Counting Time Temporal Logic for Multi-Robot Path Planning in Finite Horizons

Peng Lv, *Student Member,* Shaoyuan Li, *Senior Member,* Cristian Mahulea, *Senior Member,* Bruno Denis, Gregory Faraut, *Member,* and Xiang Yin, *Member*

***Abstract*—In this paper, we consider multi-robot path planning problems for high-level tasks with a finite horizon. In many situations, there is a need to *count how many times* a sub-task is satisfied in order to achieve the overall task. However, existing temporal logic languages, such as linear temporal logic, is not efficient in describing such requirements. To address this issue, we propose a new temporal logic language called *Counting Time Temporal Logic* (CTTL) that extends linear temporal logic by explicitly counting the number of times that some tasks are satisfied. To solve the CTTL path planning problem, we propose an efficient integer linear programming-based method to encode task satisfaction. We show that our approach is both sound and complete, while achieving higher efficiency than direct encodings of such requirements. Moreover, we study several variants of the problem. To validate our results, we present several numerical experiments to show the scalability of the proposed approach and a simulation case study of a team of autonomous robots to illustrate the feasibility of the synthesis procedure. Finally, to evaluate the real-world feasibility of our method, we conduct a hardware experiment with two Turtlebot3-Burger mobile robots.**

***Note to Pracitioners*—This work is motivated by a class of task planning problems in which the completion of the overall task depends on the number of times certain sub-tasks are completed. However, the existing approaches does not offer an convenient way to describe such requirements. To address this challenge, this paper proposes a new method to count how many times a task is satisfied. We extend the existing linear temporal logic framework by adding a counting operator, which not only facilitates the expression of the counting requirements but also preserves the ability to describe all tasks that can be expressed by the original framework. We propose an integer linear programming (ILP) method to solve the planning problem and provide a comprehensive process for translating the problem into an ILP formulation. The experimental results have demonstrated the superiority of the proposed algorithm compared with the existing methods.**

***Index Terms*—Counting time temporal logic, multi-robot path planning, integer linear programming**

This work was supported by the National Natural Science Foundation of China (62573291,62173226); by the Science Center Program of National Natural Science Foundation of China under Grant 62188101; and by the ONR Global grant N62909-24-1-2081, and by PID2024-159284NB-I00 funded by MCIN/AEI/10.13039/501100011033, by ERDF “A way of making Europe”, the European Union NextGenerationEU/PRTR, and DGA T64-23R. (Corresponding Author: Xiang Yin)

Peng Lv, Shaoyuan Li and Xiang Yin are with School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China. {lv-peng,syli,yinxiang}@sjtu.edu.cn.

Bruno Denis and Gregory Faraut are with University Research Laboratory in Automated Production, ENS Paris-Saclay, University of Paris-Saclay, Paris 91190, France. {bruno.denis,gregory.faraut}@ens-paris-saclay.fr.

Cristian Mahulea is with Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza 50018, Spain. cmahulea@unizar.es.

I. Introduction

Multi-robot systems have found widespread applications in various fields, including data gathering [14], manufacturing systems [20], autonomous warehouses [1], [2], [12], [35], and environmental surveillance [26], [33]. Traditionally, algorithms for multi-robot coordination problems have focused on synthesizing a trajectory for each robot to meet low-level task requirements, such as obstacle avoidance or target reaching [29], [39]. However, with the rapid development of cyber-physical systems (CPS), many studies have focused on developing algorithms to solve high-level complex tasks in multi-robot systems. These studies often involve formal methods, such as temporal logic, to specify task requirements and ensure correct and efficient coordination among robots [11], [18], [21], [38].

The desired high-level requirements in multi-robot path planning problem can be described using various temporal logic languages. Linear temporal logic (LTL) is a widely used formal language in robotics that provides a natural framework to specify desired properties such as response, safety, liveness, priority, and stability. Many works in multi-robot path planning have utilized LTL as the specification language. For instance, some researchers have focused on finding an optimal infinite trajectory in prefix-suffix form that satisfies an LTL formula [3]–[5], [8], [13], [28], [34], [36], [37], [40]. Others have considered planning problems for probabilistic satisfaction of LTL tasks under transition uncertainties [8], [15]. Additionally, some researchers have used probabilistic computation tree logic (PCTL) as an optimality metric to describe the task [23]. Controller synthesis problems for heterogeneous robots subject to graph temporal logic specifications (GTL) have also been studied [9]. To count the number of robots achieving a task, robust trajectory planning for multi-robot systems is studied under counting temporal logic (cLTL) [30]. However, most of these works focus on path planning problems in an infinite horizon, and the solving processes often depend on specific structural properties of the solutions, such as the prefix-suffix form trajectory.

In real-world planning problems, infinite paths are usually not practical due to limited energy or time constraints. As a result, considerable attention has been paid in the robotics research community on finite-horizon task planning problems. For instance, in [16], the authors proposed a mixed-integer linear programming-based method to control swarm robots to achieve specifications specified by spatial temporal logic (SpaTeL). Moreover, in [24], the authors developed a task batch planning decision tree plus based allocation framework for heterogeneous multi-robot system under capability LTL (CaLTL$^{\mathcal{T}}$). Additionally, in [10], a decentralized and probabilistic algorithm is proposed to control robot teams moving along graph nodes for finite-horizon planning for GTL. In [22], the authors propose an approach to specify tasks and synthesize optimal policies for Markov decision processes under co-safe linear temporal logic (scLTL), which needs to be satisfied within a finite horizon. Furthermore, in [6], a general class of LTL$_f$ is proposed to describe a specification for a finite trajectory, which interprets LTL over finite traces. Based on this interpretation, [19] studies the finite planning problem when the labeling function assigns a set of sets of atomic propositions to each state with the specification being described by LTL$_f$ formulas.

In the field of multi-robot path planning, one usually encounters the formal counting requirements in many applications. For example, the robots may have capacity constraints such that tasks cannot be fulfilled one-time. Also, in manufacturing systems, some production processes needs to be repeated to enhance the quality. Furthermore, for surveillance robots, sometimes they need to inspect some region certain number of times. The common characteristic of these missions is that the completion of the task depends on the frequency of some sub-tasks' completion. Unfortunately, current temporal logic languages, such as LTL, do not offer an convenient way to express these requirements. Therefore, the development of new formal specifications is necessary, which should integrate both temporal logic and counting constraints.

In this paper, we propose a new temporal logic called Counting Time Temporal Logic (CTTL) to address the challenge of efficiently handling tasks that require counting how many times a sub-formula has been satisfied within a finite horizon. CTTL introduces a new temporal operator called "k-until", which requires a sub-task to be satisfied for more than k times before some condition holds. We then investigate the multi-robot path planning problem using the proposed CTTL. Our approach is to encode the dynamics of the robots and CTTL specifications as integer constraints and propose an integer linear programming (ILP)-based method to solve the problem. Although the semantics of standard LTL can also express such a requirement, our experimental results demonstrate that the newly proposed ILP encoding method for CTTL is much more efficient for the purpose of path planning. In summary, the main contributions of our work as as follows. First, we introduce a new operator $\mathcal{U}^k$ in CTTL, which provides a more concise way to capture the counting requirements than the linear temporal logic. Second, we provide a more efficient ILP encoding method for the dynamics of the robot team and the CTTL formula. Under this encoding framework, we can solve the CTTL path planning problem efficiently.

The rest of the paper is organized as follows. Section II propose the syntax and semantics of the CTTL, and formulate the path planning problem. In Section III, we solve the path planning problem based on an equivalent ILP problem. All the experiment results are provided in Section IV, and we conclude the paper in Section V. Part of this work has been published on [25]. In this journal version, we provide more technical details on the implementation of the ILP-based method, Furthermore, two variants of the original problem are investigated and we also conduct new hardware experiments to evaluate the real-word feasibility of our algorithm.

II. Preliminary and Problem Formulation

This section begins by introducing some basic definitions. Then we present the new counting time temporal logic, Finally, we present the problem formulation.

We use $\mathbb{Z}^+$ to denote the set of all positive integers. For any $Z \in \mathbb{Z}^+$, we use $[Z]$ to denote the set of all integers t such that $1 \leq t \leq Z$. Let S be a finite set. We denote by $|S|$ the cardinality of S and S^* the set of all finite sequences over S including the empty string ε. For any sequence $\rho = s(1)s(2)...s(n) \in S^*$, we use $|\rho| = n$ to denote the length of ρ. Moreover, we use $\rho(i)$ to denote the i-th element $s(i)$. For any matrix v, we use v' to denote the transposition of v.

A. System Model

We first define the *deterministic transition system* (DTS) to model the dynamics of robot.

