2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Multi-Robot Path Planning With Boolean Specification Tasks Under Motion Uncertainties

Zhe Zhang[1], , Zhou He[2], , Ning Ran[3], , Michel Reniers[4],

***Abstract*— This paper studies the path planning problem of multi-robot systems under motion uncertainties with high-level tasks that are expressed as Boolean specifications. The specification imposes logical constraints on robot trajectories and final states. First, a global Markov decision process model of the multi-robot system is constructed to provide its current state. In order to tackle the state explosion problem, at each stage, we construct a local Markov decision process for every individual agent in sequence to compute the local optimal movement strategy and update the global Markov decision process accordingly (i.e., compute locally and update globally). Next, we propose a heuristic reward function design method that provides different rewards for visiting different task points by introducing the estimated distance to complete the global task. Finally, a series of numerical experiments are conducted to demonstrate the computational efficiency and scalability of our developed approach.**

Discrete event systems, Markov decision process, Multi-robot system, High-level task, Reinforcement learning.

I. INTROUDCTION

A. Motivation

Multi-robot systems (MRS) are widely used in many applications, such as search and rescue, autonomous warehouses, and manufacturing systems [1]. One of the central challenges for the controlling and coordination of MRS is the path planning that typically intends to complete some tasks by minimizing costs such as time, distance, or energy. The goal is to determine an optimal route for each robot to complete assigned tasks sequentially and achieve the global task ultimately [2]. Traditionally, most existing path planning algorithms focus on collision avoidance, target access, and path smoothing [3]–[5].

With the increasing complexity of tasks, considerable attention has been paid to multi-robot path planning with high-level tasks in recent years [6]. Particularly, linear temporal logic (LTL) and Boolean specifications are widely used as the formal language to express the high-level task requirements [7], [8], e.g., "finish job A in the regions a or b, avoid reaching some unsafe regions, and finally staying in the region c". In addition, MRS operating in complex environments are often subject to a variety of uncertainties (e.g., potential sensing noise or actuation failures). In [9], [10], the path planning problem under motion uncertainties is investigated and a control strategy that maximizes the probability of LTL task satisfaction is proposed. For the path planning of MRS with Boolean specifications, it aims to find the optimal paths with minimal cost such that the Boolean specifications are satisfied. In particular, Petri net models are employed to model the MRS and integer linear programming methods are proposed [11].

In this paper, we study the path planning problem of multi-robot systems with Boolean specification tasks under motion uncertainties. We model the mobility of an MRS as an Markov decision process (MDP) and employ a reinforcement learning technique to determine a control strategy that satisfies the Boolean specifications. An iterative reinforcement learning strategy that eliminates the dependence of state computations on the number of robots is developed to improve the computational efficiency. Moreover, in order to improve the solution quality, a heuristic reward function is designed. The scalability and computational efficiency of the proposed method is demonstrated by numerical experiments.

B. Related works

In [12], [13], the authors consider the optimal path planning problem with linear temporal logic constraints. In [14], [15], path planning problem with Boolean specifications are discussed by using Petri net models. A simulated annealing based algorithm is presented in [16] to find a near-optimal solution with relative low computational cost. More recently, the security-preserving multi-robot path planning with Boolean specifications is discussed in [17], which aims to plan an optimal path for each robot such that the Boolean tasks are finished, while some key information of the systems are hidden for an external intruder.

In summary, the existing optimal approaches [15] have high computational cost since the numbers of constraints and variables of the integer linear programming problem increase significantly as the task quantity and map size increase. The heuristic approach developed in [16] has relative low computational cost, but the accuracy decreases as the map size and task quantity increase. Moreover, theses approaches are limited by the fact that the modeling approach cannot be applied to MRSs with motion uncertainties. Thus, finding an efficient method for large-scaled MRSs with high-level tasks and under motion uncertainties is still a challenging problem.

II. PRELIMINARIES

An MDP is a tuple $\mathcal{P} = (S, s_0, S_f, A, P)$, where S is a finite set of states, $s_0 \in S$ is an initial state, $S_f \subseteq S$ is a finite set of final states, A is a finite set of actions, and $P : S \times A \times S \to [0, 1]$ is the transition probability function. It captures the motion uncertainties of MRSs. Let $\mathcal{A} : S \to 2^A$ denote the set of actions enabled at state $s \in S$ (i.e., $\mathcal{A}(s) = \{a \in A | \exists_{s' \in S} \in P(s, a, s') > 0\}$). The reward function $R : S \times A \times S \to \mathbb{R}$ represents the task objective.

We assume that the MDP is controllable and let $\boldsymbol{\zeta} : S \to A$ be the policy that maps a state $s \in S$ to an action $a \in \mathcal{A}(s)$. Given a policy $\boldsymbol{\zeta}$, a possible evolution of MDP satisfies: $s_0 \overset{\boldsymbol{\zeta}(s_0)}{\to} s_1 \overset{\boldsymbol{\zeta}(s_1)}{\to} \cdots \overset{\boldsymbol{\zeta}(s_{T-1})}{\to} s_T$, where $P(s_i, \boldsymbol{\zeta}(s_i), s_{i+1}) > 0$, $\forall i = 0, 1, \ldots, T-1$, $s_T \in S_f$, and T is the length of the evolution.

In this paper, notation $\mathbb{E}^{\boldsymbol{\zeta}}[X]$ denotes the expectation of a random variable X when an MDP evolves under policy $\boldsymbol{\zeta}$, and $\mathbb{E}^{\boldsymbol{\zeta}}[X|Y = y]$ denotes the expectation of X given value y for another random variable Y.

Given a discount factor $\gamma \in [0, 1]$, the value function $V_{\boldsymbol{\zeta}}(s)$ of state s under policy $\boldsymbol{\zeta}$ is defined as: $V_{\boldsymbol{\zeta}}(s) = \mathbb{E}^{\boldsymbol{\zeta}}\left[\sum_{i=0}^{T} \gamma^i R(s_i, \boldsymbol{\zeta}(s_i), s_{i+1}) \middle| s_0 = s\right]$. It represents the expected return (i.e. cumulative reward) of state s.

