Scalable Multi-Robot Task Allocation and Coordination under Signal Temporal Logic Specifications

Wenliang Liu, Nathalie Majcherczyk, and Federico Pecora

***Abstract*— Motion planning with simple objectives, such as collision-avoidance and goal-reaching, can be solved efficiently using modern planners. However, the complexity of the allowed tasks for these planners is limited. On the other hand, signal temporal logic (STL) can specify complex requirements, but STL-based motion planning and control algorithms often face scalability issues, especially in large multi-robot systems with complex dynamics. In this paper, we propose an algorithm that leverages the best of the two worlds. We first use a single-robot motion planner to efficiently generate a set of alternative reference paths for each robot. Then coordination requirements are specified using STL, which is defined over the assignment of paths and robots' progress along those paths. We use a Mixed Integer Linear Program (MILP) to compute task assignments and robot progress targets over time such that the STL specification is satisfied. Finally, a local controller is used to track the target progress. Simulations demonstrate that our method can handle tasks with complex constraints and scales to large multi-robot teams and intricate task allocation scenarios.**

I. Introduction

Temporal logics such as Linear Temporal Logic (LTL) [1] and Signal Temporal Logic (STL) [2] provide a formal way to specify complex and time-related requirements for a given system, and have been widely used in robotics. In this paper, we focus on multi-robot systems subject to STL specifications. Deploying a fleet of autonomous robots to meet any form of specification or objective requires overcoming four challenges: (1) determining how tasks should be distributed among robots (task assignment); (2) deciding how robots should negotiate to collaboratively finish a task or use the shared resources (coordination); (3) computing trajectories that robots should follow to complete their tasks (motion planning); (4) ensuring that these trajectories are executed accurately respecting the robots' dynamics (control).

Numerous STL control synthesis algorithms have been proposed [3]–[6]. These methods aim to find a controller to make a system satisfy a given STL specification. Although called "STL control synthesis", these methods tackle the motion planning and control problems simultaneously, i.e., they aim to find the state trajectory that satisfies the STL specification and the control inputs that realize this trajectory. When the system dynamics are complex and the planning horizon is large, these methods become computationally expensive, and cannot scale to large multi-robot systems.

Most methods for coordinating robots using temporal logic [7]–[12] discretize the state space to a graph or automaton, so they can decouple the control problem and focus on jointly solving the task allocation, coordination, and motion planning problems on the graph. However, an expressive enough abstraction of the state space can result in a very large graph, which again makes the problem computationally intractable. In [13], [14], continuous state space is considered and the control problem is solved jointly, which also scales badly. Another drawback of the above approaches is that the time steps needed might be too large for long-horizon planning. Unfortunately, STL specifications often span long horizons. A method based on time-stamped waypoints is proposed in [15] which decouples control from motion planning without discretizing the state space, and is able to perform long-horizon planning with a relatively small number of time stamps.

All the above methods attempt to solve problems (1)–(4) simultaneously. Although this enlarges the search space, potentially improving the quality of the solution, computational complexity restricts scalability. In this paper, we propose a multi-robot task allocation and coordination algorithm that is decoupled from motion planning and control. This is motivated by the insights that most multi-robot tasks require each robot to move from the source location to the target location (or a sequence of target locations) while avoiding obstacles and to obey temporal and logical rules induced by their coordination. The former can be efficiently solved using heuristic- or sampling-based single-robot motion planning algorithms [16]–[18]. However, these planners cannot enforce the latter, which is where STL is truly needed.

In this paper, we first use single-robot motion planning to efficiently generate a set of alternative reference paths for each robot. Then we use STL to define coordination specifications over the progress of robots along their paths. In this way, each robot only decides which path to take and how fast to track the path, without considering the entire state space or its dynamics. A similar idea is introduced in [19], [20], which only consider simple constraints rather than STL specifications. Similar to [15], we search for a sequence of time-stamped target progress points for the robots to track along their paths, which can be solved using Mixed Integer Linear Program (MILP). Finally, the target progress points on a reference path are tracked using a local controller of each robot, which guarantees the satisfaction of the STL specification. Our approach breaks down the STL-based coordination problem into three parts: task allocation and coordination subject to STL specifications; single-robot motion planning; and single-robot control. Although this approach slightly reduces the search space of the overall problem due to the decoupling, it significantly reduces the computational cost. We find that in practice, it is enough to

All authors are with Amazon Robotics, North Reading, MA, USA {liuwll,majcherc,fpecora}@amazon.com

realize many interesting multi-robot use cases.

The contributions of this paper are threefold. (1) We propose a scalable algorithm to operate fleets of robots subject to STL specifications by decoupling motion planning and control from the task allocation and coordination problems. (2) We prove formally that our algorithm satisfies the STL specification. (3) We evaluate our algorithm on a variety of realistic applications, showing that it outperforms state-of-the-art methods in terms of running time and is able to scale to large teams and complex STL specifications.

II. Problem Formulation

In this paper, we use $\mathbb{R}$ and $\mathbb{B}$ to denote real values and binary values, respectively. For a vector $x \in \mathbb{R}^n$, let $B_l(x, \epsilon) := \{y \in \mathbb{R}^n \mid \|y - x\|_l < \epsilon\}$ be the ϵ-ball centered at x, where $\|\cdot\|_l$ denotes the l-norm.

