Task Allocation of Heterogeneous Robots Under Temporal Logic Specifications With Inter-Task Constraints and Variable Capabilities

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

***Abstract*—Multi-Robot task allocation (MRTA) exploits different capabilities of heterogeneous robots to facilitate collaborative tasks. However, existing works are mainly built on a key assumption that the robot capabilities are invariant and few consider variable capabilities (e.g., task-dependent or time-dependent capabilities). Besides, there may also exist a variety of inter-task constraints (e.g., unrelated tasks, compatible tasks, and exclusive tasks). Motivated by this practical need, we develop a novel task allocation framework for heterogeneous multi-robot systems with variable capabilities subject to inter-task constraints and temporal logic task specifications. Specifically, we extend conventional linear temporal logic (LTL) to capability LTL, namely CaLTL$^{\mathcal{T}}$, to describe heterogeneous multi-robots systems with variable capabilities and inter-task constraints. The Task Batch Planning Decision Tree Plus (TB-PDT^{+}) is then developed, which encodes the states of Büchi automaton, the system states, and the task process into a tree structure to represent the exploration progress. Based on the TB-PDT^{+}, the Variable Capability and Inter-task Constraints Search (Var-CICS) is developed to find feasible task allocations and plans. Rigorous analysis shows that Var-CICS is valid (i.e., the generated task allocation is guaranteed to satisfy the task requirements) and complete (i.e., if a feasible task allocation exists, it is ensured to be found by Var-CICS). The complexity analysis also shows that the computation time of finding a satisfactory task allocation scales only quadratically with the number of automaton states, versus the exponential growth due to the product automaton in standard model checking methods. Numerical simulations and experiments demonstrate the effectiveness of Var-CICS.**

***Note to Practitioners*—Real-world applications often require a heterogeneous multi-robot system working collaboratively on a variety of tasks. Within such applications, robots can have diverse capabilities which may vary depending on the task at hand or over time, and are subject to inter-task constraints. Thus, in this work, we propose a new temporal logic to enrich the expressiveness in describing heterogeneous multi-robots systems with variable capabilities and inter-task constraints. We then develop the task batch planning decision tree plus and the variable capability and inter-task constraints search (Var-CICS) for the task allocation of heterogeneous multi-robot system. In contrast to automata-based methods, our method does not require sophisticated product automaton, which enables efficient and effective search of feasible task allocations. Furthermore, theoretical analyses show that Var-CICS is both complete and valid, while experimental results demonstrate its validity, efficiency and scalability.**

***Index Terms*—Task allocation, heterogeneous multi-robot systems, linear temporal logic.**

Received 14 April 2024; revised 21 November 2024; accepted 6 April 2025. Date of publication 8 April 2025; date of current version 24 April 2025. This article was recommended for publication by Associate Editor J. Li and Editor P. Rocco upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62173314. *(Corresponding author: Zhen Kan.)*

The authors are with the Department of Automation, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: zkan@ustc.edu.cn).

Digital Object Identifier 10.1109/TASE.2025.3558977

I. Introduction

ONE of the ultimate goals of multi-robot systems is to collaborate for a wide range of tasks that cannot be completed by individual robots alone. Towards this goal, significant research effort has been devoted in multi-robot task allocation (MRTA) by exploiting different capabilities of the robots. However, existing works are mainly built on a key assumption that the robot capabilities are invariant and few considers variable capabilities (e.g., task-dependent or time-dependent capabilities). For example, as shown in Fig. 1, assistive robots in the hospital may have multiple capabilities such as loading, disinfection, and virus-detection capabilities, which can be time-dependent or task-dependent. Besides variable capabilities, there may also exist a variety of inter-task constraints (e.g., unrelated tasks, compatible tasks, and exclusive tasks). For instance, the robots that have been to patient rooms are not allowed to visit the therapeutic department to avoid potential infection. Motivated by this practical need, this work aims to develop a task allocation framework for heterogeneous multi-robot systems with inter-task constraints and variable capabilities.

MRTA has shown great potentials in a wide range of applications such as search and rescue [1], medical assistance [2], and collective construction [3]. Owing to the rich expressiveness and resemblance to human language, linear temporal logic (LTL) is capable of specifying complex robotic tasks [4], [5], [6], [7]. Recently, LTL is being increasingly used with MRTA to describe collaborative behaviors of multi-robot systems [8], [9], [10], [11], [12]. In literature, the approaches to MRTA with temporal logic specifications can generally be classified in two categories: the bottom-up architecture and the top-down architecture. In bottom-up architectures, the sub-tasks are assigned to each robot and the group of robots collaborate to satisfy the global task [13], [14], [15]. Though effective, the complexity of bottom-up architectures may increase exponentially with the number of robots, limiting its applications in practice. Differently, the top-down

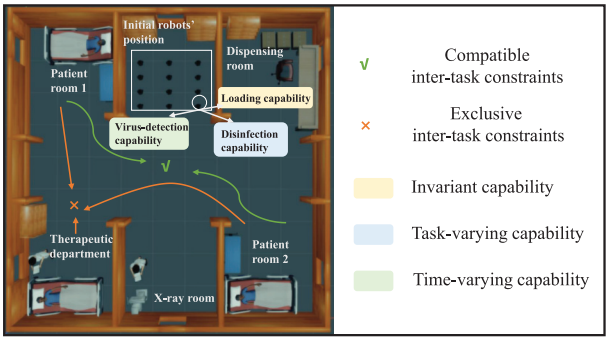

Fig. 1. The simulated hospital scenario. The tasks of visiting different patient rooms are considered as compatible while visiting the patient rooms is considered as exclusive with the task of visiting the therapeutic department due to potential infection. The robots are assumed to have different capabilities, such as the loading capability (i.e., an invariant capability), disinfection capability (i.e., a task-dependent capability), and virus-detection capability (i.e., a time-dependent capability). More explanations about the robot capabilities and the inter-task constraints are provided in Sec. III-B.

architectures [16], [17], [18] typically decompose the global task specified by temporal logic language into independent sub-tasks and select appropriate robots to perform. However, such methods usually require the construction of a product automaton, which can be computationally expensive especially for a large group of robots. In addition, inter-task constraints such as compatible and exclusive tasks are seldom considered in the aforementioned works.

Since robots with different capabilities are capable of a variety of tasks, growing research attention has been attracted to the task allocation for heterogeneous multi-agent systems [19]. For instance, signal temporal logic (STL) was extended to the capability temporal logic (CaTL) and CaTL+ in [20] and [21], respectively, to address the task allocation of heterogeneous multi-robot systems with temporal logic constraints. In [22], time window temporal logic (TWTL) was applied to specify time-bounded serial tasks for a multi-agent system. In [23], [24], and [25], the LTL formula was converted to a Büchi automaton to construct a directed tree, which is used to assign tasks to robots. In [10], the nondeterministic Büchi automaton (NBA) was relaxed by removing negative atomic propositions, based on which subtasks are extracted for the task allocation of heterogeneous multi-robot systems. The works of [8] and [26] extended LTL to counting linear temporal logic (cLTL) and counting linear temporal logic plus (cLTL+) to solve MRTA for the homogeneous multi-robot system. Compared to cLTL, cLTL+relaxes the synchronization requirements for robots and has richer expressivity. The above works, however, neither address the MRTA problem with variable capabilities nor inter-task constraints, which are common to multi-robot tasks.

Recent studies have explored task allocation and planning challenges arising from dynamic capabilities caused by robot failures. For instance, the works in [27] and [28] considered robot failure during task execution in single-robot systems, while [29], [30], [31], [32], [33] explored the challenges of such failures in multi-robot systems. Specifically, [29] proposed synthesized strategies for reacting to robot failures in homogeneous multi-robot systems. In heterogeneous multi-robot systems, [30] focused on minimizing violations when task cannot be completed due to robot failures, while [31], [32], [33] focused more on task reallocation mechanisms. For example, [31] utilized a team Markov decision process to reallocate tasks during robot failures, while [33] allocated subtasks based on the robots' current capabilities, minimizing disruptions to existing plans. However, these studies typically treated the robot failure as a binary event, either complete failure or full functionality, thus failing to address more subtle scenarios where a robot might lose a specific capability or experience degraded performance. Furthermore, in practical applications, robots often possess diverse capabilities that can dynamically change due to factors such as environmental conditions, task progress, or resource depletion. This work specifically focuses on the task allocation and planning subject to such dynamic capabilities. By tracking task progress in real-time within the motion planning module, we detect unexpected losses, degradation, or failures in robot capabilities and perform re-planning based on the current task status and the state of the robots.

To address these MRTA problems, the task objectives and constraints are often formulated in an optimization problem, such as mixed integer linear programming (MILP) [20], [34], [35]. However, optimization-based methods are generally computationally expensive and thus hard to be applied online for timely task allocation. In [36], MT* was developed to reduce the computation burden using a reduced version of the product graph. But the use of the product automaton can still lead to the state-space explosion with the increase of the task complexity and the number of robots. To alleviate this issue, sampling-based methods were exploited [24], which build a tree to approximate the product automaton. However, sampling-based methods suffer from strong randomness, and due to the existence of a large number of useless samples, excessive exploration is often needed. Again, few of the aforementioned works addresses the tasks with inter-task constraints and variable capabilities for heterogeneous multi-robot systems. As an exception, a planning decision tree was developed in our previous work [37], which takes into account inter-task constraints and achieves fast task allocation of heterogeneous robots with different categories. Nevertheless, each robot only has one capability and the considered capabilities are invariant in [37], which cannot handle task-dependent or time-dependent capabilities.

In this work, we develop a novel task allocation framework for heterogeneous multi-robot systems under temporal logic task specifications with inter-task constraints and variable capabilities. Specifically, the inter-task constraints include unrelated, compatible, and exclusive tasks. The compatible tasks allows the involved robots to be re-assigned directly (e.g., the robots that deliver medicine in patient room 1 can be re-assigned for delivery tasks in patient room 2) while the exclusive tasks restrict the task allocation to the involved robots (e.g., the robots operating in patient rooms are not allowed to get involved in any tasks in the therapeutic department). For unrelated tasks, the involved robots can be assigned freely based on the task requirement. Besides inter-task constraint, we further consider a variety of capabilities,

such as the invariant capability (i.e., the capability that does not change with time or task), the task-dependent capability (i.e., the capability that varies with task operation), and the time-dependent capability (i.e., the capability that varies with time). To describe heterogeneous multi-robots systems with variable capabilities and inter-task constraints, we extend conventional LTL to capability linear temporal logic CaLTL$^{\mathcal{T}}$. By embedding these task-specific attributes into conventional atomic propositions, CaLTL$^{\mathcal{T}}$ significantly simplifies the task allocation problem, which ensures that the planning process accounts for these critical factors upfront, reducing the complexity of subsequent solutions and leading to more efficient synthesis algorithms. The Task Batch Planning Decision Tree Plus (TB-PDT$^+$) is then developed, which encodes the states of Büchi automaton, the system, and the task process into a tree structure to represent the exploration progress. Based on the TB-PDT$^+$, the Variable Capability and Inter-task Constraints Search (Var-CICS) is developed, which can incrementally build a tree, explore, and prune the TB-PDT$^+$ to find feasible plans. Instead of formulating LTL formulas into ILP, we only need to formulate the task requirements at each node in TB-PDT$^+$, which can significantly reduce the complexity of the calculation. Additionally, the resulting plan is executed by the motion planning module in our proposed framework, which tracks the task progress online to ensure the execution of tasks. When an unexpected failure occurs, Var-CICS will regenerate a feasible plan based on the current task state and robots' state. Comprehensive numerical and simulation experiments demonstrate the effectiveness of Var-CICS.

The main contributions are summarized as follows.

- This work presents a novel task allocation framework for heterogeneous multi-robot systems with both variable capabilities and inter-task constraints. Rigorous analysis shows that the developed search algorithm Var-CICS is valid (i.e., the generated task allocation is guaranteed to satisfy the task requirements) and complete (i.e., if a feasible task allocation exists, it is ensured to be found by Var-CICS).
- We extend LTL to CaLTL$^{\mathcal{T}}$, which enhances its expressiveness and organization in describing heterogeneous multi-robots systems with variable capabilities and inter-task constraints, for more efficient task allocation and planning.
- The developed TB-PDT$^+$ encodes the automaton states generated from CaLTL$^{\mathcal{T}}$, the robots' states, and the task progression in a hierarchical tree. It is worth pointing out that TB-PDT$^+$ does not require sophisticated product automaton as in standard formal methods, which enables efficient and effective search of feasible task allocations and plans. Analytical analysis and empirical studies demonstrate that the computation time of finding a satisfactory task allocation scales only quadratically with the number of automaton states.

The rest of this paper is organized as follows. The background knowledge is presented in Section II. In Section III, we first present the models of the environment and the heterogeneous multi-robot system and then extend LTL to CaLTL$^{\mathcal{T}}$ to specify tasks with variable capabilities and inter-task constraints. A running example and the problem formulation are presented. In Section IV, we introduce the developed search algorithm Var-CICS, which is based on the planning decision tree TB-PDT$^+$. The validity and completeness of Var-CICS are analyzed in Section V. The validity, efficiency, and scalability of the Var-CICS are verified by numerical and simulation experiments in Section VI. Section VII concludes this paper.

II. Preliminaries

Linear temporal logic (LTL) has been widely applied to describe robot motion with temporal and logic constraints. The syntax of an LTL formula is defined over a set of atomic propositions $\mathcal{AP}$ as

$$\varphi ::= true \mid \pi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \cup \varphi_2,$$

where $\pi \in \mathcal{AP}$ is an atomic proposition that can be true or false, *true* is the boolean value, φ_1 and φ_2 are LTL formulas composed of atomic propositions and operators, $\neg$ (negation) and $\wedge$ (conjunction) are Boolean operators, and $\bigcirc$ (next) and $\cup$ (until) are temporal operators. The Boolean connective $\vee$ (disjunction) can be defined as $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$ and the Boolean connective $\rightarrow$ (implication) can be defined as $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$. The temporal operators $\Diamond$ (eventually) and $\Box$ (always) can be defined as $\Diamond\varphi = true \cup \varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$, respectively, i.e., $\Diamond\varphi$ means φ will be eventually satisfied and $\Box\varphi$ means φ is always satisfied. More details about LTL syntax and semantics are referred to [38].

