Efficient Coordination and Synchronization of Multi-Robot Systems Under Recurring Linear Temporal Logic

Davide Peron[1], Victor Nan Fernandez-Ayala[2], Eleftherios E. Vlahakis[2], and Dimos V. Dimarogonas[2]

***Abstract*— We consider multi-robot systems under recurring tasks formalized as linear temporal logic (LTL) specifications. To solve the planning problem efficiently, we propose a bottom-up approach combining offline plan synthesis with online coordination, dynamically adjusting plans via real-time communication. To address action delays, we introduce a synchronization mechanism ensuring coordinated task execution, leading to a multi-agent coordination and synchronization framework that is adaptable to a wide range of multi-robot applications. The software package is developed in Python and ROS2 for broad deployment. We validate our findings through lab experiments involving nine robots showing enhanced adaptability compared to previous methods. Additionally, we conduct simulations with up to ninety agents to demonstrate the reduced computational complexity and the scalability features of our work.**

I. Introduction

Successful coordination and synchronization in multi-agent systems (MAS) is essential for completing complex, interdependent tasks that individual agents cannot handle alone, making it one of the key challenges in robotic applications such as in logistics [1] and precision agriculture [2]. As the number of agents and the complexity of tasks increase, ensuring scalability and conflict-free operation becomes more challenging, especially when tasks are expressed through temporal logic constraints. Therefore, finding an efficient and scalable solution is necessary to manage computational complexity and maintain robust performances.

Linear Temporal Logic (LTL) is a logical formalism suitable for defining linear-time properties, widely used in formal verification, particularly in computer systems, and increasingly in robotics for specifying tasks in MAS [3], [4]. The introduction of Product Büchi Automata (PBA) has significantly advanced LTL compliance in MAS by systematically generating control strategies [5]. However, scalability remains a challenge as task complexity and the number of agents increase. Many recent approaches address this computational limitation by avoiding the direct construction of the PBA and relying on alternative control techniques, such as the sampling-based synthesis in [6], [7] or reinforcement learning (RL) techniques in [8]. Although efficient, these techniques introduce a stochastic element to planning, whereas we aim to maintain the deterministic guarantees of graph search methods [5]. Other approaches focus on task assignment to individual agents to limit the exponential growth of the state space, rather than using centralized planning [9]. These methods often target specific robotic functions, such as motion [10], which hinders their applicability to more complex tasks, given the specificity of their context. Recent efforts, such as [11], introduce a task grammar that enables flexible, high-level task specification for heterogeneous agents, alleviating the need for explicit task assignments while maintaining correctness guarantees. Furthermore, synchronization mechanisms are becoming increasingly critical to ensure coordinated actions in MAS [12].

In [13] local tasks defined by syntactically cosafe LTL (sc-LTL) formulas [14] are considered, with agents navigating a grid-structured workspace and performing local, collaborative, and assistive actions. An offline planner generates initial plans for each agent, which are then adapted through a Request, Reply, and Confirmation cycle. The latter utilizes the PBA to incorporate assistive actions into the agents' plans. While robust, this approach is limited to finite-time tasks defined by sc-LTL formulas. In this work, we adapt [13] by addressing its limitations and enhancing computational efficiency. Unlike the earlier study, which relied solely on simulations and did not fully capture edge cases, our work integrates both experiments and more exhaustive simulations. Specifically, we introduce a region of interest (ROI) based representation to reduce the size of the finite transition system (FTS) [5] (Sec. III-A.1), and subsequently define a new subclass of LTL called recurring LTL to specify tasks that repeat infinitely often (Sec. III-A.4). In Alg. 2 we eliminate the use of the PBA for plan adaptation in favor of the simpler FTS. Additionally, we reduce the complexity of selecting collaborative agents by using filtering Proc. 1 to warm-start the underlying mixed integer program (MIP) [15]. Lastly, we present a robust synchronization mechanism within the ROS2 [16] framework that further enables effective collaboration among agents (Sec IV-F). Our approach is computationally efficient for real-world usage as proven with a team composed of nine commercially available robotic platforms (Sec. V-C) and suitable for large-scale robotic deployments as shown by a simulation with ninety robots (see Sec. V-D). The code is available at [17].

II. Preliminaries

A. LTL and Büchi Automaton

Atomic propositions are Boolean variables that can either be true or false. An LTL formula is defined over a set of atomic propositions (Ψ), the Boolean connectors: negation

This work was supported by the ERC CoG LEAFHOUND, the EU CANOPIES project, the Knut and Alice Wallenberg Foundation (KAW) and the Digital Futures Smart Construction project.

[1] Department of Information Engineering, University of Padova, 35122, Padova, Italy. Email: davide.peron.3@studenti.unipd.it

[2] Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 10044, Stockholm, Sweden. Emails: {vnfa,vlahakis,dimos}@kth.se

($\neg$), conjunction ($\wedge$), and the temporal operators, *next* ($\bigcirc$) and *until* (U). It is specified according to the following syntax [5]: $\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$ where $a \in \Psi$, and $\top \triangleq True$. Operators *always* ($\Box$), *eventually* ($\Diamond$) can be derived from the syntax above [5, Ch. 5]. The satisfaction of an LTL formula φ is achieved over words and the language of such words can be captured through a nondeterministic Büchi automaton (NBA) [18], defined as $\mathcal{B} = \left(S, S_0, 2^{\Psi}, \delta, \mathcal{F}\right)$, where S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, 2^{Ψ} is the set of all alphabets, $\delta : S \times 2^{\Psi} \to 2^{S}$ is the transition function, $\mathcal{F} \subseteq S$ is the set of accepting states.

B. LTL Planning

We briefly highlight the key points of the initial planning (implemented offline), which was developed in [19]. The planner node takes as input an LTL task φ, and an FTS $\mathcal{T}_w$. First, φ is used to generate the NBA $\mathcal{B}_\varphi$ via the LTL2BA software [20]. Building upon the work of [21], a PBA $\mathcal{A}_\mathcal{P}$ [19] is built, then through model checking techniques [3] the optimal run that satisfies the given LTL task and the corresponding sequence of actions are found. For all the details we refer to [19], [21], [3], [13].

III. System Setup

Let a group of heterogeneous agents $\mathcal{N} = \{a_i \mid i = 1, 2, ..., N\}$ operate within a partially known workspace. Each agent can execute primitive actions that may require assistance from others. The agents are connected via a shared network. Within this framework, agents can directly exchange messages with any other agent in the workspace.

