

# Fast Task Allocation of Heterogeneous Robots With Temporal Logic and Inter-Task Constraints

Lin Li, Ziyang Chen, Hao Wang , and Zhen Kan , *Senior Member, IEEE*

**Abstract**—This work develops a fast task allocation framework for heterogeneous multi-robot systems subject to both temporal logic and inter-task constraints. The considered inter-task constraints include unrelated tasks, compatible tasks, and exclusive tasks. To specify such inter-task relationships, we extend conventional atomic proposition to batch atomic propositions, which gives rise to the LTL<sup>T</sup> formula. The Task Batch Planning Decision Tree (TB-PDT) is then developed, which is a variant of conventional decision tree specialized for temporal logic and inter-task constraints. The TB-PDT is built incrementally to represent the task progress and does not require sophisticated product automaton, which significantly reduces the search space. Based on TB-PDT, the search algorithm, namely Intensive Inter-task Relationship Tree Search (IIRTS), is developed for the fast task allocation of heterogeneous multi-robot systems. It is shown that the solution time of finding a satisfactory task allocation grows almost quadratically with the number of automaton states. Extensive simulation and experiment demonstrate the validity, the effectiveness, and the transferability of IIRTS.

**Index Terms**—Heterogeneous multi-robot systems, linear temporal logic, task allocation.

## I. INTRODUCTION

IMAGINE a medical scenario where a heterogeneous multi-robot system with different capabilities is desired to provide medical services in a hospital environment, as shown in Fig. 1. Example services, such as medicine delivery and navigation tasks, can be specified by temporal logic constraints. In addition to the task-level constraints, there might exist inter-task constraints. For instance, the robots that deliver medicine among patient rooms are not allowed to be reassigned with tasks operating in the therapeutic department due to potential infection risks. Such an inter-task constraint can indicate a class of mutually exclusive tasks. The inter-task constraints can also indicate compatible tasks, e.g., the medicine delivery tasks among patient rooms are compatible and the involved robots can be assigned freely based on the task requirement. However, few existing multi-robot task allocation (MRTA) problems jointly consider both temporal logic and inter-task constraints. In addition, the

Manuscript received 11 February 2023; accepted 21 June 2023. Date of publication 29 June 2023; date of current version 7 July 2023. This letter was recommended for publication by Associate Editor K. N. Kaipa and Editor L. Pallottino upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grants 62173314 and U2013601. (*Corresponding Author: Zhen Kan.*)

The authors are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: linlee1006@mail.ustc.edu.cn; chen ziyang19981220@126.com; whyy@mail.ustc.edu.cn; zkan@ustc.edu.cn).

Digital Object Identifier 10.1109/LRA.2023.3290531

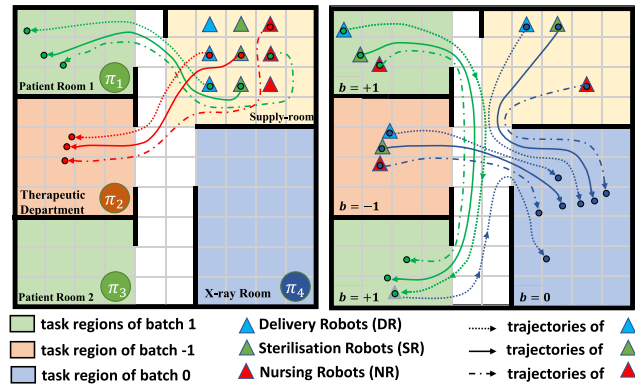


Fig. 1. The task allocation and path planning of a heterogeneous multi-robot system in the medical scenario. See Example 1 for more details.

task is desired to be allocated fast to enable real-time implementation. Hence, this work is particularly motivated to develop a fast task allocation framework for the heterogeneous multi-robot system subject to both temporal logic and inter-task constraints.

MRTA has been widely investigated in agriculture [1], warehousing scenes [2], rescue scenes [3], and medical scenarios [4]. To describe robotic tasks, linear temporal logic (LTL) is often employed [5], [6], [7], [8], [9], [10], [11], [12], owing to its rich expressiveness in specifying complex robotic behaviors. To describe the task allocation and motion planning of multi-robot systems, conventional LTL was extended to cLTL and cLTL+ in [13] and [14], respectively. However, only homogeneous robots were considered. To deal with heterogeneous robots, CaTL, a special STL-based formal language, was developed in [15], [16], which was then extended to CaTL+ in [17] to describe more content-rich tasks. However, the aforementioned approaches do not consider inter-task constraints, such as compatible or exclusive tasks. Motivated by this practical need, we extend conventional atomic propositions to batch atomic propositions to better express heterogeneous multi-robot system tasks. In addition, these works either rely on Integer Linear Program [13], [14] or Mixed Integer Linear Program [15], [16], [17], which are computationally expensive, especially when considering multi-robot systems. Hence, these approaches are hard to be applied in real-time for task allocation and mission planning. Recently, sampling-based methods [10], [11] were developed, which build a tree to approximate the product automaton. However, sampling-based methods assume that the tasks are already allocated to the robots and its mission planning generally suffers from strong randomness and poor stability. In addition,

due to the existence of a large number of useless samples, long exploration time is often needed, making it impossible to solve MRTA problems efficiently. To overcome the limitations of sampling-based and MILP-based methods, a planning decision tree was proposed in our work.

In this work, we develop a novel task allocation framework that can handle heterogeneous multi-robot systems under both temporal logic and inter-task constraints. Specifically, we consider three types of inter-task constraints, i.e., unrelated tasks, compatible tasks, and exclusive tasks. Compatible tasks are referred to the tasks with the same type and the same assignment requirements (e.g., the delivery tasks with the same delivery requirements among patient rooms), while the exclusive tasks restrict the task allocation to the involved robots (e.g., the batch of delivery robots in patient rooms cannot participate in any tasks in the therapeutic department). To specify such inter-task relationships, we extend conventional atomic proposition to batch atomic propositions, which gives rise to the  $LTL^T$  formula. The Task Batch Planning Decision Tree (TB-PDT) is then developed, which is a variant of conventional decision tree specialized for temporal logic and inter-task constraints. Based on TB-PDT, the search algorithm, namely Intensive Inter-task Relationship Tree Search (IIRTS), is developed for fast task allocation that takes into account both temporal logic and inter-task constraints for the heterogeneous multi-robot system. Extensive simulation and experiment demonstrate the validity, the effectiveness, and the transferability of IIRTS.

*Contributions:* First, the developed IIRTS is effective in handling both temporal logic and inter-task constraints. The TB-PDT encodes the automaton states generated by  $LTL^T$  formula and task progression in a hierarchical tree, which is built incrementally during mission operation. Leveraging the tree structure, satisfactory task allocations can be searched effectively. Second, the developed IIRTS can search for a satisfactory planning much faster. By encoding the system and  $LTL^T$  formula as node properties, the TB-PDT does not require sophisticated product automaton, which significantly reduces the search space. It is shown that the computation time of finding a satisfactory task allocation grows almost quadratically with the number of automaton states. In addition, the developed task allocation framework can be conveniently plugged in and used with existing path planning algorithms, such as Probabilistic Roadmaps Methods (PRM) [18], Rapidly exploring Random trees (RRT) [19], RRT\* [20], etc, which indicates that our framework is highly flexible and adaptable for various scenarios.