An LTL formula can be converted to a Non-deterministic Büchi Automata (NBA) [39].

Definition 1: An NBA is defined as $\mathcal{B} = (S, S_0, \Sigma, \delta, \mathcal{F})$, where S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, $\Sigma = 2^{\mathcal{AP}}$ is the alphabet with $2^{\mathcal{AP}}$ representing the power set of $\mathcal{AP}$, $\delta : S \times \Sigma \rightarrow 2^S$ is the transition function, $\mathcal{F} \subseteq S$ is the set of accepting states.

The word $\boldsymbol{\sigma} = \sigma_0\sigma_1\sigma_2\sigma_3\ldots$ is an infinite sequence where $\sigma_i \in \Sigma$, $\forall i \in \mathbb{Z}_{\geq 0}$. The words that satisfy the LTL formula over $\mathcal{AP}$ can be captured by the NBA $\mathcal{B}$. Let $\boldsymbol{s} = s_0s_1s_2s_3\ldots$ be a run of the NBA induced by the word $\boldsymbol{\sigma}$, where $s_i \in S$ and $s_{i+1} \in \delta(s_i, \sigma_i)$, $\forall i \in \mathbb{Z}_{\geq 0}$. If the word $\boldsymbol{\sigma}$ can generate at least one run $\boldsymbol{s}$ that intersects the accepting states $\mathcal{F}$ infinitely many times, $\mathcal{B}$ is said to accept the word $\boldsymbol{\sigma}$. Generally, an accepting run $\boldsymbol{s}$ can be represented in a prefix-suffix structure, where the prefix stage is only visited once and starts from the initial state and ends with the accepting state. The suffix stage is a cycle path that visits the accepting states infinite times. Throughout this work, we will use $\mathcal{B}_\varphi$ to denote the NBA generated from the LTL formula φ.

III. System Model and Problem Formulation

To facilitate problem formulation, we first introduce the notations used throughout this article. Let $\mathbb{N}$, $\mathbb{Z}_{\geq 0}$, and $[N]$ denote the set of natural numbers, the set of non-negative integers, and the shorthand notation for $\{1, \ldots, N\}$, respectively. Given a set A, denote by $|A|$ and 2^A the cardinality and power set of A, respectively. Given a sequence $\boldsymbol{s} = s_0s_1\ldots$, denote by $s[j\ldots] = [s_js_{j+1}\ldots]$ and $s(j) = [s_0\ldots s_j]$. Given two sets A and B, the set difference is denoted as $A \backslash B = \{x | x \in A, x \notin B\}$.

A. Environment

The operation environment is defined as a 5-tuple $\bar{Env} = (M, Q, \mathcal{AP}, \mathcal{L}_m, \mathcal{L}_a)$, where M is the workspace composed of $l \in \mathbb{N}$ non-overlapped task regions M_i, $i \in [l]$, and a non-task region $\bar{M}$ (e.g., the regions that the robots can not traverse or operate within), Q is a finite set of task states and each $q \in Q$ indicates a sub-task, $\mathcal{L}_a : Q \to \mathcal{AP}$ is a labeling function that indicates the atomic proposition associated with the task state (i.e., $\mathcal{L}_a(q) = \pi$ with $q \in Q$ and $\pi \in \mathcal{AP}$), and $\mathcal{L}_m : Q \to M$ indicates the executable position of a task state in M (i.e., $\mathcal{L}_m(q) = p$ with $p = (x, y) \in \mathbb{R}^2$ indicating the execution position of $q \in Q$). Using $\bar{Env}$, the mission can be abstracted into a set of sub-tasks, where each $q \in Q$ uniquely corresponds to an atomic proposition $\mathcal{L}_a(q)$ executed at $\mathcal{L}_m(q)$ in M.

B. Heterogeneous Multi-Robot Systems

Consider a fleet of N heterogeneous robots $A = \{a_n\}_{n=1}^{N}$. Each robot in A is defined as a tuple $a_n = \left(p_0, p, \{Cap_j\}_{j\in[N_C]}, f\right)$, where p_0 and p represent the initial and current position of a_n, respectively, and f indicates whether the robot a_n functions properly, i.e., the robot is out of order and will not be able to participate in the subsequent tasks if $f = 0$ and $f = 1$ otherwise.

Let $\{Cap_j\}_{j\in[N_C]}$ represents a finite set of N_C different capabilities. In this work, there are three types of capabilities: invariant capabilities, time-dependent capabilities and task-dependent capabilities. For example, in a hospital scenario, an assistive robot might have the loading capability (LC) (e.g., the capability of lifting and packing objects such as medicines or medical instruments), the disinfection capability (DC) (e.g., spraying sanitizer), and virus-detection capability (VC) (e.g., the capability of identifying the virus using detection reagent). The loading capability is an invariant capability, as it does not change with task or time. Differently, the disinfection capability is a task-dependent capability, as its capability decreases with the consumption of sanitizer during task operation. The virus-detection capability is a time-dependent capability, since its detection reagent can gradually be ineffective with time, even if it does not perform any detection tasks. Besides, we assume that all robots are sufficiently charged, which ensures that the robots' capabilities are not affected by power. To this end, each capability is further specified as $Cap_j = (n_j, v_j)$, where n_j represents the capability type and v_j represents the corresponding capability value which can be either a constant to reflect the invariant capability or a variable over time or tasks. The variable capabilities can be represented as functions changing with tasks or time. When the value of this function falls below zero, it indicates that the capability has exceeded its effective horizon. The capability values reflect the available capability of the robot at the current time, such as the weight of package can be delivered or the left volume of sanitizer. For instance, given a robot $a = (p_0, p_1, \{(LC, v_1), (DC, f_{DC}(\bar{\pi})), (VC, f_{VC}(t))\}, 1)$, it indicates that, starting from the initial position p_0, the robot is currently at p_1 and functions well (i.e., $f = 1$) with the loading capability of a constant value v_1, the disinfection capability with a task-dependent value $f_{DC}(\bar{\pi})$, where $\bar{\pi} = \pi_0\pi_1\ldots$ indicates the disinfection tasks that have been executed so far, and the virus-detection capability with a time-dependent value $f_{VC}(t)$. In this work, due to the practical implications of capabilities, we assume that the values of capabilities possessed by robots are non-negative at any given time.

Besides different capabilities, there exist inter-task constrains, which can be classified as compatible, exclusive, and unrelated tasks. Compatible tasks are a class of tasks requiring the same robot type with the same task requirements. Hence, compatible tasks allow the reuse of the same group of robots. For instance, if visiting patient room 1 and visiting patient room 2 are compatible tasks, the robots that have visited patient room 1 can be reassigned to visit patient room 2. Differently, robots are restricted to participate in exclusive tasks. For instance, visiting patient room and visiting therapeutic department are mutually exclusive tasks, since the robots that operate in patient rooms are not allowed to be reassigned with tasks operating in the therapeutic department due to potential infection risks.

The task allocation is denoted by $Z = [z_1, z_2, z_3, \ldots, z_N]$, where $z_n = 1$, $n \in [N]$, means robot a_n participates in the task and $z_n = 0$ otherwise. The plan is defined as $\Pi = (\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{Z})$, where $\boldsymbol{s} = s_0 s_1 \ldots$, $\boldsymbol{q} = q_0 q_1 \ldots$, and $\boldsymbol{Z} = Z_1 Z_2 \ldots$ are the sequences of automaton states, task states, and task allocations, respectively, where $\boldsymbol{s}$ and $\boldsymbol{q}$ represent the mission planning. Based on the prefix-suffix structure, a plan Π can be rewritten as $\Pi = \Pi_{pre}\Pi_{suf}\Pi_{suf}\Pi_{suf}\ldots$, where Π_{pre} is the finite prefix stage and Π_{suf} is the finite suffix stage.

C. Linear Temporal Logic With Capability and Task Batch

To facilitate the description of tasks with different capabilities and inter-task constrains, inspired by [18] and [20], we extend the conventional atomic propositions $\mathcal{AP}$ to the task propositions $\mathcal{TP}$.

Definition 2: A task proposition is a tuple $tp = (\pi, \{(n_i, v_i)\}_{i\in[N_C]}, \pm b) \in \mathcal{TP}$, where $\pi \in \mathcal{AP}$ is the conventional atomic proposition and $b \in \mathbb{Z}_{\geq 0}$ is the task batch that describes inter-task constraints (i.e., tasks with the same $+b$ are compatible, two tasks with $+b$ and $-b$ are mutually exclusive, and the tasks are unrelated if $b = 0$ or they have different magnitude of b), $\{(n_i, v_i)\}_{i\in[N_C]}$ indicates the types and corresponding capability values required for the task π.

Since the focus of this work is on the problem of task allocation, for simplifying subsequent solutions, we assume that once robots reach the designated task region, they can execute the corresponding behaviors. Therefore, by Def. 2, tp is true if the robots within the task region labeled as π possess the required capabilities $\{(n_i, v_i)\}_{i\in[N_C]}$ and comply with the specified inter-task constrains $\pm b$, which ensures that the same group of robots can be reassigned for compatible tasks while the robots participating in a task cannot be reassigned to a mutually exclusive task. Based on Def. 2, we augment $\bar{Env}$ in Sec. III to $Env = \{M, Q, \mathcal{AP}, \mathcal{L}_m, \mathcal{L}_a, \mathcal{L}_t\}$ by introducing $\mathcal{L}_t : Q \to \mathcal{TP}$, which maps the task state to a task proposition, i.e., $\mathcal{L}_t(q) = tp$ with $q \in Q$ and $tp \in \mathcal{TP}$. Based on the developed $\mathcal{TP}$, we develop CaLTL$^{\mathcal{T}}$, a variant of LTL, to describe tasks with multiple capabilities and inter-task constraints.

Definition 3: (CaLTL$^{\mathcal{T}}$). The syntax of CaLTL$^{\mathcal{T}}$ is given in the Backus-Naur form of

$$\phi := true|tp|\phi_1 \wedge \phi_2|\neg\phi|\bigcirc\phi \mid \phi_1 \cup \phi_2, \tag{1}$$

where $tp \in \mathcal{TP}$ is a task proposition that can be true or false, ϕ_1 and ϕ_2 are CaLTL$^{\mathcal{T}}$ formulas composed of task propositions, and the definition of Boolean and temporal operators are the same as conventional LTL.

Given a set of robots $A = \{a_n\}_{n\in[N]}$, the trajectory of robot a_n is denoted as $\boldsymbol{\sigma}_{a_n} = \sigma_0^{a_n}\sigma_1^{a_n}\sigma_2^{a_n}\ldots$ where $\sigma_i^{a_n} \in 2^{\mathcal{AP}}$ and the trajectory at time t is denoted as $\left(\boldsymbol{\sigma}_{a_n}, t\right) = \sigma_t^{a_n}$. The satisfaction of π for robot a_n at time t is similar to that of LTL, which is denoted as $\sigma_t^{a_n} \models \pi$. That is, for any atomic proposition $\pi \in \mathcal{AP}$, $\sigma_t^{a_n}$ satisfies π if and only if $\pi \in \boldsymbol{\sigma}_{a_n}(t)$. The collection of the trajectories of all robots at time t is then denoted as (Σ_A, t), where $\Sigma_A = \{\boldsymbol{\sigma}_{a_1}, \boldsymbol{\sigma}_{a_2}\ldots, \boldsymbol{\sigma}_{a_N}\}$.

Definition 4: The semantics of the CaLTL$^{\mathcal{T}}$ are defined over (Σ_A, t) as follows:

- $(\Sigma_A, t) \models true$;
- $(\Sigma_A, t) \models tp$, where $tp = (\pi, \{(n_j, v_j)\}_{j\in[N_C]}, \pm b)) \Leftrightarrow \left\{\sum_{\sigma_t^{a_n}\models\pi} a_n.v_j \geq tp.v_j\right\}_{j\in[N_C]}$ and $\{a_n|\sigma_t^{a_n} \models \pi\}_{n\in[N]} \models tp.\pm b$;
- $(\Sigma_A, t) \models \phi_1 \wedge \phi_2 \Leftrightarrow (\Sigma_A, t) \models \phi_1$ and $(\Sigma_A, t) \models \phi_2$;
- $(\Sigma_A, t) \models \phi_1 \vee \phi_2 \Leftrightarrow (\Sigma_A, t) \models \phi_1$ or $(\Sigma_A, t) \models \phi_2$;
- $(\Sigma_A, t) \models \neg\phi \Leftrightarrow (\Sigma_A, t) \not\models \phi$;
- $(\Sigma_A, t) \models \bigcirc\phi \Leftrightarrow (\Sigma_A, t+1) \models \phi$;
- $(\Sigma_A, t) \models \phi_1 \cup \phi_2 \Leftrightarrow \exists l \geq 0$, $(\Sigma_A, t+l) \models \phi_2$, $\forall 0 \leq l' < l$, $(\Sigma_A, t+l') \models \phi_1$.

We denote by $(\Sigma_A, 0) \models \phi$ if Σ_A satisfies ϕ and write $\Sigma_A \models \phi$ as a shorthand notation. By Def. 4, if a plan $\Pi = (\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{Z})$ satisfies the CaLTL$^{\mathcal{T}}$ formula ϕ, one has $\mathcal{L}_a(\boldsymbol{q}) \models \varphi$ where φ is the LTL formula extracted from ϕ. In Def. 4, $\left\{\sum_{\sigma_t^{a_n}\models\pi} a_n.v_j \geq tp.v_j\right\}_{j\in[N_c]}$ means the capability values possessed by the assigned robots satisfy the task requirements for capabilities, where $\sum_{\sigma_t^{a_n}\models\pi} a_n.v_j$ is the total available capability values possessed by the robots whose assigned task is π. If such a value is equal to or greater than that of the task requirements, the assignment scheme satisfies the task. For instance, if there exists a task tp_1, which requires to carry 21 boxes of medicines, the total lifting capability of the assigned robots need to be equal to or greater than 21 to meet the task requirement of tp_1 (i.e., $tp_1.v_{LC}$). We denote by $\{a_n|\left(\boldsymbol{\sigma}_{a_n}, t\right) \models \pi\}_{n\in[N]} \models tp.\pm b$ if the robots with the assigned task π satisfy the task batch requirements for the corresponding task, i.e., robots assigned to perform the task π meet the requirements of its inter-task constraints including the fact that the assigned robots are identical to those performing in its mutually compatible tasks, or cannot belong to the collections of robots already assigned for its exclusive tasks. Note that, due to the prefix-suffix structure mentioned above, if Π satisfies ϕ, its finite plan $\Pi^f = \Pi_{pre}\Pi_{suf}$ also satisfies ϕ, which is a fragment of Π.

