2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)
August 28 - September 1, 2024. Bari, Italy

2024 IEEE 20th International Conference on Automation Science and Engineering (CASE) | 979-8-3503-5851-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/CASE59546.2024.10711490

Multi-Agent Path Planning for Finite Horizon Tasks with Counting Time Temporal Logics

Peng Lv, Shaoyuan Li and Xiang Yin

***Abstract*— In this paper, we consider the multi-agent path planning problem for high-level tasks with finite horizons. In many situations, there is the 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, may not be 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 use integer linear programming to encode the satisfaction of the task. We demonstrate that our approach is both sound and complete. To validate our results, we present numerical experiments to show the scalability of the proposed approach. Furthermore, we provide a simulation case study of a team of autonomous robots to illustrate the synthesis procedure.**

I. Introduction

Multi-agent systems have found widespread applications in various fields, including data gathering [1], manufacturing systems [2] and autonomous warehouses [3], [4]. Traditionally, algorithms for multi-agent coordination problems have focused on synthesizing a trajectory for each agent to meet low-level task requirements, such as obstacle avoidance or target reaching [5], [6]. However, with the rapid development of cyber-physical systems (CPS), many recent studies have focused on developing algorithms to solve high-level complex tasks in multi-agent systems. These studies often involve formal methods, such as temporal logic, to specify task requirements and ensure correct and efficient coordination among agents [7]–[9].

The desired high-level requirements in multi-agent 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. Many works in multi-agent 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 [10]–[13]. Others have considered planning problems for probabilistic satisfaction of LTL tasks under transition uncertainties [13], [14]. Controller synthesis problems for heterogeneous agents subject to graph temporal logic specifications (GTL) have also been studied [15]. To count the number of robots achieving a task, robust trajectory planning for multi-agent systems is studied under counting temporal logic (cLTL) [11]. However, most of these works focus on path planning problems in an infinite horizon, and the algorithms often depend on specific structural properties of the solutions, such as the prefix-suffix form of trajectories.

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). Additionally, in [17], a decentralized and probabilistic algorithm is proposed to control agent teams moving along graph nodes for finite-horizon planning for GTL. In [18], 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 [19], 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, [20] 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$_f$ formulae.

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-agent path planning problem using the proposed CTTL. Our approach is to encode the dynamics of the agents 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 using the proposed CTTL is much more efficient for the purpose of path planning compared with the standard LTL-based approaches.

II. Counting Time Temporal Logic Planning Problem

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

This work was supported by the National Natural Science Foundation of China (62173226, 62061136004).

Peng Lv, Shaoyuan Li and Xiang Yin are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. {lv-peng, syli,yinxiang}@sjtu.edu.cn.

Definition 1. A DTS T is a five-tuple $T = (X, x_0, E, \Pi, L)$, where X is the set of all states, x_0 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 $x(1) = x_0$ 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 agent team $\mathcal{R}$ consisting of $N \in \mathbb{Z}^+$ agents operating synchronously in an environment that in consistent of a set of regions with connectivity constraints. The DTS and h-path set associated with agent R^n, $n \in [N]$, is denoted by $T^n = (X^n, x_0^n, E^n, \Pi, L^n)$ and P_h^n respectively, which means that we allow the agents have different dynamics but with the identical atomic proposition set. Note that this hypothesis is without lose 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 agent, 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 $i \in [h]$.

We introduce a variant 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 $t \leq h-1$ and $\mathcal{P}_h(t+1) \models \phi$;
- $\mathcal{P}_h(t) \models \phi_1 \mathcal{U}^k \phi_2$ iff $t \leq h-k+1$ and $\exists t \leq t_1 < \cdots < t_k \leq h, \mathcal{P}_h(t_i) \models \phi_2, \forall i \in [k]$ and $\forall t \leq t' < t_k, \mathcal{P}_h(t') \models \phi_1$.

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 agent 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 $\neg \bigcirc \neg\phi \not\equiv \bigcirc\phi$ [21]. And when $k = 1$, the semantics for all the above three operators are just the same as the "until", "eventually" and "always" operators in LTL.