Definition 1. A DTS T is a five-tuple $T = (X, x_1, E, \Pi, L)$, where X is the set of all states, x_1 is the initial state, $E \subseteq X \times X$ is the set of all edges, Π is the set of all atomic propositions and $L : X \to 2^{\Pi}$ is the labeling function.

A *h-path* in T with $h \in \mathbb{Z}^+$ is a finite sequence $\rho = x(1)x(2)\cdots x(h) \in X^*$ such that $x(1) = x_1$ and $\langle x(i), x(i+1)\rangle \in E, \forall i \in [h-1]$. We use P_h to denote the set of all the h-paths in T. In this paper, we consider an robot team $\mathcal{R}$ consisting of $N \in \mathbb{Z}^+$ robots operating synchronously in an environment that in consistent of a set of regions with connectivity constraints. The DTS and h-path set associated with robot R^n, $n \in [N]$, is denoted by $T^n = (X^n, x_1^n, E^n, \Pi, L^n)$ and P_h^n respectively, which means that the robots have different dynamics but with the identical atomic proposition set. Note that this assumption is without loss of generality as we can always define Π as the union of all Π_n. Given N h-paths $\rho_h^n \in P_h^n$ with $n \in [N]$, one for each robot, we use $\mathcal{P}_h = \langle \rho_h^1, \rho_h^2, \cdots, \rho_h^N \rangle$ to denote the team sequence and use $\mathcal{P}_h(i) = \langle \rho_h^1(i), \rho_h^2(i), \cdots, \rho_h^N(i) \rangle$ to denote the i-th element of $\mathcal{P}_h$ with $i \in [h]$. Without loss of generality, we assume that each robot starts from a state with empty proposition, which means that $\forall n \in [N], L^n(x_1^n) \cap \Pi = \emptyset$.

Remark 1. Note that maybe all the robots operate in the same environment, but we still consider different DTS models for each robot, as we want to model their different mobility capabilities. For example, in a factory, the cargo robot operates within the warehouse and equipment rooms, the inspection robot moves through production areas and equipment rooms, and the cleaning robot covers all regions. While their operational areas may overlap in some areas, each robot is subject to distinct mobility constraints.

B. Counting Time Temporal Logic

In this paper, we introduce a new kind of temporal logic called *counting time temporal logics* (CTTL). CTTL is an extension of Linear Temporal Logic (LTL) and is especially useful for specifying and reasoning about the completion times of certain tasks over finite sequences.

Definition 2. A CTTL formula ϕ over a given set of atomic proposition Π is recursively defined as follows:

$$\phi = \top \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \mathcal{U}^k \phi_2, \tag{1}$$

where $\pi \in \Pi$ is an atomic proposition, $k \in \mathbb{Z}^+$ is an index and ϕ_1 and ϕ_2 are CTTL formulas.

The symbols $\top$ (true), $\wedge$ (conjunction) and $\neg$ (negation) above are standard Boolean operators, while $\bigcirc$ (next) and $\mathcal{U}^k$ (k-until) are temporal operators. The above operators can also induce additional operators such as $\phi_1 \vee \phi_2 = \neg(\neg\phi_1 \wedge \neg\phi_2)$ (disjunction), $\phi_1 \to \phi_2 = \neg\phi_1 \vee \phi_2$ (implication), $\Diamond^k \phi = \top \mathcal{U}^k \phi$ (k-eventually) and $\Box^k \phi = \neg \Diamond^k \neg\phi$ (k-always).

CTTL formulas are interpreted over finite sequence from $(2^\Pi)^*$, whose semantics over the team sequence are defined as follows.

Definition 3. Given team sequence $\mathcal{P}_h = \langle \rho_h^1, \rho_h^2, \cdots, \rho_h^N \rangle$, the satisfaction of CTTL formula ϕ by $\mathcal{P}_h$ at instant t with $t \in [h]$, denoted by $\mathcal{P}_h(t) \models \phi$, is defined recursively as follows:

- $\mathcal{P}_h(t) \models \top$;
- $\mathcal{P}_h(t) \models \pi$ iff $\pi \in \bigcup_{j=1}^N L(\rho_h^j(t))$;
- $\mathcal{P}_h(t) \models \phi_1 \wedge \phi_2$ iff $\mathcal{P}_h(t) \models \phi_1$ and $\mathcal{P}_h(t) \models \phi_2$;
- $\mathcal{P}_h(t) \models \neg\phi$ iff $\mathcal{P}_h(t) \not\models \phi$;
- $\mathcal{P}_h(t) \models \bigcirc\phi$ iff $t \le h-1$ and $\mathcal{P}_h(t+1) \models \phi$;
- $\mathcal{P}_h(t) \models \phi_1 \mathcal{U}^k \phi_2$ iff $t \le h-k+1$ and $\exists t_1 < \cdots < t_k, t \le t_1, t_k \le h$, and $\mathcal{P}_h(t_i) \models \phi_2, \forall i \in [k]$ and $\forall t' < t_k, t \le t' \Rightarrow \mathcal{P}_h(t') \models \phi_1$.

If $\mathcal{P}_h(1) \models \phi$, then we say that $\mathcal{P}_h$ satisfies ϕ, written as $\mathcal{P}_h \models \phi$. Certainly, we can also define the satisfaction of ϕ by any concrete robot R^n at t according to the above semantics by defining $N = 1$, where we write $\rho_h^n(t) \models \phi$ for short. Notice that the above semantics for operators $\bigcirc$ and $\mathcal{U}^k$ are not defined in the full horizon $[h]$, but a subset. As it is impossible for the team to satisfy the corresponding formula at other instants. Moreover, note that unlike in LTL on infinite sequence, here $\neg\bigcirc\neg\phi \not\equiv \bigcirc\phi$ [7]. The intuition behind the CTTL semantics are as follows: $\phi_1 \mathcal{U}^k \phi_2$ is satisfied if ϕ_1 is satisfied "until" ϕ_2 is satisfied for at least k times before the last instant (included); $\Diamond^k \phi$ is satisfied if ϕ is satisfied for at least k times before the last instant (included); $\Box^k \phi$ is satisfied if $\neg\phi$ is satisfied for at most $k-1$ times before the last instant (included). Note that when $k = 1$, the semantics for all the above three operators are just the same as the "until", "eventually" and "always" operators in LTL. Therefore, the semantics of CTTL formulas allow us to express constraints on the desired frequency of events or behaviors in a concise manner. Some common examples are as follows:

- visit a for at least 5 times: $\Diamond^5 a$;
- visit a, b and c for at least 4 times in turn: $(\neg b\, \mathcal{U}^4\, a) \wedge (\neg c\, \mathcal{U}^4\, b) \wedge \Diamond^4 c$;
- visit c for at most twice: $\Box^3 \neg c$.

C. Problem Formulation

In the real-world scenario, infinite path planning problems are usually not practical, because that the robots are often constrained by the energy capacity, including battery volume and fuel tank size, which limits their moving ability to a certain distance. Therefore, it is hard for an robot team to follow an infinitely long reference trajectory synthesized from specifications that describe infinite behaviors, such as $\Box\Diamond$ (always eventually) in LTL syntax used in existing works. Instead, these specifications actually dictate the frequency with which tasks need to be completed within the robot team's limited energy. Motivated by the above situation, we investigate the problem of multi-robot path planning in finite sequences under CTTL in this paper, which is formally defined as follows:

Problem 1. Given N robots operating synchronously with dynamics $T^n = (X^n, x_1^n, E^n, \Pi, L^n)$ with $n \in [N]$, a finite time domain $h \in \mathbb{Z}^+$ and a CTTL formula ϕ in forms of (1), synthesize a feasible team sequence $\mathcal{P}_h = \langle \rho_h^1, \rho_h^2, \cdots, \rho_h^N \rangle$ satisfying ϕ, i.e., $\mathcal{P}_h \models \phi$.

Therefore, our objective here is to synthesize a finite horizon plan for the team of multi-robot such that the CTTL formula is satisfied.

III. Synthesis Procedure

We propose an integer linear programming (ILP) based method in this section, which is inspired by the concept of bounded LTL model checking [32]. Our approach involves encoding the dynamics of the robot team and the CTTL specifications as a set of ILP constraints. We then solve the corresponding feasibility problem to obtain a solution. In the following parts, we provide a detailed explanation of our encoding process.