Remark 1: The expressiveness of CaLTL$^{\mathcal{T}}$ ϕ covers conventional LTL φ. If the task batch $b = 0$ and v_i in tp is a sufficiently small and non-zero positive real number (i.e., a single robot can fulfill task requirements), tp is reduced to a conventional atomic proposition, leading to conventional LTL φ.

TABLE I
THE SET OF ATOMIC PROPOSITIONS

Index	Content of atomic propositions
π_1	Visiting the patient room 1 and carrying 21 boxes of medicines, consuming 18 L of sanitizer and 191 mg of reagent.
π_2	Visiting the therapeutic department and carrying 18 boxes of medicines, consuming 22 L of sanitizer and 141 mg of reagent.
π_3	Visiting the X-ray room and carrying 12 boxes of medicines, consuming 12 L of sanitizer and 173 mg of reagent.
π_4	Visiting the patient room 2 and carrying 21 boxes of medicines, consuming 18 L of sanitizer and 191 mg of reagent.
π_5	Visiting the medication room and carrying 13 boxes of medicines, 13 L of sanitizer and 138 mg of reagent.

D. Problem Formulation

In this section, we first introduce a motivating example to illustrate the problem to be addressed.

Example 1: Consider a fleet of medical robots with different capabilities operating in a hospital environment. The desired task is $\varphi = \square\lozenge\pi_1 \wedge \square\lozenge\pi_2 \wedge \square\lozenge\pi_3 \wedge \square\lozenge\pi_4 \wedge \square\lozenge\pi_5$, which requires the robots to repeatedly perform the set of atomic propositions $\{\pi_i\}_{i=1}^5$ defined in Table I. While performing φ, there are inter-task constraints, e.g., π_2 is mutually exclusive with π_1 and π_4, thus the robots assigned to π_2 are not allowed to execute π_1 or π_4 and vice versa. Note that π_1 and π_4 are considered as compatible tasks, since they have the same task requirements requiring robots of the same type. The above tasks can be described by a CaLTL$^{\mathcal{T}}$ formula as

$$\begin{aligned}\phi = {} & \square\lozenge(\pi_1, \{LC, 21; DC, 18; VC, 191\}, +1)\wedge \\ & \square\lozenge(\pi_2, \{LC, 18; DC, 22; VC, 141\}, -1)\wedge \\ & \square\lozenge(\pi_3, \{LC, 12; DC, 12; VC, 173\}, +2)\wedge \\ & \square\lozenge(\pi_4, \{LC, 21; DC, 18; VC, 101\}, +1)\wedge \\ & \square\lozenge(\pi_5, \{LC, 13; DC, 13; VC, 138\}, +3).\end{aligned}$$

The task ϕ can be simplified to $\phi = \square\lozenge tp_1 \wedge \square\lozenge tp_2 \wedge \square\lozenge tp_3 \wedge \square\lozenge tp_4 \wedge \square\lozenge tp_5$, where $tp_1 = (\pi_1, \{LC, 21; DC, 18; VC, 191\}, +1)$ and the other tp_i, $i = 2, \ldots, 5$, are defined similarly.

Before formally presenting the problem, the following mild assumption is introduced.

Assumption 1: There exists at least one plan $\Pi = (\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{Z})$ whose induced collective trajectories Σ_A satisfying the CaLTL$^{\mathcal{T}}$ formula ϕ.

Problem 1: Given a CaLTL$^{\mathcal{T}}$ formula ϕ, the environment *Env*, and the heterogeneous multi-robot system A with various capabilities, the goal is to find a finite plan Π^f that can complete the CaLTL$^{\mathcal{T}}$ formula ϕ (i.e., the mission planning satisfying the temporal requirement, while the task allocation satisfying variable capabilities and inter-task constraints).

Remark 2: The above definitions are only for pairs of compatible or exclusive tasks. If three and more tasks are involved, the task batch can be set up as a collection that contains all inter-task constraints. For example, if both π_1 and

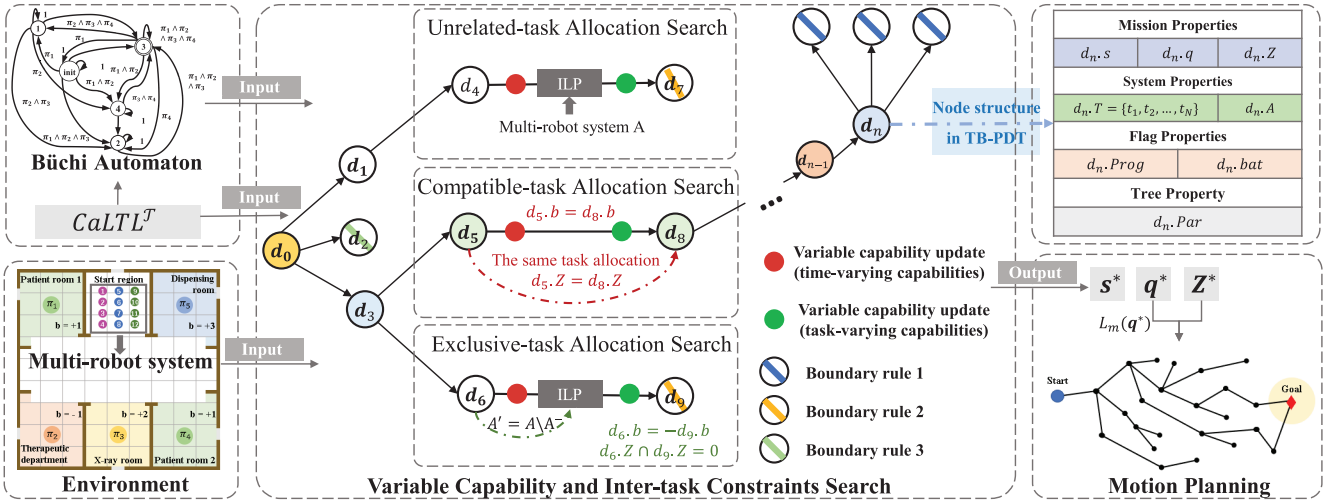

Fig. 2. The architecture of the proposed task allocation and planning. TB-PDT$^+$ is constructed based on the Büchi automaton, the states of environment, multi-robot system and tasks. The Var-CICS explores and searches for the satisfactory plan over the TB-PDT$^+$. The plan is then fed to the motion planning controller for the path planning of robots. The nodes in the TB-PDT$^+$ have four properties, namely mission properties, system properties, flag properties and the tree property. Different colors of nodes represent different task batches.

π_2 are exclusive with π_3, we can set their task batches to be $\{+1\}$, $\{+2\}$ and $\{-1, -2\}$, respectively.

IV. Task Allocation

To address Problem 1, the Task Batch Planning Decision Tree Plus (TB-PDT$^+$) and the Variable Capability and Inter-task Constraints Search (Var-CICS) are developed for the task allocation of heterogeneous multi-robot systems subject to time/task-dependent capabilities and inter-task constraints. The architecture is shown in Fig. 2.

A. Task Batch Planning Decision Tree Plus

This section presents a novel decision tree TB-PDT$^+$, a variant of the planning decision tree in our earlier work [37], to facilitate the task allocation of heterogeneous multi-robot systems.

Definition 5: The 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 $d_i = (s, q, Z, Par, T, A, Prog, bat)$, where

- $s \in S$ is the automaton state;
- $q \in Q$ is the task state, i.e., sub-tasks;
- $Z = [z_1, z_2, \ldots, z_N]$ denotes the task allocations corresponding to $\mathcal{L}_a(q)$;
- Par is the parent node of d_i and the parent node of d_0 is an empty set;
- $T = \{t_1, t_2, \ldots t_N\}$ is the estimated times of each robot completing the task of the current node d_i;
- $A = \{a_n\}_{n=1}^N$ is the multi-robot system, where $a_n = (p_0, p, \{Cap_j\}_{j\in[N_C]}, f)$;
- $Prog = \{pre, suf_s, suf_{cyc}\}$ is the flag indicating mission stages, where pre represents the prefix stage, as shown in Fig. 3, $suf_s \in S$ represents the start and end states in the first suffix stage and suf_{cyc} indicates the rest of the cycle path excluding the first suffix stage;

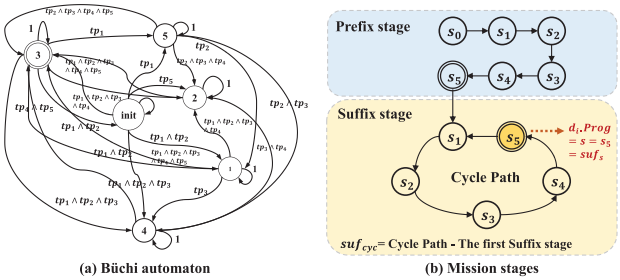

Fig. 3. (a) The Büchi automaton and (b) the schematic diagram of mission stages for $\mathcal{T}_B^+$ corresponding to the CaLTL$^{\mathcal{T}}$ formula $\phi = \Box\Diamond tp_1 \wedge \Box\Diamond tp_2 \wedge \Box\Diamond tp_3 \wedge \Box\Diamond tp_4 \wedge \Box\Diamond tp_5$ in Example 1.

- $bat \in \mathbb{Z}$ denotes the task batch, which reflects the inter-task constraints of the task propositions.

As shown in Fig. 2, each node in $\mathcal{T}_B^+$ has four types of properties: the mission properties, the system properties, the flag properties, and the tree property. The mission properties contain the automaton state, the task state, and the task allocation of the node, which indicate the mission progress. The system properties contain the estimated times and the multi-robot system, which are used to update the system states. The flag properties include the task batch and the mission stages, indicating whether an inter-task constraint is satisfied or the exploration process is stopped. An example of mission progress is shown in Fig. 3 (b). The tree property indicates the relationship between the parent node and the child node in $\mathcal{T}_B^+$. To facilitate the analysis, data structure is used throughout this work to represent the node properties, e.g., $d_i.s$ means the automaton state of the node d_i.

Assumption 2: Robots are assumed to have sufficient capabilities to perform tasks (i.e., the robots' total capability values are equal to or greater than the required task capabilities), where each capability value is constrained to be non-negative.

B. Variable Capability and Inter-Task Constraints Search

The overall idea behind Var-CICS is to incrementally construct a TB-PDT$^+$, based on which feasible task allocation can be identified. To deal with time/task-dependent capabilities and inter-task constraints, three search methods, i.e., unrelated-task allocation search, compatible-task allocation search, and exclusive-task allocation search, are developed. We select the set of robots that meets the corresponding capability types and capability values of the CaLTL$^{\mathcal{T}}$ specification with minimum total completion time. The boundary rules are also developed to prune redundant nodes based on its corresponding mission stages to further reduce the search space.

Algorithm 1 Variable Capability and Inter-Task Constraints Search (Var-CICS)

Input: CaLTL$^{\mathcal{T}}$ formula ϕ, Environment Env, the multi-robot system A
Output: $\boldsymbol{s}^*, \boldsymbol{q}^*, \boldsymbol{Z}^*$
1 Convert ϕ to Büchi automaton B_ϕ;
2 Initialize the tree $\mathcal{T}_B^+ = \{d_0\}$;
3 **while** *true* **do**
4 **for** $d_i \in \mathcal{T}_B^+$ *and* $d_i \notin \mathcal{T}_{tra}$ **do**
5 $d_i^{sub}, \mathcal{T}_B^{sub^+} \leftarrow$ **ICT**$(d_i, \mathcal{T}_B^+)$;
6 **for** $d_i^{sub} \in \mathcal{T}_B^{sub^+}$ **do**
7 Prune subtrees according to **Boundary Rules**;
8 Add $\mathcal{T}_B^{sub^+}$ to $\mathcal{T}_B^+$;
9 Add d_i in $\mathcal{T}_{tra}$;
10 Select $d^* = \mathbf{argmin}_{d_i \in \mathcal{T}_B} J$ and $d^*.Prog = suf_{cyc}$;
11 **return** $\boldsymbol{s}^*, \boldsymbol{q}^*, \boldsymbol{Z}^* \leftarrow \mathcal{T}_B^{*^+}.s, \mathcal{T}_B^{*^+}.q, \mathcal{T}_B^{*^+}.Z$;

The pseudo-code of Var-CICS is outlined in Alg. 1. Specifically, given a CaLTL$^{\mathcal{T}}$ formula ϕ, we first construct the corresponding NBA $\mathcal{B}_\phi$ using the extracted conventional LTL formula φ from ϕ (line 1). To incrementally build the tree $\mathcal{T}_{\mathrm{B}}^+$, its root node is initialized as $d_0 = (s, q, Z, Par, T, A, Prog, bat)$, where $d_0.s = s_0$, $d_0.q = q_0$, $d_0.Z = [0, 0, .., 0]$, $d_0.Par = \emptyset$, $d_0.T = \{0, 0, .., 0\}$, $d_0.Prog = pre$ and $d_0.bat = 0$ (line 2). Expanding from the root d_0, the tree $\mathcal{T}_{\mathrm{B}}^+$ grows by traversing the nodes and generating their child nodes. For each node d_i in $\mathcal{T}_{\mathrm{B}}^+$ that has not been traversed (i.e., $d_i \in \mathcal{T}_B^+$ and $d_i \notin \mathcal{T}_{tra}$ where $\mathcal{T}_{tra}$ indicates the set of nodes that has been traversed), the Inter-task Constraints Traverse (ICT) (Alg. 2) is developed to generate its sub-nodes to expand $\mathcal{T}_{\mathrm{B}}^+$ (lines 4-5). During the traversal, the boundary rules in Def. 6 are employed for pruning redundant nodes to reduce the exploration space and accelerating the traversal process (lines 6-7). The sub-tree after pruning is then added to $\mathcal{T}_B^+$ and d_i is added to $\mathcal{T}_{tra}$ (lines 8-9), which indicates that d_i has been traversed. Among all robots, the maximum time to complete the task is treated as the cost of the current node, i.e., $J = \max\{d_i.T\}$. We choose the node of the lowest cost with mission stage of suf_{cyc} as a feasible solution node d^* (line 10). The path from d^* backward to the root node is a feasible solution generated by Var-CICS. The corresponding sequences of automaton states $\boldsymbol{s}^*$, task states $\boldsymbol{q}^*$, and task allocations $\boldsymbol{Z}^*$ comprise the feasible plan Π^f that satisfies CaLTL$^{\mathcal{T}}$ for the heterogeneous multi-robot system.

