{\mathrm{A}}-r_{22}^{\mathrm{A}}\\ 1 & 1 \end{bmatrix}\begin{bmatrix} p_1\\ p_2 \end{bmatrix}=\begin{bmatrix} 0\\ 1 \end{bmatrix}. \tag{13}$$

Accordingly, the MNE policy is given in Algorithm 1. When the patrolman is within the observation range of our robot, actions are selected according to the probability distribution given by the MNE policy, as shown in Fig. 5. Different actions give different rewards to the intelligent agent. The reward matrix is obtained at each step based on the surrounding environment. Then, we solve Eq. (13) and obtain $[p_1,p_2]$. We select action 1 with a probability of p_1 and action 2 with a probability of p_2, using a

Player A (robot)	Player B (patrolman)	
	B_1 (patrol)	B_2 (track)
A_1 (avoid)	r_{11}^{A}, r_{11}^{B}	r_{12}^{A}, r_{12}^{B}
A_2 (explore)	r_{21}^{A}, r_{21}^{B}	r_{22}^{A}, r_{22}^{B}

Fig. 9 Structure of the game played between players A and B

Algorithm 1 Mixed-strategy Nash equilibrium (MNE) policy

```
Input: state of the robot $S$; state of the risk $S_\mathrm{e}$;
    $Q_\mathrm{e}(s_\mathrm{e}, a)$; $Q = \{Q_1(s,a), Q_2(s,a), ..., Q_k(s,a)\}$;
    $m = \arg\max Q_j(S), Q_j \in Q$; $\forall s_\mathrm{e} \in S_\mathrm{e}$, $\forall s \in S$, $\forall a \in A$
Output: action $A$
1: Initialize: Probability of the robot avoiding obstacles immediately $p_1$; probability of the robot continuing to explore targets $p_2$; probability of the patrolman patrolling $q_1$; probability of the patrolman tracking $q_2$; action $A$
2: $r_{ij} \leftarrow S, S_\mathrm{e}$,
   $[p_1, p_2] \leftarrow \arg\max \left(\sum_i \sum_j p_i q_j r_{ij}\right)$
3: if rand(1)$< p_1$, then
4:     Avoid the patrolman immediately,
       $A \leftarrow \varepsilon$-greedy$(S, Q_\mathrm{e})$
5: else
6:     Continue to explore targets,
       $A \leftarrow \varepsilon$-greedy$(S, Q_m)$
7: end if
```

random function. Furthermore, when the patrolman is outside the observation range of the robot, actions are selected according to the greedy algorithm. Algorithm 2 specifies the complete MNDQ in a procedural form. Model(S, S_e, A) denotes the contents of the planning model.

6 Simulations and analysis

We describe the learning and validation of MNDQ for motion planning in an unknown dynamic environment. The proposed algorithm is applied to the ablation studies to prove its efficiency and superiority. Furthermore, MNDQ is compared and evaluated with classic algorithms.

6.1 Simulation setting

The environment is a 2D grid world map, as shown in Fig. 3. The speeds of the robot and the patrolman are set to move one unit length per unit of time. The parameter configurations adopted in the simulations are listed in Table 1. The simulation environment is a computer with an Intel i7-8700 central processing unit (CPU) and 32 GB memory. All the algorithms are implemented with Python 3.6. The simulations are designed based on the OpenAI Gym Python library (Brockman et al., 2016). We compare our method with classic Q-learning algorithm (Mirchevska et al., 2021), Dyna-Q algorithm (Liu SJ and Tong, 2021), and DDPG (Dong and Zou, 2020), which are typical methods to perform search-and-rescue tasks.