A. System Model

Consider a set of N robots $\{r_1, \ldots, r_N\}$ sharing an obstacle-free space $\mathcal{W}_{\text{free}} \subseteq \mathbb{R}^3$. Each robot r_i is defined as a tuple $\langle Q_i, q_i^0, R_i, \{p_i^j\}_{j=1}^{M_i}\rangle$ where:

- Q_i is the space of obstacle-free configurations of r_i;
- $q_i^0 \in Q_i$ is the initial configuration of r_i;
- $R_i : Q_i \to 2^{\mathcal{W}_{\text{free}}}$ maps the configuration of r_i to a geometry describing the space occupied by r_i;
- $\{p_i^j\}_{j=1}^{M_i}$ is a set of M_i reference paths assigned to r_i, where a reference path $p_i^j : [0, g_i^j] \to Q_i$ maps progress between 0 and g_i^j to a configuration, g_i^j is the maximum progress corresponding to the goal configuration which is proportional to the path length. Here, $\{p_i^j\}_{j=1}^{M_i}$ corresponds to potential tasks assigned to r_i.

We assign a vector $\mathbf{z}_i \in \mathbb{B}^{M_i}$ consisting of M_i binary variables $\mathbf{z}_i = [z_i^1, \ldots, z_i^{M_i}]^\top$ to each robot r_i, indicating which reference path is selected. Specifically, $z_i^j = 1$ indicates p_i^j is selected. Since each robot can follow one and only one reference path, we have the following constraint:

$$\sum_{j=1}^{M_i} z_i^j = 1, \ i = 1, \ldots, N. \tag{1}$$

Let the joint selection vector be $\mathbf{z} = [\mathbf{z}_1^\top, \ldots, \mathbf{z}_N^\top]^\top$, which is the concatenation of all selection vectors for all robots.

The temporal profile of r_i is $\sigma_i : [0, T] \to R_{\geq 0}$, which maps time to the progress on its selected path, T is the time bound. We assume $\sigma_i(t)$ is monotonically increasing. We denote the joint temporal profile as $\boldsymbol{\sigma} : [0, T] \to \mathbb{R}^N_{\geq 0}$, which maps time to the joint progress of all robots. For simplicity, we occasionally omit the variable t and directly use $\sigma_i \in \mathbb{R}_{\geq 0}$ to denote the progress when the context makes it clear.

B. Reference Path STL

In this paper, we define a fragment of STL over $\mathbf{s} = [\mathbf{z}, \boldsymbol{\sigma}]$, which is the concatenation of the joint selection vector $\mathbf{z}$ and the joint temporal profile $\boldsymbol{\sigma}$. In the following, we refer to this STL fragment as reference-path STL (RP-STL).

Different from the general case of STL, the predicate of RP-STL μ_i is restricted to a specific robot r_i in the form of: $\sigma_i(t) - \mathbf{b}_i^\top \mathbf{z}_i \geq 0$, where $\mathbf{b}_i = [b_i^1, \ldots, b_i^{M_i}]^\top \in \mathbb{R}^{M_i}$. Depending on the value of $\mathbf{b}_i$, the predicate μ_i can have different interpretations. For example, if $b_i^1 = 10$ and $b_i^j = M \ \forall j \neq 1$, where M is a large value, then μ_i is evaluated as *true* at time t if and only if the path p_i^1 is selected and $\sigma_i(t) \geq 10$. On the contrary, if $b_i^j = -M$, $\forall j \neq 1$, then μ_i is automatically satisfied if the path p_i^1 is not selected. It is only evaluated as *false* if p_i^1 is selected but $\sigma_i(t) < 10$.

We also define a counting formula for RP-STL as a tuple $(\{\varphi_l\}_{l=1}^L, m)$ where $\{\varphi_l\}_{l=1}^L$ is a set of L RP-STL formulas and $m \leq L$ is a positive integer. A counting formula is evaluated as *true* if and only if there are at least m subformulas φ_l evaluated as *true*. Although a counting formula can be translated into standard STL using combinatorially many disjunctions, our definition provides a concise way to formulate this kind of requirements, which is very useful in practice. In addition, in Sec. III we propose a MILP encoding for counting formulas, which avoids introducing combinatorially many binary variables.

We recursively define the *syntax* of RP-STL as:

$$\begin{aligned}\varphi =& \mu_i \mid \neg\mu_i \mid (\{\varphi_l\}_{l=1}^L, m) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\ & \mid F_{[a,b]}\varphi \mid G_{[a,b]}\varphi \mid \varphi_1 U_{[a,b]}\varphi_2,\end{aligned} \tag{2}$$

where φ, φ_1, φ_2, φ_l are RP-STL formulas, $\neg$, $\wedge$, $\vee$ are the *negation*, *conjunction*, and *disjunction*, $F_{[a,b]}$, $G_{[a,b]}$, $U_{[a,b]}$ are the temporal operators *eventually*, *always*, and *until*.

The fact that a signal $\boldsymbol{s}$ satisfies an RP-STL formula φ at time t is denoted as $(\boldsymbol{s}, t) \models \varphi$. Intuitively, $(\boldsymbol{s}, t) \models F_{[a,b]}\varphi$ states that φ must become true at some time point in $[t+a, t+b]$, $(\boldsymbol{s}, t) \models G_{[a,b]}\varphi$ means that φ must be true at all time points in $[t+a, t+b]$, and $(\boldsymbol{s}, t) \models \varphi_1 U_{[a,b]}\varphi_2$ requires that φ_2 becomes true at some time in $[t+a, t+b]$ and φ_1 is true at all time before that. The satisfaction of the counting formula $(\boldsymbol{s}, t) \models (\{\varphi_l\}_{l=1}^L, m)$ is defined as $\sum_{l=1}^L \mathbf{1}((\boldsymbol{s}, t) \models \varphi_l) \geq m$, where $\mathbf{1}(\text{true}) = 1$ and $\mathbf{1}(\text{false}) = 0$. The operators and counting formula can be arbitrarily nested to express more complex requirements. For simplicity, we will omit t when $t = 0$ and denote $(\boldsymbol{s}, 0) \models \varphi$ as $\boldsymbol{s} \models \varphi$.