The action-value function $Q_{\boldsymbol{\zeta}}(s, a)$ of state-action pair (s, a) under policy $\boldsymbol{\zeta}$ is defined as:

$$Q_{\boldsymbol{\zeta}}(s, a) = \mathbb{E}^{\boldsymbol{\zeta}}\left[\sum_{i=0}^{T} \gamma^i R(s_i, \boldsymbol{\zeta}(s_i), s_{i+1}) \middle| s_0 = s, \boldsymbol{\zeta}(s_0) = a\right].$$

It represents the expected return of taking action a at state s.

The optimal policy $\boldsymbol{\zeta}^* : S \to A$ can be generated as $\boldsymbol{\zeta}^* = \arg\max_{\boldsymbol{\zeta}(s)} V_{\boldsymbol{\zeta}}(s) = \arg\max_{a \in \mathcal{A}(s)} Q^*(s, a)$.

III. Task specification and problem statement

A. Boolean specification

In this paper, we consider a team of k identical robots $\mathcal{R} = \{r_1, \ldots, r_k\}$ working in a known environment $C = \{c_1, c_2, \ldots, c_n\}$ that contains n cells. We denote by c_{s_i} and $\vec{C_I} = [c_{s_1}, c_{s_2}, \ldots, c_{s_k}]^{1\times k}$ the initial cell of robot r_i and *vector of initial cells*, respectively. We define the set of regions of interest $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\} \subseteq 2^C$, where region $\omega_i \subseteq$
C denotes a set of cells of interest. A *trajectory* $\sigma \in S^*$ of MDP is a finite sequence of states and function $\lambda : S \times \mathcal{R} \to C$ represents the positions of each robot at state s.

The task requirements of the MRS expressed as a *Boolean specification* whose elementary unit is an atomic proposition in $P_t \cup P_f$, where $P_t = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Omega|}\}$ and $P_f = \{\Pi'_1, \Pi'_2,$
$\ldots, \Pi'_{|\Omega|}\}$. Given a *trajectory* $\sigma = s_0 s_1 \cdots s_T$, $\forall i = 1, \ldots, |\Omega|$, $\Pi_i \in P_t$ evaluates to *True* if $\lambda(s_j) \cap \omega_i \neq \emptyset$, $\exists j = 0, \ldots, T$ and $\Pi'_i \in P_f$ evaluates to *True* if $\lambda(s_T) \cap \omega_i \neq \emptyset$.

By using logical connectors $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation), the atomic propositions can be connected. The Boolean specification φ contains the following three sub-specifications:

$$\varphi = \mathcal{M} \wedge \mathcal{N} \wedge \mathcal{F}. \tag{1}$$

The sub-specification $\mathcal{M}$ denotes the logic requirements on the *task regions* such that

$$\mathcal{M} = m_1 \wedge m_2 \wedge \cdots \wedge m_p, m_i = \bigvee_{\Pi_j \in P_{m_i}} \Pi_j, \tag{2}$$

where $P_{m_i} \subseteq P_t$ indicates that along the trajectory, one of the regions corresponding to atomic propositions in P_{m_i} should be visited. We denote by $\Omega_{\mathcal{M}} = \bigcup_{i=1}^{p} \Omega_{m_i}$ the *set of task cells*, where $\Omega_{m_i} = \{c | c \in \bigcup_{\Pi_j \in P_{m_i}} \omega_j\}$ represents the set of task cells that correspond to atomic propositions $\Pi_j \in P_{m_i}$.

The sub-specification $\mathcal{N}$ denotes the logic requirements on the *forbidden regions* such that

$$\mathcal{N} = \bigwedge_{\Pi_i \in P_n} (\neg \Pi_i) = \neg(\bigvee_{\Pi_i \in P_n} \Pi_i), \tag{3}$$

where $P_n \subseteq P_t$ means that regions which corresponding to atomic propositions in P_n should be avoided along the trajectories. We denote by $\Omega_{\mathcal{N}} = \{c | c \in \bigcup_{\Pi_i \in P_n} \omega_i\}$ the *set of forbidden cells*.

The sub-specification $\mathcal{F}$ denotes the logic requirements on the *final regions* such that

$$\mathcal{F} = f_1 \wedge f_2 \wedge \cdots \wedge f_q, f_i = \bigvee_{\Pi'_j \in P_{f_i}} \Pi'_j, \tag{4}$$

where $P_{f_i} \subseteq P_f$ indicates that there exists at least one robot which stays in one of the regions corresponding to atomic propositions in P_{f_i} at last. We denote by $\Omega_{\mathcal{F}} = \bigcup_{i=1}^{q} \Omega_{f_j}$ the *set of final cells*, where $\Omega_{f_i} = \{c | c \in \bigcup_{\Pi'_j \in P_{f_i}} \omega_j\}$ represents the set of final cells that correspond to atomic proposition $\Pi'_j \in P_{f_i}$.

B. Problem statement

In summary, we formulate the path planning problem of MRS with Boolean specification tasks under motion uncertainties as follows.

Problem 1: Given a known environment $C = \{c_1, \ldots, c_n\}$, a team of k identical mobile robots $\mathcal{R} = \{r_1, \ldots, r_k\}$, a Boolean specification task φ in the form of Eq. (1). We model the MRS as an MDP and aim to find a *control policy* such that the Boolean specification φ is fulfilled (i.e. $\varphi = True$) eventually while the value function is maximized (i.e., travel distance is minimized).