A. System description

1) Motion Transition System: Agent a_i's motion within the workspace is modeled as an FTS. Our approach focuses on a set of ROIs while a low-level controller handles obstacle avoidance and inter-region movement. This approach significantly reduces computational complexity compared to a fully partitioned workspace but sacrifices the ability to track, at any time, the agent's exact location within the FTS. Each agent a_i is aware only of the set of M^{a_i} ROIs, denoted by $\Pi^{a_i}_{\mathcal{M}} = \{\pi^{a_i}_1, \pi^{a_i}_2, ..., \pi^{a_i}_{M^{a_i}}\}$. The FTS assigned agent a_i is:

$$\mathcal{T}^{a_i}_{\mathcal{M}} \triangleq \left(\Pi^{a_i}_{\mathcal{M}}, \Pi^{a_i}_{\mathcal{M},0}, \Psi^{a_i}_{\mathcal{M}}, \Sigma^{a_i}_{\mathcal{M}}, \longrightarrow^{a_i}_{\mathcal{M}}, \mathrm{L}^{a_i}_{\mathcal{M}}, \mathrm{T}^{a_i}_{\mathcal{M}}\right), \quad (1)$$

where $\Pi^{a_i}_{\mathcal{M},0} \in \Pi^{a_i}_{\mathcal{M}}$ is the initial ROI, $\Psi^{a_i}_{\mathcal{M}}$ is the set of atomic propositions describing the properties of the workspace, $\Sigma^{a_i}_{\mathcal{M}}$ is the set of movement actions, $\longrightarrow^{a_i}_{\mathcal{M}} \subseteq \Pi^{a_i}_{\mathcal{M}} \times \Sigma^{a_i}_{\mathcal{M}} \times \Pi^{a_i}_{\mathcal{M}}$ is the transition relation, $\mathrm{L}^{a_i}_{\mathcal{M}} : \Pi^{a_i} \to 2^{\Psi^{a_i}_{\mathcal{M}}}$ is the labeling function, indicating the properties held by each ROI, and $\mathrm{T}^{a_i}_{\mathcal{M}} : \longrightarrow^{a_i}_{\mathcal{M}} \to \mathbb{R}^+$ is the transition time function, representing the estimated time necessary for each transition.

2) Action Model: In addition to its movement actions, agent a_i can perform actions $\Sigma^{a_i}_{\mathscr{A}} \triangleq \Sigma^{a_i}_{l} \cup \Sigma^{a_i}_{c} \cup \Sigma^{a_i}_{h}$, where $\Sigma^{a_i}_{l}$ are *local* actions performed independently, $\Sigma^{a_i}_{c}$ are *collaborative* actions requiring assistance from other agents, and $\Sigma^{a_i}_{h}$ are *assisting* actions carried out to help others. Lastly, $\sigma_0 = None \in \Sigma^{a_i}_{l}$ indicates that a_i remains idle. The action model for agent a_i is defined as the tuple:

$$\mathscr{A}^{a_i} \triangleq \left(\Sigma^{a_i}_{\mathscr{A}}, \Psi^{a_i}_{\mathscr{A}}, \mathrm{L}^{a_i}_{\mathscr{A}}, \mathrm{Cond}^{a_i}, \mathrm{Dura}^{a_i}, \mathrm{Depd}^{a_i}\right), \quad (2)$$

where $\Psi^{a_i}_{\mathscr{A}}$ is the set of atomic propositions, $\mathrm{L}^{a_i}_{\mathscr{A}}$ is the labeling function as in [13], Cond^{a_i} is the region properties required to execute an action, Dura^{a_i} is the action duration, with $\mathrm{Dura}^{a_i}(\sigma_s) = T_s > 0$, and $\mathrm{Depd}^{a_i} : \Sigma^{a_i}_{\mathscr{A}} \to 2^{\Sigma^{\sim a_i}_{h}} \times 2^{\Pi^{\mathcal{N}}}$ denotes the dependence function, where $\Sigma^{\sim a_i}_{h}$ is the set of *external* assisting actions that agent a_i depends on, and $\Pi^{\mathcal{N}} = \cup_{a_i \in \mathcal{N}} \Pi^{a_i}_{\mathcal{M}}$. Given $\sigma_c \in \Sigma^{a_i}_{c}$, we define the set of actions involved in a collaboration as:

$$\mathcal{C}(\sigma_c) = \{\sigma_c\} \cup \mathrm{Depd}^{a_i}(\sigma_c). \quad (3)$$

Definition 1. A collaboration is considered successful if all actions involved are synchronized; i.e. to complete $\sigma_c \in \Sigma^{a_i}_{c}$, it is necessary that all actions in $\mathcal{C}(\sigma_c)$ start simultaneously.

3) Agent Transition System: The planner in Sec. II-B requires to define agent a_i's FTS by combining (1) and (2).

Definition 2. Given $\mathcal{T}^{a_i}_{\mathcal{M}}$ and $\mathscr{A}^{a_i}$, a valid FTS for agent a_i, according to [5], can be constructed as follows:

$$\mathcal{T}^{a_i}_{\mathcal{G}} \triangleq \left(\Pi^{a_i}_{\mathcal{G}}, \Pi^{a_i}_{\mathcal{G},0}, \Psi^{a_i}_{\mathcal{G}}, \Sigma^{a_i}_{\mathcal{G}}, \longrightarrow^{a_i}_{\mathcal{G}}, \mathrm{L}^{a_i}_{\mathcal{G}}, \mathrm{T}^{a_i}_{\mathcal{G}}\right), \quad (4)$$

where $\Pi^{a_i}_{\mathcal{G}} = \Pi^{a_i}_{\mathcal{M}} \times \Sigma^{a_i}_{\mathscr{A}}$ is the set states, $\Pi^{a_i}_{\mathcal{G},0} = \langle \Pi^{a_i}_{\mathcal{M},0}, None \rangle$ is the initial state, $\Psi^{a_i}_{\mathcal{G}}$ is the set of atomic propositions, $\Sigma^{a_i}_{\mathcal{G}} = \Sigma^{a_i}_{\mathcal{M}} \bigcup \Sigma^{a_i}_{\mathscr{A}}$, with $\Sigma^{a_i}_{\mathcal{G},l} = \Sigma^{a_i}_{\mathcal{M}} \bigcup \Sigma^{a_i}_{l}$, $\longrightarrow^{a_i}_{\mathcal{G}}$ is the transition relation, $\mathrm{L}^{a_i}_{\mathcal{G}}$ is the labeling function, and $\mathrm{T}^{a_i}_{\mathcal{G}}$ is the transition estimated duration [13].