A. Encoding for the Dynamics of $\mathcal{R}$

Given the DTS $T^n = (X^n, x_1^n, E^n, \Pi, L^n)$ describing the dynamics of R^n, we first define the *transition matrix* of T^n as $A^n \in \{0,1\}^{|X^n| \times |X^n|}$, where the (i,j)-th element $A^n(i,j)$ of A^n is defined by

$$A^n(i,j) = \begin{cases} 1, & \text{if } (x_i^n, x_j^n) \in E^n, \\ 0, & \text{otherwise.} \end{cases}$$

Then, we introduce h state binary vectors $v^n(t) = [v_1^n(t), v_2^n(t), \cdots, v_{|X^n|}^n(t)]' \in \{0,1\}^{|X^n|}, \forall t \in [h]$, to represent the state of R^n at time instant t as follows: $\forall i \in [|X^n|]$, we have

$$v_i^n(t) = \begin{cases} 1, & \text{if } R^n \text{ is at } x_i^n \text{ at instant } t, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, based on the above transition matrix A^n and the binary vectors $v^n(t)$, the dynamics of R^n can be captured as the following state constraint equations: $\forall t \in [h]$,

$$\begin{cases} v^n(t+1) \le (A^n)' v^n(t), \\ (\mathbf{1}^n)' v^n(t) = 1, \\ v_1^n(1) = 1, \end{cases} \tag{2}$$

where $\mathbf{1}^n$ is the $|X^n|$ dimensional vector with all elements being 1. Therefore, the first constraint ensures the development of the trajectory of R^n must comply with the transition relation; the second one ensures that R^n can only appear at a state at each instant, while the third one requires that R^n must respect the initial state.

B. Encoding for the CTTL Specifications

Next, we show how to encode the CTTL formula in a recursive way. The basic idea is as follows:

- given CTTL formula ϕ and each time instant $t \in [h]$, introduce a formula satisfaction binary variable $y_\phi(t) \in \{0,1\}$ s.t. $y_\phi(t) = 1 \Leftrightarrow \mathcal{P}_h(t) \models \phi$, to encode the satisfaction of ϕ by the team of all robots at instant t;
- if $y_\phi(1) = 1$, then the CTTL task ϕ is satisfied;
- define $y_{\phi'}(t)$ incrementally for all sub-formula ϕ' of ϕ.

Specifically, for any atomic proposition π, we first introduce the following $N \times h$ individual satisfaction binary variables $z_\pi^n(t)$ with $n \in [N]$ and $t \in [h]$ to encode the satisfaction of π by the R^n at instant t, such that

$$z_\pi^n(t) = \begin{cases} 1, & \text{if } \rho_h^n(t) \models \pi, \\ 0, & \text{otherwise.} \end{cases}$$

Then, for any CTTL formula ϕ, we define the following h formula variables $y_\phi(t)$ with $t \in [h]$ to encode the satisfaction of ϕ by $\mathcal{R}$ at instant t, such that

$$y_\phi(t) = \begin{cases} 1, & \text{if } \mathcal{P}_h(t) \models \phi, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the satisfaction of ϕ by $\mathcal{R}$ can be described by $y_\phi(1) = 1$.

1) Atomic Proposition π for R^n: For every R^n, we introduce a binary vector $\pi^n = [\pi_1^n, \pi_2^n, \cdots, \pi_{|X^n|}^n]'$, where $\forall i \in [|X^n|]$, we have

$$\pi_i^n = \begin{cases} 1, & \text{if } \pi \in L^n(x_i^n), \\ 0, & \text{otherwise,} \end{cases}$$

which means that π^n encodes the satisfaction of π at X^n. Then, the individual variable and the state of the robot should satisfy the following equations, which encode the satisfaction of π by R^n at $t \in [h]$:

$$\begin{cases} (\pi^n)' v^n(t) \geq z_\pi^n(t), \\ (\pi^n)' v^n(t) < z_\pi^n(t) + 1. \end{cases} \tag{3}$$

Note that when R^n is in a state with π at t, the left side of the second equation is 1, which makes $z_\pi^n(t)$ being 1. Otherwise, the left side of the first equation is 0, making $z_\pi^n(t)$ being 0.

2) Atomic Proposition π for $\mathcal{R}$: The formula variable (by the team) and the individual variable (by each single robot) should satisfy the following constraints, which encode the satisfaction of π by $\mathcal{R}$ at $t \in [h]$:

$$\begin{cases} y_\pi(t) \geq z_\pi^i(t), \forall i \in [N], \\ y_\pi(t) \leq \sum_{i=1}^N z_\pi^i(t). \end{cases} \tag{4}$$

Note that if any one of the robot R^n satisfies π at t, then the right side of the first equation is 1, which makes $y_\pi(t)$ being 1. Otherwise, the right side of the second equation is 0, making $y_\pi(t)$ being 0.

Actually, the above equations define the satisfaction of π by $\mathcal{R}$ by the disjunction operation such that $y_\pi(t) = \bigvee_{i=1}^N z_\pi^i(t)$ as defined below.

3) Disjunction $\vee$: For $\mathcal{R}$ with $\phi = \bigvee_{i=1}^I \phi_i$, their relation constraints at $t \in [h]$ are given as follows:

$$\begin{cases} y_\phi(t) \geq y_{\phi_i}(t), \forall i \in [I], \\ y_\phi(t) \leq \sum_{i=1}^I y_{\phi_i}(t). \end{cases} \tag{5}$$

4) Conjunction $\wedge$: For $\mathcal{R}$ with $\phi = \bigwedge_{i=1}^I \phi_i$, their relation constraints at $t \in [h]$ are given as follows:

$$\begin{cases} y_\phi(t) \leq y_{\phi_i}(t), \forall i \in [I], \\ y_\phi(t) \geq 1 - I + \sum_{i=1}^I y_{\phi_i}(t). \end{cases} \tag{6}$$

5) Negation $\neg$: For $\mathcal{R}$ with $\psi = \neg\phi$, their relation constraint at $t \in [h]$ is given as follows:

$$y_\psi(t) = 1 - y_\phi(t). \tag{7}$$

6) Next $\bigcirc$: For $\mathcal{R}$ with $\psi = \bigcirc\phi$, their relation constraints at $t \in [h]$ are given as follows:

$$\begin{cases} y_\psi(t) = y_\phi(t+1), \forall t \in [h-1], \\ y_\psi(h) = 0. \end{cases} \tag{8}$$

Note that the above encoding equations from (5) to (8) for the Boolean and temporal operators are consistent with the encodings in [30], [32]. However, we still present them here for the purpose of completeness.

Next, we give the encoding method for the "k-until" operator. For the convenience of narration, when $\phi = \bigvee_{i=1}^I \phi_i$, we just write $y_\phi(t) = \bigvee_{i=1}^I y_{\phi_i}(t)$ instead of the equations in (5) and treat the "conjunction" operator as well.

7) k-Until $\mathcal{U}^k$: For $\mathcal{R}$ with $\phi = \phi_1 \mathcal{U}^k \phi_2$, their relation constraints are given as follows:

- if $k = 1$, then we have

$$\begin{cases} y_\phi(t) = y_{\phi_2}(t) \vee \Big(y_{\phi_1}(t) \wedge y_\phi(t+1)\Big), \forall t \in [h-1], \\ y_\phi(h) = y_{\phi_2}(h). \end{cases} \tag{9}$$

- if $k > 1$, then we have

$$\begin{cases} y_{\phi^k}(t) = y_{\phi_1}(t) \wedge \Bigg(\Big(y_{\phi^{k-1}}(t+1) \wedge y_{\phi_2}(t)\Big) \vee y_{\phi^k}(t+1)\Bigg), \\ \qquad \forall t \in [h-k], \\ y_{\phi^k}(h-k+1) = y_{\phi_1}(h-k+1) \wedge \Big(y_{\phi^{k-1}}(h-k+2) \wedge \\ \qquad y_{\phi_2}(h-k+1)\Big). \\ y_{\phi^k}(t') = 0, \forall t' \in [h] \setminus [h-k+1], \end{cases} \tag{10}$$

where we define $\phi^k = \phi_1 \mathcal{U}^k \phi_2$.