Algorithm 2 Mixed-strategy Nash equilibrium based Dyna-Q (MNDQ)

```
Input: state of the robot $S$; state of the risk $S_\mathrm{e}$; number of planning steps $n_\mathrm{p}$; number of smaller learning subtasks $k$; radius of observation range $r_\mathrm{o}$
Output: the learned state–action value function
1: Initialize: $Q = \{Q_1(s,a), Q_2(s,a), ..., Q_k(s,a)\}$; $Q_\mathrm{e}(s_\mathrm{e}, a)$
2: $K = 0$
3: $S \leftarrow$ current state, $S_\mathrm{e} \leftarrow$ current risk
4: while $K < k$ do
5:     $m = \arg\max_j Q_j(S), Q_j \in Q$
6:     $K = K + 1$, remove $Q_m$ from $Q$
7:     $A \leftarrow \varepsilon$-greedy$(S, Q_m)$; take action $A$; observe resultant reward $R$, state $S'$, and risk $S'_\mathrm{e}$
8:     for each $i \in \{1, 2, \ldots, k\}$ do
9:         $Q_i(S,A) \leftarrow Q_i(S,A)$
               $+\alpha\left[R + \gamma \max_a Q_i(S', a) - Q_i(S,A)\right]$
10:    end for
11:    $Q_\mathrm{e}(S_\mathrm{e}, A) \leftarrow Q_\mathrm{e}(S_\mathrm{e}, A)$
           $+\alpha\left[R + \gamma \max_a Q_\mathrm{e}(S'_\mathrm{e}, a) - Q_\mathrm{e}(S_\mathrm{e}, A)\right]$
12:    Model$(S, S_\mathrm{e}, A) \leftarrow R, S', S'_\mathrm{e}$
13:    for $N$=1:$n_\mathrm{p}$ do
14:        $S \leftarrow$ random previously observed state
15:        if the enemy is within the observation range then
16:            $A \leftarrow$ MNE policy in $S, S_\mathrm{e}$
17:        else
18:            $A \leftarrow \varepsilon$-greedy$(S, Q_m)$
19:        end if
20:        $R, S', S'_\mathrm{e} \leftarrow$ Model$(S, S_\mathrm{e}, A)$
21:        for each $i \in \{1, 2, \ldots, k\}$ do
22:            $Q_i(S,A) \leftarrow Q_i(S,A)$
                   $+\alpha\left[R + \gamma \max_a Q_i(S', a) - Q_i(S,A)\right]$
23:        end for
24:        $Q_\mathrm{e}(S_\mathrm{e}, A) \leftarrow Q_\mathrm{e}(S_\mathrm{e}, A)$
               $+\alpha\left[R + \gamma \max_a Q_\mathrm{e}(S'_\mathrm{e}, a) - Q_\mathrm{e}(S_\mathrm{e}, A)\right]$
25:    end for
26: end while
```

If the robot crashes into the obstacles, a reward of -10 is obtained. A reward of $+1$ is obtained if the robot reaches the goal or collects the targets. When the robot is trained, the patrolman takes the behavior with the highest rewards for every episode. At each time step, the environment sends a reward

Table 1 Parameter configurations adopted in the simulations

Parameter	Value
Discount rate γ	0.95
Exploring rate ε	0.1
Number of planning steps n_{p}	5
Radius of observation range r_{o}	5
Learning rate α	0.1
Episode number N_{e}	300

to the robot. The robot's sole objective is to maximize the total rewards it receives over the long run. Furthermore, average learning curves indicate the training efficiency. To compare the efficacy, we process the curves and show rewards and the number of steps taken by the robot to accomplish the task in each episode. All the simulations are obtained by an average of 10 repeat runs. The standard deviation is indicated by the shadow regions.

6.2 Ablation studies

Ablation studies are designed to prove the effectiveness of the key technologies proposed in this study. MNDQ is compared with MNDQ without MNE policy (Exp. 1), MNDQ without MNE policy or risk-monitoring mechanism (Exp. 2), and MNDQ without MNE policy, risk-monitoring mechanism, or multi-objective layered structure (Exp. 3). The simulations are performed in an unknown environment map (map size: 20×20) with obstacles and a patrolman, as shown in Fig. 1. The ablation study curves of rewards and the number of steps taken by the robot to accomplish the task in each episode are presented. Fig. 10 shows the learning curves using the four algorithms and reveals the differences in the performances of the algorithms. According to the learning curves of rewards (Fig. 10a), the MNDQ algorithm increases the average rewards to a positive number in the 100th episode. However, it is difficult for other algorithms (Exp. 1, 2, and 3) to achieve positive rewards. In Fig. 10b, the average numbers of steps taken by the robot to accomplish the task are shown.

The results illustrate that the MNDQ algorithm improves the learning performance. The multi-objective layered structure simplifies the condition of multiple objectives and effectively prevents the path from falling into local optimum. The MNE policy helps the robot decide between obstacle avoidance and exploration to ensure that the task is accomplished robustly and efficiently. The complete trajectory is shown in Fig. 11. Starting from the starting point, the robot collects all targets while avoiding obstacles and avoiding tracking by the patrolman. Fig. 12 shows the trajectories after 300 training episodes using the proposed algorithm. Several key steps during the whole navigation process are selected. The large red circle indicates the observation range of the robot in its current state. The large blue circle indicates the observation range of the patrolman. The results show that the three key technologies contained in MNDQ are favorable to raising the searching ability and learning efficiency. According to the simulation results, MNDQ makes the robot effectively collect targets while avoiding collision, with a higher learning rate.