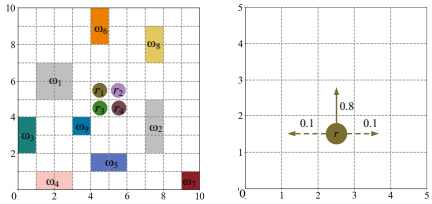
Example 1: Let us consider an MRS depicted in Fig. 1 (a) that contains four robots $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. The motion uncertainties as shown in Fig. 1 (b) (i.e., it is assumed to successfully take the desired action with a probability of 0.8, and there is a probability of 0.2 to take other perpendicular actions). The environment consists of 100 cells, i.e., $C = \{c_1, c_2, \ldots, c_{100}\}$[1]. The set of regions of interest is $\Omega = \{\omega_1, \omega_2, \ldots, \omega_9\}$, where $\omega_1 = \{c_{52}, c_{53}, c_{62}, c_{63}\}$, $\omega_2 = \{c_{28}, c_{38}, c_{48}\}$, and $\omega_3 = \{c_{21}, c_{31}\}$, etc.

The Boolean specification φ is given as follows:

$$\varphi = (\Pi_3 \wedge \Pi_4 \wedge \Pi_5 \wedge \Pi_6 \wedge \Pi_7) \wedge (\Pi'_8 \wedge \Pi'_9) \wedge \neg(\Pi_1 \vee \Pi_2) \tag{5}$$

Sub-specification $\mathcal{M} = \Pi_3 \wedge \Pi_4 \wedge \Pi_5 \wedge \Pi_6 \wedge \Pi_7$ imposes that at least one of cells c_{21} and c_{31} should be visited, at least one of cells c_2 and c_3 should be visited, at least one of cells

[1]Cells are numbered in a left-to-right bottom-to-top manner, e.g. cell c_1 corresponds to the coordinate $(1, 1)$, c_{100} corresponds to the coordinate $(10, 10)$, and so on.



(a) The environment and the task requirements. (b) Motion uncertainties.

Fig. 1. A multi-robot system of Example 1.

c_{15} and c_{16} should be visited, and at least one of cells c_{85}, c_{95}, and c_{10} should be visited along the planned trajectories. Sub-specification $\mathcal{F} = \Pi'_8 \wedge \Pi'_9$ means that at least one robot should end up on either cell c_{78} or cell c_{88}, and one robot should eventually stop at cell c_{34}. Sub-specification $\mathcal{N} = \neg(\Pi_1 \vee \Pi_2)$ indicates that all robots are prohibited from entering any of the cells c_{52}, c_{53}, c_{62}, c_{63}, c_{28}, c_{38}, and c_{48}.

IV. Iterative Reinforcement Learning Solution

A. Construction of MDP

For each sub-specification $\mathcal{M}$, $\mathcal{N}$, $\mathcal{F}$, we define a characteristic vector as follows:

- $V_{\mathcal{M}} = [v_{m_1}, v_{m_2}, \ldots, v_{m_p}] \in \{0,1\}^{1\times p}$, where $v_{m_i} = 1$ if one of propositions P_{m_i} is true, i.e., once any robot r has visited a cell from $c_{m_i} \in \Omega_{m_i}$; otherwise $v_{m_i} = 0$.
- $V_{\mathcal{F}} = [v_{f_1}, v_{f_2}, \ldots, v_{f_q}] \in \{0,1\}^{1\times q}$, where $v_{f_i} = 1$ if sub-specification $\mathcal{M}$ is fulfilled (i.e., $V_{\mathcal{M}} = \vec{1}$) and any robot stays in a Ω_{f_i} at last; otherwise $v_{f_i} = 0$.
- $V_{\mathcal{N}} = [v_n] \in \{0,1\}$, where $v_n = 1$ if each proposition $P_n \subseteq P_t$ is true, i.e., no robot has visited the forbidden regions corresponding to atomic propositions in P_n; otherwise $v_n = 0$.

We denote by (x_i, y_i) the coordinate of cell c_i. The distance between cells c_i and c_j is calculated by *Manhattan distance* as: $D(c_i, c_j) = |x_i - x_j| + |y_i - y_j|$.

The estimated distance matrices $\mathbf{Z}$ and $\mathbf{Z}'$ are computed for designing the reward function, where matrix $\mathbf{Z}$ denotes the shortest distance between each cell and the regions of each sub-specification as shown in Algorithm 1, and matrix $\mathbf{Z}'$ denotes the shortest distance between the regions corresponding to every two sub-specifications that can be computed as follows:

1) For each sub-specification $m_i \subseteq \mathcal{M}$, a matrix $\mathbf{V}_i$ is determined from the row of $\mathbf{Z}$ corresponding to the cell $c_j \in \Omega_{m_i}$: $\mathbf{V}_i(j,\cdot) = \mathbf{Z}(j,\cdot)$, $c_j \in \Omega_{m_i}$.
2) For each column $x = 1, 2, \ldots$, compute the minimum value across all rows of $\mathbf{V}_i$ to form a vector $\mathbf{W}_i(x) = \min_{y=1}^{|m_i|} \mathbf{V}_i(y,x)$, $x = 1, 2, \ldots$.
3) Vectors from step 2 are aggregated to form the matrix $\mathbf{Z}' = \left[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_p\right]^T$.