As in [13], the path is denoted by $\tau^{a_i} = \pi^{a_i}_{\mathcal{G},0}\pi^{a_i}_{\mathcal{G},1}\ldots$ its trace by $trace(\tau^{a_i}) = L^{a_i}_{\mathcal{G}}(\pi^{a_i}_{\mathcal{G},0})L^{a_i}_{\mathcal{G}}(\pi^{a_i}_{\mathcal{G},1})\ldots$ and, the associated sequence of actions by $\rho^{a_i} = \sigma^{a_i}_0, \sigma^{a_i}_1, \ldots$, i.e., the actions that allow transition between the states of τ^{a_i}.

4) Task Specification: Our focus is on implementing recurring tasks i.e., tasks that repeat infinitely often. We will consider the following syntax $\varphi' ::= \top \mid a \mid \neg a \mid \varphi'_1 \wedge \varphi'_2 \mid \Diamond\varphi'$, and for agent a_i we define the recurring task as

$$\varphi^{a_i}_r = \varphi'_1 \wedge \Box\Diamond\varphi'_2. \quad (5)$$

Note that φ'_2 cannot start with $\Diamond$ to guarantee the validity of the LTL formula. Given any satisfying word of $\varphi^{a_i}_r$, inserting a detour i.e., a finite sequence of states, between two consecutive states results in a satisfying word.

Problem 1. Given $\mathcal{T}^{a_i}_{\mathcal{G}}$ and the locally assigned task $\varphi^{a_i}_r$, design a distributed coordination and synchronization scheme such that $\varphi^{a_i}_r$ is satisfied for all $a_i \in \mathcal{N}$.

IV. Main Results

Our approach consists of an offline initial planning and an online adaptation phase. The latter involves agents exchanging request, reply, and confirmation messages to identify the best candidates for assisting with collaborative actions.

A. Offline Initial Planning

Given the FTS of an agent $\mathcal{T}^{a_i}_{\mathcal{G}}$ and the locally assigned task $\varphi^{a_i}_r$, the offline planner (see Sec. II-B) outputs the

Algorithm 1: Check in horizon and Request

Input : l, ρ^{a_i}, H^{a_i}, T_{rem}
Output : $\mathbf{Req}^{a_i}$

$s = 1$, $T_c^{a_i} = T_{rem}$
while $T_c^{a_i} < H^{a_i}$ **do**
  if $\rho^{a_i}[l+s] \in \Sigma_c^{a_i}$ **then**
    forall $\sigma_d \in \mathrm{Depd}^{a_i}(\rho^{a_i}[l+s])$ **do**
      $\pi_d =$`Choose_ROI()`
      add $(\sigma_d,\ \pi_d, T_c^{a_i})$ to $\mathbf{Req}^{a_i}$
    return $\mathbf{Req}^{a_i}$
  $T_c^{a_i} = T_c^{a_i} + T_{\mathcal{G}}^{a_i}(\rho^{a_i}[l+s])$, $s = s+1$
return $\emptyset$

optimal initial plan for the agent, $\tau_{init}^{a_i}$, namely a sequence of states that satisfy $\varphi_r^{a_i}$, and the associated sequence of actions $\rho_{init}^{a_i}$. The resulting plan guarantees that $trace\,(\tau_{init}^{a_i}) \models \varphi_r^{a_i}$ where the satisfaction relation is defined in Sec. II-A.

B. Collaboration Request

Given a collaborative action $\sigma_c^{a_i} \in \Sigma_c^{a_i}$, the request message sent by agent a_i to all the other agents is

$$\mathbf{Req}^{a_i} = \{(\sigma_d, \pi_d, T_c^{a_i}) \forall \sigma_d \in \mathrm{Depd}^{a_i}(\sigma_c^{a_i})\}.$$

Here, σ_d are the assistive actions required to complete $\sigma_c^{a_i}$, π_d are the regions where σ_d takes place, $T_c^{a_i}$ is the time required before $\sigma_c^{a_i}$ starts according to a_i's plan. Algorithm 1 generates $\mathbf{Req}^{a_i}$ by assuming a generic time instant where agent a_i is in state $\pi_{\mathcal{G},l}^{a_i}$, the l-th element of its plan τ^{a_i} (possibly different from $\tau_{init}^{a_i}$ due to modification from the application of our approach). The agent executes action $\sigma_l^{a_i}$, corresponding to $\rho^{a_i}[l]$. The algorithm inputs the current state and action index (l), the action sequence ρ^{a_i}, the time remaining for the current action (T_{rem}), and the horizon length (H^{a_i}), considered to be of a similar order of magnitude of the actions in the experimental scenario. The algorithm checks future actions within the horizon for any collaborative tasks. If found, it creates a request, otherwise, it returns an empty set, sending no message. The `Choose_ROI` function selects an ROI based on the specific experimental setup. As a result, we cannot provide a general implementation; however, the specific one used in our experiment is detailed in Sec. V-A. Lastly, at the end of each iteration, we update $T_c^{a_i}$ by adding the duration of the last action.

C. Reply

Given a request for collaboration, $\mathbf{Req}^{a_i}$, the reply message sent by agent a_j to a_i is

$$\mathbf{Reply}^{a_j} = \{(\sigma_d, \pi_d, b_d^{a_j}, t_d^{a_j}) \forall (\sigma_d, \pi_d, T_c^{a_i}) \in \mathbf{Req}^{a_i}\}.$$

Here, $b_d^{a_j}$ is a Boolean variable indicating that agent a_j can execute σ_d at region π_d and at time $t_d^{a_j}$.