Remark 1: In syntax (2), negation can only be applied to predicates, known as the Negation Normal Form (NNF). This is not restrictive, as any STL formula can be put into NNF [21]. For the counting formula, $\neg(\{\varphi_l\}_{l=1}^L, m)$ is equivalent to $(\{\neg\varphi_l\}_{l=1}^L, L - m + 1)$. Hence, any RP-STL can be put into NNF. By applying negation to a counting formula, i.e., $\neg(\{\varphi_l\}_{l=1}^L, m)$, we can require that less than m subformulas φ_l are *true*.

C. Interference Constraints

One important kind of coordination constraints that can be expressed by RP-STL is the interference constraints, i.e., how robots traverse the shared space without collision. Consider two robots r_i and $r_{i'}$, and two paths p_i^j and $p_{i'}^{j'}$ as in Fig. 1 (left). The two paths have one critical section defined as a pair of intervals $([l, u], [l', u'])$ which satisfies (1) $\forall \sigma_i \in [l, u]$, $\exists \sigma_{i'} \in [l', u']$ such that $R_i(p_i^j(\sigma_i)) \cap R_{i'}(p_{i'}^{j'}(\sigma_{i'})) \neq \emptyset$, and vice versa; (2) the intervals $[l, u]$ and $[l', u']$ are maximal, i.e., there is no way to grow them and still satisfy the first

requirement. A collision is only possible if both robots are in the critical section. The interference constraint requires that when one robot is in the critical section, the other robot should not enter it. Formally, this can be written as:

$$\begin{aligned}\varphi_{ii'}^{jj'} =&(\sigma_{i'} < [M \cdots \overset{j'\text{-th}}{l'} \cdots M]\mathbf{z}_{i'}^\top \; U_{[0,T]} \\ &\sigma_i \geq [-M \cdots \overset{j\text{-th}}{u} \cdots - M]\mathbf{z}_i^\top) \\ &\vee (\sigma_i < [M \cdots \overset{j\text{-th}}{l} \cdots M]\mathbf{z}_i^\top \; U_{[0,T]} \\ &\sigma_{i'} \geq [-M \cdots \overset{j'\text{-th}}{u'} \cdots - M]\mathbf{z}_{i'}^\top)).\end{aligned} \tag{3}$$

In English, $\varphi_{ii'}^{jj'}$ means that if paths p_i^j and $p_{i'}^{j'}$ are selected, then $r_{i'}$ cannot enter the critical section *until* r_i leaves it, *or* r_i cannot enter the critical section *until* $r_{i'}$ leaves it. If any one of p_i^j and $p_{i'}^{j'}$ is not selected, then $\varphi_{ii'}^{jj'}$ is satisfied.

We add the interference constraints $\varphi_{ii'}^{jj'}$ for all critical sections in all path pairs $(p_i^j, p_{i'}^{j'})$ where $i \neq i'$. These constraints avoid any collisions between robots.

Remark 2: The above STL formula requires exclusive use of the critical section, which might be too conservative. We can partially rely on the robot's own autonomy (the local controller) and relax this constraint by replacing u and u' in (3) with δ and δ' where

$$\delta = \inf\{\sigma \in [l, u] \mid \forall d > \sigma, R_i(p_i^j(d)) \cap R_{i'}(p_{i'}^{j'}(l')) = \emptyset\},$$
$$\delta' = \inf\{\sigma \in [l', u'] | \forall d > \sigma, R_{i'}(p_{i'}^{j'}(d)) \cap R_i(p_i^j(l)) = \emptyset\}.$$

Here, δ is the smallest progress for r_i such that it will not block $r_{i'}$ from entering the critical section.

Example 1: Consider a team of robots $\mathcal{R} = \{r_1, r_2, r_3\}$ in an environment where a bridge (occupying the space $\mathcal{B}$) goes across a river, as shown in Fig. 1 (right). Robot r_1 is assigned two reference paths p_1^1 and p_1^2, and r_2 and r_3 are assigned p_2^1 and p_3^1, respectively. There are totally 3 critical sections among these 4 paths. In addition to the interference constraints, we require that no more than 2 robots can be on the bridge at the same time due to the weight limit of the bridge. Let $[l_i^j, u_i^j]$ be an interval on the path p_i^j such that $\sigma \in [l_i^j, u_i^j]$ if and only if $R_i(p_i^j(\sigma)) \cap \mathcal{B} \neq \emptyset$. These requirements can be written as an RP-STL formula:

$$\begin{aligned}\varphi =&\varphi_{12}^{11} \wedge \varphi_{12}^{21} \wedge \varphi_{13}^{11} \wedge G_{[0,T]} \\ \neg\Bigg(\Big\{&(\sigma_1 \geq [l_1^1 \; M]\begin{bmatrix} z_1^1 \\ z_1^2 \\ z_1^1 \end{bmatrix}) \wedge (\sigma_1 < [u_1^1 \; -M]\begin{bmatrix} z_1^1 \\ z_1^2 \\ z_1^1 \end{bmatrix}), \\ &(\sigma_1 \geq [M \; l_1^2]\begin{bmatrix} z_1^1 \\ z_1^2 \\ z_1^1 \end{bmatrix}) \wedge (\sigma_1 < [-M \; u_1^2]\begin{bmatrix} z_1^1 \\ z_1^2 \\ z_1^1 \end{bmatrix}), \\ &(\sigma_2 \geq l_2^1 \wedge \sigma_2 < u_2^1), \; (\sigma_3 \geq l_3^1 \wedge \sigma_3 < u_3^1)\Big\}, 3\Bigg).\end{aligned} \tag{4}$$

Here we omit z_2^1 and z_3^1 as they are always equal to 1. The counting formula includes 4 subformulas corresponding to 4 paths (and 4 intervals $[l_i^j, u_i^j]$). The (negative) counting constraint requires that the progress along less than 3 paths can be in the interval $[l_i^j, u_i^j]$ at the same time. If a path is not selected, the corresponding subformula is automatically violated (not counted as on the bridge) regardless of progress.

