Optimal Control Synthesis with Relaxed Global Temporal Logic Specifications for Homogeneous Multi-robot Teams

Disha Kamale and Cristian-Ioan Vasile

Abstract— In this work, we address the problem of control synthesis for a homogeneous team of robots given a global temporal logic specification and formal user preferences for relaxation in case of infeasibility. The relaxation preferences are represented as a Weighted Finite-state Edit System and are used to compute a relaxed specification automaton that captures all allowable relaxations of the mission specification and their costs. For synthesis, we introduce a Mixed Integer Linear Programming (MILP) formulation that combines the motion of the team of robots with the relaxed specification automaton. Our approach combines automata-based and MILP-based methods and leverages the strengths of both approaches, while avoiding their shortcomings. Specifically, the relaxed specification automaton explicitly accounts for the progress towards satisfaction, and the MILP-based optimization approach avoids the state-space explosion associated with explicit product-automata construction, thereby efficiently solving the problem. The case studies highlight the efficiency of the proposed approach.

I. Introduction

One challenge in planning with complex temporal logic specifications in real-world applications concerns dealing with infeasible missions due to perhaps one small part of the requirements. Recent advances in high-level planning using formal specifications have explored the idea of minimally relaxing the specification to ensure satisfaction. Moreover, the problem is compounded when employing teams of robots due to combinatorial explosion of the problem search space.

Traditionally, the control synthesis problem has been addressed using automata-theoretic approaches owing to their rich expressivity. In [1], the authors introduced the problem of *minimal revision* for Buchi automata obtained from Linear Temporal Logic (LTL) specifications by quantifying the closeness between the automata. These revisions, however, are challenging to interpret. Moreover, the problem is shown to be NP-complete [2], [3]. Another automata-based notion of minimum revision was proposed in [4]. The problem of *minimum violation* based on preference-based violation was considered in [5]–[8]. In [9], [10], temporal relaxations was defined for timed specifications using a parameterized abstraction. The problem of *partial satisfaction* was tackled in [11], [12] for Signal Temporal Logic and in [13] for co-safe LTL specifications. Similar problems were tackled in [14]–[16] via hard and soft constraints. In [17], we introduced an automata-based approach that combines various existing notions of relaxation into a unified framework and allows for complex word-word relaxation. Although the automata-based approaches are highly expressive, they cannot be readily used for multi-robot teams.

Disha Kamale and Cristian-Ioan Vasile are with the Mechanical Engineering and Mechanics Department, Lehigh University, Bethlehem, PA 18015 {ddk320, cvasile}@lehigh.edu

On the other hand, a rich body of work exists in planning for large robot teams given temporal logic specifications [18]–[23]. In [18], [19], the authors proposed an informed sampling-based approach for the satisfaction of LTL specifications. With the advent of off-the-shelf solvers [24], Mixed-Integer Linear Programming (MILP) formulations have been utilized for large-scale planning problems [20]–[23] These approaches do not consider for relaxation of the specifications if infeasibilities arise.

In this work, we consider the problem of planning for a team of homogeneous robots tasked with a global temporal logic specification and with formal user preferences. We consider complex temporal logic goals with explicit timing constraints expressed as Time Window Temporal Logic (TWTL) formulae [9]. The user preferences are a set of complex rules for relaxations that are formally represented using a Weighted-Finite State Edit System (WFSE) [17]. Using the automata corresponding to the mission and preferences, we construct a relaxed specification automaton that captures the original task alongside all permissible relaxations and their penalties. To avoid the product of motion models which leads to exponential runtime with respect to the number of robots, we formulate an MILP over the transitions of the motion and relaxed specification models. The encoding efficiently finds the globally optimal solution, bringing together the strengths of automata-based and optimization-based methods.

Our approach differs from the existing MILP-based and relaxation techniques in the following aspects. The MILP formulations defined in [11], [12], [22] optimize for robustness of satisfaction, whereas we are interested in optimally relaxing a specification by modifying it using given user-defined rules. As opposed to spatial preferences as considered in [25], we consider complex preferences of revisions. In [26], the problem of distributed control synthesis for individual TWTL specifications allowing finite temporal relaxations is considered; whereas we consider planning for a global TL task with word-to-word revisions. Finally, the main difference as compared to our previous work [17] is that instead of considering a general problem of optimal relaxation over a 3-way product automaton for a single robot, we use an MILP formulation that simultaneously accounts for motion and specification relaxation for teams of robots.

The main contributions of this work are 1) We pose the problem of optimal planning with relaxation for a homogeneous robot team subject to a global temporal logic specification. 2) We define a *relaxed specification automaton* that represents the given global specifications with all allowable revisions. 3) We propose a framework that combines automata and MILP-based approaches to address the

minimal relaxation planning problem. The relaxed specification automaton explicitly accounts for the progress towards satisfaction, while the MILP formulation helps to avoid scalability issues, thereby bringing together the strengths of both approaches while avoiding their limitations. 4) We demonstrate the functionality of the proposed framework with the help of case studies and provide a runtime analysis. We compare the proposed MILP formulation with a baseline for their efficiency and expressivity.

Notation. The set of real, integer, and binary numbers are represented by $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{B}$. The set of integer numbers greater or equal to a is defined by $\mathbb{Z}_{\geq a}$. The integer range from a to b is denoted by $[[a, b]]$. For a set X, 2^X and $|X|$ represent its power set and cardinality, respectively. For an alphabet Σ, a language of all finite words over Σ is denoted by Σ^*. We refer to a directed acyclic graph as a DAG.

II. PROBLEM SETUP

In this section, we introduce the problem of optimal planning for a global temporal logic goal with formal preferences for relaxations. Using a finite system abstraction for the environment and automata to capture the progress towards goal and preferences, we define a cost function that combines the costs incurred for motion and relaxations. Next, we pose the formal problem definition.

A. Robot and Environment Models

We consider a team of N identical robots, denoted by $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$, deployed in an environment to perform a global task. Each robot can move in the environment deterministically. We assume that the robots move in the environment synchronously. This is achieved either by assuming robots have access to a global clock, synchronize via inter-robot communication, or receive and execute commands from a global controller.