Then, we construct the global MDP $\mathcal{P}^g = (S^g, s_0^g, S_f^g, A^g, P^g)$. It is defined as follows:

- The state space $S^g = \{s_1^g, s_2^g, \ldots\}$, where each state $s_i^g = [\vec{C_{\mathrm{R}}^g}, V_{\mathcal{M}}^g, V_{\mathcal{F}}^g, V_{\mathcal{N}}^g]$, $\vec{C_{\mathrm{R}}^g} = [c_{r_1}^g, c_{r_2}^g, \ldots, c_{r_k}^g]^{1\times k}$, $c_{r_j}^g \in C$ represents the position of robot r_j, and $V_{\mathcal{M}}^g, V_{\mathcal{F}}^g, V_{\mathcal{N}}^g$ represent the satisfaction of each sub-specification in global MDP $\mathcal{P}^g$.
- The initial state $s_0^g = [\vec{C_{\mathrm{I}}}^{1\times k}, \vec{0}^{1\times p}, \vec{0}^{1\times q}, 1]$.
- The set of final states $S_f^g = \{s_{f_1}^g, s_{f_2}^g, \ldots\} \subseteq S^g$, where $s_{f_i}^g$ represents the states that satisfy the Boolean specification, i.e., $V_{\mathcal{M}}^g = \vec{1}$, $V_{\mathcal{F}}^g = \vec{1}$, and $V_{\mathcal{N}}^g = \vec{1}$.
- The action space $A^g = \{a_1^g, a_2^g, \ldots\}$, where $a_i^g = \langle a_{i1}^g, a_{i2}^g, \ldots, a_{ik}^g \rangle$ is joint actions for all robot, $a_{ij}^g = c_x \in C$ signifies that the robot r_j move to cell c_x on the map. The function $\mathcal{A}^g(s)$ restricts the available actions of the robot moving to adjacent cells, and prohibits to move to a cell that is currently occupied by another robot.
- The transition probability $P^g(s_i^g, a_i^g, s_{i+1}^g) = p^g \in [0,1]$.
- The reward function are defined as shown in Eq (6).

$$R = \begin{cases} -h, & \text{if } V_{\mathcal{N}}^g = \vec{0}, \\ +h, & \text{if } V_{\mathcal{M}}^g = \vec{1}, V_{\mathcal{F}}^g = \vec{1}, \text{and} \quad V_{\mathcal{N}}^g = \vec{1}, \\ -0.1, & \text{otherwise}, \end{cases} \tag{6}$$

when $V_{\mathcal{N}}^g = \vec{0}$, it indicates that the sub-specification $\mathcal{N}$ is violated. Therefore, it results in a reward of $-h$, where h is a positive integer. Reach one of final states, i.e. $V_{\mathcal{M}}^g = \vec{1}$, $V_{\mathcal{F}}^g = \vec{1}$, and $V_{\mathcal{N}}^g = \vec{1}$, will result in a reward of h. In all other cases a -0.1 reward will be obtained.

It is evident that the number of state-action pairs increases exponentially with the number of robots in the global MDP framework. In scenarios with multi-robots, it becomes challenging to find the optimal strategy within a reasonable time frame. Therefore, we propose an iterative MDP framework by constructing an individual MDP for each robot at each stage (referred to as the local MDP $\mathcal{P}^l = (S^l, s_0^l, S_f^l, A^l, P^l)$). The local MDP $\mathcal{P}^l$ is defined as follows:

Algorithm 1: Estimated Distance Matrix Calculation

Input: Sets of regions of interest $\Omega_{\mathcal{M}}$, $\Omega_{\mathcal{N}}$, and $\Omega_{\mathcal{F}}$
Output: Estimated distance matrix $\vec{\mathbf{Z}}$

Initialization: Set task distance matrix $\vec{A} \in \mathbb{R}^{n\times p}$ and final distance matrix $\vec{B} \in \mathbb{R}^{n\times q}$;
for *each cell* $c_i \in C$ **do**
  if *cell* $c_i \in \Omega_{\mathcal{N}}$ **then**
    $\mathbf{Z}(i,\cdot) \leftarrow \infty$;
  else
    for *each subset* Ω_{m_j} *in* $\Omega_{\mathcal{M}}$ **do**
      Initialize set $U_{Temp} \leftarrow \emptyset$;
      for *each cell* $c_k \in \Omega_{m_j}$ **do**
        if $c_i \in \Omega_{m_j}$ **then**
          $U_{Temp} \leftarrow U_{Temp} \cup \{\infty\}$;
        else
          $U_{Temp} \leftarrow U_{Temp} \cup \{D(c_i, c_k)\}$;
        end
      end
      $\vec{A}(i,j) \leftarrow \min(U_{Temp})$;
    end
    for *each subset* Ω_{f_j} *in* $\Omega_{\mathcal{F}}$ **do**
      Initialize set $U_{Temp} \leftarrow \emptyset$;
      for *each cell* $c_l \in \Omega_{f_j}$ **do**
        if $c_i \in \Omega_{f_j}$ **then**
          $U_{Temp} \leftarrow U_{Temp} \cup \{\infty\}$;
        else
          $U_{Temp} \leftarrow U_{Temp} \cup \{D(c_i, c_l)\}$;
        end
      end
      $\vec{B}(i,l) \leftarrow \min(U_{Temp})$;
    end
  end
end
$\mathbf{Z} \leftarrow \begin{bmatrix} \vec{A} & \vec{B} \end{bmatrix}$

- The state space $S^l = \{s_1^l, s_2^l, \dots\}$ where $s_i^l = [c_{r_j}^l, V_{\mathcal{M}}^l, V_{\mathcal{F}}^l, V_{\mathcal{N}}^l]$, $c_{r_j}^l$ denotes the position of robot r_j in local MDP $\mathcal{P}^l$, $V_{\mathcal{M}}^l$, $V_{\mathcal{F}}^l$, $V_{\mathcal{N}}^l$ present the satisfaction of each sub-specification in local MDP $\mathcal{P}^l$.
- The initial state $s_0^l = [c_{r_j}^{l_0}, V_{\mathcal{M}}^{l_0}, V_{\mathcal{F}}^{l_0}, V_{\mathcal{N}}^{l_0}]$, where $c_{r_j}^{l_0} = c_{r_j}^g$ represents the current position of robot r_j in global MDP $\mathcal{P}^g$, $V_{\mathcal{M}}^{l_0} = V_{\mathcal{M}}^g$, $V_{\mathcal{F}}^{l_0} = V_{\mathcal{F}}^g$, and $V_{\mathcal{N}}^{l_0} = V_{\mathcal{N}}^g$ denote the current satisfaction of each sub-specification $\mathcal{M}$, $\mathcal{F}$, and $\mathcal{N}$ in global MDP $\mathcal{P}^g$, respectively.
- The set of final states $S_f^l = \{s_{f_1}^l, s_{f_2}^l, \dots\}$, where $s_{f_i}^l$ denotes the states that meet a sub-specification within $V_{\mathcal{M}}^l$ or $V_{\mathcal{F}}^l$, or those satisfy all sub-specifications corresponding to $V_{\mathcal{M}}^l$ and stay at the cell corresponding to a completed sub-specification within $V_{\mathcal{F}}^l$.
- The action space $A^l = \{a_1^l, a_2^l, \dots, a_n^l\}$ where $a_i^l = c_x \in C$ signifies that robots move to cell c_x on the map. The function $\mathcal{A}^l(s)$ is the same as $\mathcal{A}^g(s)$.
- The transition probability $P^l(s_i^l, a_i^l, s_{i+1}^l) = p^l \in [0, 1]$.
- The reward function can be defined as shown in Eq. (7).