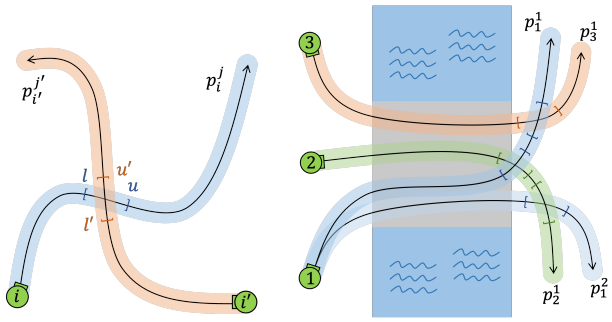


Fig. 1: Left: The critical section between two robots r_i and $r_{i'}$. Right: 3 robots crossing a bridge (gray) on a river (blue).

D. The Task Allocation and Coordination Problem

Our goal is to find the joint selection vector $\mathbf{z}$ and the joint temporal profile $\boldsymbol{\sigma}$ that satisfy a given RP-STL specification φ and minimizes a given cost function. Let a time-stamped joint progress (TSJP) be a pair $(t_k, \boldsymbol{\sigma}_{(k)})$, where $k \in \{0, 1, \ldots, K\}$, $t_k \in [0, T]$ is a time stamp and $\boldsymbol{\sigma}_{(k)} = [\sigma_{1,(k)}, \ldots, \sigma_{N,(k)}] \in \mathbb{R}_{\geq 0}^N$ is a corresponding joint progress. Note that we use bold font and parentheses on k to distinguish $\boldsymbol{\sigma}_{(k)}$ from σ_i, the latter being the temporal profile of r_i. Instead of searching for an accurate joint temporal profile over all possible functions over time, we search for a set of joint temporal profiles constructed from a sequence of TSJPs $\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K$, where t_k is the k-th time stamp and $\boldsymbol{\sigma}_{(k)}$ is the k-th joint progress, $0 = t_0 < t_1 < \ldots < t_K \leq T$, $\mathbf{0} = \boldsymbol{\sigma}_{(0)} \leq \boldsymbol{\sigma}_{(1)} \leq \ldots \leq \boldsymbol{\sigma}_{(K)} = [[g_1^1, \ldots, g_1^{M_1}] \cdot \mathbf{z}_1, \ldots, [g_N^1, \ldots, g_N^{M_N}] \cdot \mathbf{z}_N]$. Note that the equality and inequality signs for $\boldsymbol{\sigma}_{(k)}$ are interpreted element-wise. Specifically, given a sequence of TSJPs and an $\epsilon > 0$, we construct a set of joint temporal profiles as

$$\begin{aligned}&\mathcal{S}_\epsilon\Big(\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K\Big) := \\ &\qquad \{\boldsymbol{\sigma} \mid \boldsymbol{\sigma}(t_k) \in B_\infty(\boldsymbol{\sigma}_{(k)}, \epsilon), \; k = 0, \ldots, K\}.\end{aligned} \tag{5}$$

Now, our goal becomes finding a joint selection vector $\mathbf{z}$ and a sequence of TSJPs $\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K$ such that $\forall \boldsymbol{\sigma} \in \mathcal{S}_\epsilon\big(\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K\big)$, $[\mathbf{z}, \boldsymbol{\sigma}] \models \varphi$, and a cost is minimized.

Since the robots cannot make progress arbitrarily fast, we add the following constraint on the TSJPs for all robots r_i:

$$|\sigma_{i,(k+1)} - \sigma_{i,(k)}| \leq v_i^{max} \cdot (t_{k+1} - t_k), k = 0, \ldots, K-1, \tag{6}$$

where $v_i^{max} \in \mathbb{R}_{\geq 0}$ is the maximum speed for robot i, which is assumed to be a constant for different paths of a same robot, but can be different for different robots. Now we formally state the task allocation and coordination problem.

Problem 1: Consider a set of N robots. Given a set of reference paths $\{p_i^j\}_{j=1}^{M_i}$ for each robot r_i and an RP-STL specification φ, find the joint selection vector $\mathbf{z}$ and a sequence of TSJPs $\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K$ such that $\forall \boldsymbol{\sigma} \in \mathcal{S}_\epsilon\big(\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K\big)$, $[\mathbf{z}, \boldsymbol{\sigma}] \models \varphi$, and a cost $\mathcal{L}$ is minimized:

$$\begin{aligned}\min_{\mathbf{z}, \{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K} \quad &\mathcal{L}(\mathbf{z}, \{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K) \\ \text{s.t. } &[\mathbf{z}, \boldsymbol{\sigma}] \models \varphi, \; \forall \boldsymbol{\sigma} \in \mathcal{S}_\epsilon\big(\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K\big), \\ &\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^K \text{ satisfies (6)}, \mathbf{z} \text{ satisfies (1)}.\end{aligned} \tag{7}$$

Some examples of the cost function include the makespan t_K or the sum of travel time $\sum_{i=1}^{N} T_i$ where

$$T_i = \inf\{t_k \mid \sigma_{i,(k)} = [g_i^1, \dots, g_i^{M_i}] \cdot \mathbf{z}_i\}. \tag{8}$$

III. MILP-Based Solution

We solve Problem 1 by encoding (7) into a Mixed-Integer Linear Program (MILP), which can be solved efficiently using off-the-shelf solvers such as Gurobi [22].