The robots' motion in the environment is abstracted as a transition system $\mathcal{T}$ defined as follows:

Definition 1 (Transition System). *A weighted transition system (TS) is a tuple $\mathcal{T} = (X, \mathcal{R}, \mathcal{I}^{\mathcal{T}}, \delta_{\mathcal{T}}, AP, h, w_{\mathcal{T}})$, where X is a finite set of states and corresponds to regions in the environment; $\mathcal{I}^{\mathcal{T}} : \mathcal{R} \to X$ defines the robots' initial states; $\delta_{\mathcal{T}} \subseteq X \times X$ is a set of transitions that capture the set of allowable movements in the environment; AP is a set of labels (atomic propositions); $h : X \to 2^{AP}$ is a labeling function; $w_{\mathcal{T}} : \delta_{\mathcal{T}} \to \mathbb{R}_{\geq 0}$ is a weight function that denotes control effort incurred in traversing $(x, x') \in \delta_{\mathcal{T}}$.*

The states X of the environment represent locations connected by directed transitions $\delta_{\mathcal{T}}$. We use self-loop transitions $(x, x) \in \delta_{\mathcal{T}}$ to model robots stationary at state x. All transitions are assumed without loss of generality to have duration 1 that corresponds to a fixed time discretization step Δt. As a robot $r_i \in \mathcal{R}$ moves through the environment, it generates a sequence of states $\mathbf{x}_{r_i} = x_{r_i,0} x_{r_i,1} \ldots$, referred to as a *trajectory* (or run) of a robot, such that $(x_{r_i,k}, x_{r_i,k+1}) \in \delta_{\mathcal{T}}$ for all $k \in \mathbb{Z}_{\geq 0}, i \in [[1, N]]$ and $x_{r_i,0} = \mathcal{I}^{\mathcal{T}}(r_i)$. An atomic proposition $\pi \in AP$ is said to be true when at least one robot is at a state x labeled with

π, i.e., $x \in h^{-1}(\pi)$. Overall, the observations at states are given by the entire team of robots $\mathcal{R}$. Thus, we introduce the team state and output trajectories.

Definition 2. *The team trajectory $\tilde{\mathbf{x}}$ is a sequence of vectors of all states occupied by robots at each time instance, i.e., $\tilde{\mathbf{x}}(k) = [\mathbf{x}_{r_i}(k)]_{r_i \in \mathcal{R}} \in X^{|\mathcal{R}|}$ for a global time index k. The corresponding output word is $\mathbf{o} = o_0 o_1 \ldots$ with $o_k = \bigcup_{r_i \in \mathcal{R}} h(x_{r_i,k})$. The (generated) language of $\mathcal{T}$ is the set of all team output words, denoted by $\mathcal{L}(\mathcal{T})$.*

The control cost of a transition from state x to x' is captured by $w_{\mathcal{T}}(x, x')$. When robots are stationary, they incur zero control cost, i.e., $w_{\mathcal{T}}(x, x) = 0$ for all $x \in X$. The **control cost** for a team trajectory $\tilde{\mathbf{x}}$ is defined as $\hat{J}_{control}(\tilde{\mathbf{x}}) = \sum_{r_i \in \mathcal{R}} \sum_{k \in |\tilde{\mathbf{x}}| - 1} w_{\mathcal{T}}(x_{r_i,k}, x_{r_i,k+1})$.

This work focuses on high-level decision-making for a complex global task with relaxations, we assume that appropriate low-level controllers are provided that facilitate the execution of the high-level plans.

B. Task Specification

Owing to its expressivity and conciseness, the global task is given as a Time Window Temporal Logic (TWTL) formula, introduced in [9]. A TWTL formula is defined over a set of atomic propositions and its syntax expressed in Bakus-Naur form is

$$\phi ::= H^d s \,|\, \phi_1 \wedge \phi_2 \,|\, \phi_1 \vee \phi_2 \,|\, \phi_1 \cdot \phi_2 \,|\, [\phi_1]^{[a,b]}$$

where s is either the "true" constant true or an atomic proposition in AP; $\wedge$ and $\vee$ are the conjunction and disjunction Boolean operators, respectively; $\cdot$ is the concatenation operator; H^d with $d \in \mathbb{Z}_{\geq 0}$ is the *hold* operator; and $[\,]^{[a,b]}$ is the *within* operator, $a, b \in \mathbb{Z}_{\geq 0}$ and $a \leq b$. For a thorough description of the syntax and semantics of TWTL, we refer the reader to [9], [10].

For simplicity, we disallow negations of atomic propositions, i.e., $H^d \neg s$, which require additional bookkeeping in our solution in Sec. III. We leave this for future work. However, we still require the formulas without ambiguous concatenation, see [10].

C. User preferences

The user preferences govern the modifications to the original mission specification ϕ if infeasibilities arise. Formally, let L denote a language over an alphabet 2^{AP}. A *user task preference* is a pair (R, w_R), where $R \subseteq L \times (2^{AP})^*$ is a relation specifying how words in L can be transformed to words from $(2^{AP})^*$; and $w_R : R \to \mathbb{R}_{\geq 0}$ represents the cost associated with the word transformations. Alternatively, preferences are given by a set of word-rewrite rules with associated weights. We denote a weighted word-rewrite rule as $\rho = \boldsymbol{\sigma} \mapsto^{w_{\sigma,\sigma'}} \boldsymbol{\sigma'}$ where $\boldsymbol{\sigma}_{pre} \boldsymbol{\sigma} \boldsymbol{\sigma}_{suf} \in L, \boldsymbol{\sigma}_{pre} \boldsymbol{\sigma'} \boldsymbol{\sigma}_{suf} \in (2^{AP})^*$ for some prefix and suffix words $\boldsymbol{\sigma}_{pre}$ and $\boldsymbol{\sigma}_{suf}$, respectively. To allow satisfaction without relaxation, we require that $(\boldsymbol{\sigma}, \boldsymbol{\sigma}) \in R$ with $w_R(\boldsymbol{\sigma}, \boldsymbol{\sigma}) = 0$ for all $\boldsymbol{\sigma} \in L$. The example below illustrates the definition.