## II. PRELIMINARIES AND PROBLEM FORMULATION

### A. Linear Temporal Logic

Linear temporal logic is widely used to describe complex tasks. Detailed descriptions of the syntax and semantics of LTL can be found in [21]. An LTL formula can be translated to a nondeterministic Büchi automaton (NBA).

*Definition 1:* The NBA is a tuple  $\mathcal{B} = (S, S_0, \Sigma, \delta, S_F)$ , where  $S$  is a finite set of states;  $S_0 \subseteq S$  is the initial state;  $\Sigma = 2^{\mathcal{AP}}$  is the alphabet, where  $\mathcal{AP}$  is the set of atomic propositions;  $\delta : S \times \Sigma \rightarrow 2^S$  is the transition function;  $S_F \subseteq S$  is the set of accepting states.

The word  $\sigma = \sigma_0\sigma_1\sigma_2\sigma_3\dots$  is an infinite sequence where  $\sigma_i \in \Sigma, \forall i \in \mathbb{Z}_{\geq 0}$ . We denote by  $s_i \xrightarrow{\sigma_i} s_{i+1}$  if  $s_i, s_{i+1} \in S$  and  $s_{i+1} \in \delta(s_i, \sigma_i)$ . The trajectory generated by  $\sigma$  from  $s_0$  is  $s = s_0s_1s_2s_3\dots$ . If the input word  $\sigma$  can generate at least one run  $s$  that intersects  $S_F$  infinitely many times,  $\mathcal{B}$  is said to accept  $\sigma$ . Let  $\mathcal{B}_\varphi$  denote the NBA generated from  $\varphi$  [22]. In general, the accepting run  $s$  can be represented by a prefix-suffix structure, where the prefix stage starts from an initial state and ends with an accepted state and is visited only once. The suffix stage is a cyclic path of accepted states, which is visited infinitely many times.

### B. System and Task Requirements

Consider a fleet of  $N$  heterogeneous robots  $A = \{a_i\}_{i=1}^N$  consisting of  $N_c$  different categories to capture the different capabilities of robots. Each robot is defined as a tuple  $a = \{p_0, p, c\} \in A$ , where  $p_0$  is the initial position,  $p$  is the current position, and  $c \in [N_c]$  is the robot category, where  $[N_c]$  is a shorthand of  $\{1, 2, \dots, N_c\}$ . The operating environment is given as a tuple  $Env = \{M, Q, \mathcal{AP}, \mathcal{L}, \mathcal{L}_m\}$ , where  $M$  is the continuous map composed of  $l$  non-overlapped task regions  $M_i, i \in [l]$ , and the non-task region  $M'$  (i.e., the region that robots cannot traverse),  $Q$  is a finite set of task states in the environment,  $\mathcal{L} : Q \rightarrow \mathcal{AP}$  maps the task state to an atomic proposition, i.e.,  $\mathcal{L}(q) = \pi_i$  with  $q \in Q$  and  $\pi_i \in \mathcal{AP}$  and  $\mathcal{L}_m : Q \rightarrow M$  maps the task state to a task region in the map. In this work, we assume  $\Sigma = \mathcal{AP}$  and each task region has only one label in  $\mathcal{AP}$  and no multiple regions share the same label. The task allocation is denoted as  $V = [v_1, v_2, v_3, \dots, v_N]$ , where  $v_i = 1$  indicates the robot  $a_i$  is participating in the task and  $v_i = 0$  otherwise.

*Definition 2:* (Batch atomic propositions). The batch atomic proposition is a tuple  $\pi^T = (\pi, \{(c_i, n_i)\}_{i \in [N_c]}, \pm b)$ , where  $\pi \in \mathcal{AP}$ , the pair  $(c_i, n_i)$  indicates  $n_i$  robots with category  $c_i$  are required for the task,  $b \in \mathbb{Z}$  is called the task batch that indicates the inter-task constraints, i.e., two tasks are compatible if both are  $+b$ , exclusive if one is  $+b$  and the other is  $-b$ , or unrelated if  $b = 0$  or the magnitudes of the two task batches are different.

In Definition 2,  $\pi^T$  is true if the region labeled as  $\pi$  contains at least  $n_i$  robots of category  $c_i$  and these robots satisfy the batch requirement (i.e., inter-task constraints). The negation of a batch atomic proposition  $\neg\pi^T = (\neg\pi, \{(c_i, n_i)\}_{i \in [N_c]}, \pm b)$  means  $\pi^T$  is false, e.g., less than  $n_i$  robots of type  $c_i$  are in the region  $\mathcal{L}_m(\pi)$  or the robots in this region violate the inter-task constraints. In this work, an LTL formula  $\varphi^T$  defined based on  $\pi^T$  is called an  $LTL^T$  formula, which is developed to describe the complex inter-task relationships for heterogeneous multi-robot systems. We define a plan  $\Pi = (s, q, V)$ , where  $s = s_0s_1\dots$ ,  $q = q_0q_1\dots$  and  $V = V_0V_1\dots$  are the sequence of automaton states, task states, and task allocations, respectively. If  $\mathcal{L}(q) \models \varphi^T$  and  $V$  satisfies the inter-task constraints of  $\varphi^T$ , it is said that the plan  $\Pi$  satisfies  $\varphi^T$ .

### C. Problem Formulation

To elaborate the considered MRTA with temporal logic and inter-task constraints, the following example will be used as a running example throughout this work.

*Example 1:* Consider a medical scenario as shown in Fig. 1. The collaborative task is  $\varphi = \square\Diamond\pi_1 \wedge \square\Diamond\pi_2 \wedge \square\Diamond\pi_3 \wedge \square\Diamond\pi_4$ ,

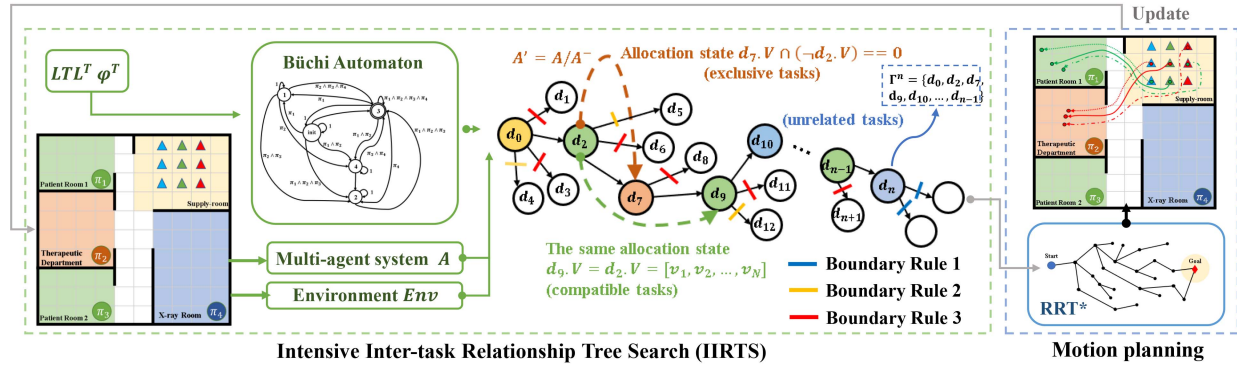


Fig. 2. The architecture of IIRTS. The TB-PDT is constructed based on  $\mathcal{B}_\varphi$  generated by  $\varphi^T$ , the environment states, and the multi-robot system  $A$ . The IIRTS then searches for the satisfactory task allocation over the TB-PDT, which is then combined with RRT\* to realize motion planning in  $Env$ .