To encode all constraints in (7) as mixed-integers linear constraints, we first encode these constraints into a Linear and Counting Constraints Formula (LCCF), which is a logic sentence of atomic formulas connected by conjunctions, disjunctions, and counting operators. Each atomic formula is in the form of $\texttt{LE} \geq 0$, where LE is a linear expression of the continuous and binary variables, including the TSJPs and the selection vector. Second, we eliminate all the disjunctions and counting operators in the LCCF, which makes the LCCF a conjunction of mixed integer linear constraints. Finally, we add all these constraints into the MILP solver to find the solution. The encoding is inspired by [15], which searches for a piece-wise linear path connecting time-stamped waypoints in the state space to satisfy STL specifications. Different from [15], here we do not require linearity on the segments between TSJPs. We only assume the temporal profile for each robot is monotonically increasing.

A. Constructing the LCCF

a) Selection vector encoding: Since constraints (1) are already in the form of $\texttt{LE} \geq 0$, we just conjunct them together to form a single LCCF.

b) RP-STL encoding: We inductively construct an LCCF that encodes the RP-STL formula φ in (7) by constructing an LCCF z_k^{φ} for each segment between two adjacent TSJPs such that z_k^{φ} is true if and only if $\forall t \in [t_k, t_{k+1}]$ and $\forall \boldsymbol{\sigma} \in \mathcal{S}_\epsilon\big(\{(t_k, \boldsymbol{\sigma}_{(k)})\}_{k=0}^{K}\big)$, $([\mathbf{z}, \boldsymbol{\sigma}], t) \models \varphi$. We refer to this property as the soundness property. Then z_0^{φ} will be the LCCF we want. Specifically, z_k^{φ} is recursively encoded as:

$$z_k^{\mu_i} = (\sigma_{i,(k)} - \boldsymbol{b}_i^\top \boldsymbol{z}_i \geq \epsilon) \wedge (\sigma_{i,(k+1)} - \boldsymbol{b}_i^\top \boldsymbol{z}_i \geq \epsilon), \tag{9}$$

$$z_k^{(\{\varphi_l\}_{l=1}^{L}, m)} = (\sum_{l=1}^{L} \mathbf{1}(z_k^{\varphi_l}) \geq m), \tag{10}$$

$$\begin{aligned} z_k^{G_{[a,b]}\varphi} = \bigwedge_{l=0}^{K-1} ([t_l, t_{l+1}] \\ \cap [t_k + a - \epsilon_t, t_{k+1} + b] \neq \emptyset \Rightarrow z_l^{\varphi}), \end{aligned} \tag{11}$$

$$\begin{aligned} z_k^{F_{[a,b]}\varphi} = (t_{k+1} - t_k \leq b - a - \epsilon_t) \wedge \bigvee_{l=0}^{K-1} ([t_l, t_{l+1}] \\ \cap [t_{k+1} + a, t_k + b - \epsilon_t] \neq \emptyset \wedge z_l^{\varphi}), \end{aligned} \tag{12}$$

where ϵ_t is a positive value that adds robustness to the timimg, $\Rightarrow$ is the *implication* operator defined as $\varphi_1 \Rightarrow \varphi_2$ iff $\neg\varphi_1 \vee \varphi_2$. The construction laws for *Until* (as a combination of (11) and (12)) and Boolean operators are omited and as reported in [15]. By using these and (9)-(12), we can recursively construct z_0^{φ} that encodes the entire formula φ.

We prove the aforementioned soundness property by induction. We start from the predicate μ_i. Intuitively, if $z_k^{\mu_i}$ defined in (9) is true, then any σ_i that satisfies $\sigma_i(t_k) \in B_\infty(\sigma_{i,(k)}, \epsilon)$ and $\sigma_i(t_{k+1}) \in B_\infty(\sigma_{i,(k+1)}, \epsilon)$ also satisfies μ_i at t_k and t_{k+1}. Since $\sigma_i(t)$ is monotonically increasing, μ_i is satisfied at all $t \in [t_k, t_{k+1}]$. The encoding for the counting formula (10) is sound by definition. The proof for other Boolean and temporal operators is similar as in [15]. Hence, the LCCF constructed above is sound.

c) Sum of travel time encoding: When using makespan t_K as the cost function, we do not need further encoding. For sum of travel time, we introduce additional variables T_i with the following constraints to enforce (8):

$$\begin{aligned} (\sigma_{i,(k)} \leq [g_i^1, \dots, g_i^{M_i}]^\top \mathbf{z}_i) \wedge (T_i \geq t_{k+1}) \\ \vee (\sigma_{i,(k)} \geq [g_i^1, \dots, g_i^{M_i}]^\top \mathbf{z}_i) \wedge (T_i \leq t_k). \end{aligned} \tag{13}$$

B. Eliminating Disjunctions and Counting Operators

Finally, we eliminate disjunction and counting formulas using the Big M method. This introduces new binary variables proportional to the number of disjunctions and subformulas in counting formulas but does not increase the number of constraints. Details are omitted here for brevity.

C. Overall MILP Approach

Now we have transformed the constraints in (7) into linear constraints of the continuous and binary variables, which makes (7) a MILP problem. Solving this MILP gives us the optimal solution. The computational cost of solving a MILP depends on the number of binary variables, which is $O(K^2 \cdot |\varphi| + N \cdot M_i)$ in our encoding, where $|\varphi|$ is the number of operators in φ, and M_i is the number of paths per robot.

IV. Local Controllers

We use a local controller for each robot to track its selected reference path according to the sequence of TSJPs obtained by solving Problem 1. We make the following assumption on the local controller. Relaxation of it is discussed later.