Notice that when $k = 1$, the encoding (9) for $\mathcal{U}^1$ is the same as the "$\mathcal{U}$" operator in LTL [32]. When $K > 1$, we further define $k-1$ auxiliary formulae from ϕ^{k-1} to ϕ^1. If $\mathcal{R}$

satisfies ϕ^k at $t \in [h-k]$, then the first equation requires that $\mathcal{R}$ must satisfy ϕ_1 at t and further satisfy one of the following conditions: 1) satisfying ϕ_2 at t and continuing to satisfy only ϕ^{k-1} at the next instant; 2) no additional constraints at t, but still needing to satisfy ϕ^k at the next instant. Another point to keep in mind is that the $\mathcal{U}^k$ operator is only defined for instants within $[h-k+1]$ according to Definition 3 and the first equation is only suited for encoding the satisfaction of ϕ^k at $t \leq h-k$. When $t = h-k+1$, the second condition above is not allowed as there is no chance for $\mathcal{R}$ to further satisfy ϕ^k at the next instant $h-k+2$. Finally, as it is impossible for $\mathcal{R}$ to satisfy ϕ^k at other time instants $t' \in [h] \setminus [h-k+1]$, we define $y_{\phi^k}(t') = 0$. Therefore, by the above recursive equations, $y_{\phi^k}(t) = 1$ if and only if there exists at least k instants t_i with $i \in [k]$ between t and h such that $y_{\phi_2}(t_i) = 1$ and before the instant t_k that $y_{\phi_2}(t_k) = 1$, $y_{\phi_1}(t_k)$ has always been 1, which means that there exists at least k instants in $[h] \setminus [t-1]$ such that $\mathcal{R}$ satisfies ϕ_2 and before the last instant, $\mathcal{R}$ always satisfies ϕ_1. Therefore, the above equations (10) are correct and consistent with the semantics of CTTL.

C. Problem Reformulation as an ILP Problem

Given a CTTL formula ϕ, we use $\{v, z, y\}_\phi$ and ILP(ϕ) to denote the sets of all the binary variables and all the constraint equations from (2) to (10) created during the encoding process respectively. Based on $\{v, z, y\}_\phi$ and ILP(ϕ), we now reformulate Problem 1 as the following ILP problem:

$$\begin{aligned} \textbf{Find}: &\quad \{v, z, y\}_\phi \\ \textbf{Subject to}: &\quad \text{ILP}(\phi), \text{ and } y_\phi(1) = 1. \end{aligned} \tag{11}$$

Next, we show that the solutions found by (11) is sound and complete by the following theorem.

Theorem 1. Given $h \in \mathbb{Z}^+$ and a CTTL formula, there exists a solution for Problem 1 if and only if there is a solution for the ILP problem (11).

Proof. ($\Rightarrow$) The completeness of encoding equations from (5) to (9) of CTTL formulae follows the results in [32]. Moreover, the proof of completeness for (10) is also trivial according to Definition 3. This completes the proof of this part.

($\Leftarrow$) The encoding equations (2) ensure that the trajectories of every robot must respect their dynamics and initial conditions. The encoding equations (3) and from (5) to (9) of CTTL formulae are sound according to [32]. Regarding equations (4) and (10), we have proven their soundness by showing that they are correct and consistent with the semantics of CTTL above. Therefore, the constraint $y_\phi(1) = 1$ together with ILP(ϕ) ensures that $\mathcal{P}_h \models \phi$. □

Therefore, if (11) has a solution, then the solution for Problem 1 can be synthesized as follows: $\forall n \in [N], t \in [h], [v_i^n(t) = 1] \Rightarrow [\rho_h^n(t) = x_i^n]$. Note that, in this work, our main objective is to find a *feasible solution*. In other words, optimality is not explicitly considered here. A naive approach to handle the optimal planning problem is to add a new optimization objective, for example, the total cost of the plan, in addition to the Boolean constraints in (11). This direct extension does not change our technical approach as it is still an ILP problem. However, it may make the ILP problem more difficult to solve since the original problem only has constraints and the optimization objective is essentially constant.

D. Variants of the Problem

In this section, we enumerate two potential variants and extensions to the above problems and engage in a discussion on implementing these extensions. We consider the following scenario such that the solution to (11) has been solved and the robot team starts to move, but during the execution of the robot team, some accidents occurred, called *environment change* or *robot fault*. We would like to study the trajectory re-planning problems for the above two types of accidents online taking into account the existing completed trajectories, rather than re-planing from the beginning, and the specific process is shown in detail as the following two parts. Here, for any instant $t \in [h]$ and any number $n \in [N]$, we use $\boldsymbol{v}^n(t)$ to denote the binary vector for R^n in the re-planning process and use $\boldsymbol{v}_i^n(t)$ to denote the specific element of $\boldsymbol{v}^n(t)$.

1) Environmental Change Cases: Suppose that during the execution of the robot team, the dynamics of R^n have changed to $T_r^n = (X^n, x_1^n, E_r^n, \Pi, L^n)$ at some instant $t \in [h] \setminus \{0\}$, which means that transition relations of R^n have changed. Given $x, x' \in X^n$ and $(x, x') \in E^n$, before t, R^n can arrive at x' from x. However, if $(x, x') \notin \hat{E}_r^n$, then the above transition will become infeasible for R^n after t. We often encounter the above scenarios such that a robot needs to deliver some files to the finance office and it will reach there by passing through the meeting room, but on its way forward, an emergency meeting suddenly convenes and the door is closed. Then, how should we re-plan their trajectories?

Specifically, we supplement some additional constraints and adjust some constraint equations for the ILP problem (11), which are shown by the following two items.

- In order to take into account the existing completed trajectories of the robot team until instant t, we add some constraint equations as follows: $\forall 0 \leq t' \leq t, m \in [N], \boldsymbol{v}^m(t') = v^m(t')$, which means that the synthesis trajectories in the re-planning process should comply with the previous ones until t.
- As the dynamics of R^n change after t, we update the constraint equations in (2) for R^n to incorporate with the new dynamics A_r^n, which denotes the transition matrix corresponding to T_r^n, as follows: $\forall t < t' \leq h, \boldsymbol{v}^n(t') \leq (A_r^n)' \boldsymbol{v}^n(t'-1)$.

Note that if the dynamics of multiple robots change instead of just one, we only need to repeat the above process for every robot at the corresponding instant and the problem can be solved.

2) Robot Fault Cases: Suppose that during the execution of the robot team, robot R^n is broken at some instant $t \in [h] \setminus \{0\}$, which means that the R^n cannot contribute to the satisfaction of the task in the remaining part of the trajectory after time instant t. Then, how should we re-plan the trajectories of the remaining robots to further finish the task?

Specifically, we make two changes for the ILP problem (11) as shown in follows.

- We also add the following constraint equations, $\forall 0 \leq t' \leq t, m \in [N], \boldsymbol{v}^m(t) = v^m(t)$, to ensure that the synthesis trajectories in the re-planning process must comply with the previous ones until t.
- As R^n failed at t, it can not collect any atomic propositions from $t+1$. Therefore, we directly move it to a state without any atomic propositions from $t+1$, that is the initial state x_1^n. To achieve this, we update equation (2) from $t+1$ for R^n as follows: $\forall t < t' \leq h, \boldsymbol{v}_1^n(t') = 1$.

Suppose that R^n arrives at x' at instant t. Even though it may be infeasible for R^n to move from x' to x_1^n and move from x_1^n to x_1^n, which means that it might be the case such that $(x', x_1^n), (x_1^n, x_1^n) \notin E^n$, we do not care about this, as R^n is already broken, it do not need to comply with its dynamics anymore.

Moreover, in the above two cases, it might be the situation that the robot team can not achieve ϕ in the original horizon h after the accident, such that when R^n got broken at t, the remaining steps $h-t$ are not enough for the remaining robots to further finish ϕ. In this case, we can directly extend h to a larger one h' and supplement constraints.

Remark 2. It is important to notice that, mathematically speaking, the proposed CTTL is no more expressive than the standard LTL. In other words, we can also express the counting time requirement by LTL formulae. Specifically, for CTTL formula $\phi = (\phi_1 \mathcal{U}^k \phi_2)$, we construct an LTL formula ϕ' as follows:

$$\phi' = \langle (\phi_1 \mathcal{U} (\phi_1 \wedge \phi_2 \wedge \bigcirc \rangle^{k-1} + \langle (\phi_1 \mathcal{U} \phi_2) \rangle + \langle) \rangle^{2k-2},$$

where $\langle * \rangle^{k-1}$ represents writing $*$ for $k-1$ times and $A+B$ represents writing B after A. For example, for $\phi = \phi_1 \mathcal{U}^2 \phi_2$, we can write its equivalent LTL expression by

$$\phi' = (\phi_1 \mathcal{U} (\phi_1 \wedge \phi_2 \wedge \bigcirc (\phi_1 \mathcal{U} \phi_2))).$$

Also for $\phi = \phi_1 \mathcal{U}^3 \phi_2$, we can write its equivalent LTL

$$\phi' = (\phi_1 \mathcal{U} (\phi_1 \wedge \phi_2 \wedge \bigcirc (\phi_1 \mathcal{U} (\phi_1 \wedge \phi_2 \wedge \bigcirc (\phi_1 \mathcal{U} \phi_2))))).$$

Therefore, any trajectory synthesis problem for CTTL formula can be reduced to an equivalent problem for LTL formula. However, this reduction process results in a formula that is linearly larger in size than the original CTTL formula. In the following section, we will also provide numerical results that demonstrate the advantages of our CTTL language over the equivalent LTL formula.