$$R = \begin{cases} -h, & \text{if } V_{\mathcal{N}}^{l_i} = \vec{0}, \\ \frac{1}{1+X}, & \text{if } (V_{\mathcal{M}}^{l_i} - V_{\mathcal{M}}^{l_{i-1}}) \times \vec{1}^T \geq 1, \\ +h, & \text{if } (V_{\mathcal{F}}^{l_i} - V_{\mathcal{F}}^{l_{i-1}}) \times \vec{1}^T \geq 1, \\ 0, & \text{if } a = c_{r_j}^{l_{i-1}}, \\ -0.1, & \text{otherwise,} \end{cases} \tag{7}$$

when $V_{\mathcal{N}}^{l_i} = \vec{0}$, it indicates that the sub-specification $\mathcal{N}$ is violated. It results in a reward of $-h$, where h is a positive integer. Meeting an atomic proposition in $\mathcal{F}$ yields a reward of $+h$. Taking action "stay" results a reward of 0. In all other instances, a reward of -0.1 is given. Upon fulfilling an atomic proposition m_j in $\mathcal{M}$, a reward of $\frac{1}{1+X}$ is given. The parameter X denotes the estimated cost for the current robot r_l to complete this task. It can be computed as Eq. (8).

$$\begin{cases} X = X_1 + X_2, \\ X_1 = \vec{\mathbf{G}} \times (\vec{1} - [V_{\mathcal{M}}^l, V_{\mathcal{F}}^l])^T + \min\limits_{i=1}^{p+q} \vec{\mathbf{H}}(i), \\ \vec{\mathbf{G}}(x) = \min\limits_{y=1}^{p} \mathbf{Z}'(y, x), \quad x = 1, 2, \dots, p+q, \\ \vec{\mathbf{H}}(i) = \mathbf{Z}(c_{r_l}^g, i) \times (L \cdot (1 - [V_{\mathcal{M}}^l, V_{\mathcal{F}}^l](i)) + 1), \\ X_2 = \begin{cases} 0, & \text{if } \Delta_1 < \Delta_2, \\ \infty, & \text{otherwise,} \end{cases} \\ \Delta_1 = \min\limits_{c_y \in \Omega_{m_j}} D(c_{r_l}^g, c_y), \\ \Delta_2 = \max\limits_{c_y \in \Omega_{m_j}} D(c_x, c_y), c_x \in C_{\mathrm{R}}^g \setminus c_{r_l}^g, \end{cases} \tag{8}$$

where L represents a large enough constant.

The parameter X_1 includes the expected distance to complete all uncompleted tasks from an uncompleted task cell and the shortest distance from $c_{r_l}^g$ to all uncompleted task cells in global MDP $\mathcal{P}^g$. The parameter X_2 represents the cost of the robot r_l completing the task m_j, which includes Δ_1 as the minimum distance from $c_{r_l}^g$ to the cells contained in task m_j, and Δ_2 as the maximum of the minimum distances from all other robots to the cells contained in task m_j. If Δ_1 is less than Δ_2, we consider that the current robot can be used to complete this task and assign X_2 a value of 0, which will not cause a change in the estimated cost X. Conversely, if the task should not be completed by the robot r_l, the estimated cost X will be infinite.

The iterative MDP framework is shown in Fig. 2. At each stage, just one robot take action and get its own reward. Therefore, the state and action space that this framework needs to explore will no longer grow exponentially with the increase in the number of robots, but will grow linearly with the increase in stages.

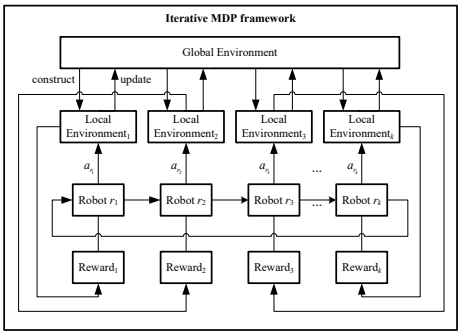


Fig. 2. Iterative MDP framework.

B. Path planning for MRS with Q-learning

After constructing MDPs, we employ the Q-learning algorithm to determine the optimal policy ζ^* of local MDP $\mathcal{P}^l$, as shown in Algorithm 2. Based on Q-learning, the action-value function updates after transiting from s_i to s_{i+1} under an action a_i according to

$$\begin{aligned} Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha[R(s_i, a_i, s_{i+1}) \\ + \gamma \max(Q(s_{i+1}, \cdot)) - Q(s_i, a_i)], \end{aligned} \tag{9}$$

where $\alpha \in [0, 1]$ is the learning rate. To balance the exploration-exploitation issue, we adopt a dynamic ε-greedy action selection strategy:

$$a = \begin{cases} \arg\max\limits_{a \in \mathcal{A}(s)} Q(s, a), & \text{if } \varepsilon < \delta, \\ \text{random choose in } \mathcal{A}(s), & \text{otherwise,} \end{cases} \tag{10}$$

where $\varepsilon \in [0, 1]$ represents the exploration factor and $\delta \in [0, 1]$ is a random number.