Assumption 1: Consider a set of N robots which have reached the joint progress $\boldsymbol{\sigma}_{(k)}$ at time t_k within a distance of ϵ, i.e., $\boldsymbol{\sigma}(t_k) \in B_\infty(\boldsymbol{\sigma}_{(k)}, \epsilon)$. Given a target TSJP $(t_{k+1}, \boldsymbol{\sigma}_{(k+1)})$ that satisfies (6), the local controllers can always make robots reach the joint progress $\boldsymbol{\sigma}_{(k+1)}$ within the distance ϵ at t_{k+1}, i.e., $\boldsymbol{\sigma}(t_{k+1}) \in B_\infty(\boldsymbol{\sigma}_{(k+1)}, \epsilon)$.

Theorem 1: Under Assumption 1, given an RP-STL specification φ and a sequence of TSJPs from solving (7), the joint temporal profile $\boldsymbol{\sigma}$ executed by the local controller that tracks these TSJPs always satisfies φ, i.e., $[\mathbf{z}, \boldsymbol{\sigma}] \models \varphi$.

Proof: Since $\boldsymbol{\sigma}(0) = \boldsymbol{\sigma}_{(0)}$, the above theorem can be proved through induction based on Assumption 1. ∎

Assumption 1 is not restrictive as many existing techniques can provide us such a local controller, e.g., the control Lyapunov functions [23]. However, in practice it is possible that robots make tracking deviations $> \epsilon$, e.g., robots stalling or significantly reducing their speed. In such cases, it is hard to guarantee the satisfaction of the specification. However, under the following relaxed assumption, we can ensure that anomalies are detectable before the specification is violated:

Assumption 2: Consider a set of N robots which reaches the joint progress $\boldsymbol{\sigma}_{(k)}$ at time t_k within a distance of ϵ, i.e., $\boldsymbol{\sigma}(t_k) \in B_\infty(\boldsymbol{\sigma}_{(k)}, \epsilon)$. Given a target TSJP $(t_{k+1}, \boldsymbol{\sigma}_{(k+1)})$ that satisfies (6), the local controllers never make robots exceed the joint progress $\boldsymbol{\sigma}_{(k+1)}$ by ϵ before t_{k+1}, i.e., $\sigma_i(t_{k+1}) < \sigma_{i,(k+1)} + \epsilon$, $\forall i = 1, \ldots, N$.

Theorem 2: Under Assumption 2, given an RP-STL specification φ and a sequence of TSJPs from solving (7), if the joint progress made by the local controllers satisfies $\boldsymbol{\sigma}(t_k) \in B_\infty(\boldsymbol{\sigma}_{(k)}, \epsilon)$, then φ is not violated before t_{k+1}.

Proof: (sketch) To prove this, we need to show that when the TSJP is violated for the first time, the specification φ is still not violated. Assumption 2 ensures that $\neg\mu_i$, i.e., $\sigma_i(t) < \mathbf{b}_i^\top \mathbf{z}_i$, is never violated. So the violation can only come from $\sigma_i(t) \geq \mathbf{b}_i^\top \mathbf{z}_i$. For $G_{[a,b]}\varphi$, (11) requires satisfaction of φ over an interval containing $[a - \epsilon_t, b]$. Since $\boldsymbol{\sigma}$ is monotonically increasing, once φ is satisfied, it is satisfied all the time later. So the first violation can only happen at the beginning of the interval, which is earlier than a. Hence, $G_{[a,b]}\varphi$ is still not violated. Similarly, $F_{[a,b]}\varphi$ needs to be satisfied over an interval with non-empty intersection with $[a, b - \epsilon_t]$. The first violation can only happen at the beginning of the interval, which is earlier than b, so $F_{[a,b]}\varphi$ is still not violated. Same for $\varphi_1 U_{[a,b]} \varphi_2$, which can be seen as a combination of *always* and *eventually*. ■

In practice, we can check if every robot reaches the TSJP at each time stamp t_k. If the deviation is less than ϵ, then we continue to track the current sequence of TSJPs, ensuring that φ is not violated before t_{k+1}. If the deviation exceeds ϵ due to unforeseen factors, we replan taking these factors into account. Replanning is beyond the scope of this paper and will be explored in future work.

V. Experimental Evaluation

In this section, we evaluate the proposed approach on several benchmark scenarios and compare it with other methods. All experiments were run in the ARGoS simulator [24] on a Mac computer with M3 Pro CPU and 18 GB RAM. We use the ARA* algorithm [25] in SBPL [26] for motion planning.

A. Scenarios

a) Single robot: We first test the algorithm on a single-robot single-path scenario (Fig. 2a) introduced in [27], referred to as `stlcg`, where the robot needs to visit and stay in the red region for $20s$, then visit and stay in the green region for $20s$, and always avoid the blue region. Let the intervals on the path corresponding to the red and green regions be I_r and I_g. The specification is expressed using RP-STL as $F_{[0,T]}G_{[0,20]}(\sigma_1 \in I_r) \wedge F_{[0,T]}G_{[0,20]}(\sigma_1 \in I_g)$. Since always avoiding the blue region can be enforced by the motion planner, we omit it in the formula.

b) Interference: We then test the algorithm for interference constraints using a similar example as in [15] (Fig. 2b), referred to as `door`, where 4 robots need to pass through a narrow door to reach the other side of the map. To compare with [15], we assume each robot has only one reference path. The RP-STL specification is the conjunction of all interference constraints for all critical sections, see (3).

c) Counting formula: We extend the scenario in Example 1 (Fig. 2c), referred to as `bridge`, to include 9 robots, each with one reference path. The specification requires ≤ 3 robots can be on the bridge simultaneously, similar to (4).