Example II.1. *Consider the specification "Visit A and B for 2 and 3 time units within the first 5 minutes followed by C*

for 1 time unit within the next 1 minute." Expressed as a TWTL formula, we have $\phi = [H^2 A \wedge H^3 B]^{[0,5]} \cdot [H^1 C]^{[0,1]}$. The relaxation preferences are given as follows - 1) C can be substituted by a visit to region D with a penalty of 3. 2) A can be substituted by simultaneously visiting B and D with a penalty of 2. Formally, $\rho_1 = C \mapsto^3 D, \rho_2 = A \mapsto^2 BD$.

The multi-robot control synthesis problem is as follows.

Problem II.1 (Optimal Control Synthesis with Relaxation). *Given a team $\mathcal{R}$, a discrete abstraction of the environment $\mathcal{T}$, a global task specification expressed as a TWTL formula ϕ, and user preferences for specification relaxation as (R, w_R), the problem of optimal control synthesis with minimal relaxations corresponds to finding a team trajectory $\tilde{\mathbf{x}}$ that satisfies a potentially relaxed version of the mission ϕ while minimizing the control and relaxation costs. Formally,*

$$\min_{\tilde{\mathbf{x}}} \hat{J}(\tilde{\mathbf{x}}) = \hat{J}_{control}(\tilde{\mathbf{x}}) + \lambda \cdot w_R(\mathbf{o}, \mathbf{o}^{relax}),$$

$$s.t. \; \mathbf{o}^{relax} = h(\tilde{\mathbf{x}}), \quad (\mathbf{o}, \mathbf{o}^{relax}) \in R, \quad \mathbf{o} \models \phi,$$

where λ is a blending parameter, $\mathbf{o}^{relax} = h(\tilde{\mathbf{x}})$ is the output word of the $\mathcal{T}$, and $\mathbf{o} \in (2^{AP})^*$ is a word satisfying ϕ.

III. APPROACH

In this section, we present a solution to Problem II.1. Our approach combines the expressivity of automata with the efficiency of optimization-based approaches. We proceed by translating the specification and user preferences to automata and then constructing a product automaton that encapsulates the original specification and all its permissible revisions. We present an algorithm to compute this relaxed specification product which is defined using a set of input-output symbol pairs in the WFSE. We then present the main contribution of this work – an MILP formulation that combines the motion model with the relaxed specification automaton as a solution to Problem II.1. Critically, we avoid the construction of product transition systems, a common in automata-based methods [9], [27], [28].

A. Specification and User Preferences Automata

In order to solve the problem of control synthesis for a given mission ϕ, a deterministic finite state automaton, (defined below), is recursively constructed from ϕ using the procedure discussed in [10].

Definition 3 (Deterministic Finite State Automaton). *A deterministic finite state automaton (DFA) is a tuple $\mathcal{A}_\phi = (S_{\mathcal{A}_\phi}, s_0^{\mathcal{A}_\phi}, \Sigma, \delta_{\mathcal{A}_\phi}, F_{\mathcal{A}_\phi})$, where $S_{\mathcal{A}_\phi}$ is a finite set of states; $s_0^{\mathcal{A}_\phi} \in S_{\mathcal{A}_\phi}$ is the initial state; Σ is the input alphabet; $\delta_{\mathcal{A}_\phi} : S_{\mathcal{A}_\phi} \times \Sigma \to S_{\mathcal{A}_\phi}$ is the transition function; $F_{\mathcal{A}_\phi} \subseteq S_{\mathcal{A}_\phi}$ is the set of accepting states.*

A trajectory of the DFA $\mathbf{s} = s_0 s_1 \ldots s_{n+1}$ is generated by a finite sequence of symbols $\boldsymbol{\sigma} = \sigma_0 \sigma_1 \ldots \sigma_n$ if $s_0 = s_0^{\mathcal{A}_\phi}$ is the initial state of $\mathcal{A}_\phi$ and $s_{k+1} = \delta_{\mathcal{A}_\phi}(s_k, \sigma_k)$ for all $k \geq 0$. A finite input word $\boldsymbol{\sigma}$ over Σ is said to be accepted by a finite state automaton $\mathcal{A}_\phi$ if the trajectory of $\mathcal{A}_\phi$ generated by $\boldsymbol{\sigma}$ ends in a state belonging to the set of accepting states, i.e., $F_{\mathcal{A}_\phi}$. The *(accepted) language* of a DFA $\mathcal{A}_\phi$ is the set of accepted input words denoted by $\mathcal{L}(\mathcal{A}_\phi)$.

In [17], we showed that formal user preferences can be represented as a special class of automata called Weighted-Finite State Edit System (WFSE), defined as follows.

Definition 4 (Weighted Finite State Edit System). *A weighted finite state edit system (WFSE) is a weighted DFA $\mathcal{E} = (Z_\mathcal{E}, z_0^\mathcal{E}, \Sigma_\mathcal{E}, \delta_\mathcal{E}, F_\mathcal{E}, w_\mathcal{E})$, where $\Sigma_\mathcal{E} = 2^{AP} \times 2^{AP}$, and $w_\mathcal{E} : \delta_\mathcal{E} \to \mathbb{R}_{\geq 0}$ is the transition weight function.*

The alphabet $\Sigma_\mathcal{E}$ captures the substitution operations. A weighted transition $z' = \delta_\mathcal{E}(z, (\sigma, \sigma'))$ has input, output symbols σ and σ'. Given a word $\vec{\boldsymbol{\sigma}} = (\sigma_0, \sigma_0')(\sigma_1, \sigma_1') \ldots (\sigma_r, \sigma_r') \in \mathcal{L}(\mathcal{E})$, $r = |\vec{\boldsymbol{\sigma}}| - 1$, we refer to $\boldsymbol{\sigma} = \sigma_0 \sigma_1 \ldots \sigma_r \in (2^{AP})^*$ as *input word* and to $\boldsymbol{\sigma}' = \sigma_0' \sigma_1' \ldots \sigma_r' \in (2^{AP})^*$ as the *output word*. Moreover, we say that $\mathcal{E}$ transforms $\boldsymbol{\sigma}$ into $\boldsymbol{\sigma}'$.