TABLE I  
LISTS OF TASKS IN A MEDICAL SCENE

stamp	Tasks
1	One delivery robot transports medicines from the supply-room to the patient room 1. One sterilization robot and one nursing robot go to the patient room 1.
2	One delivery robot, one sterilization robot, and one nursing robot go to the therapeutic department. The robots that entered the patient rooms cannot enter the therapeutic department.
3	One delivery robot, one sterilization robot, and one nursing robot go to the patient room 2.
4	Three delivery robots, two sterilization robots, and two nursing robots go to the X-ray room.

where  $\pi_1, \pi_2, \pi_3, \pi_4$  are explained in Table I. Since  $\varphi$  contains exclusive tasks (i.e., the robots that have been to the patient rooms are not allowed to go to the therapeutic department), the batch value  $+b$  and  $-b$  are used to indicate such exclusive tasks. Since  $\pi_1$  and  $\pi_3$  are the same type of task and require the same type and number of robots, they are considered as compatible tasks and thus the same batch value  $b$  is used. The task  $\varphi$  can be rewritten in an  $LTL^T$  formula as

$$\begin{aligned} \varphi^T = & \square\Diamond(\pi_1, \{(DR, 1), (SR, 1), (NR, 1)\}, +1) \wedge \\ & \square\Diamond(\pi_2, \{(DR, 1), (SR, 1), (NR, 1)\}, -1) \wedge \\ & \square\Diamond(\pi_3, \{(DR, 1), (SR, 1), (NR, 1)\}, +1) \wedge \\ & \square\Diamond(\pi_4, \{(DR, 3), (SR, 2), (NR, 2)\}, 0), \end{aligned} \quad (1)$$

which can be abbreviated as  $\varphi^T = \square\Diamond\pi_1^{T_1} \wedge \square\Diamond\pi_2^{T_2} \wedge \square\Diamond\pi_3^{T_3} \wedge \square\Diamond\pi_4^{T_4}$ .

Based on the prefix-suffix structure mentioned above, the plan  $\Pi$  can be written as  $\Pi = \Pi_{pre}\Pi_{tra}\Pi_{suf}\Pi_{suf}\dots$ , where  $\Pi_{pre}$  is the finite prefix stage,  $\Pi_{tra}$  is the finite transition stage, and  $\Pi_{suf}\Pi_{suf}\dots$  is the infinite cycle stage. If the plan  $\Pi = \Pi_{pre}\Pi_{tra}\Pi_{suf}\Pi_{suf}\dots$  satisfies the  $LTL^T$  formula  $\phi^T$ , the  $\Pi^f = \Pi_{pre}\Pi_{tra}\Pi_{suf}$  satisfies the  $\phi^T$ . Formally, we first

introduce a mild Assumption 1, which is widely employed in the literature, and then Problem 1 is considered.

*Assumption 1:* Assuming that there are sufficient robots and exists at least one plan  $\Pi^f$  satisfying the  $LTL^T$  formula  $\varphi^T$ .

*Problem 1:* Given an  $LTL^T$  task formula  $\varphi^T$ , the heterogeneous multi-robot system  $A$ , and the environment  $Env$ , the goal is to find a finite plan  $\Pi^f$  that satisfies  $\varphi^T$ .

Since there might exist many satisfactory finite plans, we are also interested in plans that can complete  $\varphi^T$  fast.

### III. TASK ALLOCATION

To address Problem 1, we first present the developed Task Batch Planning Decision Tree (TB-PDT), based on which the Intensive Inter-task Relationship Tree Search (IIRTS) algorithm is developed for fast task allocation. The approach overview is shown in Fig. 2.

#### A. Task Batch Planning Decision Tree

To address MRTA with inter-task constraints, the Task Batch Planning Decision Tree (TB-PDT) is presented in this section, which encodes the automaton states generated by  $\varphi$  and the system states in a hierarchical tree structure. By growing the tree from the root node to leaf nodes, TB-PDT can be searched for satisfactory task allocations.

*Definition 3:* A TB-PDT  $\mathcal{T}_B$  is constructed based on a set of nodes  $\{d_i\}$ ,  $i \in \mathbb{Z}_{\geq 0}$ , where  $d_0$  represents the root of the tree. Each node in  $\mathcal{T}_B$  is defined as a tuple  $d_i = (s, q, V, Par, T, Prog, bat)$ , where

- $s \in S$  denotes the automaton state,
- $q \in Q$  denotes the task state,
- $V = [v_1, v_2, \dots, v_N]$  denotes the allocation corresponding to the node task  $\mathcal{L}(d_i.q)^T$ ,
- $Par$  is the parent node of  $d_i$ ,
- $T = \{t_1, t_2, \dots, t_N\}$  is the set of estimated times that each robot completes the current node task,
- $Prog = \{pre, tra, s, oth\}$  indicates the mission stages, where  $pre, tra, oth$  represent the prefix stage, the transition stage and other cases respectively, and  $s \in S$  is the start and end accepting state of the cyclic path in the first suffix stage,
- $bat \in \mathbb{Z}$  indicates the task batch, which reflects the inter-task constraints of the node  $d_i$ .

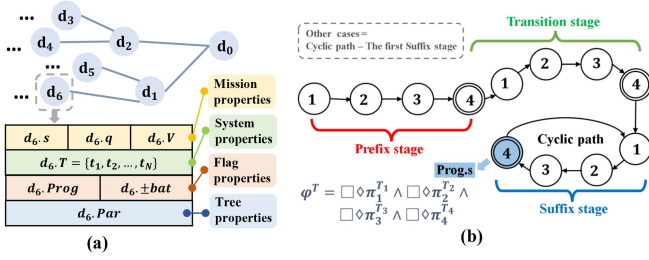


Fig. 3. (a) The node structure in TB-PDT. (b) The mission stages of  $\varphi^T$ .

As shown in Fig. 3(a), each node contains four properties. The mission properties (i.e.  $s, q, V$ ) indicate the mission states of the node. The system property  $T$  indicates the estimated task completion times of all robots. The tree property  $Par$  indicates the relationship between the parent node and the child nodes. The flag property  $Prog$  indicates the current mission stage, as shown in Fig. 3(b). As for the batch  $bat$ , we use the same  $b \in \mathbb{Z}_{>0}$  to indicate compatible tasks (i.e., the task allocation can be the same), and use opposite batch values (e.g.,  $+b$  and  $-b$ ) to indicate exclusive tasks (i.e., the robots in the complement of the set are chosen for task assignment). If batch values are different in magnitude or equal to 0, such tasks are considered as unrelated. Throughout this work, we will adopt the data structure to represent the node properties, e.g.,  $d_i.s$  represents the automaton states of node  $d_i$ .