Algorithm 2 Inter-Task Constraints Traverse (ICT)

Input: The current node d_i and the tree $\mathcal{T}_B^+$
Output: The sub-node d_i^{sub}
1 Initialize the subtree $\mathcal{T}_B^{sub^+} = \emptyset$;
2 **for** $s \in S \backslash \{\Gamma_i.s\}$ **do**
3 **for** q *such that* $s \in \delta(d_i.s, \mathcal{L}_a(q))$ **do**
4 Initialize the sub-node d_i^{sub};
5 **if** $|d_i^{sub}.bat| \in \{d_s.bat\}$ *with* $d_s \in \mathcal{T}_B^+$ *and* $d_i^{sub}.bat \neq 0$ **then**
6 **if** $d_i^{sub}.bat > 0$ *and* $d_i^{sub}.A.v_j \geq \mathcal{L}_t(q).v_j$, $j \in N_C$ **then**
7 $d_i^{sub} \leftarrow$ **CAS**$(d_i, d_i^{sub}, \mathcal{T}_B^+, q)$;
8 **else if** $d_i^{sub}.bat < 0$ **then**
9 $d_i^{sub} \leftarrow$ **EAS**$(d_i, d_i^{sub}, \mathcal{T}_B^+, q)$;
10 **else**
11 $d_i^{sub} \leftarrow$ **UAS**$(d_i, d_i^{sub}, \mathcal{T}_B^+, q)$;
12 **else**
13 $d_i^{sub} \leftarrow$ **UAS**$(d_i, d_i^{sub}, \mathcal{T}_B^+, q)$;
14 $d_i^{sub}.Prog =$ **PROG**$(d_i.Prog, s)$;
15 **if** $d_i^{sub}.Prog = d_i.Prog$ **then**
16 Add s into the $\{\Gamma_i.s\}$;
17 **else**
18 $\Gamma_i \leftarrow \emptyset$;
19 Add d_i^{sub} to Γ_i;
20 Add d_i^{sub} to $\mathcal{T}_B^{sub^+}$;
21 **return** The sub-node d_i^{sub} and the sub-tree $\mathcal{T}_B^{sub^+}$.

Definition 6: To prune out the branches and nodes that are unnecessarily traversed in $\mathcal{T}_{\mathrm{B}}^+$, the boundary rules are developed as follows:

1) If a node d has the mission stage $d.Prog = suf_{cyc}$, node d will stop being traversed and will be added to the traversed set $\mathcal{T}_{tra}$;
2) The node whose mission stage remains unchanged and the automaton state has been traversed will not be sampled by Var-CICS;
3) If two nodes d_1 and d_2 have the same automaton state and mission stage, i.e. $d_1.s = d_2.s$, $d_1.Prog = d_2.Prog$, the node with a larger cost will stop being traversed and will be added to the traversed set $\mathcal{T}_{tra}$.

The boundary rules are designed to avoid unnecessary exploration in $\mathcal{T}_{\mathrm{B}}^+$. The boundary rule 1 means that if the mission stage of the node is suf_{cyc}, the first suffix stage has been completed and no further exploration is needed. And the boundary rule 2 guarantees that the traversed automaton state has not been sampled if its mission stage remains the same. The boundary rule 3 ensures that $\mathcal{T}_{\mathrm{B}}^+$ will select the node with a lower cost for expansion when the nodes have the same automaton state and mission stage.

C. Inter-Task Constraints Traverse

The Inter-task Constraints Traverse (ICT) is a core algorithm in Var-CICS, which is developed to expand $\mathcal{T}_{\mathrm{B}}^+$ by generating sub-nodes as outlined in Alg. 2. Let Γ_i be the node path from d_0 to d_i excluding d_i, i.e., $\Gamma_i = d_0 \ldots d_{i-1}$, and $\{\Gamma_i.s\}$ be the set of automaton states associated with Γ_i. Given the current node d_i, its sub-nodes and sub-trees are created and initialized (lines 1-4). Specifically, we first identify all possible

task states q such that $s \in \delta(d_i.s, \mathcal{L}_a(q))$ for all automaton states $s \in S \setminus \{\Gamma_i.s\}$ that have not been traversed. The sub-node d_i^{sub} is then initialized as $d_i^{sub}.s = s$ and $d_i^{sub}.q = q$. The task allocation Z, the estimated times T, the multi-robot system A, the mission stage $Prog$, and the task batch bat of sub-node d_i^{sub} are initialized the same with the parent node d_i (line 4).

The task allocation is searched for different inter-task constraints. If there exist inter-task constraints for the current sub-node d_i^{sub}, the type of constraints (i.e., compatible, exclusive, unrelated) is determined by investigating the constraints between the task batch of d_i^{sub} and that of nodes in the tree (lines 5-13). Due to variable capabilities of robots, previously compatible tasks may no longer be compatible. Therefore, we first evaluate whether the robot capability values applied in the same task batch satisfy the current task requirement. If satisfied, i.e., $d_i^{sub}.A.v_j \geq \mathcal{L}_t(q).v_j$ for the same $j \in [N_C]$, the compatible-task allocation search (CAS) is invoked, otherwise the unrelated-task allocation search (UAS) is invoked. If the task batch and that of an explored node has the same magnitude but with different sign, it indicates exclusive inter-task constraints and thus the exclusive-task allocation search (EAS) is applied (lines 8-9). If the current sub-node d_i^{sub} does not have inter-task constraints with the any other nodes in the tree (i.e., $|d_i^{sub}.bat| \notin \{d_s.bat\}, d_s \in \mathcal{T}_B^+$) or $d_i^{sub}.bat = 0$, the unrelated-task allocation search (UAS) is performed, where $\{d_s.bat\}$ contains the task batches of all nodes in $\mathcal{T}_B^+$. The mission stage of d_i^{sub} is then determined using the function

$$\mathbf{PROG}(Prog, s) = \begin{cases} Prog, & \text{if } s \notin \mathcal{F}, \\ suf_s, & \text{if } s \in \mathcal{F}, Prog = pre, \\ suf_{cyc}, & \text{if } Prog = s. \end{cases} \tag{2}$$

If d_i^{sub} and the parent node d_i are in the same mission stage, it indicates that the current mission stage has not been finished yet and we need to continue the exploration and add the current traversed automaton state s into $\{\Gamma_i.s\}$. After that, the sub-node d_i^{sub} is added to Γ_i and the sub-tree $\mathcal{T}_B^{sub^+}$ respectively.

1) Unrelated-Task Allocation Search (UAS): For tasks without inter-task constraints or with compatible constraints but the current capabilities possessed by robots assigned to its compatible tasks do not satisfy the task requirements, the sub-nodes are traversed using UAS. Let t_n^{i-1} and $p_{a_n}^{i-1}$ be the expected time and position after completing Γ_i of robot a_n, respectively. For each robot, the expected time after completing the task $\mathcal{L}_a(q)$ corresponding to the current node d_i^{sub} is $t_n^i = t_n^{i-1} + \frac{1}{v_a}\|p_{a_n}^{i-1} - \mathcal{L}_m(q)\|_2$, where v_a is the maximal linear velocity of all robots and $\mathcal{L}_m(q)$ is the target position.

For time-dependent capability (i.e., $marker = 0$), the capability update (CU) in Alg. 6 is applied to update the capability values of the current robot system $d_i^{sub}.A$ before task allocation. We update the current multi-robot system A and obtain the capability values required for the current task state q (lines 2-3). In order to obtain a feasible task allocation with the minimum overall completion time at the current node, we formulate the task allocation problem as the following ILP

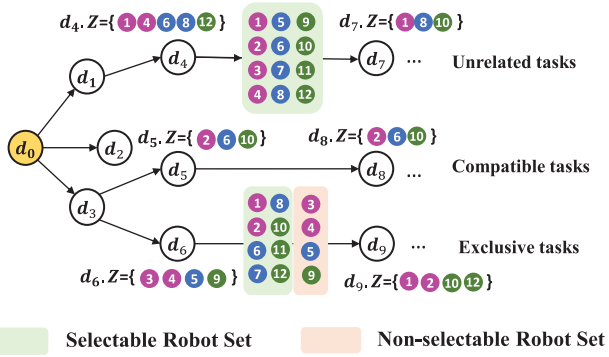

Fig. 4. The schematic diagram of different inter-task constraints on robot selections in Var-CICS.

(line 4)

$$\begin{aligned} \min_{Z} \quad & \lceil n = 1 \rceil |A| \sum d_i^{sub}.Z(n) \times t_n^i \\ \text{s.t.} \quad & \lceil n = 1 \rceil |A| \sum d_i^{sub}.Z(n) \times a_n.Cap_k.v_k \geq \mathcal{L}_t(q).v_k, k \in [N_C], \end{aligned} \tag{3}$$

where $|A|$ is the number of robots in system A. As indicated in (3), if $d_i^{sub}.Z(n) = 1$, the robot a_n is employed to complete the current task, and vice versa.

For task-dependent capabilities (i.e., $marker = 1$), the algorithm of CU is exploited to detect and update the robots selected to complete the current task (line 6). To ensure that all selected robots reach the target position $\mathcal{L}_m(q)$ before the next traversal, the expected time of the current sub-node is assigned as the maximum time among all the selected robots (line 7). The expected positions corresponding to the selected robots are updated to the current target position $\mathcal{L}_m(q)$ (line 8). As the expected times of the unselected robots remain to be t_n^{i-1}, to avoid task conflict, the expected time of the current node is set as the maximum of that of all robots (line 9).

Algorithm 3 Unrelated-Task Allocation Search (UAS)

Input: $d_i, d_i^{sub}, \mathcal{T}_B^+, q$
Output: d_i^{sub}
1 $t_n^i = t_n^{i-1} + \frac{1}{v_a}\|p_{a_n}^{i-1} - \mathcal{L}_m(q)\|_2 \in d_i.T$ for each robot $a_n \in d_i.A$;
2 $d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 0, \mathcal{T}_B^+)$;
3 Obtain the capability values required for the current task $\mathcal{L}_t(q).\{n_j, v_j\}_{j\in[N_C]}$;
// Formulate capability allocation problem as an ILP
4 $d_i^{sub}.Z \leftarrow \mathbf{ILP}(A, \mathcal{L}_t(q), t^i)$ in (3);
5 **if** $d_i^{sub}.Z(n) = 1$ **then**
6 $d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 1, \mathcal{T}_B^+)$;
7 $t_n^i \leftarrow \max\{t_j^i\}$, where $j \in \{d_i^{sub}.Z(j) = 1\}$;
8 $p_{a_n}^i \leftarrow \mathcal{L}_m(q)$;
9 $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$;
10 **return** d_i^{sub}.

2) Compatible-Task Allocation Search (CAS): For compatible tasks, if the current capabilities possessed by robots assigned to its mutually compatible tasks, these robots can be re-assigned for the compatible tasks to reduce the traversal

exploration time. For instance, as shown in Fig. 4, the robots labeled as 2, 6, and 10 are re-assigned for the compatible task corresponding to d_8.

Specifically, if there exists a node d_p in the tree $\mathcal{T}_B^+$ such that its task batch is the same as the current node d_i^{sub} (i.e., $d_i^{sub}.bat = d_p.bat$) and the capability values of robots applied in the same task batch satisfy the current task requirements, the algorithm of compatible-task allocation search (CAS) in Alg. 4 is applied. The node with the same task batch is first extracted from the tree $\mathcal{T}_B^+$ and its task allocation is assigned to the current node d_i^{sub}. The time-dependent capability values are updated by the algorithm of CU (line 3). In the algorithm of CAS, since there is no need to re-explore the allocation scheme, only their expected times need to be calculated for the selected robots (line 5). Afterwards the task-dependent capabilities of the selected robots are updated by CU (line 6). Similar to the algorithm of UAS, the expected positions and completion times are determined (lines 7-8) and used to update the node d_i^{sub} (line 9).

Algorithm 4 Compatible-Task Allocation Search (CAS)

Input: $d_i, d_i^{sub}, \mathcal{T}_B^+, q$
Output: d_i^{sub}
1 Initialize t_n^{i-1}, $p_{a_n}^{i-1}$;
2 $d_i^{sub}.Z = d_p.Z$ if $\exists d_p \in \mathcal{T}_B^+$ s.t. $d_i^{sub}.bat = d_p.bat$;
3 $d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 0, \mathcal{T}_B^+)$;
4 **if** $d_i^{sub}.Z(n) = 1$ **then**
5 $\quad t_n^i = t_n^{i-1} + \frac{1}{v_a}\|p_{a_n}^{i-1} - \mathcal{L}_m(q)\|_2$;
6 $\quad d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 1, \mathcal{T}_B^+)$;
7 $\quad t_n^i \leftarrow \max\{t_j^i\}$, where $j \in \{d_i^{sub}.Z(j) = 1\}$;
8 $\quad p_{a_n}^i \leftarrow \mathcal{L}_m(q)$;
9 $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$;
10 **return** d_i^{sub}.