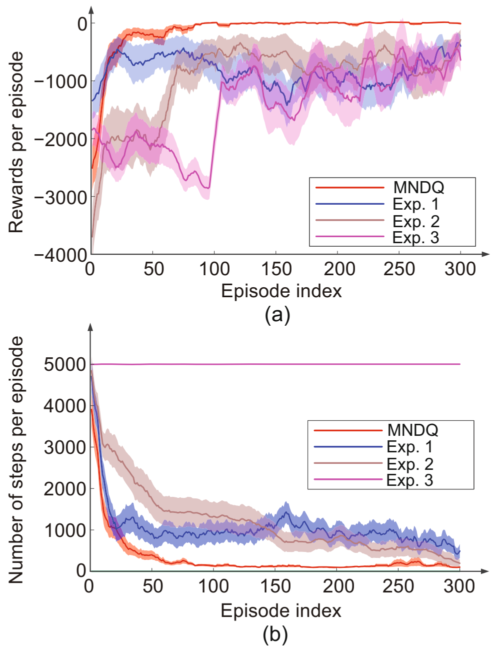

Fig. 10 Average learning curves of the ablation study: (a) rewards; (b) number of steps

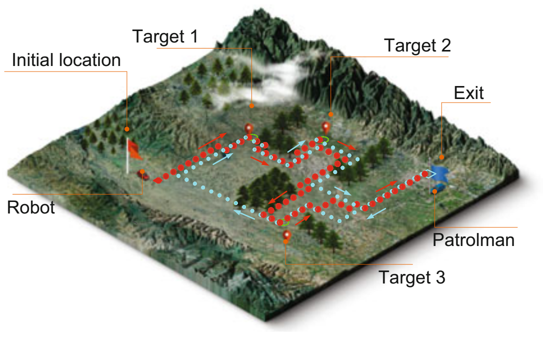

Fig. 11 The complete trajectory in an unknown environment map with obstacles and a patrolman

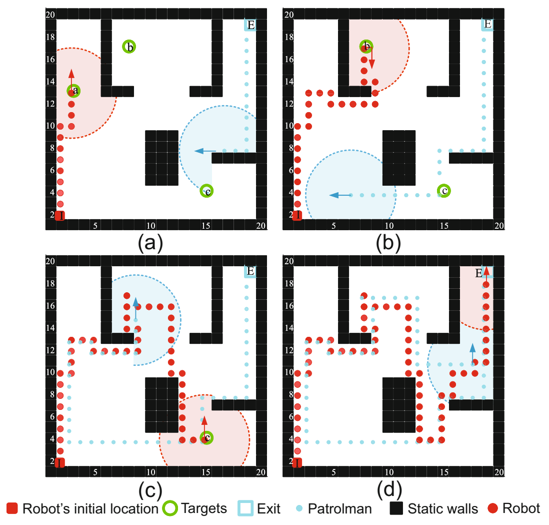

Fig. 12 Planned trajectory with static obstacles: (a–c) current positions of the robot (solid red dot) and the patrolman (solid blue dot) when the robot collects the targets; (d) the robot reaching the exit. References to color refer to the online version of this figure

6.3 Comparison in unknown dynamic environments

1. Scene 1: environment without static obstacles

The additional static obstacles are removed here to give prominence to the interaction between the robot and the patrolman. The robot needs to collect the targets in parallel with obstacle avoidance according to Section 4.

We compare the performances of different methods using various maps (size: 20×20, 10×10). The performance of MNDQ is significantly enhanced, regarding the learning curves shown in Fig. 13. It can be found that the size of the map has an impact on the completion of the task. As shown in Figs. 13a and 13c, the learning curves of the rewards taken by the robot to accomplish the task in each episode are presented. The MNDQ algorithm increases the average rewards to a positive number in the 20th episode. However, other algorithms (Q-learning with MNE policy, Dyna-Q, Q-learning, and DDPG) tend to stabilize after 100 episodes. Figs. 13b and 13d present the learning curves of the number of steps. Our method reaches convergence in the 20th episode. This means that we complete the search-and-rescue task with fewer steps. Other algorithms fail to achieve rapid convergence.

The simulation results show that the improved Dyna-Q algorithm exhibits global searching ability,