The path planning algorithm is shown in Algorithm 3. Initially, the global MDP $\mathcal{P}^g$ is constructed according to Problem 1. Then, the robots from r_1 to r_k are selected cyclically to construct their local MDP $\mathcal{P}^l$, and the optimal policy for the local MDP $\mathcal{P}^l$ is solved by the Algorithm 2. Selected Robot performs a single move and updates the global MDP $\mathcal{P}^g$ based on the current optimal policy. Repeat until a final state in global MDP $\mathcal{P}^g$ is reached, i.e., the Boolean specification φ is satisfied.

V. Case Studies

In this section, all experiments are implemented by Python 3.8 on a computer installed with the Windows 10 operating system with a CPU of AMD Ryzen 7 at 3.2 GHz and 16 GB of RAM.

Algorithm 2: ε-greedy Q-learning Algorithm

Input: Local MDP $\mathcal{P}^l$, learning rate α, discount factor γ, maximum number of episodes E
Output: Optimal policy $\boldsymbol{\zeta}_l^*$

Initialization $\varepsilon_0 = 1$, for each state-action pairs (s, a), set action-value function $Q(s, a) \leftarrow 0$
$\varepsilon = \varepsilon_0$;
for $e := 1, 2, \ldots, E$ **do**
  $s_i = s_0^l$;
  while $s_i \notin S_f^l$ **do**
    choose action a_i by Eq. (10);
    transition to a new state $s_{i+1} = s_i \times a_i$;
    get reward r_{i+1} by Eq. (7);
    update action-value function $Q(s_i, a_i)$ by Eq. (9);
    update state $s_i \leftarrow s_{i+1}$;
  end
  update $\varepsilon = 1 - (\frac{e}{E})^2$;
end
optimal policy $\boldsymbol{\zeta}_l^* = \arg\max_{a \in A(s)} Q^*(s, a)$.

Algorithm 3: Iterative Path Planning Algorithm

Input: The known environment C, the set of robot R, the Boolean specification φ
Output: The set of paths $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$

Construct global MDP $\mathcal{P}^g$;
Initialization current state of the global MDP $\mathcal{P}^g$ $s_i^g = s_0^g$, set of paths $\mathcal{T}_j \cup \{c_{r_j}\}, r_j \in \mathcal{R}$
while $s_i^g \notin S_f^g$ **do**
  for *each robot* $r_j \in R$ **do**
    construct local MDP $\mathcal{P}^l$;
    obtain the optimal policy $\boldsymbol{\zeta}_l^*$ of local MDP $\mathcal{P}^l$ by Algorithm 2;
    The robot r_j moves in the environment according to $\boldsymbol{\zeta}_l^*$;
    update state of the global MDP $\mathcal{P}^g$, $s_i^g = s_i^g \times \boldsymbol{\zeta}_l^*(s_i^g)$;
    update set of paths $\mathcal{T}_j \cup \{c_{r_j}\}$;
  end
end

A. Simulation experiment

In this subsection, we present the simulation results for Example 1. As shown in Fig. 3, the path of each robot is as follows: r_1 : $c_{55}c_{54}c_{44}c_{43}c_{33}c_{32}c_{31}$, r_2 : $c_{56}c_{66}c_{67}c_{57}c_{67}c_{66}c_{65}c_{75}c_{85}c_{86}c_{87}c_{88}$, r_3 : $c_{45}c_{35}c_{25}c_{24}c_{25}$ $c_{15}c_5c_4c_3c_4c_5c_6c_7c_8c_9c_{10}$, r_4 : $c_{46}c_{45}c_{44}c_{34}$.

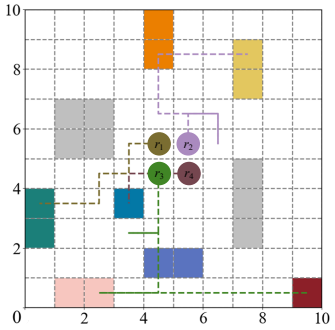
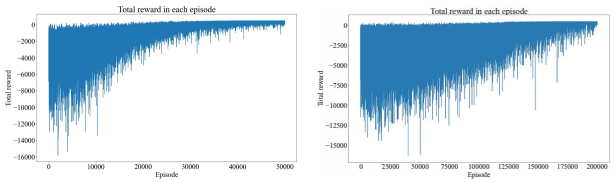


Fig. 3. Simulation result for Example 1.

It can be observed that robot r_1 visited the dark green area (satisfying the sub-specification m_1), robot r_2 visited the orange area and stopped in the yellow area (satisfying sub-specifications m_4 and f_1), robot r_3 visited the pink, purple, and dark red areas (satisfying sub-specifications m_2, m_3 and m_5), and robot r_4 stopped in the blue area (satisfying the sub-specification f_2). Note that all robots avoided the grey areas along the trajectories (satisfying sub-specifications $\mathcal{N}$).

B. Scalability result

In this section, we compare the proposed iterative framework with the global MDP framework in same scenario with a 10×10 grid environment and 10 tasks. As shown in Fig. 4, the global MDP framework requires 50,000 episodes (taking 2355.85s) for the total reward to converge in one robot scenario. When the number of robots reaches two, the total reward still does not converge, even after 200,000 episodes (more than 10,000s). In contrast, the iterative MDP framework converges within 100 episodes, with the total reward record for one such round shown in Fig. 5.



(a) One robot scenario. (b) Two robots scenario.

Fig. 4. Total reward in each episode of global MDP framework.

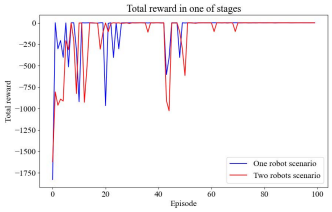


Fig. 5. Total reward in one of stages of iterative MDP framework.