3) Exclusive-Task Allocation Search (EAS): For tasks with mutually exclusive inter-task constraints (i.e., robots participated in some specific tasks cannot be assigned to some other tasks), the exclusive-task allocation search (EAS) is developed, which is outlined in Alg. 5. First we find the node d_p in $\mathcal{T}_B^+$ whose task batch has the same magnitude but with different sign to that of the current node. Then we get the task allocations of d_p and add the robots selected by the node d_p with the task $\mathcal{L}_a(d_p.q)$ to the non-selectable set A^- (lines 1-3). Based on the multi-robot system A and the non-selectable set A^-, we obtain a new selectable set $A_{new} = d_i^{sub}.A \backslash A^-$ (line 4). We then solve for the expected times for each robot in the new multi-robot set A_{new} as shown in Fig. 4 to reach the target position of the current task state q (line 5). After applying the algorithm of CU to update the time-dependent capabilities and obtaining the task requirements corresponding to the current task state, similar to (3), the ILP is formulated using t^i and A_{new} to obtain the task allocation Z (lines 6-8), such that the selected robots' capability values meet the task requirements and minimizes the sum of expected times to complete the task $\mathcal{L}_a(q)$. The selected robots are updated with their task-dependent capabilities, while assigning their expected positions and expected times to the target position corresponding to the task state and the maximum expected time for all selected robots, respectively (lines 9-12). Finally, like Alg. 3 and Alg. 4, we take the maximum value of t_n^i of the selected robots and t_n^{i-1} of the unselected robots as the expected time of the current node (line 13).

Algorithm 5 Exclusive-Task Allocation Search (EAS)

Input: $d_i, d_i^{sub}, \mathcal{T}_B^+, q$
Output: d_i^{sub}
1 **if** *there exist* $d_p \in \mathcal{T}_C$ *s.t.* $d_p.bat = -d_i^{sub}.bat$ **then**
2 $\quad$ **if** $d_p.Z(n) = 1$ **then**
3 $\quad\quad$ Add a_n into A^-;
4 Generate the new multi-robot system $A_{new} = d_i^{sub}.A \backslash A^-$;
5 $t_n^i = t_n^{i-1} + \frac{1}{v_a}\|p_{a_n}^{i-1} - \mathcal{L}_m(q)\|_2$, $a_n \in A_{new}$;
6 $d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 0, \mathcal{T}_B^+)$;
7 Obtain the capability values required for the current task $\mathcal{L}_t(q).\{n_j, v_j\}_{j \in N_C}$;
// Formulate capability allocation problem as an ILP with the t_n^i and the new multi-robot system A_{new}
8 $d_i^{sub}.Z \leftarrow \mathbf{ILP}(A_{new}, \mathcal{L}_t(q), t^i)$ in (3)
9 **if** $d_i^{sub}.Z(n) = 1$ **then**
10 $\quad d_i^{sub}.A \leftarrow \mathbf{CU}(d_i^{sub}, t^{i-1}, q, marker = 1, \mathcal{T}_B^+)$;
11 $\quad t_n^i \leftarrow \max\{t_j^i\}$, where $j \in \{d_i^{sub}.Z(j) = 1\}$;
12 $\quad p_{a_n}^i \leftarrow \mathcal{L}_m(q)$;
13 $d_i^{sub}.t_n^i \leftarrow \max\{t_n^i, t_n^{i-1}\}$;
14 **return** d_i^{sub}.

D. Capability Update (CU)

Since the robot capability can be varying (e.g. time or task dependent), the algorithm of capability update (CU) is developed. As outlined in Alg. 6, given the current sub-node d_i^{sub}, the expected times $t^{i-1} = \{t_1^{i-1}, t_2^{i-1}, \ldots, t_N^{i-1}\}$ of all robots in completing Γ_i, and the marker for time-dependent capability (i.e., $marker = 0$) or task-dependent capability (i.e., $marker = 1$), the algorithm of CU updates the corresponding capabilities. Specifically, for time-dependent capabilities, given a node $d_i \in \mathcal{T}_B^+$ we define

$$f_{time}^k \triangleq \begin{cases} d_0.A.v_k, & i = 0, \\ f_{time}^k(d_i.T), & i = 1, \\ f_{time}^k(d_i.T - d_{i-1}.T), & i \geq 2, \end{cases} \tag{4}$$

where $f_{time}^k(\cdot)$ is user specified indicating how the value of kth capability changes and $k \in [N_C]$. In (4), If the node is the root (i.e., $i = 0$), since no task has been performed at this point, the capability values remain unchanged. If the current node is the first node in $\mathcal{T}_B^+$ (i.e., $i = 1$), f_{time}^k is a function of the expected time corresponding to d_1. Otherwise (i.e., $i \geq 2$), f_{time}^k is a function of the difference between $d_i.T$ and $d_{i-1}.T$. For the task-dependent capabilities, after selecting robots, the corresponding task-dependent capabilities of the selected robots are updated according to $\mathcal{L}_t(q)$. The function $f_{task}^k(\mathcal{L}_t(q))$ shows the change of task-dependent capability value, which is a function of the task $\mathcal{L}_t(q)$.

Remark 3: It is worth pointing out that Var-CICS can address the capabilities that are invariant, time-dependent, task-dependent, or both time and task dependent. These types of capability value can both be represented as $a + bf_1(\bar{\pi}) + cf_2(t)$, where a, b and c are constant weights, f_1 and f_2 are

Algorithm 6 Capability Update (CU)

Input: d_i^{sub}, t^{i-1}, q, $marker$, $\mathcal{T}_B^+$
Output: The multi-robot system after updating the time-dependent variation capabilities $d_i^{sub}.A$

```
if marker = 0 then
    for k ∈ [N_C] do
        if d_i^sub.A.Cap_k is time-dependent then
            d_i^sub.A.v_k = f_time^k in (4);
if marker = 1 then
    for k ∈ [N_C] do
        if d_i^sub.A.Cap_k is task-dependent then
            d_i^sub.A.v_k = f_task^k(L_t(q));
return d_i^sub.A.
```

functions dependent on the task and time, respectively. The algorithm of CU can be invoked to update the corresponding capability values according to the marker.

Remark 4: For robots with neither time-dependent nor task-dependent capabilities, i.e., general task allocation problems for heterogeneous multi-robot systems with capability as in [35] and [37], the algorithm of CU can be skipped when traversing the nodes. It is sufficient to set *marker* as an arbitrary number other than 0 or 1. Therefore, heterogeneous robotic systems with constant capability values are a special case in this paper.

E. Motion Planning

Once the Var-CICS completes the task allocation, we take the sequences of automaton states $\boldsymbol{s}^*$, task states $\boldsymbol{q}^*$, and task allocations $\boldsymbol{Z}^*$ from the plan $\boldsymbol{\Pi} = (\boldsymbol{s}^*, \boldsymbol{q}^*, \boldsymbol{Z}^*)$ as inputs for the motion planning, which tracks the execution of the generated plan. If the robot accidentally loses some capabilities or failures that prevent it from completing the subsequent tasks, we will set the values of these lost capabilities to zero or set $a.f$ to zero. Besides, the current task state and the robots' position are updated to restart a new exploration and generate the new plan. Since the sequences of task states and task allocations contain the target position of each robot, existing path planning methods can be conveniently employed to generate trajectories of all robots to reach their target positions. For instance, common path planning algorithms, such as probabilistic roadmaps methods (PRM) [40], rapidly exploring random trees (RRT) [41], RRT* [42], etc, or existing novel path planning methods such as [43] can be conveniently plugged in and used with our framework for path planning.

F. Example

To illustrate the mechanism of Var-CICS, the following example is provided.

Example 2: Suppose there are 6 robots of 3 types, which can be represented as

$$\begin{aligned} a_{1:2} &= (p_0, p, \{n_1, f_{n_1}^1(\pi_1,\pi_2); n_2, f_{n_2}^1(\pi_1,\pi_2); \\ &\quad n_3, f_{n_3}^1(\pi_3,t)\}, 1), \\ a_{3:4} &= (p, p, \{n_1, f_{n_1}^2(\pi_1,\pi_2); n_2, f_{n_2}^2(\pi_1,\pi_2); \\ &\quad n_3, f_{n_3}^2(\pi_3,t)\}, 1), \\ a_{5:6} &= (p_0, p, \{n_1, f_{n_1}^3(\pi_1,\pi_2); n_2, f_{n_2}^3(\pi_1,\pi_2); \\ &\quad n_3, f_{n_3}^3(\pi_3,t)\}, 1) \end{aligned}$$

respectively. The initial positions of robots are different and f is the function indicating the robots' capability values. The capability n_1 and n_2 vary with tasks π_1 and π_2, while n_3 varies with task π_3 and time t. The required task for the robots are encoded in the following CaLTL$^{\mathcal{T}}$ formula

$$\begin{aligned} \phi = &\Diamond(\pi_1, \{(n_1, 17), (n_2, 10), (n_3, 120)\}, +1) \wedge \\ &\Diamond(\pi_2, \{(n_1, 7), (n_2, 14), (n_3, 121)\}, -1) \wedge \\ &\Box\Diamond(\pi_3, \{(n_1, 17), (n_2, 10), (n_3, 120)\}, +1) \wedge \\ &\Box\Diamond(\pi_4, \{(n_1, 10), (n_2, 10), (n_3, 110)\}, +2), \end{aligned} \tag{5}$$

which means robots are required to visit the regions of π_1, π_2, π_3, and π_4, where π_3, and π_4 need to be visited infinitely often. The tasks in ϕ require different capability values. For instance, π_1 requires the value of 17 for capability n_1, 10 for n_2 and 120 for n_3. Besides, π_1 and π_3 are mutually compatible tasks, while π_1 is an exclusive task of π_2.

First we convert the conventional linear temporal logic in ϕ to B_ϕ. The root node d_0 and the tree $\mathcal{T}_B^+$ are initialized as in Sec. IV-B, and, starting from d_0, ICT is invoked to incrementally build $\mathcal{T}_B^+$ by generating sub-nodes for the nodes that have not been traversed. Suppose d_1 is a sub-node of d_0 with $d_1.s = s_1$, $d_1.q = q_1$ and $\mathcal{L}_t(q_1) = tp_1$. Due to $tp_1.b = +1$, the task batch of d_1 is set as $d_1.bat = +1$. Since the tree $\mathcal{T}_B^+$ only contains the root node now, (i.e., $|d_1.bat| \notin \{d_s.bat\}$ with $d_s \in \mathcal{T}_B^+$), the task allocation $d_1.Z$ is obtained by UAS. After applying UAS, the estimated times of d_1 are obtained and, before computing ILP, the algorithm of CU updates the time-dependent capabilities of all robots. After obtaining the task allocation, the task-dependent capabilities of selected robots are update by CU. The generated task allocation, the updated state of multi-robot system, and the estimated times of robots are assigned to $d_1.Z$, $d_1.A$ and $d_1.T$, respectively. Since $d_1.s$ is not an accepting state, the mission stage of d_1 is set as $d_1.Prog = pre$ according to (2). And the capability values of robots are updated according to the corresponding function rules and CU in Alg. 6.

Following the similar procedure, d_0 can generate another sub-node d_2, which has the same task batch and mission stage as d_1. But the automaton state and the task state of d_2 are s_2 and q_2, respectively. The task allocation $d_2.Z$ is determined by UAS as well. The sub-node d_3 of parent node d_1 is then generated similarly, and the automaton state and task state of d_3 are s_1 and q_1, respectively. Since d_3 and d_2 have the same automaton state and mission stage and d_3 has a larger cost than d_2, d_3 will be pruned according to the boundary rule 3 and will not be expanded afterwards. Continue to expand $\mathcal{T}_B^+$ starting from d_2, and expand to nodes d_4, d_5 and d_6 sequentially in the similar way, with their automaton states and task states s_2, s_3, s_4 and q_2, q_3, q_4, respectively. Since the task batch of d_4 is -1, it has a mutually exclusive constraint with the task of d_2. Thus exploring the sub-node d_4 requires the algorithm of EAS, which adds the selected robot of d_2 to the set A^- and selects robots required by $\mathcal{L}_t(d_4.q)$ in $A \backslash A^-$. The task batch of d_5 is

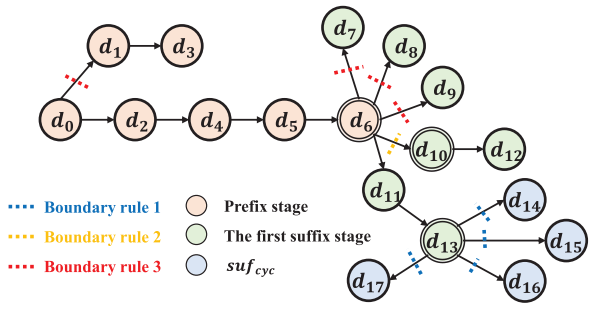

Fig. 5. The TB-PDT$^+$ generated by the Var-CICS to solve the CaLTL$^\mathcal{T}$ formula $\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Box\Diamond tp_3 \wedge \Box\Diamond tp_4$.

TABLE II
THE CORRESPONDING PROPERTIES OF NODES IN TB-PDT$^+$

Mission stage	node	$d_i.s$	$\mathcal{L}_a(d_i.q)$	$d_i.V$	$d_i.bat$
Prefix stage	d_0	$\emptyset$	$\emptyset$	$[0,0,0,0,0,0]$	0
	d_2	s_1	π_1	$[1,1,0,0,1,0]$	+1
	d_4	s_2	π_2	$[0,0,1,1,0,1]$	-1
	d_5	s_3	π_3	$[1,1,0,0,1,0]$	+1
	d_6	s_4	π_4	$[0,0,1,1,0,1]$	+2

Mission stage	node	$d_i.s$	$\mathcal{L}_a(d_i.q)$	$d_i.V$	$d_i.bat$
Suffix stage	d_{11}	s_3	π_3	$[1,1,0,0,1,0]$	+1
	d_{13}	s_4	π_4	$[0,0,1,1,0,1]$	+2

the same as that of d_2, so they have a compatible relationship. Before performing the algorithm of CAS, it is necessary to determine whether the current capability values of the robots selected for the same the task batch of node (i.e. d_2) satisfy the current task requirements. If satisfied, CAS is applied, which directly employs the task allocation for the same task batch of node (i.e. $d_5.Z = d_2.Z$), and UAS is invoked otherwise. According to the task batch of d_6, we can know that d_6 is solved by UAS. Since the automaton state of d_6 is an accepting state, the mission stages of the sub-node d_7, d_8, d_9, d_{10} and d_{11} generated by d_6 are within the first suffix stage according to (2). The sub-nodes d_7, d_8, d_9 and d_{10} have the same automaton state and mission stage. Since the costs of the sub-nodes d_7, d_8 and d_9 are larger than that of d_{10}, the nodes d_7, d_8 and d_9 are pruned according to the boundary rule 3. Then d_{10} generates the sub-node d_{12}, which has the same automaton state and mission stage as d_{11} (i.e. $d_{12}.s = d_{11}.s = s_3$ and their mission stages are both the suffix stage). Since the automaton state of d_{11} is the traversed automaton state, Var-CICS prunes all nodes on the branch of d_{12} according to the boundary rule 2.