### B. Intensive Inter-Task Relationship Tree Search

Based on the developed TB-PDT  $\mathcal{T}_B$ , this section presents the search algorithm IIRTS for fast task allocation that considers inter-task constraints. The pseudo-code of IIRTS is outlined in Algorithm 1. The tree  $\mathcal{T}_B$  grows by taking into account the  $LTL^T$  formula  $\varphi^T$ , the environment  $Env$ , and the multi-robot system  $A$ . The IIRTS explores over  $\mathcal{T}_B$  to search for satisfactory allocations and ends if all nodes have been traversed (lines 2–3). If not, nodes are continuously traversed to obtain branches that satisfy the task requirements. Let  $\Gamma_i$  denote the path from  $d_0$  to  $d_i$  that excludes  $d_i$  and the set of automaton states associated with  $\Gamma_i$  is denoted by  $\{\Gamma_i.s\}$ . After initializing the sub-tree  $\mathcal{T}_B^{sub}$  (line 5), we find all possible task states  $q$ , which satisfy  $s \in \delta(d_i.s, \mathcal{L}(q))$  in the set of untraversed automaton states  $s \in S \setminus \{\Gamma_i.s\}$ .<sup>1</sup> Then we set the task state, the automaton state, and the node path of sub-node  $d_i^{sub}$  as  $q, s$ , and  $\Gamma_i$ , respectively. The task batch  $d_i^{sub}.bat$  is initialized to the batch  $\pm b$  in the  $LTL^T$  formula. If the task corresponding to the current sub-node is unrelated to the tasks in the sub-tree  $\mathcal{T}_B^{sub}$  (i.e.,  $|d_i^{sub}.bat| \notin \{\mathcal{T}_B^{sub}.bat\}$  or  $d_i^{sub}.bat = 0$ ), the unrelated-task search (US) (Algorithm 2) is applied (line 10). If the task corresponding to the current sub-node is interleaved with the tasks in  $\mathcal{T}_B^{sub}$  (i.e.,  $d_i^{sub}.bat \in \{\mathcal{T}_B^{sub}.bat\}$ ), we need to determine whether they are compatible ( $d_i^{sub}.bat > 0$ ) or exclusive ( $d_i^{sub}.bat < 0$ ). The sub-node  $d_i^{sub}$  is then generated by compatible-task search (CS) (Algorithm 3) or exclusive-task search (ES) (Algorithm 4). If the mission stage of  $d_i^{sub}$  is the same as  $d_i$ , which means the current

<sup>1</sup>Given two sets  $A$  and  $B$ , let  $A \setminus B = \{x|x \in A, x \notin B\}$  denote the set difference.

### Algorithm 1: Intensive Inter-Task Relationship Tree Search.

**Input:**  $LTL^T$  formula  $\varphi^T$ , Environment  $Env$ , the multi-robot system  $A$   
**Output:**  $s^*, q^*, V^*$   
1: Convert  $\varphi^T$  to  $\mathcal{B}_\phi$  and initialize  $\mathcal{T}_B = \{d_0\}$ ;  
2: **if** all nodes are traversed **then break**  
3: **end if**  
4: **for**  $d_i \in \mathcal{T}_B$  that has not been traversed **do**  
5: Initialize the sub-tree  $\mathcal{T}_B^{sub} = \emptyset$ ;  
6: **for**  $s \in S \setminus \{\Gamma_i.s\}$  **do**  
7: **for**  $q$  such that  $s \in \delta(d_i.s, \mathcal{L}(q))$  **do**  
8: Initialize the sub-node  $d_i^{sub}$ ;  
9: **if**  $|d_i^{sub}.bat| \notin \{\mathcal{T}_B^{sub}.bat\}$  **or**  $d_i^{sub}.bat = 0$  **then**  
10:  $d_i^{sub} \leftarrow \text{US}(q, d_i)$ ;  
11: **else**  
12: **if**  $d_i^{sub}.bat > 0$  **then**  
13:  $d_i^{sub} \leftarrow \text{CS}(q, d_i, \mathcal{T}_B^{sub}, \mathcal{T}_B)$ ;  
14: **end if**  
15: **if**  $d_i^{sub}.bat < 0$  **then**  
16:  $d_i^{sub} \leftarrow \text{ES}(q, d_i, \mathcal{T}_B^{sub}, \mathcal{T}_B)$ ;  
17: **end if**  
18: **end if**  
19: **if**  $d_i^{sub}.Prog = d_i.Prog$  **then**  
20: Add  $s$  into the  $\{\Gamma_i.s\}$ ;  
21: **else**  
22:  $\{\Gamma_i.s\} \leftarrow \emptyset$ ;  
23: **end if**  
24: Add  $d_i^{sub}$  to  $\Gamma_i$  and  $\mathcal{T}_B^{sub}$ ;  
25: **end for**  
26: **end for**  
27: **for**  $d_i^{sub} \in \mathcal{T}_B^{sub}$  **do**  
28: Prune sub-trees according to **Boundary Rules**;  
29: **end for**  
30: Add  $\mathcal{T}_B^{sub}$  to  $\mathcal{T}_B$  and add  $d_i$  to  $\mathcal{T}_{tra}$ ;  
31: **end for**  
32: Select  $d^* = \arg \min_{d_i \in \mathcal{T}_B} J$  with  $d^*.Prog = oth$   
33: **Return**  $s^*, q^*, V^* \leftarrow \mathcal{T}_B^*.s, \mathcal{T}_B^*.q, \mathcal{T}_B^*.V$

mission stage has not yet ended, the exploration continues and the current automaton state  $s$  is added to  $\{\Gamma_i.s\}$  (lines 19–20). If not, i.e., the current mission stage ends, we add sub-node  $d_i^{sub}$  to  $\Gamma_i$  and the sub-tree  $\mathcal{T}_B^{sub}$ , and start a new round of traversal (lines 21–24). Since  $\mathcal{T}_B^{sub}$  contains all possible sub-nodes, which may lead to dimensional explosion, the following boundary rules are developed to prune redundant nodes in  $\mathcal{T}_B^{sub}$  (Line 28) to enable fast search of satisfactory task allocations.

*Definition 4:* The boundary rules are developed as follows:

- 1) For any node  $d$ , if its mission stage is  $d.Prog = oth$ , node  $d$  will no longer be traversed and will be added to the traversed set  $\mathcal{T}_{tra}$ ;
- 2) For any node  $d$ , the traversed automaton state is not be sampled if its mission stage remains the same;
- 3) For any nodes with the same automaton state and the same mission stage, i.e.  $d_1.s = d_2.s, d_1.Prog = d_2.Prog$ , the node with larger cost will no longer be traversed and will be added to the traversed set  $\mathcal{T}_{tra}$ .

Boundary rules can accelerate the search by limiting the tree expansion. To evaluate the node performance, the longest estimated time to complete the task among all robots is treated as the cost in this work, i.e.,  $J = \max\{d_i.T\}$ , where the estimated time is the lower bound for robots to complete the corresponding task. To find an allocation that can complete the LTL<sup>T</sup> formula fast, the optimal node  $d^*$  with the minimum cost  $J$  and the mission stage *oth* is then selected (line 32). By backtracking according to  $d^*.Par$  until the root node  $d_0$ , the sequences of Büchi automaton states, trajectories, and task allocation from the root  $d_0$  to  $d^*$  can be determined. The TB-PDT from  $d_0$  to  $d^*$  is called the optimal tree  $\mathcal{T}_B^*$ .

### C. Inter-Task Relationship Search

As the main components of IIRTS in Algorithm 1, the unrelated-task search, compatible-task search, and exclusive-task search are explained in detail subsequently.