The global MDP framework needs to explore all state-action pairs ($\sum_{i=1}^{|C|^k \cdot 2^{p+d}} |\mathcal{A}^g(s_i^g)|$) in each episode. Consider the maximum number of episodes E, it turns out that a total of $E \cdot \sum_{i=1}^{|C|^k \cdot 2^{p+d}} |\mathcal{A}^g(s_i^g)|$ state-action pairs need to be explored which leads to significant computational cost. The iterative framework requires exploring $\sum_{i=1}^{|C|} |\mathcal{A}^l(s_i^l)|$ state-action pairs in each episode. The number of state-action pairs to be explored is no longer related to the number of robots but rather to the number of stages, totaling $(\sum_{j=1}^{k} |\mathcal{T}_j|) \cdot E \cdot \sum_{i=1}^{|C|} |\mathcal{A}^l(s_i^l)|$. Therefore, our method demonstrates superior scalability when dealing with multi-robot path planning problems.

C. Comparison result

In [16], a simulated annealing algorithm (SAA) based approach is developed for the multi-robot path planning with Boolean specification tasks without considering the motion uncertainties, which is special situation of our considered problem. Thus, we increase the number of cells, the length of sub-specifications (i.e., the complexity of tasks), and the number of robots respectively and compare our approach with the SAA in [16]. Each tested case is randomly generated by using the Robot Motion Toolbox RMTool [18] and run for ten times. For our approach, the design parameters are set as: the learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.9$, and the maximum number of episode $E = 100$. For approach of [16], the parameters are set as: the initial temperature $T_0 = 100$, the final temperature $T_f = 0.1$, the temperature attenuation coefficient $\tau = 0.99$, and we test different number of neighborhood searches $\mathcal{K}$.

TABLE I

SUMMARY STATISTICS FOR DIFFERENT NUMBER OF CELLS n WITH 15 TASKS AND FIVE ROBOTS

Number of cells n	Our approach		Approach developed in [19]									
			$\mathcal{K}=100$		$\mathcal{K}=200$		$\mathcal{K}=300$		$\mathcal{K}=400$		$\mathcal{K}=500$	
	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s
10×10	24.5	2.47	29.1	21.02	27.7	26.32	27.9	32.15	26.3	39.24	26.6	47.2
15×15	25.8	5.61	35.6	59.32	34.5	65.27	34.1	71.39	33.8	88.21	33.8	93.99
20×20	45.8	20.59	67.7	162.21	65.9	170.38	64.2	178.05	63.8	184.98	63.9	192.35
25×25	50.1	54.22	75.2	410.52	73.4	417.52	72.8	424.58	72.6	432.09	72.5	439.85
30×30	63.9	183.21	104.6	861.43	97.8	883.28	96.4	902.58	95.6	925.67	95.7	942.07
35×35	83.7	442.04	109.4	1600.52	103.8	1625.45	101.5	1649.78	99.7	1676.18	99.5	1692.43
40×40	103.7	748.53	138.2	2801.54	135.7	2827.08	128.2	2855.42	125.8	2878.27	125.6	2903.72
45×45	101.5	1029.51	134.4	4114.84	128.5	4148.12	125.2	4175.57	123.4	4207.49	123.8	4235.86
50×50	107.9	2446.58	174.9	6854.79	168.0	6894.85	154.1	6937.40	138.1	6972.25	138.8	7008.56

TABLE II

SUMMARY STATISTICS FOR DIFFERENT LENGTH OF SUB-SPECIFICATION $\mathcal{M}$ WITH 20×20 CELLS AND FIVE ROBOTS

Length of sub-specification $\mathcal{M}$	Our approach		Approach developed in [19]									
			$\mathcal{K}=100$		$\mathcal{K}=200$		$\mathcal{K}=300$		$\mathcal{K}=400$		$\mathcal{K}=500$	
	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s
5	31.4	10.78	32.5	51.11	31.9	59.24	31.6	66.89	31.5	72.30	31.5	78.97
10	38.8	11.79	46.8	129.61	44.6	137.80	44.3	145.38	44.2	152.19	44.3	159.87
15	44.1	15.39	78.5	247.68	74.5	259.44	71.7	260.92	68.9	273.16	68.5	283.81
20	61.7	21.54	101.6	375.89	97.4	388.89	94.6	405.72	92.1	417.65	91.9	441.37
25	57.8	25.71	116.4	561.79	106.8	576.27	97.7	580.73	96.4	595.08	96.3	609.15
30	63.4	32.44	122.9	740.81	116.0	757.49	107.4	773.13	102.1	787.42	101.8	799.82
35	72.0	46.41	142.6	993.63	134.2	1014.23	127.5	1032.57	124.3	1049.87	123.8	1072.02
40	82.9	54.65	176.4	1274.71	165.1	1293.47	157.8	1314.73	151.4	1331.29	150.9	1350.87
45	87.4	59.18	187.2	1585.51	169.7	1597.76	162.6	1616.79	158.7	1637.14	157.4	1659.02
50	92.9	64.34	196.4	1971.39	187.2	1995.41	172.3	2017.81	165.4	2038.26	164.3	2057.15

Comparison in the number of cells: In this experiment, we increase the number of cells from 10×10 to 50×50 while maintaining the number of robots at five and keeping the length of sub-specifications at 15. The experimental results in Table I indicate that the number of cells can have a significant effect on computation time for both methods. However, our approach can provide higher-quality solution (smaller objective function) and lower computation time. Since the computational cost for preprocessing the map in [16] grows significantly as the number of cells increases.

Comparison in the length of task specifications: In this experiment, we increase the length of sub-specification $\mathcal{M}$ (i.e., increase the number of tasks) while maintaining the number of cells at 20×20 cells and keeping the number of robots at five. As shown in Table II, the computation time for both methods is not significantly effected by the increase of length of sub-specification $\mathcal{M}$. However, our method consistently outperforms the method of [16] in solution quality while being more time-efficient.