Overall, we address the problem of multi-agent trajectory planning in finite sequences under CTTL, which is formally defined as follows:

Problem 1. Given N agents operating synchronously with dynamics $T^n = (X^n, x_0^n, E^n, \Pi, L^n)$ with $n \in [N]$, a finite time domain $h \in \mathbb{Z}^+$ and a CTTL formula ϕ in forms of (1), synthesize a team sequence $\mathcal{P}_h = \langle \rho_h^1, \rho_h^2, \cdots, \rho_h^N\rangle$ satisfying ϕ, i.e., $\mathcal{P}_h \models \phi$.

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 [22]. Our approach involves encoding the dynamics of the agent team and the CTTL specifications as a set of ILP constraints.

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

Given the DTS $T^n = (X^n, x_0^n, E^n, \Pi, L^n)$ 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 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 t-th state of R^n 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 follows: $\forall t \in [h]$,

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

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 dynamics; the second one ensures that R^n can only appear at a state at any time, while the third one requires that R^n must respect the initial condition.

B. Encoding for the CTTL Specifications

Next, we show how to encode the CTTL formula in a recursive way.

For any atomic proposition π, we first introduce the following $N \times h$ 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 binary 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, we give the following equations to describe 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 $\mathcal{R}$: For $\mathcal{R}$, we give the following equations to describe the satisfaction of π 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 agent 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 $\phi = \bigvee_{i=1}^I \phi_i$, the following equations are given to describe the satisfaction of ϕ at $t \in [h]$:

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

4) Conjunction $\wedge$: For $\mathcal{R}$ with $\phi = \bigwedge_{i=1}^I \phi_i$, the following equations are given to describe the satisfaction of ϕ at $t \in [h]$:

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

5) Negation $\neg$: For $\mathcal{R}$ with $\psi = \neg\phi$, the following equation is given to describe the satisfaction of ψ at $t \in [h]$:

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

6) Next $\bigcirc$: For $\mathcal{R}$ with $\psi = \bigcirc\phi$, the following equation is given to describe the satisfaction of ψ:

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

Note that the above encoding equations from (5) to (8) for the Boolean and temporal operators are consistent with the encodings in [11], [22]. 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 $\phi = \phi_1 \mathcal{U}^k \phi_2$, if $k = 1$, then the following equations are given to describe the satisfaction of ϕ:

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

If $k > 1$, we define $\phi^k = \phi = \phi_1 \mathcal{U}^k \phi_2$, then the following equations are given to describe the satisfaction of ϕ:

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

Notice that when $k = 1$, the encoding (9) for $\mathcal{U}^1$ is the same as the "$\mathcal{U}$" operator in LTL [22]. 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 $\hat{t}$ between t and h such that $y_{\phi_2}(\hat{k}) = 1$ and before the k-th 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 $\hat{k}$ 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).

Therefore, if (11) has a solution, then the solution for Problem 1 can be synthesized as follows: $\forall n \in [N], t \in [h], [v_i^n(t) = 1] \Rightarrow [\rho_h^n(t) = x_i^n]$.

Remark 1. 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:

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

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.

Table 1: Statistics for different number of agents.

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.

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

IV. Experiment Results

In this section, we provide a set of experiments to illustrate our results. 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` [23] to setup the ILP problem and `GUROBI` [24] as the underlying ILP solver. The simulation video is also available[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 agents 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. For each set of parameters, we repeated the experiment for 20 times and recorded the average value of all experimental data.

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

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

which requires to 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 shown in the table, 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 increase linearly with the increase of $|X^n|$. However, the solving time is significantly affected by the system size, which seems to exhibit polynomial growth as $|X^n|$ increases.

[1] https://github.com/Lv-Peng/CTTL

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

Next, we further study the effect of the length of planning horizon h on running time. In this experiment, we fix $|X^n| = 50$ and $N = 20$ and increase h from 20 to 100. Besides, note that just increasing the planning horizon might result in trivial solutions, such that the agents choose to stay in place for many steps in the trajectory. 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 1. Some parameters are given as follows: $N = 10, |X^n| = 50, \forall n \in [N]$ and $h = 100$.

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. As the complexity of the specification increases, the difference between the execution times 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.