1) *Unrelated-Task Search*: As outlined in Algorithm 2, given a node  $d_i$ , the estimated time stamp that a robot  $a_n$ ,  $n \in 1, \dots, N$ , reaches the target position  $p_{pos_i}$  corresponding to  $d_i.\mathcal{L}_m(q)$  is given as  $t_n^i = t_n^{i-1} + \frac{1}{v_a} \|p_{a_n}^{i-1} - p_{pos_i}\|_2$ , where  $t_n^{i-1}$  is the estimated time of completing  $\Gamma_i$  by the robot  $a_n$ ,  $p_{a_n}^{i-1}$  is the expected position of the robot  $a_n$  after completing  $\Gamma_i$ , and  $v_a$  is the maximal linear velocity of the robot. Then we sort  $d_i.T = \{t_1^i, t_2^i, \dots, t_N^i\}$  in the ascending order (line 2). Depending on the robot types and their numbers required by  $\varphi^T$ , the robots with the shortest completion time are selected from the sorted  $T_i$  (line 3). If  $d_i^{sub}.V(n) = 1$ , it means robot  $a_n$  is selected to complete the corresponding task of  $d_i^{sub}$ . Thus, we update the estimated time stamp  $t_n^i$  and the expected position  $p_{a_n}^i$  of the assigned robot  $a_n$  (lines 4–6). To enable the selected robots wait until all selected robots reach their target positions, the estimated time stamps of all selected robots are the maximum of the time stamps of all selected robots. The robots that are not selected remain the same estimated time as the previous node. Thus, the time stamp of sub-node is updated to the maximum of the arrival times of all robots at their respective target positions (line 8). The mission stage corresponding to the sub-node is determined by

$$\text{PROG}(Prog, s) = \begin{cases} Prog, & \text{if } s \notin \mathcal{F} \\ tra & \text{if } s \in \mathcal{F}, Prog = pre \\ s & \text{if } s \in \mathcal{F}, Prog = tra \\ oth & \text{if } Prog = s \end{cases} \quad (2)$$

where *pre* indicates that the automaton state is from the initial state to the accepting state, *tra* indicates that the automaton state is from the accepting state at the end of prefix stage to the next accepting state, *oth* indicates other cases after completion of the first suffix cycle stage, and  $s \in \mathcal{S}$  is the start and end accepting state of the first suffix stage. The illustration of mission stages is shown in Fig. 3(b).

2) *Compatible-Task Search*: Since robots can be re-assigned with the same allocation for compatible tasks, the computation time used for reallocation can be reduced. The compatible-task search in Algorithm 3 takes advantage of this property to enable efficient robots' task allocation. For each robot  $a_n$ ,  $n \in 1, \dots, N$ , we first initialize the estimated completion time

---

#### Algorithm 2: Unrelated-Task Search (US).

---

**Input:**  $q, d_i, \varphi^T$

**Output:**  $d_i^{sub}$

- 1: Update  $t_n^i = t_n^{i-1} + \frac{1}{v_a} \|p_{a_n}^{i-1} - p_{pos_i}\|_2 \in T^i$  for each robot  $a_n$ , where  $p_{pos_i} = d_i.\mathcal{L}_m(q)$ ;
  - 2: Sort  $d_i.T$ ;
  - 3: Set  $d_i^{sub}.V$  according to the sorted  $T^i$  based on  $d_i.V$  and  $\varphi^T$ ;
  - 4: **if**  $d_i^{sub}.V(n) = 1$  **then**
  - 5:  $t_n^i \leftarrow \max\{t_n^i | n \text{ s.t. } d_i^{sub}.V(n) = 1\}$ ;
  - 6: Set  $p_{a_n}^i \leftarrow p_{pos_i}$ ;
  - 7: **end if**
  - 8: Set  $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$ ;
  - 9: Set  $d_i^{sub}.Prog = \text{PROG}(d_i.Prog, s)$ ;
  - 10: **Return**  $d_i^{sub}$
- 

---

#### Algorithm 3: Compatible-Task Search (CS).

---

**Input:**  $q, d_i, \mathcal{T}_B^{sub}, \mathcal{T}_B$

**Output:**  $d_i^{sub}$

- 1: Initialize  $t_n^{i-1}, p_{a_n}^{i-1}$ ;
  - 2: Determine  $t_n^i = t_n^{i-1} + \frac{1}{v_a} \|p_{a_n}^{i-1} - p_{pos_i}\|_2$ , where  $p_{pos_i} = d_i.\mathcal{L}_m(q)$ ;
  - 3:  $d_i^{sub}.V = d_k.V$  if  $\exists d_k \in \mathcal{T}_B$  s.t.  $d_i^{sub}.bat = d_k.bat$ ;
  - 4: **if**  $d_i^{sub}.V(n) = 1$  **then**
  - 5:  $t_n^i \leftarrow \max\{t_n^i | n \text{ s.t. } d_i^{sub}.V(n) = 1\}$ ;
  - 6:  $p_{a_n}^i \leftarrow p_{pos_i}$ ;
  - 7: **end if**
  - 8:  $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$ ;
  - 9:  $d_i^{sub}.Prog = \text{PROG}(d_i.Prog, s)$ ;
  - 10: **Return**  $d_i^{sub}$
- 

$t_n^{i-1}$  and the expected position  $p_{a_n}^{i-1}$  after completing  $\Gamma_i$  (line 1). The estimated time to reach the target position  $p_{pos_i}$  for each robot  $a_n$  is obtained via  $t_n^i = t_n^{i-1} + \frac{1}{v_a} \|p_{a_n}^{i-1} - p_{pos_i}\|_2$  (line 2). If there is a compatible task in the tree  $\mathcal{T}_B$  that corresponds to the current child node task, the allocation  $V$  for such a compatible task is assigned to the current sub-node  $d_i^{sub}$  (line 3). The corresponding position and completion time are updated to the target position and the estimated time of the last selected robot to arrive (lines 4–7). The mission stage corresponding to the sub-node is updated by (2) (line 9).

3) *Exclusive-Task Search*: As indicated in Algorithm 4, we first search for the node  $d_k$  in the tree  $\mathcal{T}_B$  that has the opposite batch value to the current node  $d_i$ , i.e.,  $d_k.bat = -d_i.bat$ . If exists, a new multi-robot system  $A_{new} = A \setminus A^-$  is generated, where  $A^-$  is the set of robots specified to participate in  $d_k$  (lines 2–4). After computing the estimated completion times of each robot in  $A_{new}$  for the current task, we sort  $d_i.T$  with the corresponding robots and select robots that take the shortest time according to the requirements of  $\varphi^T$  (line 7). Following the similar procedure, the estimated completion time and the expected position of the assigned robot are updated (lines 8–11). The estimated completion time of the current sub-node is determined by  $\max\{t_n^i, t_n^{i-1}\}$ . The mission stage is updated by function PROG accordingly. Note that the new multi-robot