Algorithm 2 generates $\mathbf{Reply}^{a_j}$ by assuming a request arrives when agent a_j is in state $\pi_{\mathcal{G},m}^{a_j}$, the m-th element of its plan τ^{a_j}, currently executing action $\sigma_m^{a_j}$. Alg. 2 takes as

Algorithm 2: Reply of a_j to a request from a_i

Input : $\mathbf{Req}^{a_i}$, m, τ^{a_j}, $\mathcal{T}_{\mathcal{G}}^{a_j}$, $\overline{T}^{a_j}$, T_{rem}
Output : $\mathbf{Reply}^{a_j}$, D^{a_j}

1 **forall** $(\sigma_d, \pi_d, T_c^{a_i}) \in \mathbf{Req}^{a_i}$ **do**
2 **if** $\overline{T}^{a_j}$ *and* $\langle \pi_d, \sigma_d \rangle \in \Pi_{\mathcal{G}}^{a_j}$ **then**
3 $\pi_{\mathcal{G},init}^{a_j} = \tau^{a_j}[m+1]$, $\pi_{\mathcal{G},fin}^{a_j} = \tau^{a_j}[m+2]$
4 $\pi_{\mathcal{G},targ}^{a_j} = \langle \sigma_d, \pi_d \rangle$
5 (D_1, C_1)=`Dijkstra`$(\mathcal{T}_{\mathcal{G}}^{a_j},\ \pi_{\mathcal{G},init}^{a_j},\ \pi_{\mathcal{G},targ}^{a_j})$
6 (D_2, C_2)=`Dijkstra`$(\mathcal{T}_{\mathcal{G}}^{a_j},\ \pi_{\mathcal{G},targ}^{a_j},\ \pi_{\mathcal{G},fin}^{a_j})$
7 $D^{a_j}(\sigma_d) = D_1 + D_2[1:end]$
8 $t_d^{a_j} = \sum_{i=0}^{len(C_1)-2} C_1[i]$
9 add $(\sigma_d, \pi_d, \top, T_{rem} + t_d^{a_j})$ to $\mathbf{Reply}^{a_j}$
10 **else**
11 add $(\sigma_d, \pi_d, \bot, K)$ to $\mathbf{Reply}^{a_j}$
12 **return** D^{a_j}, $\mathbf{Reply}^{a_j}$

inputs the request from a_i, the current state and action index (m), the plan τ^{a_j}, the FTS $\mathcal{T}_{\mathcal{G}}^{a_j}$, the availability indicator $\overline{T}^{a_j}$ ($\top$ if available, $\bot$ if collaborating), and the time remaining for the current action $T_{rem} = \max(T_{\mathcal{G}}^{a_j}(\sigma_m^{a_j}) - \Delta t, 0)$ with Δt the elapsed time from the start of $\sigma_m^{a_j}$. It outputs $\mathbf{Reply}^{a_j}$ and a detour dictionary D^{a_j}, which includes possible path detours, e.g., $D^{a_j}(\sigma_d)$ represents a detour that includes the state $\langle \pi_d, \sigma_d \rangle$. The algorithm checks if a_j can assist with any requested actions, if so, from lines 3 to 7, it builds the detour between initial ($\pi_{\mathcal{G},init}^{a_j}$) and final ($\pi_{\mathcal{G},fin}^{a_j}$) states, passing through a target ($\pi_{\mathcal{G},targ}^{a_j}$) state. It uses a modified `Dijkstra` algorithm that returns the shortest path D and an array C with the associated action costs. The first call yields D_1 and C_1 i.e., the path and cost from the initial to the target state, similarly the second call produces D_2 and C_2 from the target to the final state. Then it merges the two paths into the detour saving it in the dictionary D^{a_j}. Lastly, it calculates the time $t_d^{a_j}$ for a_j to start the assistive action, excluding the current action's completion time. Note that the summation avoids adding the cost of σ_d, stopping at $len(C_1) - 2$. If a state is not feasible or a_j is occupied, it sets $t_d^{a_j} = K$, where $K \gg T_c^{a_i}$; in practice, we set $K = 10T_c^{a_i}$, but any finite value would work.

Remark 1. Note that $t_d^{a_j}$, $T_c^{a_i}$ are nominal times, but an agent might be affected by finite delays due to actions taking longer than expected in experiments.

The definition of $T_{rem} \geq 0$ ensures nonnegative times from action delays. Note a_i replies negatively to its request.

D. Confirmation

Given $\mathbf{Req}^{a_i}$, the confirmation message sent by a_i to a_j, based on $\mathbf{Reply}^{a_j}$ $\forall a_j \in \mathcal{N}$, has the following structure:

$$\mathbf{Conf}_{a_j}^{a_i} = \{(\sigma_d, \pi_d, c_d^{a_j}, T_d^{a_j})\ \forall (\sigma_d, \pi_d, T_c^{a_i}) \in \mathbf{Req}^{a_i}\}.$$

Here, $c_d^{a_j}$ is a Boolean variable indicating whether a_j is confirmed to perform σ_d at ROI π_d, and $T_d^{a_j}$ is the estimated start time for the collaboration. If a_j is not selected, $T_d^{a_j} < 0$.

To build the confirmation messages, the Boolean variables $\{c_d^{a_j},\ a_j \in \mathcal{N}\}$ must satisfy two constraints: 1) Each agent in $\mathcal{N}$ can be confirmed for at most one $(\sigma_d, \pi_d) \in \mathbf{Req}^{a_i}$; 2) Exactly one agent must be confirmed for each action (σ_d, π_d). Moreover, agents are selected based on their ability to perform the assisting action σ_d as close as possible to the target time $T_c^{a_i}$. Given $== M$, denote the assisting actions as $\{\sigma_d \mid d = 1, \ldots, M\}$. The problem of finding $\{c_d^{a_j}\}$ can be formulated as a MIP solved for $\mathcal{R} = \mathcal{N}$

$$\min_{\{c_d^{a_j}, a_j \in \mathcal{R}\}_{d=1}^{M}} \quad \sum_{d=1}^{M} \sum_{j=1}^{|\mathcal{R}|} c_d^{a_j} |t_d^{a_j} - T_c^{a_i}| \tag{6a}$$

$$\text{s.t.} \quad \sum_{d=1}^{M} b_d^{a_j} c_d^{a_j} \leq 1 \quad \forall a_j \in \mathcal{R} \tag{6b}$$

$$\sum_{j=1}^{|\mathcal{R}|} b_d^{a_j} c_d^{a_j} = 1 \quad \forall d \in \{1, ..., M\}. \tag{6c}$$

Once solved, if $c_d^{a_j} = \top$, $T_d^{a_j} = t_d^{a_j}$ otherwise, $T_d^{a_j} = -1$. To handle the complexity of (6), which grows exponentially with the number of binary variables involved, we propose a filtering procedure to reduce the number of agents involved while maintaining optimality.