Since the mission stages of nodes d_{14}, d_{15}, d_{16} and d_{17} are known as suf_{cyc} by (2), they are all pruned according to the boundary rule 1. Therefore, the whole tree TB-PDT$^+$ generated by Var-CICS is shown in Fig. 5, where the valid node paths are $\{d_0, d_2, d_4, d_5, d_6, d_{11}, d_{13}\}$ and the properties of the corresponding nodes are shown in Table II.

G. The Complexity of Var-CICS

The node will stop being traversed if its mission stage is suf_{cyc} and there are no repetitive automaton states for the same mission stage. Thus, the length of $\mathcal{T}_B^+$ is at most $3|S|$, where $|\cdot|$ is the cardinality of a set. The number of expandable nodes of $\mathcal{T}_B^+$ is at most $(2 + |\mathcal{F}|) \times |S|$ due to the boundary rule 3. Since a parent node generates no more than $|S| \times |Q|$ sub-nodes, at most $((2 + |\mathcal{F}|) \times |S|) \times (3 \times |S|) \times (|S| \times |Q|)$ nodes can be explored. ILP is performed in each node to compute the task allocation and the complexity of computing each task allocation is $\mathcal{O}\left(N_c \times 2^N\right) \approx \mathcal{O}\left(2^N\right)$. Since $|\mathcal{F}|$ is generally much smaller than $|S|$, the time complexity of Var-CICS is approximate as $\mathcal{O}(|S|^3 \times 2^N)$. It is worth pointing out that, in practice, the number of generated sub-nodes $|S| \times |Q|$ is small, and thus the complexity of Var-CICS is approximate to $\mathcal{O}(|S|^2 \times 2^N)$ in most cases. This observation is also evidenced by extensive simulations in Sec. VI. When the number of robots N does not change, the computation time of Var-CICS increases approximately linearly proportional to $|S|^2$.

H. Discussions

As there may exist many satisfying plans, we are more interested in those that complete tasks fast. Therefore, in this work, we focus more on the estimated times in the objective function. The estimated times for robots in Algs 3- 5 are calculated based on the robots' maximal linear velocity speed and the distances between the robots' current positions and the goal positions corresponding to the current node task. Such an estimate time can be conservative, as robots may not move along straight lines in the environment. Since we do not assume full knowledge of the environment and have no clues about the future robot paths before task allocation, we use the estimated times as a lower bound of the estimated arrival time in this work, based on which the task allocation is then explored over $\mathcal{T}_B^+$. Hence, the estimated time in TB-PDT$^+$ can be considered as a lower bound on the operation times, rather than the actual execution times of robots, which is used to determine the costs of task allocations and solve task allocations in Var-CICS.

V. ALGORITHM ANALYSIS

This section investigates the performance of Var-CICS in the following aspects: the validity and completeness. The validity indicates if the generated plan satisfies the task specification and the algorithm completeness indicates whether or not a feasible solution, if exits, is guaranteed to be found.

Theorem 1: Given a CaLTL$^\mathcal{T}$ *formula* ϕ*, an environment Env, and a heterogeneous multi-robot system A, the developed Var-CICS is valid, i.e., the plan* Π *generated by Var-CICS satisfies* ϕ*.*

Proof: Since the extension of TB-PDT$^+$ is based on the feasible transitions in the büchi automaton, it can be known that the generated sequence of automaton states is satisfying the temporal logic of ϕ. Therefore, in the following, to prove the Theorem 1, we just need to prove that the robots assigned by the generated task allocations satisfy the capability and inter-task constraint requirements of task propositions.

Given a TB-PDT$^+$ $\mathcal{T}_B^+$, without loss of generality, suppose that tp is the current task proposition to be executed, which corresponds to the node d_i in $\mathcal{T}_B^+$ such that $\mathcal{L}_t(d_i.q) = tp$,

$\mathcal{L}_a(d_i.q) = \pi$ and $tp = (\pi, \{Cap_j\}_{j\in N_c}, \pm b)$. To assign appropriate robots to the current task, Var-CICS expands $\mathcal{T}_B^+$ by ICT in Alg. 2. The ICT determines which search method should be applied based on the values of the task batch and the current capability values of robots.

If $|d_i.bat| \notin \{\mathcal{T}_B^+.bat\}$, $(|d_i.bat| \in \{\mathcal{T}_B^+.bat\}) \cap (d_i.bat > 0) \cap (d_i^{sub}.A.v_j < \mathcal{L}_t(q).v_j)$ or $d_i.bat = 0$, UAS in Alg. 3 is performed. It evaluates the expected arrival time at $\mathcal{L}_m(d_i.q)$ for each robot and updates the current capability values of robots in A based on Alg. 6. The required task allocation Z is then solved via ILP using the current robots' capabilities and the required capabilities for $\mathcal{L}_t(q)$. Based on Z, the task-dependent capabilities of the selected robots are updated. ILP ensures that the task allocation solved by Alg. 3 is valid.

Similarly, if $(|d_i.bat| \in \{\mathcal{T}_B^+.bat\}) \cap (d_i.bat > 0) \cap (d_i^{sub}.A.v_j \geq \mathcal{L}_t(q).v_j)$, CAS in Alg. 4 is performed. Alg. 4 is valid since ICT determines whether the robots selected in the task corresponding to the same task batch satisfy the subsequent task requirements at this point before choosing the search method of CAS. If satisfied, CAS is performed and otherwise, UAS is performed.

If $|d_i.bat| \in \{\mathcal{T}_B^+.bat\}$ and $d_i.bat < 0$, EAS is performed. Different from the UAS, EAS applies the ILP to select robots in the complementary set of task allocations that are mutually exclusive with the current task. Under Assumption 1, it is guaranteed that ILP will have a solution, so the task allocation solved by EAS is valid.

Since the task batch condition is already guaranteed when choosing which search method to apply, the task allocation obtained by the Var-CICS satisfies both capabilities and inter-task constraint requirements. Therefore, the sequences of automaton states, task states and task allocations satisfy the temporal, capability and inter-task constraints of CaLTL$^{\mathcal{T}}$ formula and Var-CICS is valid. ■

Theorem 2: Given a CaLTL$^{\mathcal{T}}$ *formula ϕ, an environment Env, a heterogeneous multi-robot system A, and Assumption 2 holds, the developed Var-CICS is complete, i.e., if a feasible plan satisfying the* CaLTL$^{\mathcal{T}}$ *formula ϕ exists, it is ensured to be found by Var-CICS.*

Proof: The plan contains the sequences of automaton states, task states and task allocations. Thus the completeness of Var-CICS will be proved in terms of task allocation and mission planning, respectively. Since the tree expands nodes through feasible automaton state transitions and compute task allocations at each node, the mission planning and task allocation do not affect each other. Therefore, the Var-CICS is complete if both mission planning and task allocation are complete individually.

We first show that the completeness of task allocation is ensured by Var-CICS. The Var-CICS first determines the task batch of the node and select the search method accordingly. For unrelated tasks and exclusive tasks in Alg. 3 and Alg. 5, respectively, Var-CICS solves for task allocation using ILP in (3). Thus as long as Assumption 2 holds, the ILP can always find a feasible solution for the current task with the minimum total estimated operation time. For compatible tasks, the same group of robots is employed. Similarly, given that Assumption 2 holds, The compatible-task allocation search in Alg. 4 can always find a task allocation such that the capability values of selected robots satisfy the task requirement. Therefore, Var-CICS guarantees the completeness of the task allocation.

Then, we need to prove that the Var-CICS guarantees the completeness of the mission planning. First, we consider the case where there are no boundary rules. We assume that there exists a plan $\Pi^f = \Pi_{pre}\Pi_{suf}$ satisfies the CaLTL$^{\mathcal{T}}$ formula ϕ in the given environment *Env* and heterogeneous multi-robot system A. Suppose the corresponding sequences of automaton states, task states and task allocations in Π^f are $\boldsymbol{s} = s_0 s_1 \dots s_{end}$, $\boldsymbol{q} = q_0 q_1 \dots q_{end}$ and $\boldsymbol{Z} = Z_1 Z_2 \dots Z_{end}$, respectively, where $\pi_i = \mathcal{L}(q_i)$, $\pi_i \in \boldsymbol{\pi}$, $q_i \in \boldsymbol{q}$ and $s_{i+1} \in \delta(s_i, \mathcal{L}_a(q))$. To see that Π^f, if exists, is ensured to be found, we start from the root node of the TB-PDT$^+$. By the definition of the TB-PDT$^+$, we can know that $d_0.s = s_0 \in \boldsymbol{s}$ and $d_0.q = q_0 \in \boldsymbol{q}$. If there are no boundary rules, then all possible nodes will be traversed by the Var-CICS. Thus, there will exist a node d_m such that $d_m.s = s_{n_1} \in \boldsymbol{s}$, $d_m.q = q_{n_1} \in \boldsymbol{q}$ and $\mathcal{L}(d_m.q) = \pi_{n_1} \in \boldsymbol{\pi}$, where $1 \leq n_1 \leq end$. Similarly, for $\forall(s_k, q_k, Z_k) \in (\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{Z})$, there always exists a node d' satisfying $d'.s = s_k$, $d'.q = q_k$ and $d'.Z = Z_k$. Therefore, the node d_{end} can be found similarly. The planning from the node d_{end} backward until the root node d_0 is the one satisfying the CaLTL$^{\mathcal{T}}$ formula ϕ.

When the boundary rules are added, since boundary rule 1 only stops traversal when the mission stage of the node is suf_{cyc} and aims to generate a finite plan $\Pi^f = \Pi_{pre}\Pi_{suf}$ rather than a plan $\Pi = \Pi_{pre}\Pi_{suf}\Pi_{suf}\Pi_{suf} \dots$, the boundary rule 1 does not affect the completeness of the mission planning. Suppose that $d_0 d_1 \dots d_{end}$ is a feasible path found by the Var-CICS in the generated TB-PDT$^+$ without using boundary rules, and the corresponding finite plan is $\Pi^f = (\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{Z})$. If there exist two nodes d_p and d_q on this path, while $d_p.s = d_q.s$, $d_p.Prog = d_q.Prog$ and $0 < p < q < end$, this path changes to $d_0 \dots d_p d_{q+1} \dots d_{end}$ and the corresponding sequences of task states and automaton states change from $s_0 s_1 \dots s_{end}$ and $q_0 q_1 \dots q_{end}$ to the $s_0 \dots s_p s_{q+1} \dots s_{end}$ and $q_0 \dots q_p q_{q+1} \dots q_{end}$, respectively, when the boundary rule 2 is applied. Since both nodes have the same mission stage and the automaton state of d_q is same as the already traversed d_p, the new sequences of task state and automaton state do not affect the feasibility of ϕ. Moreover, the task allocation and the mission planning do not affect each other. Therefore, the boundary rule 2 does not affect the completeness of the mission planning. Continue with the feasible path $d_0 d_1 \dots d_{end}$ generated by Var-CICS without using boundary rules. If there exist two nodes d_g and d_h on this path such that $d_g.s = d_h.s \in \boldsymbol{s}$, $d_g.Prog = d_h.Prog$ and d_g has a lower cost than d_h, the node path becomes $d_0 \dots d_g d_{h+1} \dots d_{end}$ when the boundary rule 3 is added. Similarly, since d_g and d_h have the same automaton state and are in the same mission stage, the new path $d_0 \dots d_g d_{h+1} \dots d_{end}$ also satisfies the CaLTL$^{\mathcal{T}}$ formula ϕ. Therefore, the boundary rule 3 does not affect the completeness of the mission planning. In summary, the boundary rules do not compromise the completeness of the Var-CICS, and they only affect the expansion of nodes in the TB-PDT$^+$. In sum, Var-CICS can guarantee the completeness of mission

TABLE III
THE TASK BATCHES FOR EACH SUBTASK IN FOUR CASES

Case	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8
UAS	+1	+2	+3	+4	+5	+6	+7	+8
CAS	+1	+2	+1	+4	+5	+4	+7	+8
EAS	+1	+2	+3	+4	-1	+6	+7	-2
CAS+EAS	+1	+2	+1	+4	-1	+4	+7	-2

planning both with and without the boundary rules, and thus Var-CICS is complete. ■

VI. EXPERIMENT

In this section, the developed task allocation is evaluated via numerical and simulation experiments using MATLAB R2019b and Gazebo on a computer with Intel Core i5-1050ti. The default solver of MATLAB is utilized to solve ILP problems. We examine: **1) the validity**: whether the Var-CICS can obtain a task allocation that satisfies the CaLTL$^{\mathcal{T}}$ formula ϕ in the presence of variable capabilities and inter-task constraints; **2) the efficiency**: whether the Var-CICS achieves high efficiency when solving MRTA problem; and **3) the scalability**: how the performance of Var-CICS varies with the number of robots and the automaton states.

A. Numerical Experiments

The performance of Var-CICS is evaluated in this section under different numbers of automaton states, atomic propositions and robots with inter-task constraints and variable capabilities.

1) Performance of Var-CICS With Inter-Task Constraints: We consider 45 robots of 3 different types. Suppose each type contains 15 robots and each robot has three kinds of capabilities, i.e.,

$$a_{1:15} = (p_0, p, \{n_1, f_1^1(\bar{\pi}); n_2, f_2^1(\bar{\pi}); n_3, f_3^1(t)\}, 1),$$
$$a_{16:30} = (p_0, p, \{n_1, f_1^2(\bar{\pi}); n_2, f_2^2(\bar{\pi}); n_3, f_3^2(t)\}, 1),$$
$$a_{31:45} = (p_0, p, \{n_1, f_1^3(\bar{\pi}); n_2, f_2^3(\bar{\pi}); n_3, f_3^3(t)\}, 1),$$

where n_1 and n_2 are task-dependent capabilities and n_3 is a time-dependent capability. We consider four different cases, namely: 1) without inter-task constraints (i.e., only unrelated-task allocation search is applied); 2) with compatible inter-task constraints (i.e., compatible-task allocation search is applied); 3) with exclusive inter-task constraints (i.e., exclusive-task allocation search is applied); and 4) with both compatible and exclusive inter-task constraints (i.e., both compatible-task and exclusive-task allocation search are applied). We run each case 10 times with slightly disturbed initial robot positions for the corresponding CaLTL$^{\mathcal{T}}$ formula ϕ and the average computation time is shown in Fig. 6 and Table IV.