Comparison in the number of robots: In this experiment, we fix the number of cells as 20×20 and the number of task regions as 50, while increasing the number of robots from 5 to 50. The experiment data in Table III indicate that the increase of the number of robots has relative small impacts for both methods, since increasing the number of robots generally results in a reduced overall movement cost for an MRS. However, the increase in the number of robots leads to a significant increase of solution space for [16] which results in a poorer solution quality than our approach.

TABLE III

SUMMARY STATISTICS FOR DIFFERENT NUMBER OF ROBOTS k WITH 15 TASKS AND 20×20 CELLS

Number of robot k	Our approach		Approach developed in [19]									
			$\mathcal{K}=100$		$\mathcal{K}=200$		$\mathcal{K}=300$		$\mathcal{K}=400$		$\mathcal{K}=500$	
	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s	Obj.	CPU time/s
5	90.2	49.29	198.9	1915.83	187.4	1929.16	184.8	1946.02	181.4	1964.43	180.9	1983.34
10	82.4	47.99	170.6	1949.51	166.4	1968.84	164.3	1989.48	164.0	2000.07	163.8	2017.87
15	70.3	44.60	163.7	1966.59	160.1	1985.18	156.2	2003.97	155.3	2021.72	155.6	2039.63
20	68.8	53.89	161.3	2012.23	150.9	2030.67	147.6	2048.38	145.3	2068.06	144.8	2087.80
25	63.0	51.34	146.9	2049.46	142.8	2070.16	136.4	2089.17	134.7	2106.63	134.2	2124.37
30	60.8	54.46	149.5	2084.18	143.5	2102.67	140.0	2120.34	136.1	2137.98	136.5	2158.39
35	59.2	62.93	150.0	2090.84	144.2	2109.38	142.7	2130.03	141.3	2151.12	141.5	2168.78
40	58.7	64.35	149.5	2147.78	143.2	2166.92	141.0	2187.13	138.7	2204.63	138.2	2223.74
45	57.4	74.33	147.0	2219.42	142.2	2238.71	138.7	2258.04	137.2	2280.27	137.1	2297.48
50	56.7	82.73	154.7	2284.79	149.3	2302.67	146.1	2319.60	143.0	2339.15	139.7	2357.53

VI. CONCLUSION

In this paper, we present a novel iterative MDP approach to the multi-robot path planning problem with Boolean specification tasks under motion uncertainties. Extensive numerical experiments demonstrate that our approach is more scalable than global MDP frameworks for multi-robot systems and outperforms existing heuristics in terms of computation time and solution quality. As a future direction, we aim to propose a method for designing optimal reward functions under the iterative MDP framework.

REFERENCES

[1] I. Kovalenko, D. Tilbury, K. Barton. "The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Engineering Practice*, vol. 86, pp. 105–117, 2019.

[2] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh. "A review on path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[3] P. Lv, Z. Xu, Y, Ji, S. Li, X. Yin. "Optimal supervisory control of discrete event systems for cyclic tasks," *Automatica*, vol. 164, Art. no. 111634, 2024.

[4] C. Li, X. Huang, J. Ding, K. Song, S. Lu. "Global path planning based on a bidirectional alternating search A* algorithm for mobile robots," *Computers & Industrial Engineering*, vol. 168, 2022, Art. no. 108123.

[5] B. Song, Z. Wang, L. Zou. "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve", *Applied Soft Computing*, vol. 100, 2021, Art. no. 106960.

[6] Y. Kantaros, M. Zavlanos. "Sampling-based optimal control synthesis for multirobot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4051–4066, 2018.

[7] P. Yu, V. Dimarogonas. "Distributed motion coordination for multirobot systems under LTL specifications," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1047–1062, 2021.

[8] F. Imeson, S. Smith. "A language for robot path planning in discrete environments: The TSP with Boolean satisfiability constraints," *In Proceedings of the 34th International Conference on Robotics and Automation*, 2014, pp. 5772–5777.

[9] M. Guo, M. Zavlanos. "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4051–4066, 2018.

[10] M. Cai, S. Xiao, Z. Li, Z. Kan. "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 301–316, 2021.

[11] C. Mahulea, M. Kloetzer, J. Lesage. "Multi-robot path planning with boolean specifications and collision avoidance," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 1101–108, 2020.

[12] P. Lv, G. Luo, Z. Ma, S. Li, X. Yin. "Optimal multi-robot path planning for cyclic tasks using Petri nets," *Control Engineering Practice*, vol. 138, Art. no. 105600, 2023.

[13] X. Yu, X. Yin, S. Li, Z. Li. "Security-preserving multi-agent coordination for complex temporal logic tasks," *Control Engineering Practice*, vol. 123, Art. no. 105130, 2022.

[14] Z. He, Y. Dong, G. Ren, C. Gu, Z. Li. "Path planning for automated guided vehicle systems with time constraints using timed Petri nets," *Measurement & Control*, vol. 53, no. 9-10, pp. 2030–2040, 2020.

[15] C. Mahulea, M. Kloetzer. "Robot planning based on boolean specifications using Petri net models," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2218–2225, 2017.

[16] W. Shi, Z. He, W. Tang, W. Liu, Z. Ma. "Path planning of multi-robot systems with boolean specifications based on simulated annealing," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6091–6098, 2022.

[17] W. Shi, Z. He, Z. Ma, N. Ran, X. Yin. "Security-preserving multi-robot path planning for Boolean specification tasks using labeled Petri nets," *IEEE Control Systems Letters*, vol. 7, pp. 2017–2022, 2023.

[18] L. Parrilla, C. Mahulea, M. Kloetzer. "Rmtool: recent enhancements," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5824–5830, 2017.