Procedure 1 (*Filtering Procedure*)**.** 1) Define $\mathcal{N}_U \subseteq \mathcal{N}$ as the set of agents who can assist in at least one action i.e., $a_j \in \mathcal{N}_U$ if $b_d^{a_j} = \top$ for any $(\sigma_d, \pi_d, b_d^{a_j}, t_d^{a_j}) \in \mathbf{Reply}_{a_i}^{a_j}$. 2) For each $\sigma_d \in \mathbf{Req}^{a_i}$, sort agents in ascending order of $\Delta_d^{a_j} = |t_d^{a_j} - T_c^{a_i}|$ and store the first M for each σ_d in $\mathcal{N}_F$.

Theorem 1. *Consider the set of agents $\mathcal{N}$ and the M actions in $\mathbf{Req}^{a_i}$. Let $\mathcal{N}_F$ be the set of filtered agents given by Proc. 1. Assume that the MIP in* (6) *for $\mathcal{R} = \mathcal{N}$ is feasible. Then, the MIP in* (6) *for $\mathcal{R} = \mathcal{N}$ and for $\mathcal{R} = \mathcal{N}_F$ have identical optimal solutions.*

Proof. Let $d \in \{1, ..., M\}$ be the set of actions as denoted in Proc. 1 and rewrite the objective function in (6a) for $\mathcal{R} = \mathcal{N}$ as $\sum_{j=1}^{|\mathcal{N}|} c_1^{a_j} \cdot \Delta_1^{a_j} + \ldots + \sum_{j=1}^{|\mathcal{N}|} c_M^{a_j} \cdot \Delta_M^{a_j}$. For each term (action), define $\mathcal{N}_d$ as the set of M agents with the smallest $\Delta_d^{a_j}$ and then define $\mathcal{N}_F = \bigcup_{d=1}^{M} \mathcal{N}_d$. Rewrite the filtered objective function as $\sum_{a_j \in \mathcal{N}_1} c_1^{a_j} \cdot \Delta_1^{a_j} + \ldots + \sum_{a_j \in \mathcal{N}_M} c_M^{a_j} \cdot \Delta_M^{a_j}$. Let $\{c_d^{a_j}\}_{\mathcal{N}}^*$ and $\{c_d^{a_j}\}_{\mathcal{N}_F}^*$ be the optimal solutions of (6) for $\mathcal{R} = \mathcal{N}$, and $\mathcal{R} = \mathcal{N}_F$ respectively. Assuming the solutions differ, there must exist an agent $a_j \in \mathcal{N} \setminus \mathcal{N}_F$ so that $c_d^{a_j} = 1$ in $\{c_d^{a_j}\}_{\mathcal{N}}^*$ would result in a smaller value for the objective function compared to any agent $a_i \in \mathcal{N}_F$ where $c_d^{a_i} = 1$ in $\{c_d^{a_j}\}_{\mathcal{N}_F}^*$, implying $\sum_{j=1}^{|\mathcal{N}|} c_d^{a_j} \cdot \Delta_d^{a_j} < \sum_{a_j \in \mathcal{N}_d} c_d^{a_j} \cdot \Delta_d^{a_j}$. Simplifying the inequality under the condition that $c_d^{a_j} = 1$ for only one a_j leads to $\Delta_d^{a_j} < \Delta_d^{a_i}$. Since a_j would then appear in $\mathcal{N}_d$, it must belong to $\mathcal{N}_F$ by Proc. 1 which contradicts the assumption. Hence, the optimal solutions are identical. □

Based on the solution of (6) for $\mathcal{R} = \mathcal{N}_F$, $\mathbf{Conf}_{a_j}^{a_i} \forall\ a_j \in \mathcal{N}$ is built as follows: if $a_j \in \mathcal{N}_F$ and (6) is feasible, add $(\sigma_d, \pi_d, c_d^{a_j}, T_d^{a_j})$ to $\mathbf{Conf}_{a_j}^{a_i}$, otherwise add $(\sigma_d, \pi_d, \perp, -1)$. Lastly, a_i sends $\mathbf{Conf}_{a_j}^{a_i}$ to all $a_j \in \mathcal{N}$.

As in [13], the focus is on loosely coupled MAS where collaborations are sporadic compared to the total actions. This permits us to make the following assumption.

Assumption 1. *(Loosely Coupled System)* There exists a finite time $\mathbf{T} > 0$ such that for each agent $a_i \in \mathcal{N}$ and any collaborative action $\sigma_c^{a_i}$ requested at time $t_c > 0$, (6) for $\mathcal{R} = \mathcal{N}_F$ will attain a solution within $t_c + \mathbf{T}$ for $\sigma_c^{a_i}$.

Assumption 1 allows for collaboration delays if no immediate solution is found. A plan adaptation process repeats until a solution is achieved, ensuring eventual success, as discussed in Sec. IV-E. This assumption is reasonable for the tasks defined by (5). We next show that based on the availability of agents and the scarcity of collaborative actions, both task progress and assistance can be accomplished.

E. Plan Adaptation

After agent a_i sends the confirmation messages to all agents $a_j \in \mathcal{N}$, two scenarios arise. 1) If (6) for $\mathcal{R} = \mathcal{N}_F$ has no solution, $\sigma_c^{a_i}$ cannot be fulfilled based on the current replies. Agent a_i will delay $\sigma_c^{a_i}$ by adding a detour of duration T_{delay} (design parameter), using the self-loop $\sigma_0 = None$, and reattempt the request-reply-confirmation (RRC) cycle after this delay, while all the other agents proceed as planned and discard the detours. By Asm. 1, a solution will eventually be found even if the RRC cycle might be repeated multiple times. 2) If (6) for $\mathcal{R} = \mathcal{N}_F$ has a solution, each assisting agent a_j checks if it has been selected (i.e., $c_d^{a_j} = \top$ for any σ_d). If selected, the agent updates τ^{a_j} to include $D^{a_j}(\sigma_d)$ and sets $\overline{T}^{a_j} = \perp$. If not selected, the agent continues with its original plan. The requesting agent a_i sets $\overline{T}^{a_i} = \perp$. Note that for assisting agents, we will set $\overline{T}^{a_j} = \top$ at the end of the detour to ensure they continue their plans. For a_i, we will set $\overline{T}^{a_i} = \top$ after $\sigma_c^{a_i}$ is completed.