Remark: In this work, we are particularly interested in addressing the minimum revision problem with complex preferences. Thus, as opposed to a more general alphabet used for WFSEs in [17], we drop the inclusion of the *empty symbol* ϵ in the current work. Investigating the problems of minimum violation, insertion and temporal relaxation are topics for future research.

The **relaxation cost** for transforming $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma}'$ is given by $\hat{J}_{revision}(\vec{\boldsymbol{\sigma}}) = \sum_{l \in [[0, |\vec{\boldsymbol{\sigma}}|]]} w_\mathcal{E}(\sigma_l, \sigma_l')$.

B. Relaxed Specification Product Automaton

We capture all relaxed versions of the TWTL specification ϕ via a product non-deterministic finite state automaton between the DFA $\mathcal{A}_\phi$ and the WFSE $\mathcal{E}$.

Definition 5 (Relaxed Specification Automaton). *Given a specification DFA $\mathcal{A}_\phi = (S_{\mathcal{A}_\phi}, s_0^{\mathcal{A}_\phi}, \Sigma, \delta_{\mathcal{A}_\phi}, F_{\mathcal{A}_\phi})$, the user task preferences represented as a WFSE $\mathcal{E} = (Z_\mathcal{E}, z_0^\mathcal{E}, \Sigma_\mathcal{E}, \delta_\mathcal{E}, F_\mathcal{E}, w_\mathcal{E})$, the relaxed specification automaton is a tuple $\mathcal{A} = (Q_\mathcal{A}, q_\mathcal{A}^0, \Sigma_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A}, w_\mathcal{A})$, where $Q_\mathcal{A} = Z_\mathcal{E} \times S_{\mathcal{A}_\phi}$ represents the state space; $q_0^\mathcal{A} = (z_0^\mathcal{E}, s_0^{\mathcal{A}_\phi})$ is the initial state; $\Sigma_\mathcal{A} = \Sigma_{\mathcal{A}_\phi}$ denotes the alphabet; $\delta_\mathcal{A} \subseteq Q_\mathcal{A} \times \Sigma_\mathcal{A} \times Q_\mathcal{A}$ is a transition relation; $F_\mathcal{A} = F_\mathcal{E} \times F_{\mathcal{A}_\phi} \subseteq Q_\mathcal{A}$ represents the set of final (accepting) states; $w_\mathcal{A} : \delta_\mathcal{A} \to \mathbb{R}_{\geq 0}$ is the weight function.*

The procedure for constructing $\mathcal{A}$ is outlined in Alg. 1. All components of the product are initialized, and the initial state is added to the stack of nodes to be processed (lines 1-3). For a given input symbol σ, a transition $((z, s), \sigma, (z', s')) \in \delta_\mathcal{A}$ if and only if there exists σ' such that $z' \in \delta_\mathcal{E}(z, (\sigma, \sigma'))$ and $s' = \delta_{\mathcal{A}_\phi}(s, \sigma')$. Given the current states (z, s), we iterate over each outgoing edge of z in $\mathcal{E}$ (line 5) with input-output symbol pair (σ, σ'). The next state in the DFA $\mathcal{A}_\phi$ is determined given the current state s and the output symbol σ' (line 6). The states and edges are added to the product and to $F_\mathcal{A}$, if applicable. The weight function is defined as $w_\mathcal{A}((z, s), \sigma, (z', s')) = w_\mathcal{E}(z, (\sigma, \sigma'), z')$. The process repeats until no states (z, s) are left to process in the stack. Thus, the relaxed specification automaton simultaneously captures the original specification, and user preferences for relaxations along with their penalties.

The relaxed specification automaton $\mathcal{A}$ is a non-deterministic model due to the potentially multiple ways to

Algorithm 1: *construct_$\mathcal{A}$()*

Input: $\mathcal{E}, \mathcal{A}_\phi$
Result: $\mathcal{A} = (Q_\mathcal{A}, q_0^\mathcal{A}, \Sigma_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A}, w_\mathcal{A})$

1 $\Sigma_\mathcal{A} \leftarrow \Sigma_{\mathcal{A}_\phi}, q_\mathcal{A}^0 \leftarrow (z_0^\mathcal{E}, s_0^{\mathcal{A}_\phi}), Q_\mathcal{A} \leftarrow \{q_\mathcal{A}^0\}, \delta_\mathcal{A} \leftarrow \emptyset, F_\mathcal{A} \leftarrow \emptyset$
2 $stack \leftarrow Q_\mathcal{A}$
3 **while** $stack \neq \emptyset$ **do**
4      $(z, s) \leftarrow stack.pop()$
5   **forall** $(z, (\sigma, \sigma'), z') \in \delta_\mathcal{E}$ **do**
6         $s' \leftarrow \delta_{\mathcal{A}_\phi}(s, \sigma')$
7   **if** $(z', s') \notin Q_\mathcal{A}$ **then**
8            $Q_\mathcal{A} \leftarrow Q_\mathcal{A} \cup \{(z', s')\}$
9            $stack.push((z', s'))$
10   **if** $z' \in F_\mathcal{E}$ *and* $s' \in F_{\mathcal{A}_\phi}$ **then**
             $F_\mathcal{A} \leftarrow F_\mathcal{A} \cup \{(z', s')\}$
11        $\delta_\mathcal{A} \leftarrow \delta_\mathcal{A} \cup \{((z, s), \sigma, (z', s')))\}$
12        $w_\mathcal{A}((z, s), \sigma, (z', s'))) \leftarrow w_\mathcal{E}(z, (\sigma, \sigma'), z')$

---

translate an input symbol σ into an output σ', see lines 5, 6 and 11 in Alg. 1. Semantically, this means that there are potentially multiple ways to relax the mission specification, as illustrated in the example below.