Remark 3. We conclude this section by discussing the complexity of the overall synthesis algorithm. Despite the fact that the computational complexity of solving ILP problems is known to be NP-complete, there exist several effective heuristic algorithms that can significantly alleviate the computational burden, such as [17]. In general cases, there are $\mathcal{O}\big(hN|\hat{X}| + h(|\phi| + n\hat{k})\big)$ decision variables and constraint equations for (11) (very roughly estimated), where h is the length of planning domain, N is the number of robots, $|\hat{X}|$ is the number of states of the largest DTS, $|\phi|$ is the length of the CTTL formula ϕ, n is the number of $\mathcal{U}^k$ in ϕ and $\hat{k}$ is the largest value among all the indexes of k-until operators.

IV. Experiment Results

In this section, we provide a set of experiments to illustrate our results. In Section IV-A, we conduct a set of numerical experiments to show the scalability of our encoding method and the efficiency of the CTTL language. In Section IV-B, we provide a simulation experiment on a 7×5 grid world to further illustrate our synthesis procedure. Finally, in Section IV-C, we conduct a hardware experiment to evaluate the real-world feasibility of our method. All simulations are implemented by `Python 3.7` and robot simulation platform `V-REP 4.2.0` on a PC with 64 cores with 3.30 GHz processors and 64 GB of RAM using PYTHON-MIP [31] to setup the ILP problem and GUROBI [27] as the underlying ILP solver. The robots used in the hardware experiment are two `Turtlebot3-Burger` mobile robots. Moreover, we use a `Vicon` Motion Capture Systems to localize each robot and use a low-level PID controller to track the planned high-level trajectory from one grid to another. All the simulation and experimental videos in later sections are available online[1].

A. Numerical Experiments

First, we demonstrate the scalability of our encoding method by varying several parameters, such as the size of DTS, the number of robots and the length of planning horizon. We also illustrate the efficiency of the CTTL by comparing it with the LTL on reasoning about missions describing completion times of some tasks. In each experiment, the DTSs T^n are generated from Erdös-Rényi graphs with edge probability being 0.75 and the number of every atomic proposition is set to $\lfloor \frac{|X^n|}{20} \rfloor$ with their locations being chosen randomly from X^n, which means that the accessible regions of each robot are considered independently and they do not overlap. For each set of parameters, we repeated the experiment for 20 times and recorded the average value of all experimental data. Moreover, in this part, we consider that all the accessible regions of all robots do not intersect.

We start by investigating the effect of the number of robots N on runtime. We set $|X^n| = 50$ for all $n \in [N]$, $h = 20$. Consider the following task:

$$\phi = (\neg b \mathcal{U}^{\frac{N}{5}} a) \wedge (\neg c \mathcal{U}^{\frac{N}{5}} b) \wedge \Diamond^{\frac{N}{5}} c \wedge \Box^{\frac{N}{5}} \neg d, \qquad (12)$$

which requires that visit a, b and c for at least $\frac{N}{5}$ times each in order and visit d at most $\frac{N}{5} - 1$ times. We increase N from 10 to 50 and the statistics are displayed in Table 1. As can be seen, all the three parameters increase linearly with the increase of N.

Then, we investigate the effect of system size $|X^n|$. We still use the above task, but set $N = 10$ and $h = 20$. The statistics are shown in Table 2. We can see that both the number of variables and constraints still linearly increase with the increase of $|X^n|$. However, the solving time is significantly

[1]https://www.youtube.com/@ILagrange-j1s

Table 1: Statistics for different number of robots.

N	10	20	50
variable	11362	22500	55674
constraint	13840	27458	67592
time (sec)	4.64	9.03	22.73

Table 2: Statistics for different size of systems.

$\|X^n\|$	50	100	200
variable	11362	21362	41362
constraint	13840	23840	43840
time (sec)	4.64	18.69	79.76

affected by the system size, which seems to exhibit polynomial growth as $|X^n|$ increases.

Next, we further study the effect of the length of planning horizon h on runtime. In this experiment, we fix $|X^n| = 50$ and $N = 20$ and increase h from 20 to 100 to see the results. Besides, note that just increasing the planning horizon might result in trivial solutions, such that many steps in the trajectory stay in place. Therefore, we also simultaneously increase the complexity of the CTTL formula. Specifically, we increase h from 20 to 50 and then to 100. Correspondingly, we simultaneously change all the parameters in (12) from $\frac{N}{5}$ to $\frac{N}{2}$ and then to $\frac{N}{1}$. The statistics are displayed in Table 3. As expected, all the three parameters increase linearly with the increase of h.

Finally, in order to demonstrate the efficiency of our CTTL language, we further carry out two additional experiments to compare the trends of the above parameters when the specification is given in the form of CTTL formula versus their equivalent LTL formula as mentioned in Remark 2. Some parameters are given as follows: $N = 10, |X^n| = 50, \forall n \in [N]$ and $h = 100$. Note that, we use equations (5)-(10) to encode the satisfactions of the CTTL formula and use equations (5)-(9) to encode the satisfactions of the equivalent LTL formula.

In the first experiment, consider the following CTTL task:

$$\phi = \Diamond^k a \wedge \Diamond^k b \wedge \Diamond^k c \wedge \Diamond^k d.$$

We increase parameter k above from 10 to 50. For each value, we construct the equivalent LTL task and use the encoding method proposed in this paper to solve the two problems. The statistics are recorded in Table 4. From Table 4, we can see that regardless of the value of k, the average value of each parameter solved based on encoding CTTL is always smaller than that based on encoding LTL. And as the complexity of the specification increases, the difference between the values

Table 3: Statistics for different length of planning horizon.

h	20	50	100
variable	22500	58944	126884
constraint	27458	76742	180482
time (sec)	9.03	24.18	59.79

Table 4: Statistics on comparative experiments between CTTL and LTL by changing k.

	k	10	25	40	50
CTTL	variable	62104	72004	80104	84504
	constraint	85904	115604	139904	153104
	time (sec)	42.78	53.33	77.25	80.88
LTL	variable	71356	95296	119236	135196
	constraint	110180	175880	241580	285380
	time (sec)	43.13	81.92	173.48	217.22

Table 5: Statistics on comparative experiments between CTTL and LTL by another way.

	ϕ	ϕ_1	ϕ_2	ϕ_3	ϕ_4
CTTL	variable	58701	67302	75903	84504
	constraint	76601	102102	127603	153104
	time (sec)	30.72	54.42	67.68	80.88
LTL	variable	71449	92698	113947	135196
	constraint	109745	168290	226835	285380
	time (sec)	45.83	96.73	134.95	217.22

of the two languages becomes more prominent. In other words, the rate at which the parameters increase for CTTL is much lower than that for LTL.

In order to further evaluate the scalability of both languages with respect to the complexity of the specification, we conducted the second experiment using a different approach to increase the complexity. We consider the following four tasks:

- $\phi_1 = \Diamond^{50} a$;
- $\phi_2 = \Diamond^{50} a \wedge \Diamond^{50} b$;
- $\phi_3 = \Diamond^{50} a \wedge \Diamond^{50} b \wedge \Diamond^{50} c$;
- $\phi_4 = \Diamond^{50} a \wedge \Diamond^{50} b \wedge \Diamond^{50} c \wedge \Diamond^{50} d$

We re-conduct the above experiments and the statistics are recorded in Table 5. The data clearly demonstrate the advantages of using the CTTL language once again.

These results are as expected since the equivalent LTL formula is always combinatorially much longer than the CTTL formula, as stated in Remark 2. Therefore, these numerical results demonstrate the scalability of our encoding method and the advantage of the CTTL language.

B. Simulation Experiments

In this part, we conduct a simulation for multi-robot path planning. Consider a factory as depicted in Fig 1. The factory can be divided into 35 grid regions, which can be further clarified into two parts, inside the building (the red regions) and outside the building (the green and blue regions). For the convenience of narration, we encode the above grids into 35 states $\{x_i : i \in [35]\}$ from top to bottom and from left to right. There are twelve grids of interest: x_1 (living quarters), x_6 (lake), x_{10} (workshop 1), x_{11} (finance office), x_{13} (lounge), x_{20} (canteen), x_{23} (warehouse), x_{24} (workshop 2), x_{25} (workshop 3), x_{26} (toilet), x_{30} (supermarket) and x_{35} (fire location).