Fig. 6 indicates the computation times for the four cases with the number of automaton states ranging from 5 to 64. As can be seen from Fig. 6, the computation time of Var-CICS is only quadratically proportional to the number of automaton states. Table IV shows the computation time of Var-CICS for a

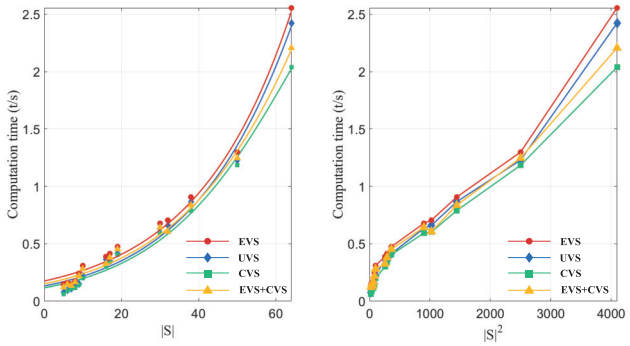

Fig. 6. The computation times of Var-CICS for four cases with the number of automaton states ranging from 5 to 64. The left plot indicates how the computation time scales with the number of automaton states, and the right plot further shows that the computation time indeed scales approximately linearly to the square of the number of automaton states.

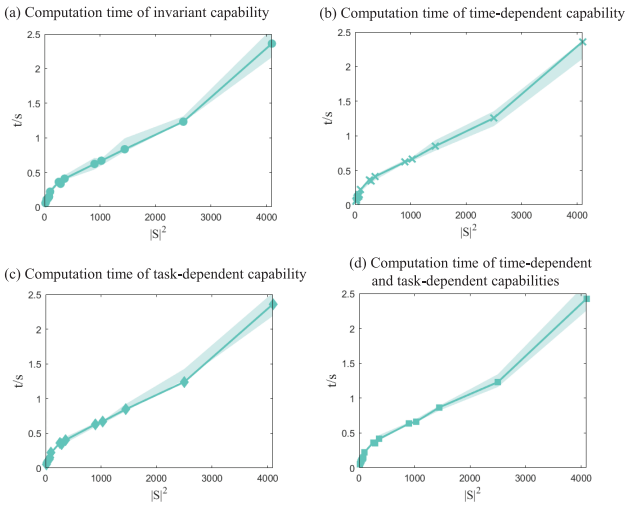

Fig. 7. The computation time of Var-CICS with the number of automaton states ranging from 4 to 64 for the cases with invariant, time-dependent, task-dependent and both time-dependent and task-dependent capabilities.

set of different CaLTL$^{\mathcal{T}}$ formula with automaton states ranging from 73 to 256 and the settings of task batches for each subtask in the four cases are detailed in Table III.

From Fig. 6 and Table IV, the following four conclusions can be drawn:

- Tasks with compatible inter-task constraints can reduce the computation time by CAS;
- Tasks with exclusive inter-task constraints increase the computation time when performing EAS due to the increased task complexity;
- The computation time of Var-CICS is approximately linearly proportional to the square of the number of automaton states;
- For the case where the number of automaton states is large or the inter-task constraints are added, Var-CICS can get the task allocation in less than $3s$ with 64 automaton states, and thus the algorithm is efficient and scalable.

2) Performance of Var-CICS With Variable Capability: We consider the same fleet of robots and the same tasks as in previous experiments, but with compatible and exclusive

TABLE IV
SOLUTION FOR DIFFERENT CaLTL$^{\mathcal{T}}$ FORMULA

CaLTL$^{\mathcal{T}}$ formula	num_{B_s}	Time(s) (*unrelated tasks*)	Time(s) (*compatible tasks*)	Time(s) (*exclusive tasks*)	Time(s) (*compatible +exclusive*)
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge (tp_5 \rightarrow \bigcirc tp_2) \wedge (\neg tp_3 \cup tp_4)$ (6)	73	1.7769	1.4406	1.8128	1.6854
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge (\neg tp_1 \cup tp_2)$ (7)	96	3.8457	3.3094	4.8772	4.0914
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7$ (7)	128	6.0076	4.5198	6.4823	5.8684
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge \Diamond tp_8 \wedge (\neg tp_1 \cup tp_2)$ (8)	192	10.9577	10.6496	13.0118	10.9043
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge \Diamond tp_8$ (8)	256	17.4186	16.3260	19.2960	18.1460

TABLE V
SOLUTION FOR DIFFERENT CaLTL$^{\mathcal{T}}$ FORMULA

CaLTL$^{\mathcal{T}}$ formula	num_{B_s}	Time(s) (*no variation*)	Time(s) (*time variation*)	Time(s) (*task variation*)	Time(s) (*time +task*)
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge (tp_5 \rightarrow \bigcirc tp_2) \wedge (\neg tp_3 \cup tp_4)$ (6)	73	1.7772	1.7844	1.7310	1.7769
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge (\neg tp_1 \cup tp_2)$ (7)	96	3.7837	3.8122	3.7053	3.8457
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7$ (7)	128	5.8771	5.9496	5.9213	6.0076
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge \Diamond tp_8 \wedge (\neg tp_1 \cup tp_2)$ (8)	192	10.9733	10.9645	10.9903	10.9577
$\Diamond tp_1 \wedge \Diamond tp_2 \wedge \Diamond tp_3 \wedge \Diamond tp_4 \wedge \Diamond tp_5 \wedge \Diamond tp_6 \wedge \Diamond tp_7 \wedge \Diamond tp_8$ (8)	256	17.1460	17.4415	17.1031	17.4186

TABLE VI
SOLUTION FOR DIFFERENT HETEROGENEOUS MULTI-AGENTSYSTEM

A	Time(s) (*unrelated tasks*)	Time(s) (*compatible tasks*)	Time(s) (*exclusive tasks*)	Time(s) (*compatible +exclusive*)
$a_{cate_1} \times 10, a_{cate_2} \times 10, a_{cate_3} \times 10$	0.0934	0.0786	0.1378	0.1211
$a_{cate_1} \times 20, a_{cate_2} \times 20, a_{cate_3} \times 20$	0.1005	0.0850	0.1419	0.1283
$a_{cate_1} \times 50, a_{cate_2} \times 50, a_{cate_3} \times 50$	0.1531	0.1381	0.2640	0.1977
$a_{cate_1} \times 100, a_{cate_2} \times 100, a_{cate_3} \times 100$	0.2708	0.2290	0.3589	0.2828

inter-task constraints and variable capabilities. Similarly, we consider four cases: 1) robots with invariant capabilities; 2) robots with invariant and time-dependent capabilities; 3) robots with invariant and task-dependent capabilities; and 4) robots with invariant, time-dependent, and task-dependent capabilities. Fig. 7 shows the evolution of the computation time of Var-CICS as the number of automaton states increases from 4 to 64 in four different cases. Table V indicates the computation time of Var-CICS for the four cases for a set of CaLTL$^{\mathcal{T}}$ formula with automaton states ranging from 73 to 256, where the computation time is the average of running 10 times with slightly disturbed initial robot positions.

Through Fig. 7 and Table V, it can be concluded that the computation time is linearly proportional to the square of the number of automaton states, but does not vary with the robot capability types, since the variable capability update in Alg. 6 does not increase the complexity of the overall algorithm.

3) Performance of Var-CICS With Robot Numbers: To show how the number of robots affects the computation time in the previous two cases, we consider 30, 60, 150 and 300 robots for each case, i.e., there are 3 types of robots and each with 10, 20, 50 and 100 robots. For each test, we also average the computation time after running 10 times with different initial robot positions. Table VI and Table VII show the average computation time for different numbers of robots with different inter-task constraints and different capability types, respectively. To further show the performance of Var-CICS with a large group of robots, we tested the average computation time of Var-CICS after running 10 times with the

TABLE VII
SOLUTION FOR DIFFERENT HETEROGENEOUS MULTI-AGENTSYSTEM

A	Time(s) (no variation)	Time(s) (time)	Time(s) (task)	Time(s) ($time$ $+task$)
$a_{cate_1} \times 10, a_{cate_2} \times 10, a_{cate_3} \times 10$	0.0931	0.0921	0.0928	0.0934
$a_{cate_1} \times 20, a_{cate_2} \times 20, a_{cate_3} \times 20$	0.0962	0.0960	0.0969	0.1005
$a_{cate_1} \times 50, a_{cate_2} \times 50, a_{cate_3} \times 50$	0.1481	0.1560	0.1566	0.1531
$a_{cate_1} \times 100, a_{cate_2} \times 100, a_{cate_3} \times 100$	0.2606	0.2644	0.2710	0.2708

TABLE VIII
SOLUTION FOR DIFFERENT HETEROGENEOUS MULTI-AGENTSYSTEM

A	Time(s)
$a_{cate_1} \times 250; a_{cate_2} \times 250; a_{cate_3} \times 250$	0.3716
$a_{cate_1} \times 500; a_{cate_2} \times 500; a_{cate_3} \times 500$	0.7140
$a_{cate_1} \times 1000; a_{cate_2} \times 1000; a_{cate_3} \times 1000$	1.3776
$a_{cate_1} \times 2000; a_{cate_2} \times 2000; a_{cate_3} \times 2000$	2.9357
$a_{cate_1} \times 5000; a_{cate_2} \times 5000; a_{cate_3} \times 5000$	9.0335

number of robots ranging from 750 to 15000 of 3 capability types without inter-task constraints in Table VIII.

According to Table VI and Table VII, the conclusions in previous two cases still hold. Besides, by Table VIII, we also conclude that the computation time increases with the increase of the number of robots, but the Var-CICS can still get the task allocation in a short time when the number of robots reaches 15000, which demonstrates the scalability of our approach.

B. Simulation Experiment

To further verify the effectiveness of the Var-CICS, we evaluate the algorithm in a hospital environment using MATLAB and Gazebo. Following Example 1, we consider three types of robots, each with four robots and each robot has three types of capabilities, i.e., the loading capability (LC), the disinfection capability (DC), and the virus-detection capability (VC). The loading capability is an invariant capability, the disinfection capability is a task-dependent capability and the virus-detection capability is time-dependent. Suppose the capability values change according to

$$\begin{aligned} f_{LC} &= v_{LC}^0, \\ f_{DC} &= v_{DC}^0 + k_{DC}^1 I(\pi_1) + k_{DC}^2 I(\pi_2) + k_{DC}^4 I(\pi_4), \\ f_{VC} &= v_{VC}^0 + k_{VC}(t), \end{aligned} \tag{6}$$

where $I(\cdot)$ is the indicator function, i.e., $I(\pi_i) = 1$ when executing π_i. In (6), $v_{LC}^0 \in \mathbb{R}^+$ is a constant indicting the invariant loading capability, $k_{DC}^i \in \mathbb{R}^+$, $i = 1, 2, 4$, are the associated weights indicating how much the capability values vary with the tasks (e.g., k_{DC}^1 is set as the area of patient room 1 to indicate the volume of sanitizer used for the disinfection task π_1 and we set $k_{DC}^1 = k_{DC}^2 = k_{DC}^4 = -1$ in this experiment), and $k_{VC}(t) \in \mathbb{R}^+$ indicates how much the value of virus-detection capability varies with time (e.g., we set $k_{VC}(t) = -0.02t$ to indicate that the effectiveness of the reagent decreases with time).

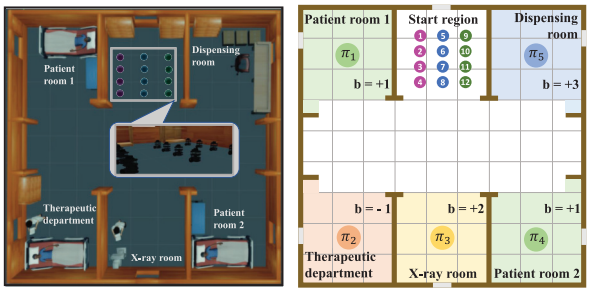

Fig. 8. The hospital scenario consists of 2 patient rooms, X-ray room, dispensing room and therapeutic department. The left is the simulated hospital environment in Gazebo. It contains 12 turtlebots, divided into 3 categories and 4 in each category. The right is the corresponding floor plan, which shows the inter-task constraints between tasks: π_2 is mutually exclusive with π_1 and π_4, π_1 and π_4 are compatible tasks, π_3 and π_5 are both unrelated tasks.

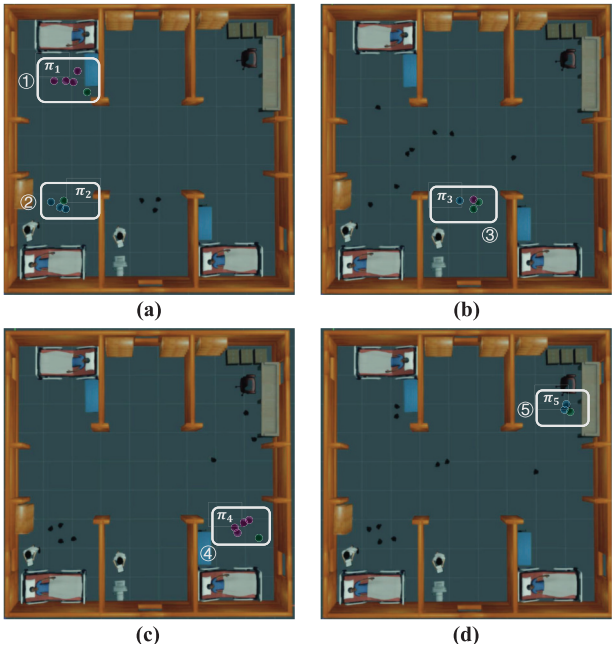

Fig. 9. The results of the simulation experiment. (a) Completions of π_1 and π_2. As π_1 and π_2 are exclusive tasks, the robots in π_1 are not involved in π_2. (b) Completion of π_3. (c) Completion of π_4. As π_1 and π_4 are compatible tasks, the same batch of turtlebots is used. (d) Completion of π_5 and start to complete the next round of tasks.