Example III.1. *Consider user preferences:* $A \mapsto^{p1} B$, $AB \mapsto^{p2} CD$, *and* $AC \mapsto^{p3} BD$. *The same input symbol A may result in different output symbols leading to non-determinism in the relaxed specification automaton.*

Proposition 1. $\mathcal{A}$ *is a directed acyclic graph (DAG).*

This follows from the fact that the recursive translation of ϕ into $\mathcal{A}_\phi$ as proposed in [10] results in a DAG. In general, if $\mathcal{A}_\phi$ is a DAG, the resulting product $\mathcal{A}$ is also a DAG. Proposition 1 offers a key insight into our MILP formulation since the product space assumes a directionality that can be leveraged for flow-based encoding.

C. Mixed Integer Linear Programming (MILP$_\mathcal{E}$)

Given $\mathcal{T}$ and $\mathcal{A}$, we now pose Problem II.1 as an instance of an MILP. Towards this, we make the following assumptions: 1) The specification ϕ does not contain any negations; 2) All transitions in $\mathcal{T}$ take unit duration for traversal. These assumptions are not restrictive since, in the case of assumption (1), the forbidden states can be eliminated during environment abstraction. Regarding assumption (2), auxiliary states and transitions can be used in case of longer durations. The MILP encoding uses the observation that we do not need to track robots in $\mathcal{T}$. It suffices to compute the number of robots at each state and transition over time.

Let $n(x)$ denote the initial number of robots at the state $x \in X$ such that $\sum_{x \in X} n(x) = N$, and $X_0^\mathcal{T} = (\mathcal{I}^\mathcal{T})^{-1}(\mathcal{R})$ the initial set of states occupied by robots. We define integer variables $z_{x,x',q,q'} \in [[0, N]]$ to denote the number of robots traversing edge (x, x') in $\delta_\mathcal{T}$ and (q, q') in $\delta_\mathcal{A}$. Similarly, the integer variables $z_{x,q}$ correspond to the number of robots present at state $x \in X$ and state $q \in Q_\mathcal{A}$.

1) Initialization Constraints: Let the set of atomic propositions satisfied at deployment be $AP_0 = h(X_0^\mathcal{T})$. We introduce a virtual node $\bowtie$ that is connected to all nodes in $X_0^\mathcal{T}$ with only outgoing edges, i.e., we add $\{(\bowtie, x) \mid x \in X_0^\mathcal{T}\}$ of weight 0. The set of states in $\mathcal{A}$ that can be reached due to the satisfaction of the atomic propositions in AP_0 is given by $Q_0 = \bigcup_{\sigma \in 2^{AP_0}} \delta_{\mathcal{A}_\phi}(q_0, \sigma)$.

$$\sum_{q \in Q_0} z_{\bowtie, x, q_0, q} = n(x), \forall x \in X_0^\mathcal{T}, \quad (1)$$

2) Flow Constraints: Since the overall number of robots should remain constant throughout the mission, the flow constraints require that the total number of robots entering a region must be equal to the total number of robots leaving that region. Let $\hat{Q}_\mathcal{A} = Q \setminus F_\mathcal{A} \setminus \{q_0^\mathcal{A}\}$. The set of predecessors of $q \in \hat{Q}_\mathcal{A}$ is $\mathcal{N}_\mathcal{A}^-(q) = \{q' \mid \exists \sigma \in 2^{AP}, q \in \delta_\mathcal{A}(q', \sigma)\}$. Similarly, the set of successors is $\mathcal{N}_\mathcal{A}^+(q) = \{q' \mid \exists \sigma \in 2^{AP}, q' \in \delta_\mathcal{A}(q, \sigma)\}$. Using these, the flow constraints are encoded as follows.

$$\sum_{(x',x) \in \delta_\mathcal{T}} \sum_{q' \in \mathcal{N}_\mathcal{A}^-(q)} z_{x',x,q',q} = \sum_{(x,x') \in \delta_\mathcal{T}} \sum_{q' \in \mathcal{N}_\mathcal{A}^+(q)} z_{x,x',q,q'}. \quad (2)$$

3) State-transition Constraints: The number of robots at a state x in the environment and q in the specification is given by the outflow from (2), except for sink nodes with $q \in F_\mathcal{A}$. For all states $x \in X$ and $q \in Q$,

$$z_{x,q} = \begin{cases} \sum_{(x,x') \in \delta_\mathcal{T}} \sum_{q' \in \mathcal{N}_\mathcal{A}^+(q)} z_{x,x',q,q'}, & q \notin F_\mathcal{A} \\ \sum_{(x',x) \in \delta_\mathcal{T}} \sum_{q' \in \mathcal{N}_\mathcal{A}^-(q)} z_{x',x,q',q}, & q \in F_\mathcal{A} \end{cases}. \quad (3)$$

4) Synchronization Constraints: Since, at any instance, the entire team is in a single state of $\mathcal{A}$, we synchronize the state transition in $\mathcal{A}$ across all robots. Let $\xi_{q,q'} \in \mathbb{B}$ denote whether a transition $(q, q') \in \delta_\mathcal{A}$ was taken. For $(q, q') \in \delta_\mathcal{A}$,

$$\sum_{q' \in \mathcal{N}_\mathcal{A}^+(q)} \xi_{q,q'} \leq 1, \quad (4)$$

$$\sum_{(x,x') \in \delta_\mathcal{T}} z_{x,x',q,q'} \leq N \cdot \xi_{q,q'}. \quad (5)$$

Since $\mathcal{A}$ is a DAG, each edge in $\delta_\mathcal{A}$ is traversed at most once. This is enforced by constraints (4) and (5), with the latter imposing the requirement on the robot flow.