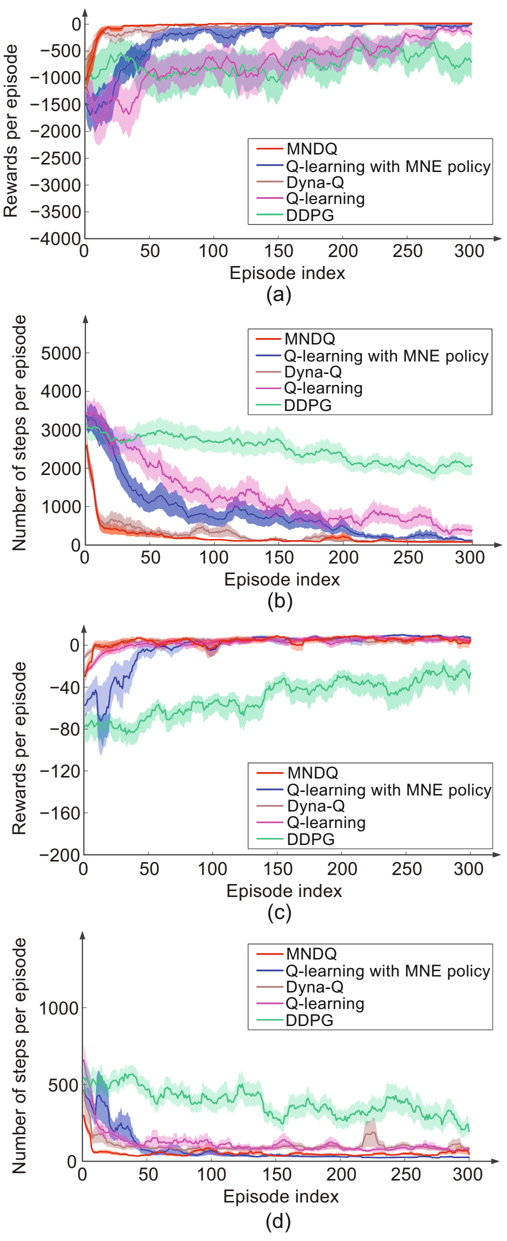

Fig. 13 Average learning curves in the unknown dynamic environments without static obstacles: (a) rewards under the map size 20×20; (b) number of steps under the map size 20×20; (c) rewards under the map size 10×10; (d) number of steps under the map size 10×10. References to color refer to the online version of this figure

excellent convergence properties, and superior learning efficiency in an unknown dynamic environment. The curves of MNDQ tend toward stability after about 100 episodes, which shows the better robustness of the model. The other methods are volatile and fail to collect all the targets or avoid obstacles. Fig. 14 shows the planned trajectory after 300 training episodes using MNDQ in an unknown environment (size: 20×20) with a patrolman.

2. Scene 2: environment with static obstacles

The simulation is performed in an unknown environment map with static obstacles and a patrolman. The map is the same as shown in Fig. 3. In addition, we compare the performances with different maps (sizes: 20×20, 10×10). The planned trajectory with MNDQ is shown by the solid red dot in Fig. 12 (size: 20×20). The learning curves of different maps (sizes: 20×20, 10×10) are shown in Fig. 15. Static obstacles inside the environment make it more difficult to achieve convergence with classic methods. The learning curves of rewards of different algorithms are shown in Figs. 15a and 15c. The learning curves of the number of steps are shown in Figs. 15b and 15d. Different from scene 1, it is more difficult to complete the task for other algorithms because of the addition of static obstacles. Especially in the map with a size of 20×20, our method reaches convergence in the 100th episode. However, other algorithms are almost unable to converge, meaning that the robot cannot complete tasks safely.

The average numbers of collisions after training are shown in Table 2. In these cases, the number of collisions in our method is always zero. The results prove the wide applicability of the proposed algorithm for different-sized maps. The MNDQ method makes the robot fulfill search-and-rescue tasks without collisions. However, other algorithms may not perform as well as expected, leading to collisions. As the size of the map increases, the number of collisions with other algorithms increases.

To validate the performance, we take a representative environment with static obstacles as an

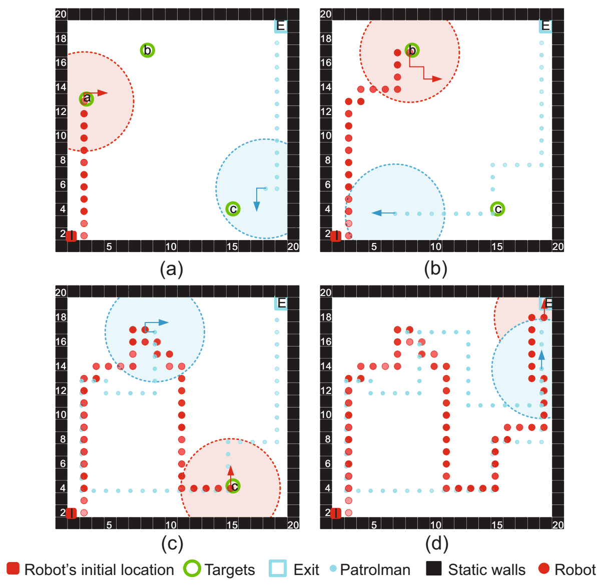

Fig. 14 Planned trajectory without static obstacles: (a–c) current positions of the robot (solid red dot) and the patrolman (solid blue dot) when the robot collects targets; (d) the robot reaching the exit