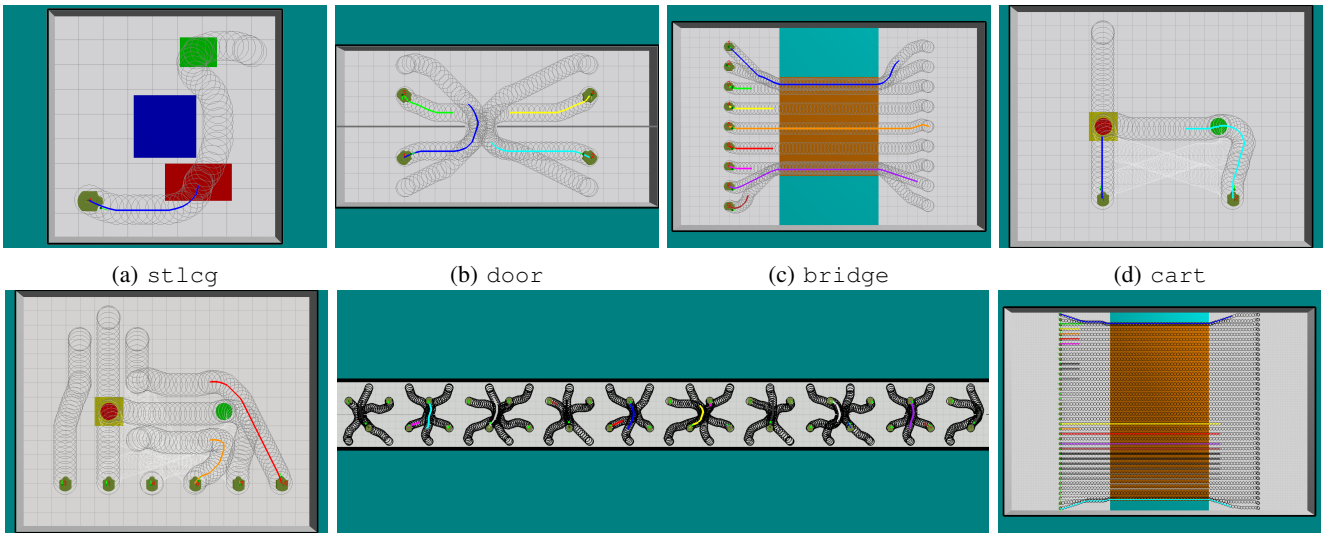
d) Task-specific requirements: We test our approach in a warehouse scenario with task-specific requirements. Here, we have work stations where packages are loaded into carts. In a first scenario (`cart`), once a cart is full, one robot must transport it to a truck for delivery, while a second robot must replenish the station with an empty cart. Each robot is assigned two paths p_i^1 and p_i^2, corresponding to transporting the full or empty cart, respectively. Task assignment is open in the specification, and is computed as part of the MILP solution. The specification states that the robot returning the empty cart cannot proceed to the station until the other robot has removed the full cart; and that the station cannot be empty for more than $20s$, in order to avoid congestion caused by incoming packages. Let $[l_i^j, u_i^j]$ be the interval on the path p_i^j where the robot is at the station (manipulating the cart). The specification is $\varphi_{int} \wedge (\varphi_{swap}^{12} \wedge \varphi_{swap}^{21})$ where φ_{int} is the conjunction of all interference constraints, and

$$\varphi_{swap}^{ii'} = (\sigma_{i'} < [M, l_{i'}^2] \cdot \mathbf{z}_{i'}^\top)\ U_{[0,T]}\ (\sigma_i \geq [u_i^1, -M] \cdot \mathbf{z}_i^\top) \wedge G_{[0,T]}(\sigma_i \geq [l_i^1, -M] \cdot \mathbf{z}_i^\top \Rightarrow F_{[0,20]}\ \sigma_{i'} \geq [-M, u_{i'}^2] \cdot \mathbf{z}_{i'}^\top).$$

We then increase the complexity of the scenario by introducing an escort task, referred to as `escort`. Since a robot may have limited visibility while carrying a cart, we require that at least two additional robots accompany it when it is carrying the cart. We introduce 4 more robots r_3, r_4, r_5, r_6 for the escorting task. Each robot is assigned multiple paths corresponding to which cart it escorts. The RP-STL specification is $\varphi_{int} \wedge (\varphi_{swap}^{12} \wedge \varphi_{swap}^{21}) \wedge \varphi_{esc}$, where φ_{esc} (omitted for brevity, see video attachment) requires that at least two unladen robots are alongside each cart-laden robot.

B. Comparison with other methods

We compare our method with the piece-wise linear (PWL) path method [15] and gradient-based methods. For `stlcg` and `door`, we compare with an enhanced gradient-based method from [5] with gradient computed analytically using STLCG [27]. In other scenarios, the PWL method is not applicable because the STL syntax in [15] does not support temporal operators involving multiple robots. For these cases, we compare with the gradient-based method from [13], which also employs the counting semantics. However, this method cannot avoid inter-agent collision, so we simplify the problem for [13] by omitting these constraints (which are not omitted in our approach). For all gradient-based methods, we set the time horizon $T = 50$ and the time step $\Delta t = 1$. We did not compare with MILP-based MPC methods, e.g., [3], as the authors of [15] have shown that their method outperforms [3] in these scenarios. The comparison results in terms of runtime are given in Table I. Note that for the PWL method, we only show the results of using sum of travel time (STT) as the objective function, as it is faster than using makespan. For gradient-based methods, the objective is to maximize the robustness and minimize the energy cost.