5) Guard Satisfaction Constraints: The progress towards satisfaction is encoded as follows. We process all transitions $(q, \sigma, q') \in \delta_\mathcal{A}$ between q and q' together based on Boolean function representation $g_{q,q'}$ called a *guard*. Formally, $g_{q,q'}(\sigma)$ is true if and only if $(q, \sigma, q') \in \delta_\mathcal{A}$. We assume that the guards are in disjunction normal form (DNF), i.e., $g = \bigvee_{i=1}^{m^+} \eta_i$, where each term $\eta_i = \bigwedge_{\ell=1}^{m^*} \pi_\ell, \pi_\ell \in AP$. Let $Terms_\mathcal{A}(q, q') = \{\eta_i\}_{i=1}^{m^+}$. Let $\xi_{q',\eta} \in [0, 1]$ indicate whether the term η of the guards leading to state q' in $\mathcal{A}$ is satisfied. In other words, η is a set of APs associated with the guard such that their simultaneous satisfaction enables the transition to q'. Similarly, we define $\xi_{q',\pi} \in [0, 1]$ indicating whether the AP π is satisfied for a guard leading to state q' in $\mathcal{A}$. Note that η and π may be common across multiple incoming transitions to q'. With this, the guard satisfaction constraints are as follows.

$$\xi_{q,q'} \leq \sum_{\eta \in Terms_\mathcal{A}(q,q')} \xi_{q',\eta}, \quad (6)$$

$$\xi_{q',\eta} \leq \xi_{q',\pi}, \forall \pi \in \eta, \eta \in Terms_\mathcal{A}(q, q'), \quad (7)$$

$$\xi_{q',\pi} \leq \sum_{x \in h^{-1}(\pi)} z_{x,q'}, \quad (8)$$

$\forall (q, q') \in \delta_\mathcal{A}$. Constraint (6) ensures that at least one of the terms associated with the guard of (q, q') is satisfied.

The satisfaction of all atomic propositions corresponding to a term is ensured by (8). Finally, the motion in $\mathcal{T}$ is connected with guard satisfaction in $\mathcal{A}$ via $\xi_{q',\pi}$ which can be 1 only if there are robots at states labeled with AP π.

6) Final state: The constraint to ensure that the final state in $\mathcal{A}$ is reached, i.e., a relaxation of ϕ is satisfied, is

$$\sum_{x \in X} \sum_{q \in F_{\mathcal{A}}} z_{x,q} > 0. \tag{9}$$

7) Cost function: Finally, we define the objective as follows $\hat{J} = \hat{J}_{control} + \lambda \hat{J}_{revision}, \lambda > 0$

$$\hat{J}_{control} = \sum_{(x,x') \in \delta_{\mathcal{T}}} \sum_{(q,q') \in \delta_{\mathcal{A}}} w_{\mathcal{T}}(x,x') z_{x,x',q,q'} \tag{10}$$

$$\hat{J}_{revision} = \sum_{(x,x') \in \delta_{\mathcal{T}}} \sum_{(q,q') \in \delta_{\mathcal{A}}} w_{\mathcal{A}}(q,q') z_{x,x',q,q'} \tag{11}$$

IV. CASE STUDIES

In this section, we show the functionality of our approach, a comparison with a baseline, and a runtime analysis of MILP$_{\mathcal{E}}$. These studies were performed on Dell Precision 3640 Intel i9 with 64 GB RAM using Python 3.9.7. The environment used across all case studies is shown in Fig. 1. The $\mathcal{T}$ is shown using blue nodes and directed edges with control costs in red, or 1 otherwise. For this environment, we have $|X| = 20, |\delta| = 63$.

Functionality. Consider a team of 30 robots deployed for scientific exploration and data collection. The zones of primary interest in the environment are *A*, *B*, and *C*. Let S denote the regions to collect samples from, M denotes the regions to be monitored, and U denotes the data upload centers. The labels follow the notation $f_{zone,i}$ where f denotes the functionality among $\{sample, monitor, upload\}$, $zone$ denotes the zone name followed by a number. For instance, for $zoneA$ we have, $\{S_{A1}, S_{A2}, M_A\}$. The goal is to gather samples, monitor the specified regions within the given time durations, and upload the data at upload centers. The specification expressed plain English is: *"Collect samples from S_{A1} for 3 minutes and S_1' for 1 minute, monitor region M_{B1} for 2 minutes within the first t_1 minutes. Next, monitor regions M_A and M_1' for 1 minute each within the next t_2 minutes or collect samples and monitor regions S_C, M_C for 2 and 3 minutes, respectively, within the next t_3 minutes. Finally, upload the collected data at the upload centers U_A, U_B, U_C within the next t_4 minutes."* Using TWTL, the specification is expressed as follows:

$$\phi = [H^3 S_{A1} \wedge H^1 S_{A3} \wedge H^2 M_{B1}]^{[0,t_1]}$$
$$\cdot \; (\; [H^1 S_{A2} \wedge H^1 M_A \wedge H^1 M_{B2}]^{[0,t_2]}$$
$$\vee [H^2 S_C \wedge H^3 M_{C1} \wedge H^1 M_{C2}]^{[0,t_3]} \;)$$
$$\cdot [U_A \wedge U_B \wedge U_C]^{[0,t_4]}.$$

Here, the infeasibilities may stem from insufficient time to reach regions or unreachable regions. In such cases, we have the relaxation preferences: *1) Region S_{A1} can be substituted by collecting samples from regions S_1' and S_2' with a penalty of 5; 2) Substitute region M_A by monitoring regions M_1' and M_{C2} and collecting samples from region S_1' for a penalty of 2; 3) Collecting samples from region S_{A3} can be substituted by monitoring regions M_{B2} and M_1' for a penalty of 3.*
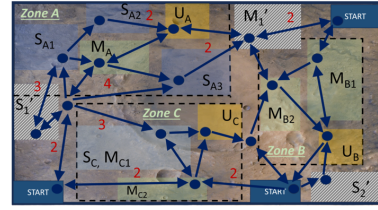

Fig. 1: Transition system for Case Study 1

Formally, 1) $S_{A1} \mapsto^5 S_1'S_2'$, 2) $M_A \mapsto^2 M_1'M_{C2}S_1'$ and 3) $S_{A3} \mapsto^3 M_{B2}M_1'$.

1) Feasible specification: With $t_1 = 7, t_2 = 3, t_3 = 4, t_4 = 3$ and all regions available, ϕ is feasible with $|Q_{\mathcal{A}}| = 44, |\delta_{\mathcal{A}}| = 76$. In this case, no revisions are required. Fig. 2(a) shows the generated plan where each plan starting at a different start location is shown using a different color. The final locations of each plan have been encircled. The cost is $\hat{J} = 92$ with $\lambda = 0.5$.

2) Insufficient time: We assign $t_1 = 6$, keeping all other values the same as above, which makes $H^3 S_{A1}$ infeasible. Fig. 2(b) shows the plans obtained using relaxation preference (1). Note that the plan passes through S_{A1} at a later time to satisfy regions M_A, S_{A2}.