**Algorithm 4:** Exclusive-task Search (ES).

---

**Input:**  $d_i, \mathcal{T}_B^{sub}, \mathcal{T}_B$   
**Output:**  $d_i^{sub}$

- 1: **if** there exists  $d_k \in \mathcal{T}_B$  s.t.  $d_k.bat = -d_i.bat$  **then**
- 2:   **if**  $d_k.V(n) = 1$  **then**
- 3:     Add  $a_n$  into  $A^-$ ;
- 4:   **end if**
- 5:   Generate the new multi-robot system  $A_{new} = A \setminus A^-$ ;
- 6:   Determine  $t_n^i = t_n^{i-1} + \frac{1}{v_a} \|p_{a_n}^{i-1} - p_{pos_i}\|_2$ ,  
 $a_n \in A_{new}, p_{pos_i} = d_i.\mathcal{L}_m(q)$ ;
- 7:   Set  $d_i^{sub}.V$  according to the sorted  $d_i.T$  based on  $d_i.V$   
and  $\varphi^T$ ;
- 8:   **if**  $d_i^{sub}.V(n) = 1$  **then**
- 9:      $t_n^i \leftarrow \max\{t_n^i | n \text{ s.t. } d_i^{sub}.V(n) = 1\}$ ;
- 10:     $p_{a_n}^i \leftarrow p_{pos_i}$ ;
- 11:   **end if**
- 12:    $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$ ;
- 13:    $d_i^{sub}.Prog = \text{PROG}(d_i.Prog, s)$ ;
- 14:   **end if**
- 15: **Return**  $d_i^{sub}$

---

system  $A_{new}$  is only for the current sub-node  $d_i^{sub}$  and does not affect the whole tree.

*Remark 1:* In this work, we focus on the MRTA with inter-task constraints. It is worth pointing out that other languages such as cLTL, STL, and CaTL, can also be leveraged by formulating the inter-task constraints as a MILP problem. However, solving MILP can be computationally expensive, especially for complex task requirements and large-scale robots. Instead, our approach can address MRTA much efficiently and effectively.

#### D. Algorithm Analysis

*Theorem 1:* Given a multi-robot system  $A$  operating in the environment  $Env$  with  $LTL^T$  formula  $\varphi^T$ , the IIRTS in Algorithm 1 is ensured to be valid (i.e., the obtained task allocation satisfies  $\varphi^T$ ) and complete (i.e., a feasible task allocation, if exists, is guaranteed to be found).

*Sketch of Proof:* As discussed in Section III-C, the developed unrelated-task search, compatible-task search, and exclusive-task search of IIRTS all guarantee that the allocations of the explored sub-nodes satisfy the  $LTL^T$  formula  $\varphi^T$ , which indicates that IIRTS is valid.

To show IIRTS is complete, we prove that IIRTS can guarantee the completeness of both task allocation and mission planning. By the design of IIRTS, it is guaranteed that the task allocation satisfies the task requirements as long as there are sufficient numbers of robots, thus IIRTS is complete for the task allocation. To prove IIRTS guarantees the completeness of the mission planning, we first consider the case without using the boundary rules. Suppose there exists a node  $d_k$  that satisfies the  $LTL^T$  formula  $\varphi^T$ . By tracing back from  $d_k$  to the root, the corresponding output sequences are  $s = s_0 s_1 \dots s_n$ ,  $V = V_0 V_1 \dots V_n$ ,  $q = q_0 q_1 \dots q_n$ , and  $\pi^T = \pi_0^T \pi_1^T \dots \pi_n^T = \mathcal{L}(q)$ , which satisfy,  $\forall \pi_i^T \in \pi^T, \pi_i^T \in \delta(s_{i-1}, s_i), i \in \mathbb{Z}_{>0}$ . That is, the output  $(s, q, V)$  is a satisfactory planning for  $\varphi^T$ . To see that  $(s, q, V)$ , if exists, is ensured to be found, we start from the root of the tree.

By the definition of TB-PDT, the root  $d_0 \in \mathcal{T}_B$  has  $d_0.s = s_0 \in s$  and  $L(d_0.q) = \pi_0^T \in \pi^T$ . Without boundary rules, all nodes will be traversed by IIRTS. Hence, there always exists a node  $d_m \in \mathcal{T}_B$  such that  $d_m.s = s_{n_1} \in s$  and  $L(d_m.q) = \pi_{n_1}^T \in \pi^T$ , ( $n_1 < n$ ). Thus, for  $\forall (s_k, q_k, V_k) \in (s, q, V)$ , there always exists a node  $d'$  satisfying  $d'.s = s_k, d'.q = q_k, d'.V = V_k$ , which indicates this satisfying planning can be found by IIRTS.

When considering boundary rules, suppose there exists a point  $d_n$  satisfying  $d_m.Prog = d_n.Prog$  and  $d_m.s = d_n.s$ , but the cost of  $d_n$  is larger than  $d_m$ . Thus  $d_n.s \in s$  and the node  $d_n$  will be removed according to boundary rules. As the mission stages and automaton states of both  $d_m$  and  $d_n$  are the same, the sequences of  $s$  and  $q$  after pruning off  $d_n$  do not affect the completion of the whole task. In addition, boundary rules ensure that no duplicated automaton states exist in the same mission stage, eliminating the dimensional explosion. Given that IIRTS guarantees the validity of the nodes' task allocation, the pruned sequences can still satisfy  $\varphi^T$ . Since the boundary rules only reduce redundant nodes and traversals to reduce the search space, while the task allocation and mission planning do not affect each other, when integrating the boundary rules, the planning that satisfies  $\varphi^T$ , if exists, is ensured to be found by IIRTS. ■

Due to the boundary rules 1) and 2), the node will no longer be traversed if its mission stage is *oth* and there are no duplicate automaton states for the same mission stage. Therefore, the length of  $\mathcal{T}_B$  is at most  $3 \times |S|$ , where  $|\cdot|$  is the cardinality of a set. The width of  $\mathcal{T}_B$  (i.e., the number of expandable nodes) is at most  $(2 + |S_F|) \times |S|$  due to the boundary rule 3). As a parent node generates no more than  $|S| \times |Q|$  sub-nodes, at most  $((2 + |S_F|) \times |S|) \times (3 \times |S|) \times (|S| \times |Q|)$  nodes will be explored. Since  $|S_F|$  is generally much smaller than  $|S|$  and the number of sub-nodes is in general smaller than  $|S| \times |Q|$ , the time complexity of IIRTS can be approximately as  $\mathcal{O}(|S|^2)$ , i.e., the computation time is approximately quadratic to  $|S|$ .

## IV. EXPERIMENT

In this section, the IIRTS is evaluated to examine: **1) the effectiveness:** whether the generated task allocation satisfies the task requirements; **2) the efficiency:** whether the task allocation can be generated in a short time; and **3) the transferability:** whether IIRTS can be combined with various other path planning algorithms. MATLAB R2019b and gazebo are used on a computer with Intel Core i5-1050ti.

### A. Numerical Experiments

The performance of IIRTS is evaluated under different numbers of atomic propositions, automaton states, and robots.

1) *Numerical Experiment 1:* We first consider 45 robots with 3 categories (i.e., 15 robots in each category) to perform different  $LTL^T$  tasks. There are four cases: 1) without related tasks; 2) with compatible tasks, which require compatible-task search; 3) with exclusive tasks, which require exclusive-task search; and 4) with both compatible and exclusive tasks, which require both compatible-task and exclusive-task search. For each case, we consider a set of  $LTL^T$  formulas such that the number of automaton states ranges from 4 to 64 as shown in Fig. 4. To further show the performance with more automaton states, we

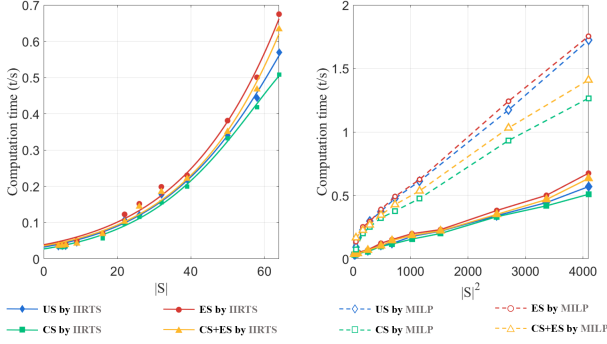


Fig. 4. The left plot shows the computation times of the IIRTS with the number of automaton states from 4 to 64 for four cases. The right plot shows the computation times of IIRTS and MILP.