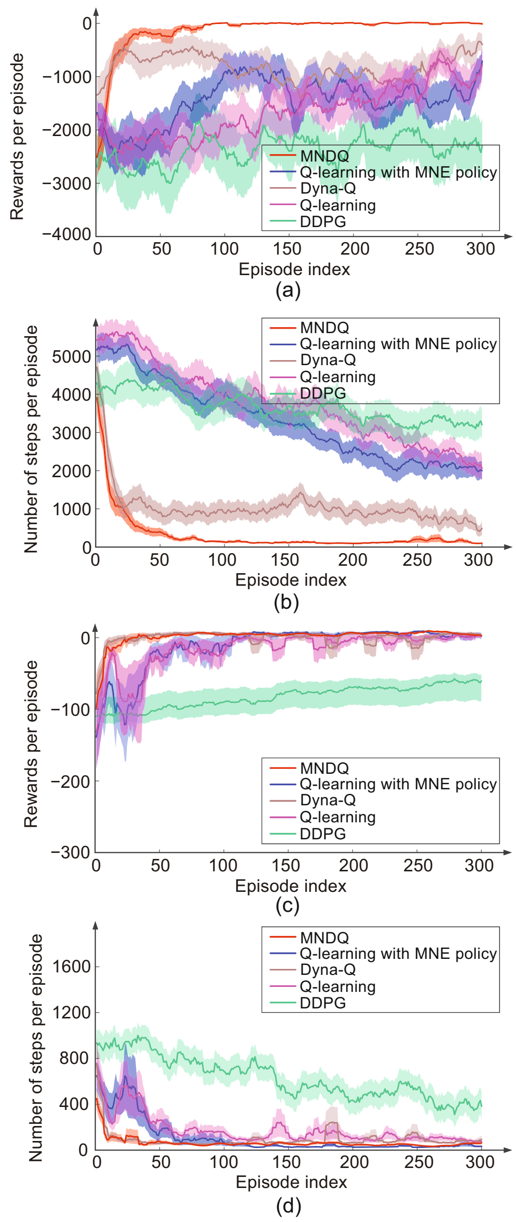

Fig. 15 Average learning curves in the unknown dynamic environments with static obstacles: (a) rewards under the map size 20 × 20; (b) number of steps under the map size 20 × 20; (c) rewards under the map size 10 × 10; (d) number of steps under the map size 10 × 10. References to color refer to the online version of this figure

example, with a map size of 20×20. The trajectories planned by different methods are shown in Fig. 16. Our method enables the robot to complete the tasks and reach the endpoint without collisions. Other methods cannot avoid collisions in most cases. When within the observation range of the patrolman, the robot with other methods is pursued, rendering the task partially or completely incomplete. In the simulations, MNDQ achieves the best performance. The robot successfully collects all targets, avoids the patrolman, and finally reaches the exit with our method.

7 Conclusions

In this paper, we investigated the robot search-and-rescue problem in unknown dynamic environments and proposed an MNDQ algorithm. The algorithm adopted a multi-objective layered structure to decompose the task into smaller learning subtasks. We proposed a risk-monitoring mechanism to help the robot generate a collision-free static trajectory. These mechanisms were designed to express and simplify the tasks. Furthermore, we combined the knowledge of the game theory and designed the MNE policy to choose better specific state–action pairs and enhance the sample efficiency. The policy enabled the agent to make decisions in the form of probability to maximize the expected rewards and improved the performance of Dyna-Q. The simulation results proved that MNDQ is feasible and effective for motion planning in an unknown dynamic environment.

In future research, we plan to run the proposed algorithm with real-world experiments. Continuous action and state spaces could be further considered to generate smooth trajectories. Besides, search-and-rescue tasks in large-scale environments are worth

Table 2 Average number of collisions

Scene	Average number of collisions				
	MNDQ	Dyna-Q	Q-learning	Q-learning with MNE policy	DDPG
Scene 1 (20×20)	0	1.4	5.5	1.6	>10
Scene 1 (10×10)	0	1.2	2.4	0	9.3
Scene 2 (20×20)	0	3.6	>10	8.2	>10
Scene 2 (10×10)	0	3.3	2.7	1.5	>10