F. Time Synchronization

Suppose (6) for $\mathcal{R} = \mathcal{N}_F$ has a solution, establishing a collaboration between the requesting agent a_i and some assisting agents $a_j \in \mathcal{N}$. To successfully complete it, according to Def. 1 it is necessary to synchronize the execution of the actions. Let $\mathcal{S} \subseteq \mathcal{N}$ denote the set of agents involved in this collaboration. Once $a_j \in \mathcal{S} \setminus a_i$ is ready to execute σ_d, it sends a $\mathbf{Ready}^{a_j}$ message to a_i. When a_i is ready to execute $\sigma_c^{a_i}$ and has received all $\mathbf{Ready}^{a_j}$ messages, it sends a $\mathbf{Start}$ message to all a_j, ensuring all agents start simultaneously. The following Lemma underpins the synchronization process.

Lemma 1. *There exists a finite time $T_s \geq 0$ such that, if t is the time when $\mathbf{Conf}_{a_j}^{a_i}$ is sent, then by $t + T_s$ $\mathbf{Start}$ will be sent, and the agents will begin their collaboration.*

Proof. Consider an agent $a \in \mathcal{S} \setminus a_i$ in state $\pi_{\mathcal{G},init}^a$ at time t. After completing the assistive action σ_d , it transitions to $\pi_{\mathcal{G},fin}^a = \langle \pi_d, \sigma_d \rangle$ via a finite sequence $\rho_d \subset \rho^a$, where ρ^a is the action sequence associated with the plan τ^a. Each action $\sigma_i \in \rho_d$ has an effective duration $T_{\sigma_i} + d_{\sigma_i}$, where $T_{\sigma_i} = T_{\mathcal{G}}^a(\sigma_i) < \infty$ and $0 \leq d_{\sigma_i} < \infty$ is the delay (Rmk. 1). The

time before starting σ_d is $T_S^a = \sum_{i=0}^{|\rho_d|-2}(T_{\sigma_i} + d_{\sigma_i}) < \infty$. After T_S^a, a starts σ_d and sends $\mathbf{Ready}^a$. The same holds for a_i. Let $T_S = \max_a\{T_S^a\} < \infty$ for $a \in \mathcal{S}$. At $t + T_S$, all agents send $\mathbf{Ready}^a$, and a_i sends $\mathbf{Start}$, synchronizing the agents. □

G. Approach Summary

Theorem 2. *Given the set of agents $\mathcal{N}$, let $a_i \in \mathcal{N}$ build the FTS $\mathcal{T}_{\mathcal{G}}^{a_i}$ (4) and get assigned a recurring LTL task $\varphi_r^{a_i}$ (5). After constructing the initial plan as in Sec. IV-A implement the RRC cycle for any collaborative action as detailed in Alg. 1, Alg. 2, and Sec. IV-D. Then, apply the plan adaptation from Sec. IV-E, followed by the synchronization method in Sec. IV-F. Under Asm. 1 this approach solves Problem 1.*

Proof. Consider an agent $a_i \in \mathcal{N}$. The initial plan satisfies the task specification $\varphi_r^{a_i}$, as ensured by the planner (Sec. IV-A). As plan adaptation is permitted by the definition in (5), task satisfaction is guaranteed. Furthermore, under Asm. 1 all collaborations are established within a finite time.By Lemma 1, every collaboration will be successfully completed (Def. 1), as there exists a finite time $T_S \geq 0$ that ensures all actions in the collaboration start simultaneously at $t + T_S$. Therefore, $\varphi_r^{a_i}$ is satisfied. Since a_i can be any agent, this holds for all $a_i \in \mathcal{N}$, ensuring that the proposed approach solves Problem 1. □

H. Complexity Reduction

Compared to [13], by using the ROI representation instead of grid partitioning, we reduce the size of $\mathcal{T}_{\mathcal{M}}^{a_i}$, which also reduces the size of $\mathcal{T}_{\mathcal{G}}^{a_i}$. Additionally, in Alg. 2, we replace the more complex PBA $\mathcal{A}_{\mathcal{P}}$ with the simpler FTS $\mathcal{T}_{\mathcal{G}}^{a_i}$ to ensure task satisfaction, leveraging the properties given by (5) to maintain LTL compliance. This reduces our approach's overall complexity, particularly since the most expensive operations in Alg. 2 are the two calls to `Dijkstra`, which have a quadratic cost on the number of states. Lastly, the filtering procedure before solving the MIP ensures that only $M \leq |\mathcal{N}_F| \leq M^2$ agents are involved, offering substantial complexity reduction when $N \gg M$. Moreover, by Thm. 1, if $M = 1$ Proc. 1 returns the optimal agent, solving the MIP (6) directly under negligible computational cost.

V. Experimental Results

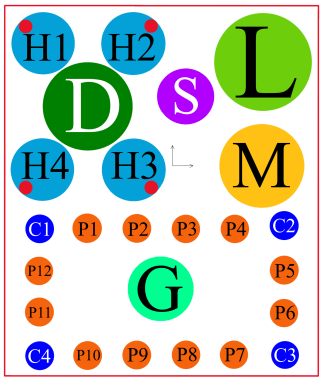
We conducted experiments, within the framework of the CANOPIES project [22], to show the advantages of our approach, most of which are based on the following system.

A. System description

We consider a workspace measuring $4.2\text{m} \times 5.2\text{m}$ (Fig. 1b), and two types of agents: Robotis Turtlebot3 Burger [23] and Hebi Rosie with robotic arm [24].

1) Turtlebot: It knows the ROIs: $C1-4$, $P1-12$, M, and G. The collaborative actions are *check_connection (cc)* at $C1$ or $C2$, *group (g)* at G, *remove_object (ro)* at M, the assistive actions are *help_check_connection (hcc)* at $C3$ or $C4$, *help_group (hg)* at G and the local actions are *patrol (p)* at $P1-12$, the movement related actions and, *None*.

(a) T. Turtlebot, B. Rosie



(b) Workspace abstraction

Fig. 1: Experimental setup

2) Rosie: It knows the ROIs: $H1-4$, D, L, M, and S. The collaborative action is *load (l)* at L, the assistive actions are *help_load (hl)* at L, *help_remove_object (hro)* at M and, the local actions are *harvest (h)* at $H1-4$, *manipulate (m)* at M, *deliver (d)* at D, *supervise (s)* at S, the movement related actions and, *None*.