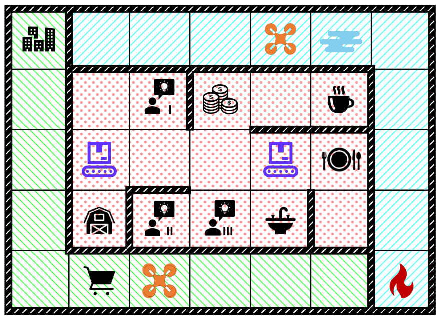


Fig. 1: The topology of the factory with two UGVs and two UAVs.

Two UGVs with G_0 being initially placed at x_{16} and G_1 at x_{19}, and two UAVs with A_0 at x_5 and A_1 at x_{31}, move in this factory. For safety reasons, G_0 and G_1 are only allowed to move inside the building, while A_0 and A_1 can only inspect outside the building to deal with the emergencies. Moreover, to ensure efficient energy usage and prevent any potential collisions, each UAV is confined to operate within a designated area, the green part for A_0, while the blue part for A_1. Both UGVs are capable of transporting parcels between these grids, while the UAVs have the added capability of addressing other incidents in addition to parcel delivery, such as extinguishing fires by fetching water. At any given moment, these four robots can either choose to move from their current location to an adjacent grid synchronously, or remain in place for one unit of time to unload a parcel or address a fire within their designated regions. It is assumed that they can immediately unload a parcel or extinguish an ignition source upon arrival, and two UGVs cannot unload at the same locations simultaneously due to the space constraints.

Now consider the following tasks:

- due to insufficient parcels left, the UGVs must replenish before delivery by proceeding to the warehouse;
- base on the order and urgency of transportation requests, the two UGVs should first deliver three parcels to workshop 1, followed by three parcels to workshop 3 and finally two parcels to workshop 2. Additionally, there are transportation requests for three parcels from the toilet, two parcels from the finance office, and two parcels from the lounge respectively;
- two UGVs should never enter canteen for food safety;
- living quarters purchase two pieces of goods from supermarket and UAVs need to transport them;
- UAVs discover two ignition sources at fire location, but they need to first go to lake to fetch water and then proceed to the fire location to put out the fire.

By the CTTL language, the above tasks can be formulated as the following CTTL formula:

$$\begin{aligned}\phi = &(\neg x_{10}\mathcal{U}^1 x_{13}) \wedge (\neg x_{25}\mathcal{U}^3 x_{10}) \wedge (\neg x_{24}\mathcal{U}^3 x_{25})\\ &\wedge\Diamond^2 x_{24} \wedge \Diamond^2 x_{11} \wedge \Diamond^2 x_{13} \wedge \Diamond^3 x_{26} \wedge \Box^1 \neg x_{20}\\ &\wedge(\neg x_1\mathcal{U}^1 x_{30}) \wedge \Diamond^2 x_1 \wedge (\neg x_{35}\mathcal{U}^1 x_6) \wedge \Diamond^2 x_{35}.\end{aligned} \tag{13}$$

We use the encoding methods proposed in this paper to formulate the optimization problem with the planning horizon being 20, which has 5186 optimization variables and 9038 constraints and is solved in 2.78 s. The simulation trajectories for the two UGVs are shown in Figure 2(a). For the trajectories of the two UAVs, we refer the readers to the simulation video. Specifically, the trajectories for the above four robots are as follows:

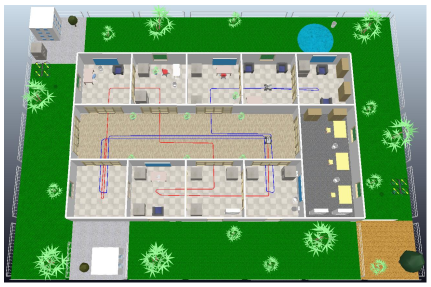
- $G_0: \ x_{16} \to x_{23} \to x_{16} \to x_9 \to x_{10} \to x_{10} \to x_{10} \to x_{17} \to x_{18} \to x_{25} \to x_{25} \to x_{25} \to x_{24} \to x_{24} \to x_{25} \to x_{26} \to x_{26} \to x_{19} \to x_{19} \to x_{19}$;
- $G_1: \ x_{19} \to x_{18} \to x_{17} \to x_{16} \to x_{23} \to x_{16} \to x_{17} \to x_{18} \to x_{19} \to x_{26} \to x_{19} \to x_{18} \to x_{11} \to x_{11} \to x_{12} \to x_{13} \to x_{13} \to x_{12} \to x_{12} \to x_{12}$;
- $A_0: \ x_5 \to x_4 \to x_3 \to x_2 \to x_3 \to x_4 \to x_5 \to x_4 \to x_5 \to x_5 \to x_6 \to x_7 \to x_{14} \to x_{21} \to x_{28} \to x_{35} \to x_{28} \to x_{35} \to x_{28} \to x_{28}$;
- $A_1: \ x_{31} \to x_{32} \to x_{33} \to x_{34} \to x_{33} \to x_{32} \to x_{31} \to x_{32} \to x_{31} \to x_{31} \to x_{30} \to x_{29} \to x_{22} \to x_{15} \to x_8 \to x_1 \to x_8 \to x_1 \to x_8 \to x_8$.

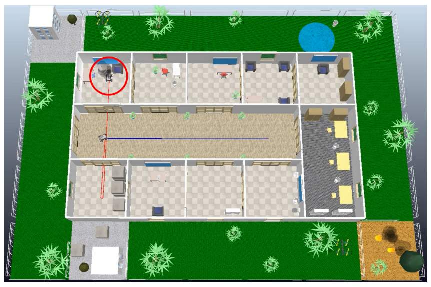
Note that the above task is satisfied by this solution with synchronous execution. Specifically, after replenishing parcels from x_{23}, G_0 first delivers three parcels to x_{10}, then three parcels to x_{25}, then two parcels to x_{24} and finally two parcels to x_{26}. As for G_1, before G_0 arriving x_{26}, G_1 has already delivered a parcel to x_{26} and then it further delivers two parcels to x_{11} and x_{13} each. Moreover, for A_0, it first proceeds to x_6 to fetch water. Then it goes to x_{35} twice to extinguish two ignition sources. Finally, for A_1, it picks up two pieces of goods from the supermarket and unloads them at x_1 twice to complete the task.

Now consider that when G_0 arrives x_9, its engine gets broken as shown in Figure 2(b), which means that it can not continue with its trajectories synthesized before anymore. Therefore, the re-planning procedure needs to be carried out. We still choose the planning horizon being 20, but no solution can be solved, which means that the robot team without G_0 can not further achieve ϕ within the remaining 16 steps. On account of this, we extend h to 30 and the simulation trajectories are shown in Figure 2(c) with the specific trajectories for the above four robots shown as follows:

- $G_0: \ x_{16} \to x_{23} \to x_{16} \to x_9 \to$ broken;
- $G_1: \ x_{19} \to x_{18} \to x_{17} \to x_{16} \to x_{23} \to x_{16} \to x_9 \to x_{10} \to x_{10} \to x_{10} \to x_{17} \to x_{18} \to x_{19} \to x_{26} \to x_{26} \to x_{26} \to x_{25} \to x_{25} \to x_{25} \to x_{24} \to x_{24} \to x_{25} \to x_{18} \to x_{11} \to x_{11} \to x_{12} \to x_{13} \to x_{13} \to x_{12} \to x_{12}$;
- $A_0: \ x_5 \to x_4 \to x_3 \to x_2 \to x_3 \to x_4 \to x_4 \to x_5 \to x_4 \to x_5 \to x_6 \to x_7 \to x_7 \to x_{14} \to x_{21} \to x_{28} \to x_{35} \to x_{28} \to x_{35} \to x_{28} \to x_{28} \to x_{28} \to x_{28} \to x_{28} \to x_{21} \to x_{21} \to x_{21} \to x_{14} \to x_{14} \to x_{14}$;
- $A_1: \ x_{31} \to x_{32} \to x_{33} \to x_{34} \to x_{33} \to x_{32} \to x_{31} \to x_{32} \to x_{32} \to x_{31} \to x_{32} \to x_{31} \to x_{30} \to x_{29} \to x_{29} \to x_{22} \to x_{15} \to x_8 \to x_1 \to x_8 \to x_1 \to x_8 \to x_{15} \to x_{15} \to x_{15} \to x_{15} \to x_{15} \to x_{15} \to x_{15} \to x_{15}$.

The simulation video for the re-planning process is also available. Note that as G_0 got broken, G_1 has to replace G_0 to complete the task that G_0 should have completed, which


(a) Simulation trajectories before re-planning.


(b) The time instant when G_0 gets broken.