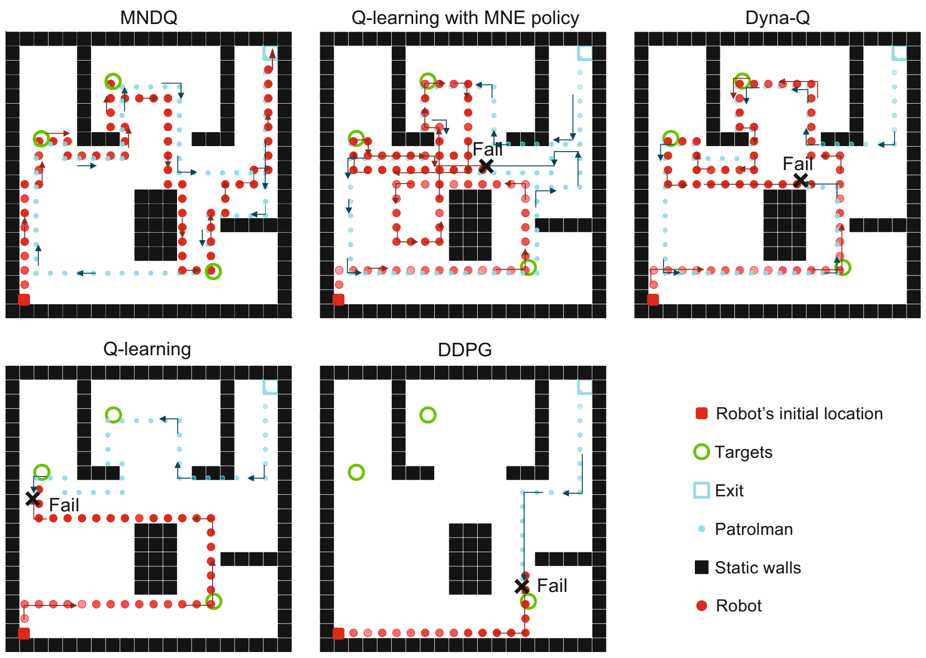

Fig. 16 Planned trajectories obtained by different algorithms (arrows indicate the directions of the trajectories)

studying further. Specifically, the focus of research is how to break the curse of dimensionality. Studying RL algorithms in large-scale dynamic environments is a highly meaningful task, and we plan to investigate it in our future work.

Contributors

Yang CHEN designed the research and drafted the paper. Yang CHEN and Dianxi SHI processed the data. Yang CHEN and Huanhuan YANG designed the simulations. Tongyue LI and Zhen WANG helped organize the paper. Yang CHEN and Dianxi SHI revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Aggarwal S, Kumar N, 2020. Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. *Comput Commun*, 149:270-299. https://doi.org/10.1016/j.comcom.2019.10.014

Banerjee C, Datta D, Agarwal A, 2015. Chaotic patrol robot with frequency constraints. Proc IEEE Int Conf on Research in Computational Intelligence and Communication Networks, p.340-344. https://doi.org/10.1109/ICRCICN.2015.7434261

Brito B, Floor B, Ferranti L, et al., 2019. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robot Autom Lett*, 4(4):4459-4466. https://doi.org/10.1109/LRA.2019.2929976

Brito B, Everett M, How JP, et al., 2021. Where to go next: learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robot Autom Lett*, 6(3):4616-4623. https://doi.org/10.1109/LRA.2021.3068662

Brockman G, Cheung V, Pettersson L, et al., 2016. OpenAI Gym. https://arxiv.org/abs/1606.01540

Chiu ZY, Richter F, Funk EK, et al., 2021. Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning. Proc IEEE Int Conf on Robotics and Automation, p.7737-7743. https://doi.org/10.1109/ICRA48506.2021.9561673

Curiac DI, Banias O, Volosencu C, et al., 2018. Novel bioinspired approach based on chaotic dynamics for robot patrolling missions with adversaries. *Entropy*, 20(5):378. https://doi.org/10.3390/e20050378

Dong YS, Zou XJ, 2020. Mobile robot path planning based on improved DDPG reinforcement learning algorithm. Proc IEEE 11th Int Conf on Software Engineering and Service Science, p.52-56. https://doi.org/10.1109/ICSESS49938.2020.9237641

Faust A, Oslund K, Ramirez O, et al., 2018. PRM-RL: long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. Proc IEEE Int Conf on Robotics and Automation, p.5113-5120. https://doi.org/10.1109/ICRA.2018.8461096

Fudenberg D, Tirole J, 1991. Game Theory. MIT Press, Cambridge, USA.

Gaertner M, Bjelonic M, Farshidian F, et al., 2021. Collision-free MPC for legged robots in static and dynamic scenes. Proc IEEE Int Conf on Robotics and Automation, p.8266-8272. https://doi.org/10.1109/ICRA48506.2021.9561326

Geng N, Meng QG, Gong DW, et al., 2019. How good are distributed allocation algorithms for solving urban search and rescue problems? A comparative study with centralized algorithms. *IEEE Trans Autom Sci Eng*, 16(1):478-485. https://doi.org/10.1109/TASE.2018.2866395

Greenwald A, Hall K, 2003. Correlated-Q-learning. Proc 20th Int Conf on Machine Learning, p.242-249.

Gregor M, Nemec D, Janota A, et al., 2018. A visual attention operator for playing Pac-Man. Proc ELEKTRO, p.1-6. https://doi.org/10.1109/elektro.2018.8398308

Hayamizu Y, Amiri S, Chandan K, et al., 2021. Guiding robot exploration in reinforcement learning via automated planning. Proc 31st Int Conf on Automated Planning and Scheduling, p.625-633. https://doi.org/10.1609/icaps.v31i1.16011

Hong LB, Wang Y, Du YC, et al., 2021. UAV search-and-rescue planning using an adaptive memetic algorithm. *Front Inform Technol Electron Eng*, 22(11):1477-1491. https://doi.org/10.1631/FITEE.2000632

Hu JL, Wellman MP, 2003. Nash Q-learning for general-sum stochastic games. *J Mach Learn Res*, 4:1039-1069.

Hubert T, Schrittwieser J, Antonoglou I, et al., 2021. Learning and planning in complex action spaces. Proc 38th Int Conf on Machine Learning, p.4476-4486.