3) Task specification: We consider a basic team composed of 3 Rosies and 6 Turtlebots. The recurring LTL tasks (5) assigned to the agents are as follows: $\varphi_r^{\text{rosie}_0} = \square\lozenge(h \wedge H1 \wedge \lozenge(h \wedge H3 \wedge \lozenge d))$, $\varphi_r^{\text{rosie}_1} = \square\lozenge(s \wedge \lozenge(l \wedge \lozenge m))$, $\varphi_r^{\text{rosie}_2} = \square\lozenge(h \wedge H2 \wedge \lozenge(h \wedge H4 \wedge \lozenge d))$, $\varphi_r^{\text{turtlebot}_0} = \square\lozenge(p \wedge P1 \wedge \lozenge(p \wedge P11))$, $\varphi_r^{\text{turtlebot}_1} = \square\lozenge(p \wedge P2 \wedge \lozenge(p \wedge P10))$, $\varphi_r^{\text{turtlebot}_2} = \square\lozenge(p \wedge P4 \wedge \lozenge(p \wedge P6))$, $\varphi_r^{\text{turtlebot}_3} = \square\lozenge(p \wedge P12 \wedge \lozenge(p \wedge P9 \wedge \lozenge(cc \wedge C1))$, $\varphi_r^{\text{turtlebot}_4} = \square\lozenge(p \wedge P3 \wedge \lozenge(p \wedge P8 \wedge \lozenge g))$ and, $\varphi_r^{\text{turtlebot}_5} = \lozenge(ro \wedge \lozenge(p \wedge P5 \wedge \lozenge(cc \wedge C2))) \wedge \square\lozenge(p \wedge P5 \wedge \lozenge(p \wedge P7))$. The starting ROIs are respectively $H1$, M, $H2$, $P1$, $P2$, $P4$, $P12$, $P3$, M. Lastly, we define `Choose_ROI`, which is relevant only when *check_connection* must be completed. If cc is completed in $C1$, then hcc is completed in $C3$. If cc is completed in $C2$, then hcc is completed in $C4$.

B. Complexity Reduction

We assess the computational gains discussed in Sec. IV-H.

1) Comparison between ROI and Grid Representation: We developed an ROI representation that focuses only on the necessary regions for the agent unlike previous approaches [13] which used a grid structure to partition the workspace. As shown in Tab. I, the grid was partitioned with cells sized to contain the specific robot (6×7 grid for Rosie and 20×25 for Turtlebots). The ROI representation led to a significant reduction in the number of states, achieving an average reduction of 88.7% compared to the grid representation. This

Agent	States in $\mathcal{T}_{\mathcal{M}}$	States Grid	Reduction
Rosie	8	42	81.0%
Turtlebot	18	500	96.4%

TABLE I: Computational gains of $\mathcal{T}_{\mathcal{M}}$ over a grid structure

Agent	States $\mathcal{T}_\mathcal{G}$	States $\mathcal{A}_\mathcal{P}$	Reduction
$\text{rosie}_{0,1,2}$	18	162	88.9%
$\text{turtlebot}_{0,1,2}$	37	148	75.0%
$\text{turtlebot}_{3,4}$	37	333	88.9%
turtlebot_{5}	37	592	93.8%

TABLE II: Computational gains by FTS against PBA

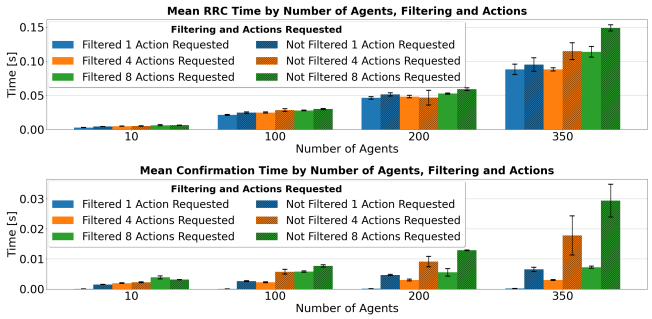


Fig. 2: Effects of agents filtering

reduction influences the size of the FTS $\mathcal{T}_\mathcal{G}$ and the PBA $\mathcal{A}_\mathcal{P}$. In the sequel, we focus on the ROI representation.

2) Gains of using the FTS: In Alg. 2, we use the FTS $\mathcal{T}_\mathcal{G}$, whereas [13] uses the PBA $\mathcal{A}_\mathcal{P}$. This results in an average reduction of 84.8% in the number of states across all agents. The PBA's state count increases with task complexity, hence, $turtlebot_5$, with the most complex task, benefits the most from using $\mathcal{T}_\mathcal{G}$ in `Dijkstra`. The results are in Tab. II.

3) MIP Filtering: Lastly, we analyzed the effect of the filtering procedure on the RRC cycle and on the confirmation step. This was tested in a centralized simulation, varying the number of agents and actions in a request. CycloneDDS [25] was used as the ROS2 middleware for stable communication between nodes. As shown in Fig. 2 (bottom plot), the filtering procedure significantly reduced the confirmation time, especially as the number of agents increased, while the time remained relatively constant for different numbers of actions. The greatest reduction occurs with a single requested action, where the MIP is solved by Proc. 1, with these results being barely visible in the plot due to their low values. However, this reduction in confirmation time has a limited impact on the overall RRC cycle, as shown in Fig. 2 (upper plot), where the bottleneck is the ROS2 communication. Significant improvements only appear with 350 agents, the maximum allowed by the workstation. This suggests that with a larger number of agents ($N \gg M$), the filtering procedure yields substantial gains.

C. Experimental Results

In this setup, we deployed our approach to the available hardware, in a decentralized way. For robot movement, we developed a model predictive controller [26] with control barrier function [27], [28] constraints for collision avoidance. Agent poses were tracked using the Qualisys Motion Capture System [29]. For the '*remove_object*' action, we used a visual servoing controller [30], [31], while all other actions were simulated. A video of the experiment is available at [32]. Fig. 3 shows the sequence of actions completed by each agent, demonstrating that all tasks were completed and that the approach synchronized collaborative and assistive actions to start simultaneously, even if some agents were ready earlier. Note that only non-movement actions are shown.