3) Zone unavailability: Assume zone *A* is not available. Thus, relaxation preferences (1) and (3) are used and the plan is shown in Fig. 2(c). Even though regions S_{A2}, M_A are not reachable, the subformula in disjunction is still feasible, and no relaxation is necessary for the subformula. Since our formulation minimizes the combined cost of control effort and relaxations, excess robots in the team are stationary.

Efficiency Vs Expressivity. To the authors' best knowledge, no other MILP formulation defined over timed specifications accommodates revision relaxations. Thus, we consider a baseline MILP encoding based on flow conservation and direct specification encoding without the use of automata, similar to [22], [29]. The total number of robots is N = 3. Table I presents the number of binary and total decision variables and the computation time for Gurobi across six specifications. Note that the baseline does not capture any notion of relaxation. Thus, the number of constraints needed to guide satisfaction is considerably less. To introduce relaxations in the baseline, additional constraints and decision variables would need to be created, increasing the computation time. For specifications that require relaxations (e.g., last 2 specifications in Table I due to time or unavailability of S_{A1}), the problem is infeasible for baseline. Thus, it is evident that MILP$_{\mathcal{E}}$ offers more expressivity in terms of handling formal user preferences for relaxations at the cost of higher computation time.

4) Runtime Analysis: We present the runtime performance of the proposed approach with respect to multiple factors, varying a single parameter at a time. Our results are in accordance with the observations made in [22].

$\mathcal{T}$ **size**. We consider $\phi = [H^1 S_{A2}]^{[0,3]} \cdot [H^2 M_A]^{[0,5]}$. The WFSE considers trivial revisions that map an input word to itself. We vary the environment size from 10 to 500 nodes, while keeping a set of APs fixed, $|AP| = 5$, from which labels are randomly assigned to the nodes. Fig. (4)
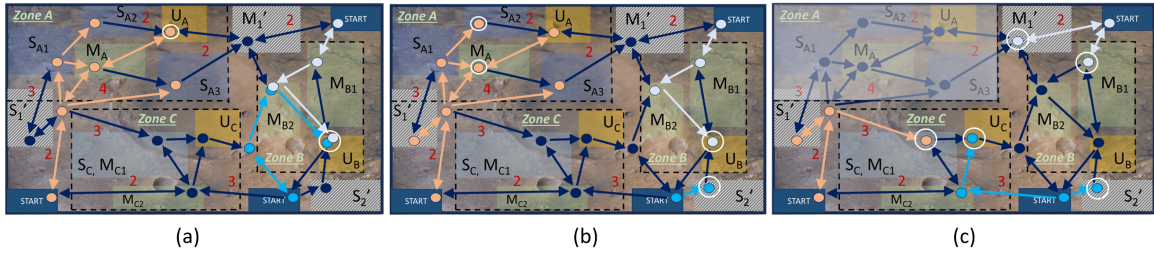
Fig. 2: Trajectories obtained in case of a) feasible specification, b) insufficient time, c) unavailability of $zoneA$.

TABLE I: Comparison with baseline encoding

Encoding	Specification	Binary	Total	Time (s)
Baseline		10	484	0.0044
MILP$_\mathcal{E}$	$[H^3 S_{A1}]^{[0,5]}$	1107	1371	0.800
Baseline		11	660	0.0069
MILP$_\mathcal{E}$	$[H^2 U_C \wedge H^3 S_{A2}]^{[0,7]}$	5147	5865	6.3797
Baseline		7	649	0.0068
MILP$_\mathcal{E}$	$[H^4 M_1']^{[0,6]} \vee [H^6 M_{B1}]^{[0,7]}$	6887	7278	4.517
Baseline		27	1212	0.021
MILP$_\mathcal{E}$	$[H^3 S_{A3}]^{[0,5]} \cdot [H^2 S_1']^{[4,9]}$	28158	30054	14.07
Baseline				infeasible
MILP$_\mathcal{E}$	$[H^3 S_{A3}]^{[0,4]} \cdot [H^2 S_1']^{[4,9]}$	25957	28840	29.382
Baseline	$[H^3 S_{A1} \wedge H^1 S_C]^{[0,5]}$			infeasible
MILP$_\mathcal{E}$	$\cdot [H^2 M_{B_2}]^{[4,8]} \cdot [H^1 U_B]^{[0,2]}$	28440	30936	30.770

(a) Runtime vs $\mathcal{A}_\phi$ size

(b) Runtime vs No. preferences

(c) Runtime vs N

(d) Runtime vs No. of APs
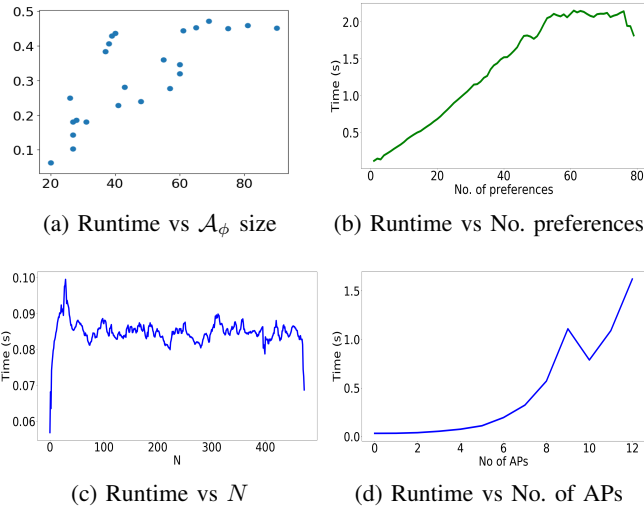
Fig. 3: Computation time for MILP$_\mathcal{E}$ for various parameters

shows the time performance of MILP$_\mathcal{E}$ for $N = 2, 10, 20$. For smaller teams ($N = 2$), the problem may take longer due to overlapping requirements. The execution time for MILPs is affected by the bounds on decision variables. Introducing more robots makes the computation faster ($N = 10$), however, presence of multiple excess robots increases the upper bound on the decision variables $z_{x,q}$ and $z_{x,x',q,q'}$ resulting in a slightly higher computation time.