Hwang KS, Lin JL, Huang HL, 2011. Dynamic patrol planning in a cooperative multi-robot system. Proc 14th FIRA RoboWorld Congress, p.116-123. https://doi.org/10.1007/978-3-642-23147-6_14

Hwang KS, Jiang WC, Chen YJ, 2015. Model learning and knowledge sharing for a multiagent system with Dyna-Q learning. *IEEE Trans Cybern*, 45(5):978-990. https://doi.org/10.1109/TCYB.2014.2341582

Jaderberg M, Czarnecki WM, Dunning I, et al., 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859-865. https://doi.org/10.1126/science.aau6249

Lei XY, Zhang Z, Dong PF, 2018. Dynamic path planning of unknown environment based on deep reinforcement learning. *J Rob*, 2018:5781591. https://doi.org/10.1155/2018/5781591

Li CH, Fang C, Wang FY, et al., 2019. Complete coverage path planning for an Arnold system based mobile robot to perform specific types of missions. *Front Inform Technol Electron Eng*, 20(11):1530-1542. https://doi.org/10.1631/FITEE.1800616

Li HQ, Huang J, Cao Z, et al., 2023. Stochastic pedestrian avoidance for autonomous vehicles using hybrid reinforcement learning. *Front Inform Technol Electron Eng*, 24(1):131-140. https://doi.org/10.1631/FITEE.2200128

Li HR, Zhang QC, Zhao DB, 2020. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Trans Neur Netw Learn Syst*, 31(6):2064-2076. https://doi.org/10.1109/TNNLS.2019.2927869

Li ZR, Lu C, Yi YT, et al., 2022. A hierarchical framework for interactive behaviour prediction of heterogeneous traffic participants based on graph neural network. *IEEE Trans Intell Transp Syst*, 23(7):9102-9114. https://doi.org/10.1109/TITS.2021.3090851

Liu SJ, Tong XR, 2021. Urban transportation path planning based on reinforcement learning. *J Comput Appl*, 41(1):185-190 (in Chinese). https://doi.org/10.11772/j.issn.1001-9081.2020060949

Liu YY, Yan SH, Zhao Y, et al., 2022. Improved Dyna-Q: a reinforcement learning method focused via heuristic graph for AGV path planning in dynamic environments. *Drones*, 6(11):365. https://doi.org/10.3390/drones6110365

Liu Z, Cao YQ, Chen JY, et al., 2023. A hierarchical reinforcement learning algorithm based on attention mechanism for UAV autonomous navigation. *IEEE Trans Intell Transp Syst*, 24(11):13309-13320. https://doi.org/10.1109/TITS.2022.3225721

Lu YL, Yan K, 2020. Algorithms in multi-agent systems: a holistic perspective from reinforcement learning and game theory. https://arxiv.org/abs/2001.06487

Lu YM, Kamgarpour M, 2020. Safe mission planning under dynamical uncertainties. Proc IEEE Int Conf on Robotics and Automation, p.2209-2215. https://doi.org/10.1109/ICRA40945.2020.9196515

Luo GY, Wang YT, Zhang H, et al., 2023. AlphaRoute: large-scale coordinated route planning via Monte Carlo tree search. Proc 37th AAAI Conf on Artificial Intelligence, p.12058-12067. https://doi.org/10.1609/aaai.v37i10.26422

Martins-Filho LS, Macau EEN, 2007. Patrol mobile robots and chaotic trajectories. *Math Probl Eng*, 2007:061543. https://doi.org/10.1155/2007/61543

McGuire KN, de Croon GCHE, Tuyls K, 2019. A comparative study of bug algorithms for robot navigation. *Robot Auton Syst*, 121:103261. https://doi.org/10.1016/j.robot.2019.103261

Mirchevska B, Hügle M, Kalweit G, et al., 2021. Amortized Q-learning with model-based action proposals for autonomous driving on highways. Proc IEEE Int Conf on Robotics and Automation, p.1028-1035. https://doi.org/10.1109/ICRA48506.2021.9560777

Nasar W, da Silva Torres R, Gundersen OE, et al., 2023. The use of decision support in search and rescue: a systematic literature review. *ISPRS Int J Geo-Inform*, 12(5):182. https://doi.org/10.3390/ijgi12050182

Nash JFJr, 1950. Equilibrium points in n-person games. *Proc Natl Acad Sci USA*, 36(1):48-49. https://doi.org/10.1073/pnas.36.1.48

Ng J, Bräunl T, 2007. Performance comparison of bug navigation algorithms. *J Intell Robot Syst*, 50(1):73-84. https://doi.org/10.1007/s10846-007-9157-6

Niroui F, Sprenger B, Nejat G, 2017. Robot exploration in unknown cluttered environments when dealing with uncertainty. Proc IEEE Int Symp on Robotics and Intelligent Sensors, p.224-229. https://doi.org/10.1109/IRIS.2017.8250126

Ohnishi M, Wang L, Notomista G, et al., 2019. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Trans Robot*, 35(5):1186-1205. https://doi.org/10.1109/TRO.2019.2920206

Osborne M, Rubinstein A, 1994. A Course in Game Theory. MIT Press, Cambridge, USA.