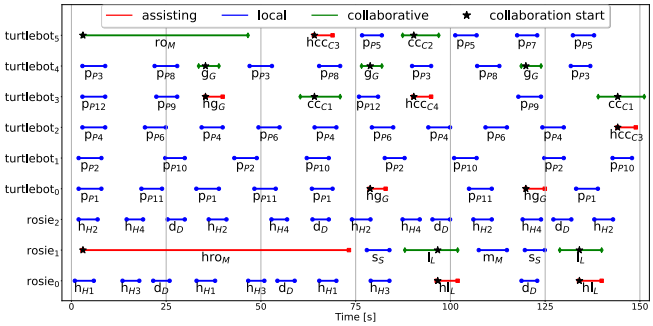


Fig. 3: Agents actions in the experimental settings

D. Scalability

To demonstrate the scalability of our approach, we created a simulation with 90 agents, representing 10 of the teams described in Sec. V-A, due to the unavailability of such a large number of robots. Fig. 4 shows the action sequence of a subset of these agents for clarity. The results indicate that the agents successfully completed their assigned tasks, collaborations, and synchronized actions. The successful completion of collaborations by the agents demonstrates the approach's effectiveness in performing well with ninety agents. Experiments with more agents could not be conducted due to RAM constraints. The results indicate that our approach scales well and can perform effectively with larger teams.

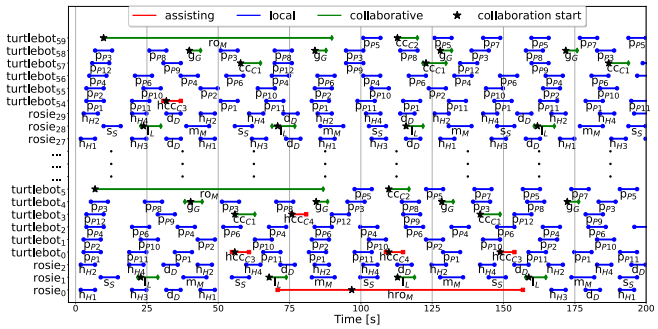


Fig. 4: 90 agents actions in the scalability simulation

VI. Conclusion

This work focuses on MAS coordination and synchronization under recurring LTL. We extended the bottom-up scheme for distributed motion and task coordination of MAS in [13], reducing computational complexity to enhance scalability and enable deployment on robotic hardware. The package was developed in ROS2, with a synchronization mechanism to handle action delays in experiments. Future work will focus on developing additional actions and incorporating human-in-the-loop scenarios.

REFERENCES

[1] J. Palanca, M. Alcaniz, and P. Gonzalez, "A multi-agent system for autonomous mobile robot coordination," *arXiv*, 2020. [Online]. Available: https://arxiv.org/abs/2109.12386

[2] B. Xie, Z. Meng, and J. Zhou, "Agricultural collaborative robots for smart farming," *Agriculture*, vol. 13, no. 1, p. 95, 2023.

[3] C. Belta, B. Yordanov, and E. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer Cham, 01 2017, vol. 89.

[4] X. Luo and M. M. Zavlanos, "Temporal logic task allocation in heterogeneous multirobot systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3602–3621, 2022.

[5] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 01 2008, vol. 26202649.

[6] Y. Kantaros and M. M. Zavlanos, "Temporal logic optimal control for large-scale multi-robot systems: 10^400 states and beyond," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 2519–2524.

[7] ——, "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. [Online]. Available: https://doi.org/10.1177/0278364920913922

[8] Y. Kantaros, "Accelerated reinforcement learning for temporal logic control objectives," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 5077–5082.

[9] M. Guo, J. Tumova, and D. V. Dimarogonas, "Cooperative decentralized multi-agent control under local ltl tasks and connectivity constraints," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 75–80.

[10] K. Grover, F. S. Barbosa, J. Tumova, and J. Křetínský, "Semantic abstraction-guided motion planning for scltl missions in unknown environments," *Robotics: Science and Systems XVII*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235651518

[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, 2022.

[12] Z. Liu, M. Guo, and Z. Li, "Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks," *Automatica*, vol. 159, p. 111377, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109823005435

[13] M. Guo and D. V. Dimarogonas, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, 2017, available at https://people.kth.se/~dimos/pdfs/tase17.pdf.

[14] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," in *Formal Methods Syst. Design*, vol. 19, no. 3, Nov. 2001, pp. 291–314.

[15] L. A. Wolsey, *Integer programming*. New York, NY, USA: Wiley, 1998.

[16] "Ros2 - robot operating system 2," available at: https://www.ros.org/.

[17] D. Peron, V. Nan Fernandez-Ayala, E. Vlahakis, and D. V. Dimarogonas, "multi_robot_coordination_recurring_ltl," Github. [Online]. Available: https://github.com/DavidePeron19/multi_robot_coordination_recurring_LTL.git.

[18] J. R. Büchi, *On a Decision Method in Restricted Second Order Arithmetic*. New York, NY: Springer New York, 1990, pp. 425–435. [Online]. Available: https://doi.org/10.1007/978-1-4613-8928-6_23

[19] R. Baran, X. Tan, P. Varnai, P. Yu, S. Ahlberg, M. Guo, W. S. Cortez, and D. V. Dimarogonas, "A ros package for human-in-the-loop planning and control under linear temporal logic tasks," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 2182–2187.

[20] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Proceedings of the 13th International Conference on Computer Aided Verification*, ser. CAV '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 53–65.

[21] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011. [Online]. Available: https://doi.org/10.1177/0278364911417911

[22] CANOPIES project, "A Collaborative Paradigm for Human Workers and Multi-Robot Teams in Precision Agriculture Systems," in *https://canopies.inf.uniroma3.it/*, european Commission under the H2020 Framework Programme. [Online]. Available: https://canopies.inf.uniroma3.it/.

[23] "Robotis turtlebot3 burger rpi4," available at: https://www.turtlebot.com/turtlebot3/.

[24] "Hebi "rosie" mobile omni-directional base with arm and gripper," available at: https://www.hebirobotics.com/robotic-mobile-platforms.

[25] "Eclipse cyclonedds," available at: https://projects.eclipse.org/projects/iot.cyclonedds.

[26] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[27] Y. Emam, P. Glotfelter, and M. Egerstedt, "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE Press, 2019, pp. 3984–3990. [Online]. Available: https://doi.org/10.1109/CDC40024.2019.9029886

[28] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017. [Online]. Available: http://dx.doi.org/10.1109/TAC.2016.2638961

[29] "Qualisys motion capture," available at: https://www.qualisys.com/.

[30] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, 01 2006.

[31] ——, "Visual servo control, part ii: Advanced approaches," *IEEE Robot. Autom. Mag.*, vol. 14, 01 2007.

[32] D. Peron, V. Nan Fernandez-Ayala, E. Vlahakis, and D. V. Dimarogonas, "Efficient coordination and synchronization of multi-robot systems under recurring linear temporal logic," Youtube. [Online]. Available: https://youtu.be/PjambeIB_ZQ.