TABLE II  
TASK BATCH VALUES IN FOUR CASES

Case	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$
US	+1	+2	+3	+4	+5	+6	+7	+8
CS	+1	+2	+1	+4	+5	+2	+7	+8
ES	+1	+2	+3	-1	+5	+6	-5	+8
CS+ES	+1	+2	+1	-1	+5	+2	-5	+8

TABLE III  
COMPUTATION TIME FOR DIFFERENT  $LTL^T$  FORMULA

$num_{B_s}$	Time(s) (US)	Time(s) (CS)	Time(s) (ES)	Time(s) (CS + ES)
96	1.1279	1.0022	1.5768	1.4740
128	1.9804	1.8143	2.6328	1.9334
192	4.7748	4.4016	6.0628	5.3489
256	8.0519	7.6519	11.1823	8.5647

consider the following  $LTL^T$  formulas in Table III :

$$\begin{aligned} \varphi_1^T &= \diamond\pi_1^{T_1} \wedge \diamond\pi_2^{T_2} \wedge \diamond\pi_3^{T_3} \wedge \diamond\pi_4^{T_4} \wedge \diamond\pi_5^{T_5} \\ &\quad \wedge \diamond\pi_6^{T_6} \wedge \diamond\pi_7^{T_7} \wedge (\neg\pi_1^{T_1} \cup \pi_2^{T_2}), \\ \varphi_2^T &= \diamond\pi_1^{T_1} \wedge \diamond\pi_2^{T_2} \wedge \diamond\pi_3^{T_3} \wedge \diamond\pi_4^{T_4} \wedge \diamond\pi_5^{T_5} \wedge \diamond\pi_6^{T_6} \wedge \diamond\pi_7^{T_7}, \\ \varphi_3^T &= \diamond\pi_1^{T_1} \wedge \diamond\pi_2^{T_2} \wedge \diamond\pi_3^{T_3} \wedge \diamond\pi_4^{T_4} \wedge \diamond\pi_5^{T_5} \\ &\quad \wedge \diamond\pi_6^{T_6} \wedge \diamond\pi_7^{T_7} \wedge \diamond\pi_8^{T_8} \wedge (\neg\pi_1^{T_1} \cup \pi_2^{T_2}), \\ \varphi_4^T &= \diamond\pi_1^{T_1} \wedge \diamond\pi_2^{T_2} \wedge \diamond\pi_3^{T_3} \wedge \diamond\pi_4^{T_4} \wedge \diamond\pi_5^{T_5} \\ &\quad \wedge \diamond\pi_6^{T_6} \wedge \diamond\pi_7^{T_7} \wedge \diamond\pi_8^{T_8}, \end{aligned}$$

whose number of atomic propositions are 7, 7, 8, and 8, and the corresponding number of automaton states are 96, 128, 192, and 256, respectively. The batch values of  $\pi_i^{T_i}$ ,  $i = 1, \dots, 8$ , in  $\varphi_1^T$ ,  $\varphi_2^T$ ,  $\varphi_3^T$ , and  $\varphi_4^T$  for each case are listed in Table II.

We run each case 20 times with slightly disturbed initial robot positions and the average computation time is shown in Fig. 4 and Table III. It indicates the computation time increases quadratically with the number of automaton states, which is consistent with the analysis of time complexity of IIRTS in Section III-D. It is observed that incorporating exclusive tasks increases the algorithm complexity, leading to longer computation time. Nevertheless, IIRTS can still generate satisfactory

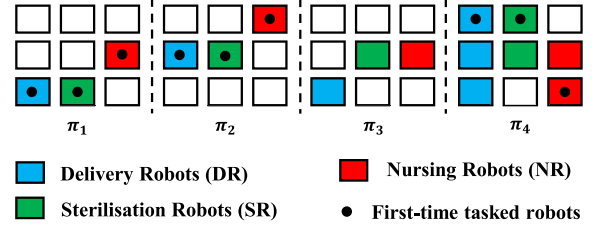


Fig. 5. The satisfactory allocation for Example 1 obtained by IIRTS. The colored squares represent the selected robots under that task. A square with a black dot represents the robot that is first assigned a task.

TABLE IV  
COMPUTATION TIME FOR DIFFERENT MULTI-ROBOT SYSTEM

A	Time(s) (US)	Time(s) (CS)	Time(s) (ES)	Time(s) (CS + ES)
$\{a_{DR}, a_{NR}, a_{SR}\} \times 15$	0.0348	0.0343	0.0376	0.0362
$\{a_{DR}, a_{NR}, a_{SR}\} \times 20$	0.0494	0.0459	0.0499	0.0473
$\{a_{DR}, a_{NR}, a_{SR}\} \times 50$	0.0638	0.0599	0.0667	0.0659
$\{a_{DR}, a_{NR}, a_{SR}\} \times 100$	0.0902	0.0872	0.1000	0.0931
$\{a_{DR}, a_{NR}, a_{SR}\} \times 300$	0.2480	0.2215	0.2690	0.2216

solutions in seconds for about 200 automaton states, which demonstrates the effectiveness of IIRTS. For compatible-task search, the computation time is reduced, both in ordinary scenarios and in complex scenarios with exclusive tasks requirements. Fig. 5 also shows that IIRTS can find a task allocation much faster than MILP. Table III reflects how the computation time of IIRTS algorithm varies with the number of automaton states.

2) *Numerical Experiment 2*: To show how the computation time varies with the robot number, we consider the  $LTL^T$  formula  $\varphi^T = \square\diamond\pi_1^{T_1} \wedge \square\diamond\pi_2^{T_2} \wedge \square\diamond\pi_3^{T_3} \wedge \square\diamond\pi_4^{T_4}$  with  $15 \times 3$ ,  $20 \times 3$ ,  $50 \times 3$ , and  $100 \times 3$  robots, respectively. Each sub-task  $\pi_i^{T_i}$ ,  $i = 1, 2, 3, 4$ , in  $\varphi^T$  requires 8, 10, 30 and 60 robots of each type, respectively. Similarly, we also consider four cases and each case is run 20 times with slightly disturbed initial robot positions. The average computation times are listed in Table IV. Although the computation times increase with the robot number, the inclusion of compatible-task search reduces the computation time and the inclusion of exclusive-search can solve complex task requirements in a shorter time. Even with 300 robots, the IIRTS can still solve complex task assignments in around 0.1s.

## B. Simulation Experiments

1) *Case 1*: Continuing with Example 1, the satisfactory allocation obtained by IIRTS is shown in Fig. 5. We mark the first-time tasked robots, i.e., at the beginning of the task. The marked robots will go directly to the task regions, which shortens the total duration of tasks and improves efficiency. In Fig. 5, the colored squares represent the selected robots under that task and the square with a dot represents the robot that is first assigned a task  $\varphi^T = \square\diamond\pi_1^{T_1} \wedge \square\diamond\pi_2^{T_2} \wedge \square\diamond\pi_3^{T_3} \wedge \square\diamond\pi_4^{T_4}$ .