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

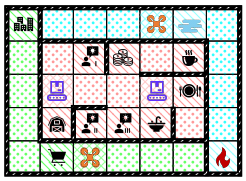
These results are as expected since the equivalent LTL formula is always combinatorially much longer than the CTTL formula, as stated in Remark 1. Therefore, these numerical results demonstrate the scalability of our encoding method and the advantage of the CTTL language.

B. Simulation Experiments

Consider a factory as depicted in Fig 1(a) with 35 grid regions and 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).

There are 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 of the building, while A_0 and A_1 can only inspect outside of the building to deal with the emergencies. Moreover, to ensure efficient energy usage and to prevent any potential collisions, each UAV is confined to operate within a designated area, the green part for A_0, and the blue part for A_1. Both UGVs are capable of transporting parcels between these grids. Furthermore, 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 to remain in the same 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: 1) due to insufficient



(a) The topology of the factory with two UGVs and two UAVs.

(b) Simulation trajectories for the two UGVs.

Fig. 1: Scenes and trajectories for the simulation experiment.

parcels left, the UGVs must replenish before delivery by proceeding to the warehouse; 2) 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; 3) two UGVs should never enter canteen for food safety; 4) living quarters purchase two parcels from supermarket and UAVs need to transport them; 5) UAVs discover two ignition sources at fire location. They need to first to fetch water from the lake and then go to the fire location to put out the fire.

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

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

The planning problem is solved in 2.78 s. The simulation trajectories for the two UGVs are shown in Figure 1(b). 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 parcels from the supermarket and unloads them at x_1 twice.

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-agent path planning problem for CTTL specifications using integer linear programming techniques. The efficiency of CTTL in describing such counting tasks was demonstrated by experiment results. In the future, we plan to investigate the robust planning problem, where some agents may be subject to execution delays or even failures.

References

[1] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1082–1097, 2018.

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

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

[4] Y. Tatsumoto, M. Shiraishi, K. Cai, and Z. Lin, "Application of online supervisory control of discrete-event systems to multi-robot warehouse automation," *Control Engineering Practice*, vol. 81, pp. 97–104, 2018.

[5] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Implan: Scalable incremental motion planning for multi-robot systems," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 1–10, IEEE, 2016.

[6] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[8] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[9] X. Yin, B. Gao, and X. Yu, "Formal synthesis of controllers for safety-critical autonomous systems: developments and challenges," *Annual Reviews in Control*, p. 100940, 2024.

[10] P. Lv, G. Luo, Z. Ma, S. Li, and X. Yin, "Optimal multi-robot path planning for cyclic tasks using petri nets," *Control Engineering Practice*, vol. 138, p. 105600, 2023.

[11] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1189–1206, 2019.

[12] Z. He, J. Yuan, N. Ran, and X. Yin, "Security-based path planning of multi-robot systems by partially observed petri nets and integer linear programming," *IEEE Control Systems Letters*, 2024.

[13] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

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

[15] F. Djeumou, Z. Xu, M. Cubuktepe, and U. Topcu, "Probabilistic control of heterogeneous swarms subject to graph temporal logic specifications: A decentralized and scalable approach," *IEEE Transactions on Automatic Control*, 2022.

[16] I. Haghighi, S. Sadraddini, and C. Belta, "Robotic swarm control from spatio-temporal specifications," in *55th IEEE Conference on Decision and Control (CDC)*, pp. 5708–5713, IEEE, 2016.

[17] F. Djeumou, Z. Xu, and U. Topcu, "Probabilistic swarm guidance subject to graph temporal logic specifications.," in *Robotics: Science and Systems*, 2020.

[18] B. Lacerda, D. Parker, and N. 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*, pp. 1511–1516, IEEE, 2014.

[19] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Twenty-Third International Joint conference on Artificial Intelligence*, pp. 854–860, 2013.

[20] T. Kinugawa and T. Ushio, "Hyper-labeled transition system and its application to planning under linear temporal logic constraints," *IEEE Control Systems Letters*, vol. 6, pp. 2437–2442, 2022.

[21] G. De Giacomo, M. Y. Vardi, *et al.*, "Synthesis for LTL and LDL on finite traces," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 1558–1564, 2015.

[22] V. Schuppan, T. Latvala, T. Junttila, K. Heljanko, and A. Biere, "Linear encodings of bounded LTL model checking," *Logical Methods in Computer Science*, vol. 2, 2006.

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

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