Padakandla S, 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Comput Surv*, 54(6):127. https://doi.org/10.1145/3459991

Patle BK, Babu LG, Pandey A, et al., 2019. A review: on path planning strategies for navigation of mobile robot. *Def Technol*, 15(4):582-606. https://doi.org/10.1016/j.dt.2019.04.011

Pei M, An H, Liu B, et al., 2022. An improved Dyna-Q algorithm for mobile robot path planning in unknown dynamic environment. *IEEE Trans Syst Man Cybern Syst*, 52(7):4415-4425. https://doi.org/10.1109/TSMC.2021.3096935

Prado J, Marques L, 2014. Energy efficient area coverage for an autonomous demining robot. Proc 1st Iberian Robotics Conf, p.459-471. https://doi.org/10.1007/978-3-319-03653-3_34

Puterman ML, 1990. Markov decision processes. *Handb Oper Res Manage Sci*, 2:331-434. https://doi.org/10.1016/S0927-0507(05)80172-0

Rosenthal RW, 1973. A class of games possessing pure-strategy Nash equilibria. *Int J Game Theory*, 2(1):65-67. https://doi.org/10.1007/BF01737559

Roughgarden T, 2010. Algorithmic game theory. *Commun ACM*, 53(7):78-86. https://doi.org/10.1145/1785414.1785439

Roughgarden T, 2016. Twenty Lectures on Algorithmic Game Theory. Cambridge University Press, New York, USA. https://doi.org/10.1017/CBO9781316779309

Shi HB, Yang SK, Hwang KS, et al., 2018. A sample aggregation approach to experiences replay of Dyna-Q learning. *IEEE Access*, 6:37173-37184. https://doi.org/10.1109/ACCESS.2018.2847048

Sutton RS, Barto AG, 1999. Reinforcement learning. *J Cognit Neurosci*, 11(1):126-134. https://doi.org/10.1162/089892999563184

Sutton RS, Barto AG, 2018. Reinforcement Learning: an Introduction (2nd Ed.). MIT Press, Cambridge, USA.

Wakayama S, Ahmed NR, 2020. Auto-tuning online POMDPs for multi-object search in uncertain environments. Proc AIAA Scitech Forum. https://doi.org/10.2514/6.2020-0391

Wang BY, Liu Z, Li QB, et al., 2020. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robot Autom Lett*, 5(4):6932-6939. https://doi.org/10.1109/LRA.2020.3026638

Wu JF, Braverman V, Yang L, 2021. Accommodating picky customers: regret bound and exploration complexity for multi-objective reinforcement learning. Proc 35th Int Conf on Neural Information Processing Systems, p.13112-13124.

Wu YX, Li XJ, Liu JJ, et al., 2019. Switch-based active deep Dyna-Q: efficient adaptive planning for task-completion dialogue policy learning. Proc 33rd AAAI Conf on Artificial intelligence, p.7289-7296. https://doi.org/10.1609/aaai.v33i01.33017289

Wyrąbkiewicz K, Tarczewski T, Niewiara Ł, 2020. Local path planning for autonomous mobile robot based on APF-BUG algorithm with ground quality indicator. In: Bartoszewicz A, Kabziński J, Kacprzyk J (Eds.), Advanced, Contemporary Control. Springer, Cham, p.979-990. https://doi.org/10.1007/978-3-030-50936-1_82

Yu Y, Tang J, Huang JY, et al., 2021. Multi-objective optimization for UAV-assisted wireless powered IoT networks based on extended DDPG algorithm. *IEEE Trans Commun*, 69(9):6361-6374. https://doi.org/10.1109/TCOMM.2021.3089476

Zhang YH, Chai ZJ, Lykotrafitis G, 2021. Deep reinforcement learning with a particle dynamics environment applied to emergency evacuation of a room with obstacles. *Phys A Stat Mech Appl*, 571:125845. https://doi.org/10.1016/j.physa.2021.125845

Zheng KY, Sung Y, Konidaris G, et al., 2021. Multi-resolution POMDP planning for multi-object search in 3D. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.2022-2029. https://doi.org/10.1109/IROS51168.2021.9636737

Zheng Z, Liu Y, Zhang XY, 2016. The more obstacle information sharing, the more effective real-time path planning? *Knowl-Based Syst*, 114:36-46. https://doi.org/10.1016/j.knosys.2016.09.021

Zou LX, Xia L, Du P, et al., 2020. Pseudo Dyna-Q: a reinforcement learning framework for interactive recommendation. Proc 13th Int Conf on Web Search and Data Mining, p.816-824. https://doi.org/10.1145/3336191.3371801