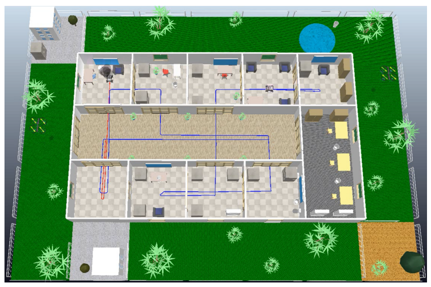
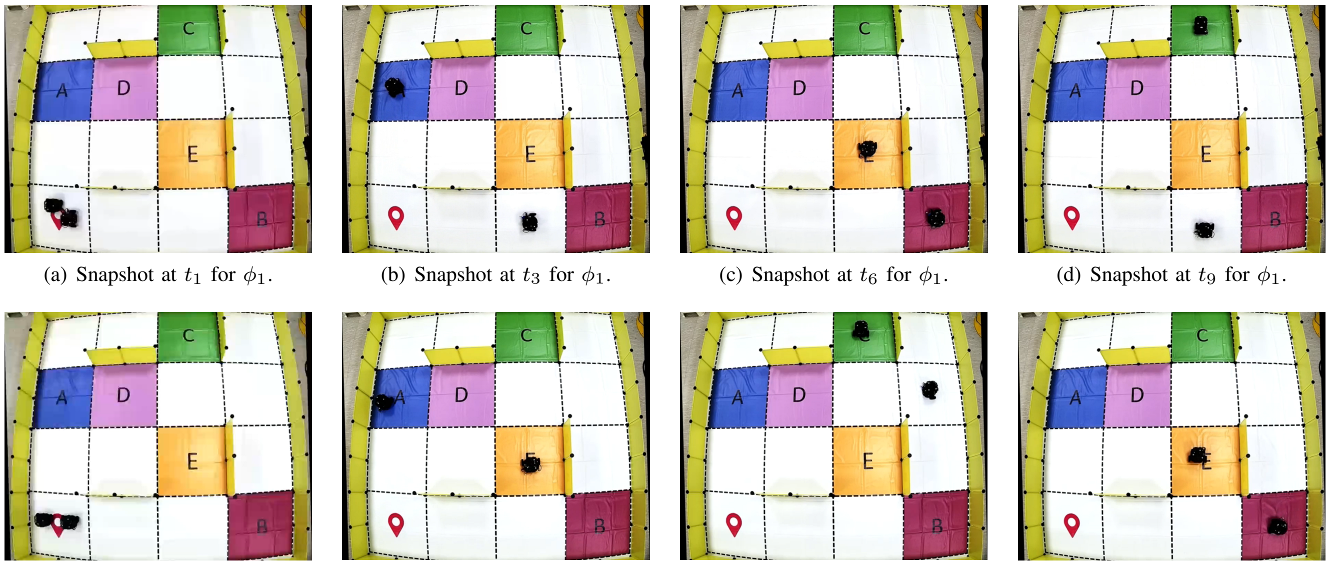

(c) Simulation trajectories after re-planning.

Fig. 2: Simulation trajectories for the two UGVs.



(a) Snapshot at t_1 for ϕ_1. (b) Snapshot at t_3 for ϕ_1. (c) Snapshot at t_6 for ϕ_1. (d) Snapshot at t_9 for ϕ_1.

(e) Snapshot at t_1 for ϕ_2. (f) Snapshot at t_4 for ϕ_2. (g) Snapshot at t_9 for ϕ_2. (h) Snapshot at t_{11} for ϕ_2.

Fig. 3: Experiment scene and snapshots of the solution for ϕ_1 and ϕ_2. Note that in (a) and (e), two mobile robots R_1 and R_2 with optical sensor are placed in the lower left corner.

causes the trajectory of G_1 to change to a very long trajectory as shown by the blue one.

C. Hardware Demonstration

In this section, we conduct two hardware experiments to validate the real-world feasibility of our method. These experiments take place in an indoor 4×4 grid motion capture environment as illustrated in Figure 3(a). We consider discrete and region-level trajectories, which are fully recorded in the experimental videos for the two experiments, and these trajectories can be further translated into individual robot movement plans using established low-level path planning algorithms.

In the first experiment, we consider the following task

$$\phi_1 = (\neg B \mathcal{U}^1 A) \wedge (\neg C \mathcal{U}^2 B) \wedge \Diamond^2 C \wedge \Diamond^2 E \wedge \Box^1 \neg D,$$

which requires that C and E should be visited for at least twice respectively, A should be visited for at least once before B being visited, B should be visited for at least twice before C being visited and D should never be visited.

We formulate the optimization problem using our encoding methods with the length of the planning horizon being 10, which has 697 optimization variables and 1212 constraints and is solved in 0.09 s. Snapshots of one of the solution for ϕ_1 are shown in Figure 3(a)-3(d). Specifically, R_1 first goes to A and R_2 waits for it before B in Figure 3(b). After R_1 visiting A once, R_2 goes to visit B twice and R_1 proceeds to visit E once while avoiding D in Figure 3(c). Finally, R_1 leaves E and goes to visit C twice and R_2 leaves B to visit E once in Figure 3(d). Therefore, ϕ_1 has been finished by the synthesized trajectories.

In order to verify the adaptability of our method to different task, we further give a more complicate task as follows:

$$\phi_2 = (\neg B \mathcal{U}^3 A) \wedge \Diamond^2 B \wedge \Diamond^4 C \wedge \Diamond^4 E \wedge \Box^1 \neg D,$$

which requires that B, C and E should be visited for at least twice, four times and four times respectively, A should be visited for at least three times before B being visited, and D should never be visited.

We formulate the optimization problem using our encoding methods with the length of the planning horizon being 15, which has 1183 optimization variables and 2266 constraints and is solved in 0.18 s. Snapshots of the solution for ϕ_2 are shown in Figure 3(e)-3(h). Specifically, R_1 first goes to visit A three times and R_2 goes to E once in Figure 3(f). Next,

R_2 leaves E to visit C twice and then leaves for B and R_1 also proceeds to visit C twice while avoiding D in Figure 3(g). Finally, R_1 leaves C to visit E three times and R_2 goes to visit B twice in Figure 3(h). Therefore, ϕ_2 has also been finished by R_1 and R_2.

V. Conclusion

In this paper, we proposed a new temporal logic language for specifying finite horizon tasks called the *counting time temporal logic* (CTTL). Compared with the standard LTL formulae in finite horizon, CTTL allows us to directly specify the number of completions for some sub-formulae. We then solved the multi-robot path planning problem for CTTL specifications using integer linear programming techniques. The efficiency of CTTL in describing such counting tasks and its feasibility in real-world scenario have been demonstrated by experiment results. Moreover, two variants of the basic problem have also been investigated. In the future, we plan to study the robust planning problem, where some robots may be subject to execution delays. In this work, we mainly focus on finding feasible plans without considering numerical optimality criteria. Although our approach can be directly extended to the qualitative setting, finding optimal plans using ILP is generally time-consuming. How to find optimal plans satisfying CTTL specifications more efficiently is also an interesting future direction.

References

[1] F Basile, P Chiacchio, and E Di Marino. An auction-based approach to control automated warehouses using smart vehicles. *Control Engineering Practice*, 90:285–300, 2019.

[2] Kai Cai. Warehouse automation by logistic robotic networks: a cyber-physical control approach. *Frontiers of Information Technology & Electronic Engineering*, 21(5):693–704, 2020.

[3] Ziyang Chen, Mingyu Cai, Zhangli Zhou, Lin Li, and Zhen Kan. Fast motion planning in dynamic environments with extended predicate-based temporal logic. *IEEE Transactions on Automation Science and Engineering*, 2024.

[4] Ziyang Chen, Zhangli Zhou, Shaochen Wang, Jingsong Li, and Zhen Kan. Fast temporal logic mission planning of multiple robots: A planning decision tree approach. *IEEE Robotics and Automation Letters*, 2024.

[5] Bohan Cui, Feifei Huang, Shaoyuan Li, and Xiang Yin. Robust temporal logic task planning for multirobot systems under permanent robot failures. *IEEE Transactions on Control Systems Technology*, 2024.

[6] Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Twenty-Third International Ioint conference on Artificial Intelligence*, pages 854–860, 2013.

[7] Giuseppe De Giacomo, Moshe Y Vardi, et al. Synthesis for LTL and LDL on finite traces. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1558–1564, 2015.

[8] Xuchu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257, 2014.

[9] Franck Djeumou, Zhe Xu, Murat Cubuktepe, and Ufuk Topcu. Probabilistic control of heterogeneous swarms subject to graph temporal logic specifications: A decentralized and scalable approach. *IEEE Transactions on Automatic Control*, 2022.

[10] Franck Djeumou, Zhe Xu, and Ufuk Topcu. Probabilistic swarm guidance subject to graph temporal logic specifications. In *Robotics: Science and Systems*, 2020.