In Example 1,  $\pi_1^{T_1}$  and  $\pi_2^{T_2}$  are mutually exclusive tasks, and  $\pi_1^{T_1}$  and  $\pi_3^{T_3}$  are compatible tasks. As indicated from the allocation scheme in Fig. 5, the robots previously participated in  $\pi_1^{T_1}$  are not selected to perform  $\pi_2^{T_2}$ , while the same batch of robots performing  $\pi_1^{T_1}$  is selected to continue perform  $\pi_3^{T_3}$ . Since

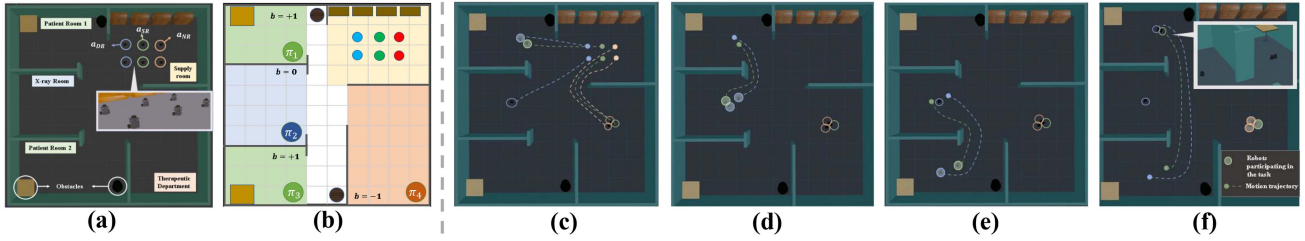


Fig. 6. The hospital scenario. (a) The Gazebo environment containing 6 turtlebots divided into 3 categories with 2 robots in each category. (b) The corresponding simulated environment, which indicates  $\pi_1^T$  and  $\pi_3^T$  are compatible tasks, and  $\pi_1^T$  and  $\pi_4^T$  are exclusive tasks. (c) The execution of  $\pi_1^T$ . The turtlebots performing  $\pi_4^T$  are marked as the first-time tasked robots, and they proceed directly to the therapeutic department. (d) The execution of  $\pi_2^T$ . (e) The execution of  $\pi_3^T$ . As  $\pi_1^T$  and  $\pi_3^T$  are compatible tasks, the same batch of turtlebots is used. (f) The execution of  $\pi_4^T$ .

$\pi_4^T$  is unrelated to any other tasks, no inter-task constraints are imposed on the selection of the robots.

2) *Case 2*: The IIRTS is then evaluated in real-time simulation experiments using Matlab and Gazebo. As shown in Fig. 6, there are 6 turtlebots divided into 3 categories, i.e., the delivery robots (DR), the sterilization robots (SR), and the nursing robots (NR). The multi-robot system is required to complete the  $LTL^T$  task

$$\begin{aligned} \varphi^T = & \square\Diamond(\pi_1, \{(DR, 1), (SR, 1), (NR, 0)\}, +1) \wedge \\ & \square\Diamond(\pi_2, \{(DR, 2), (SR, 1), (NR, 0)\}, 0) \wedge \\ & \square\Diamond(\pi_3, \{(DR, 1), (SR, 1), (NR, 0)\}, +1) \wedge \\ & \square\Diamond(\pi_4, \{(DR, 0), (SR, 1), (NR, 2)\}, -1). \end{aligned}$$

The results are shown in Fig. 6, which satisfy the  $LTL^T$  formula  $\varphi^T$  and the inter-task constraints. The robots for  $\pi_4^T$  are marked as first-time tasked robots. That is, they are not assigned a task before  $\pi_4^T$ , so they go to the therapeutic department at the beginning of the task. After task allocation using IIRTS, PRM is used to generate the robot trajectories.

Case 1 and 2 demonstrate that IIRTS can solve task allocation effectively and efficiently. In addition, the IIRTS outputs the allocation scheme and the target areas that each robot should move to, which can be conveniently integrated with existing path planning methods, such as PRM, RRT, RRT\*, for implementation in real environments. Hence, IIRTS is flexible and adaptable for various scenarios.

## V. CONCLUSION

This work develops a fast task allocation framework for heterogeneous multi-robot system subject to both temporal logic and inter-task constraints. Ongoing research will extend the single-attribute tasks to multi-attribute tasks (e.g., explicit time constraints) and consider more diverse and challenging tasks for heterogeneous multi-robot systems.

## REFERENCES

- [1] J. J. Roldán, P. Garcia-Aunon, M. Garzón, J. D. León, J. D. Cerro, and A. Barrientos, "Heterogeneous multi-robot system for mapping environmental variables of Greenhouses," *Sensors*, vol. 16, no. 7, 2016, Art. no. 1018.
- [2] N. Baras and M. Dasygenis, "An algorithm for routing heterogeneous vehicles in robotized warehouses," in *Proc. Panhellenic Conf. Electron. Telecommun.*, 2019, pp. 1–4.
- [3] K. Nagatani et al., "Emergency response to the nuclear accident at the Fukushima Daiichi nuclear power plants using mobile rescue robots," *J. Field Robot.*, vol. 30, no. 1, pp. 44–63, 2013.
- [4] Z. Zhou, D. J. Lee, Y. Yoshinaga, S. Balakirsky, D. Guo, and Y. Zhao, "Reactive task allocation and planning for quadrupedal and wheeled robot teaming," in *Proc. Int. Conf. Automat. Sci. Eng.*, 2022, pp. 2110–2117.
- [5] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal tasks," 2022, *arXiv:2208.07756*.
- [6] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.
- [7] M. Cai, K. Leahy, Z. Serlin, and C.-I. Vasile, "Probabilistic coordination of heterogeneous teams from capability temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1190–1197, Apr. 2022.
- [8] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, Jan. 2023.
- [9] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021.
- [10] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multirobot systems under global temporal tasks," *IEEE Trans. Autom. Control*, vol. 64, no. 5, pp. 1916–1931, May 2019.
- [11] Y. Kantaros and M. M. Zavlanos, "STyLuS\*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, 2020.
- [12] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multirobot systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3602–3621, Dec. 2022.
- [13] Y. E. Sahin, P. Nilsson, and N. Ozay, "Synchronous and asynchronous multi-agent coordination with cLTL+ constraints," in *IEEE Conf. Decis. Control*, 2017, pp. 335–342.
- [14] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1189–1206, Aug. 2020.
- [15] K. Leahy et al., "Scalable and robust algorithms for task-based coordination from high-level specifications (ScRATHeS)," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2516–2535, Aug. 2022.
- [16] K. Leahy, A. Jones, and C.-I. Vasile, "Fast decomposition of temporal logic specifications for heterogeneous teams," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2297–2304, Apr. 2022.
- [17] W. Liu, K. Leahy, Z. Serlin, and C. Belta, "Robust multi-agent coordination from CaTL+ specifications," 2022, *arXiv:2210.01732*.
- [18] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [19] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
- [20] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1478–1483.
- [21] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
- [22] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proc. Int. Conf. Comput. Aided Verif*, 2001, pp. 53–65.