(a) `stlcg` (b) `door` (c) `bridge` (d) `cart`

(e) `escort` (f) `door` (32 robots) (g) `bridge` (40 robots)

Fig. 2: Experiment results. Black and white circles are waypoints for the selected and unselected paths. Solid lines are the paths between the first two TSJPs. In (d), (e), the red and green circles are the full and empty carts, respectively. Videos of the experiments can be found at https://www.amazon.science/scalable-multi-robot-task-allocation-and-coordination-under-signal-temporal-logic-specifications.

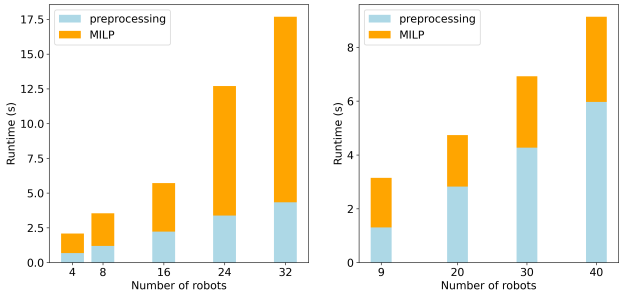
Scenarios	Ours (MS)	Ours (STT)	PWL (STT)	Gradient
`stlcg`	0.612	N/A	**0.243**	1.211
`door`	**2.087**	2.804	22.49	fail
`bridge`	**3.147**	3.914	N/A	130.4
`cart`	**1.652**	2.563	N/A	64.31
`escort`	**32.57**	247.6	N/A	269.6

TABLE I: Runtime comparison (s). MS and STT refer to the use of makespan and sum of travel time as the objective.

For all the scenarios above, our approach finds the optimal solution, with the local controller tracking the TSJPs on time. The planned paths and the first two TSJPs for each scenario (using makespan) are shown in Fig. 2. Videos can be found in the supplementary material. We can see that our approach successfully satisfies constraints involving reach and avoid (`stlcg`), interference (`door`), density (`bridge`), and real-world requirements with complex STL specification and task allocation (`cart` and `escort`). It significantly outperforms other methods in terms of runtime for all the multi-robot scenarios, with minimal compromise on solution quality. The runtime of our method consists of two parts: the preprocessing phase (motion planning, interference computation, MILP encoding, etc) and the MILP solving phase. Although the motion planner is currently run sequentially for each path, this step could be greatly accelerated by parallel computing. In the `stlcg` case, the majority of the time is spent on preprocessing, with the MILP solving taking only $0.057s$.

C. Scalability

We further test the scalability of our approach using the `door` and `bridge` scenarios. In the `door` case, we increase the number of robots and doors proportionally (as shown in Fig. 2f) so that the number of critical sections increases linearly. In the `bridge` case, we increase the number of robots, the capacity of the bridge, and the width



(a) `door` (b) `bridge`

Fig. 3: Runtime for different number of robots.

of the bridge proportionally (as shown in Fig. 2g) so that robots can always cross the bridge in 3 waves while the number of critical sections is kept constant. Testing results are shown in Fig. 3. The computation time for MILP does not increase significantly with the growth in the number of robots, especially in the `bridge` case which uses the counting constraints. Although adding robots will introduce more binary variables in the MILP, the additional constraints in these scenarios are well-structured and weakly coupled. So solvers such as Gurobi can exploit this structure to reduce computation time. The time for motion planning increases linearly with the number of robots but can be parallelized.

VI. Conclusion

We propose an algorithm to operate a multi-robot system subject to STL specifications. Compared with other methods in the literature, our approach significantly reduces computational cost by decoupling of task allocation and coordination from motion planning and control while maintaining formal guarantees. Experimental results demonstrate efficiency and scalability to large robot teams and complex specifications.

REFERENCES

[1] A. Pnueli, "The temporal logic of programs," in *18th annual symposium on foundations of computer science (sfcs 1977)*. ieee, 1977, pp. 46–57.

[2] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International symposium on formal techniques in real-time and fault-tolerant systems*. Springer, 2004, pp. 152–166.

[3] V. Raman, M. Maasoumy, and A. Donzé, "Model predictive control from signal temporal logic specifications: A case study," in *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, 2014, pp. 52–55.

[4] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 772–779.

[5] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1235–1240.

[6] I. Haghighi, N. Mehdipour, E. Bartocci, and C. Belta, "Control from signal temporal logic specifications with smooth cumulative quantitative semantics," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4361–4366.

[7] Y. Chen, X. C. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global ltl specifications," in *2011 50th IEEE conference on decision and control and european control conference*. IEEE, 2011, pp. 2718–2723.

[8] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The international journal of robotics research*, vol. 37, no. 7, pp. 818–838, 2018.

[9] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[10] 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.

[11] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2021.

[12] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Planning of heterogeneous multi-agent systems under signal temporal logic specifications with integral predicates," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1375–1382, 2021.

[13] W. Liu, K. Leahy, Z. Serlin, and C. Belta, "Robust multi-agent coordination from catl+ specifications," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3529–3534.

[14] Z. Liu, J. Dai, B. Wu, and H. Lin, "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 2516–2521.

[15] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] P. Forte, A. Mannucci, H. Andreasson, and F. Pecora, "Online task assignment and coordination in multi-robot fleets," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4584–4591, 2021.

[20] A. Mannucci, L. Pallottino, and F. Pecora, "On provably safe and live multirobot coordination with online goal posting," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1973–1991, 2021.

[21] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[22] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[23] E. J. Rodríguez-Seda, C. Tang, M. W. Spong, and D. M. Stipanović, "Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing," *The International Journal of Robotics Research*, vol. 33, no. 12, pp. 1569–1592, 2014.

[24] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[25] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," *Advances in neural information processing systems*, vol. 16, 2003.

[26] M. Likhachev, "Search-based planning with motion primitives," 2010.

[27] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 356–370, 2023.