The heterogeneous multi-robot system is required to complete the CaLTL$^{\mathcal{T}}$ formula

$$\phi = \square\Diamond tp_1 \wedge \square\Diamond tp_2 \wedge \square\Diamond tp_3 \wedge \square\Diamond tp_4 \wedge \square\Diamond tp_5,$$

Fig. 10. The experimental results without and with robot failures. (a)-(e) The results for tasks tp_1 through tp_5, respectively, without robots failures. (f)-(j) The results for tasks tp_1 through tp_5, respectively, where a_1 and a_5 fail after completing task tp_2.

TABLE IX
CAPABILITY VALUES OF ROBOTS AFTER COMPLETING TASKS

task	type 1	type 2	type 3
π_0	$(5,5,70.0000)_{a_1}$ $(5,5,70.0000)_{a_2}$ $(5,5,70.0000)_{a_3}$ $(5,5,70.0000)_{a_4}$	$(5,7,50.0000)_{a_5}$ $(5,7,50.0000)_{a_6}$ $(5,7,50.0000)_{a_7}$ $(5,7,50.0000)_{a_8}$	$(7,6,50.0000)_{a_9}$ $(7,6,50.0000)_{a_{10}}$ $(7,6,50.0000)_{a_{11}}$ $(7,6,50.0000)_{a_{12}}$
task	type 1	type 2	type 3
π_1	$(5,4,69.7398)_{a_1}$ $(5,4,69.7398)_{a_2}$ $(5,4,69.7398)_{a_3}$ $(5,4,69.7398)_{a_4}$	$(5,7,49.7398)_{a_5}$ $(5,7,49.7398)_{a_6}$ $(5,7,49.7398)_{a_7}$ $(5,7,49.7398)_{a_8}$	$(7,5,49.7398)_{a_9}$ $(7,6,49.7398)_{a_{10}}$ $(7,6,49.7398)_{a_{11}}$ $(7,6,49.7398)_{a_{12}}$
task	type 1	type 2	type 3
π_2	$(5,4,69.5706)_{a_1}$ $(5,4,69.5706)_{a_2}$ $(5,4,69.5706)_{a_3}$ $(5,4,69.5706)_{a_4}$	$(5,6,49.5706)_{a_5}$ $(5,6,49.5706)_{a_6}$ $(5,7,49.5706)_{a_7}$ $(5,6,49.5706)_{a_8}$	$(7,5,49.5706)_{a_9}$ $(7,6,49.5706)_{a_{10}}$ $(7,6,49.5706)_{a_{11}}$ $(7,5,49.5706)_{a_{12}}$
task	type 1	type 2	type 3
π_3	$(5,4,69.0678)_{a_1}$ $(5,4,69.0678)_{a_2}$ $(5,4,69.0678)_{a_3}$ $(5,4,69.0678)_{a_4}$	$(5,6,49.0678)_{a_5}$ $(5,6,49.0678)_{a_6}$ $(5,7,49.0678)_{a_7}$ $(5,6,49.0678)_{a_8}$	$(7,5,49.0678)_{a_9}$ $(7,6,49.0678)_{a_{10}}$ $(7,6,49.0678)_{a_{11}}$ $(7,5,49.0678)_{a_{12}}$
task	type 1	type 2	type 3
π_4	$(5,3,67.9356)_{a_1}$ $(5,3,67.9356)_{a_2}$ $(5,3,67.9356)_{a_3}$ $(5,3,67.9356)_{a_4}$	$(5,6,47.9356)_{a_5}$ $(5,6,47.9356)_{a_6}$ $(5,7,47.9356)_{a_7}$ $(5,6,47.9356)_{a_8}$	$(7,4,47.9356)_{a_9}$ $(7,6,47.9356)_{a_{10}}$ $(7,6,47.9356)_{a_{11}}$ $(7,5,47.9356)_{a_{12}}$
task	type 1	type 2	type 3
π_5	$(5,3,65.3330)_{a_1}$ $(5,3,65.3330)_{a_2}$ $(5,3,65.3330)_{a_3}$ $(5,3,65.3330)_{a_4}$	$(5,6,45.3330)_{a_5}$ $(5,6,45.3330)_{a_6}$ $(5,7,45.3330)_{a_7}$ $(5,6,45.3330)_{a_8}$	$(7,4,45.3330)_{a_9}$ $(7,6,45.3330)_{a_{10}}$ $(7,6,45.3330)_{a_{11}}$ $(7,5,45.3330)_{a_{12}}$

where $tp_1 = (\pi_1, \{LC, 21; DC, 18; AC, 191\}, +1)$ and the other tp_i, $i = 2, \ldots 5$, are defined similarly according to Table I. The simulation environment is shown in Fig. 8.

In Example 1, since π_1 and π_4 are mutually compatible tasks, the same batch of robots can be used if their capability values satisfy the task requirement. Since π_1 and π_2 are mutually exclusive tasks, the robots that executed π_1 cannot participate in π_2. The experimental results are shown in Fig. 9, and the capability values of robots for each task after execution are listed in Table IX, where π_0 denotes the initial capability values of robots and type 1,2 and 3 are the types of robots. For simplicity, each element in Table IX represents the values of capabilities possessed by robots, e.g., $(5,5,70.0000)_{a_1}$ indicates that the robot a_1 has the capability value of 5 for LC, 5 for DC and 70.0000 for VC.

As can be seen from Fig. 9 and Table IX, Var-CICS can effectively perform the tasks specified by the CaLTL$^{\mathcal{T}}$ formula ϕ with variable capability. And for the motion planning, in this work, the PRM and RRT* are both tried to applied in our method to perform tasks. More details can be found in the experiment video.[1]

To further show that Var-CICS can deal with unexpected robot failures. We continue with the above experiment setup and assume that robots a_1 and a_5 fail after performing task tp_2 and unable to participate in the subsequent tasks. The experiment results are shown in Fig. 10, where Fig. 10 (a)-(e) represent the completion of the task $\phi = \square\lozenge tp_1 \wedge \square\lozenge tp_2 \wedge \square\lozenge tp_3 \wedge \square\lozenge tp_4 \wedge \square\lozenge tp_5$ without robots' failures and Fig. 10 (f)-(j) represent the results with robots' failures. After tp_2, a_1 and a_5 are replaced with other available robots for the subsequent tasks. Therefore, Var-CICS can deal with environmental changes such as robot failures.

Through the above numerical experiments and simulation experiment, we can verify that Var-CICS is **valid** (simulation experiment), **efficient** (numerical experiments) and **scalable** (numerical experiments).

VII. CONCLUSION

This work presents a task allocation framework for heterogeneous multi-robot system under temporal logic specification with inter-task constraints (i.e., unrelated tasks, compatible

[1]Experiment video is available online at: https://youtu.be/9XT6qvz94lY

tasks and exclusive tasks) and variable capabilities (i.e., the invariant, time-dependent, and task-dependent capabilities). To describe heterogeneous multi-robot systems with inter-task constraints and variable capabilities, we extend conventional LTL to capability linear temporal logic CaLTL$^{\mathcal{T}}$. The TB-PDT^{+} is then developed, which encodes the states of Büchi automaton, the system and the task process to represent the exploration progress of task planning and allocation. Based on the TB-PDT^{+}, the Var-CICS is developed, which can continuously expand, explore and prune the TB-PDT^{+} to find feasible solutions. The approach is evaluated based on the analytical analysis and empirical studies. The results of numerical experiments and simulation experiment show that the Var-CICS has is valid and efficient. Besides, the experiments verified that Var-CICS can also deal with unexpected robot failures. Since the output of Var-CICS is a sequence of target positions of the assigned robots, many existing path planning methods can be conveniently plugged in for path planning, which indicates that our framework is highly flexible for various scenarios.

In practice, the operation environments and the task requirements are often unknown in advance or changing, e.g., in a disaster relief scenario, the robots may not have the full knowledge of the environment after disaster and may have no clue about the potential survivors, and thus the task allocations need to account for partially known or unknown environment with dynamic events such as the finding of survivors. Hence, future research will consider leveraging recent advances in deep learning for active inference and autonomous task generation to enable task allocation and intelligent collaboration of heterogeneous robots in dynamic and changing environments. Additionally, it is worth noting that handling robot failures in this work is limited to situations where failures occur before the planning phase. In our future work, we will focus on methods to address unexpected robot failures that arise during task execution.

REFERENCES

[1] M. Ashish, J. Vilela, G. Nejat, and B. Benhabib, "A multirobot path-planning strategy for autonomous wilderness search and rescue," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1784–1797, Sep. 2014.

[2] Z. Zhou, Z. Chen, M. Cai, Z. Li, Z. Kan, and C.-Y. Su, "Vision-based reactive temporal logic motion planning for quadruped robots in unstructured dynamic environments," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 5983–5992, Jun. 2024.

[3] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An autonomous robotic system for three-dimensional collective construction," in *Proc. Robot, Sci. Syst.*, vol. 7, Los Angeles, CA, USA, Jun. 2011, pp. 257–264.

[4] 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.

[5] H. Wang, H. Zhang, L. Li, Z. Kan, and Y. Song, "Task-driven reinforcement learning with action primitives for long-horizon manipulation skills," *IEEE Trans. Cybern.*, vol. 54, no. 8, pp. 4513–4526, Aug. 2024.

[6] Z. Zhou et al., "Local observation based reactive temporal logic planning of human–robot systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 643–655, 2025.

[7] Z. Chen and Z. Kan, "Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications," *Int. J. Robot. Res.*, vol. 44, no. 4, pp. 640–664, Apr. 2025.

[8] 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.

[9] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robot. Auto. Syst.*, vol. 122, Dec. 2019, Art. no. 103289.

[10] 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.

[11] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks," *Automatica*, vol. 159, Jan. 2024, Art. no. 111377.

[12] X. Luo and C. Liu, "Simultaneous task allocation and planning for multi-robots under hierarchical temporal logic specifications," 2024, *arXiv:2401.04003*.

[13] 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, Feb. 2015.

[14] M. Guo and D. V. Dimarogonas, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 797–808, Apr. 2017.

[15] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, Aug. 2016.

[16] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 158–171, Feb. 2012.

[17] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 889–911, 2013.

[18] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Multi-objective search for optimal multi-robot planning with finite LTL specifications and resource constraints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 768–774.

[19] A. Prasad, H.-L. Choi, and S. Sundaram, "Min-max tours and paths for task allocation to heterogeneous agents," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 3, pp. 1511–1522, Sep. 2020.

[20] K. Leahy et al., "Scalable and robust algorithms for task-based coordination from high-level specifications (ScRATCHeS)," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2516–2535, Aug. 2022.

[21] W. Liu, K. Leahy, Z. Serlin, and C. Belta, "Robust multi-agent coordination from CaTL+ specifications," in *Proc. Amer. Control Conf. (ACC)*, May 2023, pp. 3529–3534.

[22] E. Bonnah and K. A. Hoque, "Runtime monitoring of time window temporal logic," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 5888–5895, Jul. 2022.

[23] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Trans. Autom. Control*, vol. 62, no. 7, pp. 3109–3121, Jul. 2017.

[24] 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.

[25] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal logic task planning and intermittent connectivity control of mobile robot networks," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4105–4120, Oct. 2019.

[26] Y. E. Sahin, P. Nilsson, and N. Ozay, "Provably-correct coordination of large collections of agents with counting temporal logic constraints," in *Proc. ACM/IEEE 8th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2017, pp. 249–258.

[27] J. Tumova, A. Marzinotto, D. V. Dimarogonas, and D. Kragic, "Maximally satisfying LTL action planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 1503–1510.

[28] Q. Meng and H. Kress-Gazit, "Automated robot recovery from assumption violations of high-level specifications," in *Proc. IEEE 20th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2024, pp. 4154–4161.

[29] F. Huang, X. Yin, and S. Li, "Failure-robust multi-robot tasks planning under linear temporal logic specifications," in *Proc. 13th Asian Control Conf. (ASCC)*, May 2022, pp. 1052–1059.

[30] S. Kalluraya, B. Zhou, and Y. Kantaros, "Minimum-violation temporal logic planning for heterogeneous robots under robot skill failures," 2024, *arXiv:2410.17188*.

[31] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3559–3564.

[32] 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. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2022, pp. 2110–2117.
[33] S. Kalluraya, G. J. Pappas, and Y. Kantaros, "Resilient temporal logic planning in the presence of robot failures," in *Proc. 62nd IEEE Conf. Decis. Control (CDC)*, Dec. 2023, pp. 7520–7526.
[34] 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.
[35] 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.
[36] D. Gujarathi and I. Saha, "MT: Multi-robot path planning for temporal logic specifications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 13692–13699.
[37] L. Li, Z. Chen, H. Wang, and Z. Kan, "Fast task allocation of heterogeneous robots with temporal logic and inter-task constraints," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4991–4998, Aug. 2023.
[38] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[39] P. Gastin and D. Oddoux, "Fast LTL to büchi automata translation," in *Proc. Int. Conf. Comput. Aided Verif.*, Cham, Swizerland. Springer, 2001, pp. 53–65.
[40] 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.
[41] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[42] 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.*, May 2011, pp. 1478–1483.
[43] 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.

Lin Li received the B.E. degree in mechanical engineering and automation from Hefei University of Technology, Hefei, China, in 2021. She is currently pursuing the Ph.D. degree in control engineering with the University of Science and Technology of China, Hefei. Her current research interests include multi-robot systems and formal methods.

Ziyang Chen received the B.E. degree from Beihang University, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree in control science and engineering with the University of Science and Technology of China, Hefei, China. His current research interests include robotics and multi-agent systems.

Hao Wang received the B.S. degree in mechanical engineering from China University of Mining and Technology, Xuzhou, Jiangsu, China, in 2017, and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2023. His current research interests include motion planning in robotics and deep reinforcement learning.

Zhen Kan (Senior Member, IEEE) received the Ph.D. degree from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2011.

He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests include control theory, formal methods, and robotics. He currently serves on program committees for several internationally recognized scientific and engineering conferences. He is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.