[11] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[12] Maria Pia Fanti, Agostino M Mangini, Giovanni Pedroncelli, and Walter Ukovich. A decentralized control strategy for the coordination of agv systems. *Control Engineering Practice*, 70:86–97, 2018.

[13] Meng Guo and Dimos V Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[14] Meng Guo and Michael M Zavlanos. Multirobot data gathering under buffer constraints and intermittent communication. *IEEE Transactions on Robotics*, 34(4):1082–1097, 2018.

[15] Meng Guo and Michael M Zavlanos. Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Transactions on Automatic Control*, 63(12):4051–4066, 2018.

[16] Iman Haghighi, Sadra Sadraddini, and Calin Belta. Robotic swarm control from spatio-temporal specifications. In *55th IEEE Conference on Decision and Control (CDC)*, pages 5708–5713. IEEE, 2016.

[17] Taoan Huang, Aaron Ferber, Yuandong Tian, Bistra Dilkina, and Benoit Steiner. Local branching relaxation heuristics for integer linear programs. *arXiv preprint arXiv:2212.08183*, 2022.

[18] Yiannis Kantaros and Michael M Zavlanos. Distributed intermittent connectivity control of mobile robot networks. *IEEE Transactions on Automatic Control*, 62(7):3109–3121, 2016.

[19] Takuma Kinugawa and Toshimitsu Ushio. Hyper-labeled transition system and its application to planning under linear temporal logic constraints. *IEEE Control Systems Letters*, 6:2437–2442, 2022.

[20] Ilya Kovalenko, Dawn Tilbury, and Kira Barton. The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, 86:105–117, 2019.

[21] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[22] Bruno Lacerda, David Parker, and Nick Hawes. Optimal and dynamic planning for markov decision processes with co-safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1511–1516. IEEE, 2014.

[23] Morteza Lahijanian, Sean B Andersson, and Calin Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics*, 28(2):396–409, 2011.

[24] Lin Li, Ziyang Chen, Hao Wang, and Zhen Kan. Task allocation of heterogeneous robots under temporal logic specifications with inter-task constraints and variable capabilities. *IEEE Transactions on Automation Science and Engineering*, 2025.

[25] Peng Lv, Shaoyuan Li, and Xiang Yin. Multi-agent path planning for finite horizon tasks with counting time temporal logics. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 2025–2030. IEEE, 2024.

[26] Sina Sharif Mansouri, Christoforos Kanellakis, Emil Fresk, Dariusz Kominiak, and George Nikolakopoulos. Cooperative coverage path planning for visual inspection. *Control Engineering Practice*, 74:118–131, 2018.

[27] Gurobi Optimization. Gurobi optimizer reference manual; gurobi optimization. *Inc.: Houston, TX, USA*, 2016.

[28] Jiming Ren, Haris Miller, Karen M Feigh, Samuel Coogan, and Ye Zhao. Ltl-d*: Incrementally optimal replanning for feasible and infeasible tasks in linear temporal logic specifications. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4495–4502. IEEE, 2024.

[29] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. Implan: Scalable incremental motion planning for multi-robot systems. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[30] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Multirobot coordination with counting temporal logics. *IEEE Transactions on Robotics*, 36(4):1189–1206, 2019.

[31] Haroldo G Santos and T Toffolo. Mixed integer linear programming with python. *COINOR Computational Infrastructure for Operations Research*, 2020.

[32] Viktor Schuppan, Timo Latvala, Tommi Junttila, Keijo Heljanko, and Armin Biere. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2, 2006.

[33] Jongho Shin, Dongjun Kwak, and Taehyung Lee. Robust path control for an autonomous ground vehicle in rough terrain. *Control Engineering Practice*, 98:104384, 2020.

[34] Stephen L Smith, Jana Tumova, Calin Belta, and Daniela Rus. Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 30(14):1695–1708, 2011.

[35] Yuta Tatsumoto, Masahiro Shiraishi, Kai Cai, and Zhiyun Lin. Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. *Control Engineering Practice*, 81:97–104, 2018.

[36] Daiying Tian, Hao Fang, Qingkai Yang, and Yue Wei. Decentralized motion planning for multiagent collaboration under coupled LTL task specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[37] Alphan Ulusoy, Stephen L Smith, and Calin Belta. Optimal multi-robot path planning with LTL constraints: guaranteeing correctness through synchronization. In *Distributed Autonomous Robotic Systems*, pages 337–351. Springer, 2014.

[38] Shuo Yang, Xiang Yin, Shaoyuan Li, and Majid Zamani. Secure-by-construction optimal path planning for linear temporal logic tasks. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 4460–4466, 2020.

[39] Jingjin Yu and Steven M LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[40] Jianing Zhao, Keyi Zhu, Mingyang Feng, Shaoyuan Li, and Xiang Yin. No-regret path planning for temporal logic tasks in partially-known environments. *The International Journal of Robotics Research*, page 02783649251315758, 2025.

Peng Lv (S'24) was born in Heilongjiang, China, in 1998. He received the B.Eng degree in automation from Harbin Engineering University in 2020. He is currently a Doctoral student at Shanghai Jiao Tong University. He is also a visiting student at ENS Paris-Saclay, University of Paris-Saclay. His research interests include formal methods, temporal logic task planning and game theory in Discrete-Event Systems.

Shaoyuan Li (Senior Member, IEEE) was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree in automatic control theory and application from Nankai University, Tianjin, in 1997.

Since 1997, he has been with the School of Automation and Sensing, Shanghai Jiao Tong University, Shanghai, China, where he is currently a Professor. His current research interests include model predictive control, dynamic system optimization, and cyber-physical systems.

Bruno Denis received the Ph.D. degree from the University of Nancy 1, Nancy, France, in 1994. He is currently an Associate Professor of Automatic Control at the École Normale Supérieure Paris-Saclay, France. He teaches courses in control of mechatronic systems, modeling and simulation of multi-physical manufacturing systems, and fault detection and removal in discrete-event control systems. His research interests include formal methods and models for the synthesis, analysis, and diagnosis of Discrete-Event Systems (DES) and Hybrid Dynamical Systems (HDS), with applications in manufacturing systems, networked automated systems, energy production, and digital twin.

Gregory Faraut received the B.S. degree in electrical engineering and the M.S. degree in computer science from the University of Nice Sophia Antipolis, Nice, France, in 2004 and 2006, respectively, and the Ph.D. degree in automatic control from the Ampere Lab, INSA Lyon, Villeurbanne, France, in 2010. Since 2011, he has been an Associate Professor of Automatic Control with LURPA, ENS Paris-Saclay, University of Paris-Saclay, and Full Professor since 2022. His research interests concern the field of Discrete Event Systems with applications to cyber-physical systems, behavioral identification, resilient control, and, more recently, Digital Twins for cognitive systems.

Cristian Mahulea received his B.S. and M.Sc. degrees in control engineering from the Technical University of Iasi, Romania, in 2001 and 2002, respectively, and his Ph.D. in systems engineering from the University of Zaragoza, Spain, in 2007. Currently, he is a Full Professor at the University of Zaragoza, where he chaired the Department of Computer Science and Systems Engineering from 2020 to 2024. He has also served as a visiting professor at the University of Cagliari, Italy.

His research interests include discrete event systems, hybrid systems, mobile robotics, and healthcare systems. He has been a Visiting Researcher at the University of Sheffield (UK), Boston University (USA), University of Cagliari (Italy), and ENS Paris-Saclay (France).

Cristian has served as an Associate Editor for IEEE Transactions on Automation Science and Engineering (TASE) and IEEE Control Systems Letters (L-CSS). He is currently an Associate Editor for IEEE Transactions on Automatic Control (TAC), the International Journal of Robotics Research (IJRR), Discrete Event Dynamic Systems: Theory and Applications (JDES), and IEEE Robotics and Automation Letters (RA-L). Additionally, he was the General Chair of ETFA 2019.

Xiang Yin (S'14-M'17) was born in Anhui, China, in 1991. He received the B.Eng degree from Zhejiang University in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering. Since 2017, he has been with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, where he is a Full Professor. His research interests include formal methods, discrete-event systems, robotics, artificial intelligence and cyber-physical systems.

Dr. Yin is serving as the chair of the *IEEE CSS Technical Committee on Discrete Event Systems*, Associate Editors for the *Journal of Discrete Event Dynamic Systems: Theory & Applications*, *Nonlinear Analysis: Hybrid Systems*, *IEEE Control Systems Letters*, *IEEE Transactions on Automation Science and Engineering*, and a member of the *IEEE CSS Conference Editorial Board*.