$\mathcal{A}_\phi$ **size**. We consider a specification of the form

$$\phi = [H^{t_1} x_1 \wedge H^{t_2} x_2]^{[0,t_3]} \oplus [H^{t_4} x_3]^{[0,t_5]}$$

where $\oplus \in \{\vee, \cdot\}$ and the APs x_i are randomly chosen from a set $\{S_{A1}, S_{A2}, M_A, S_{A3}, S_C\}$ and the durations and time windows are randomly assigned with respect to a horizon varying from 6 to 40 time units while following the conditions necessary for feasibility of ϕ. Fig. (3a) indicates that, owing to the complex structures of ϕ, the execution
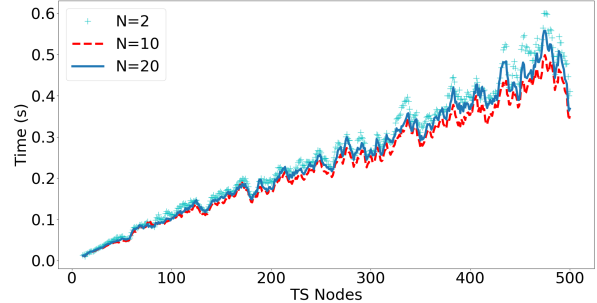

Fig. 4: Time for solving MILP$_\mathcal{E}$ where the environment size varies from 10 to 500 for $N = 2, 10, 20$.

time for MILP$_\mathcal{E}$ depends not only on the horizon of ϕ but also on its internal structure.

Preferences. Given $\phi = [H^1 S_{A2}]^{[0,3]} \cdot [H^2 M_A]^{[0,5]}$ and a fixed $\mathcal{T}$, we vary the number of preferences from 1 to 80 where $|AP| = 5$. We consider the preferences of the form $B \mapsto \sigma$ where σ is a set of propositions randomly chosen from AP with varying $|\sigma| \in [[1, 4]]$. Fig. (3b) shows that the computation time varies almost linearly with the number of preferences and tends to be constant for a large number of preferences. Note that, similar to $\mathcal{A}_\phi$, the computation for preferences also depends on their internal structure.

Number of robots. Keeping the $\mathcal{T}, \phi$ same as above, we vary the number of robots N from 1 to 400. The results, as visualized in Fig. (3c), indicate that initially, there is a rise in the computation time due to an increase in the bounds of the variables. However, the computation time remains largely unaffected when N increases further.

AP size. Finally, we note the effect of varying the size of atomic propositions on runtime. As expected, the computation time increases exponentially with $|AP|$ since the transitions in $\mathcal{A}$ are defined over 2^{AP}.

V. CONCLUSION

We present a novel approach to control synthesis for multi-robot teams task with global temporal logic specifications and with user preferences for relaxations. Preserving the explicit indication of progress towards satisfaction using a relaxed specification automaton, we propose a mixed-integer linear programming approach to avoid the computationally expensive product construction of robots' motion models. The runtime analysis indicates that our approach offers more expressivity as compared to existing MILP formulations for temporal logic planning while offering computational advantages with respect to purely automata-based methods.

REFERENCES

[1] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *2011 IEEE international conference on robotics and automation*, pp. 40–45, IEEE, 2011.

[2] K. Kim and G. E. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 265–271, IEEE, 2012.

[3] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the minimal revision problem of specification automata," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1515–1535, 2015.

[4] M. Lahijanian and M. Kwiatkowska, "Specification revision for markov decision processes with optimal trade-off," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7411–7418, IEEE, 2016.

[5] J. Tumova, L. I. R. Castro, S. Karaman, E. Frazzoli, and D. Rus, "Minimum-violation ltl planning with conflicting specifications," in *2013 American Control Conference*, pp. 200–205, IEEE, 2013.

[6] T. Wongpiromsarn, K. Slutsky, E. Frazzoli, and U. Topcu, "Minimum-violation planning for autonomous systems: Theoretical and practical considerations," in *2021 American Control Conference (ACC)*, pp. 4866–4872, IEEE, 2021.

[7] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scltl motion planning for mobility-on-demand," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1481–1488, IEEE, 2017.

[8] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 3, pp. 757–762, 2019.

[9] C. I. Vasile and C. Belta, "An automata-theoretic approach to the vehicle routing problem.," in *Robotics: Science and Systems*, pp. 1–9, Berkeley, CA, 2014.

[10] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[11] G. A. Cardona and C.-I. Vasile, "Partial satisfaction of signal temporal logic specifications for coordination of multi-robot systems," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 223–238, Springer, 2022.

[12] G. A. Cardona and C.-I. Vasile, "Preferences on partial satisfaction using weighted signal temporal logic specifications," in *2023 European Control Conference (ECC)*, pp. 1–6, IEEE, 2023.

[13] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe ltl specifications.," in *IJCAI*, vol. 15, pp. 1587–1593, Citeseer, 2015.

[14] H. Rahmani and J. M. O'Kane, "What to do when you can't do it all: Temporal logic planning with soft temporal logic constraints," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6619–6626, IEEE, 2020.

[15] S. Ahlberg and D. V. Dimarogonas, "Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 788–793, IEEE, 2019.

[16] 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.

[17] D. Kamale, E. Karyofylli, and C.-I. Vasile, "Automata-based optimal planning with relaxed specifications," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6525–6530, IEEE, 2021.

[18] 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.

[19] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multirobot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, vol. 64, no. 5, pp. 1916–1931, 2018.

[20] 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.

[21] G. A. Cardona, K. Leahy, and C.-I. Vasile, "Temporal logic swarm control with splitting and merging," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12423–12429, IEEE, 2023.

[22] 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.

[23] K. Leahy, A. Jones, and C.-I. Vasile, "Fast decomposition of temporal logic specifications for heterogeneous teams," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2297–2304, 2022.

[24] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.

[25] J. Karlsson, F. S. Barbosa, and J. Tumova, "Sampling-based motion planning with temporal logic missions and spatial preferences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15537–15543, 2020.

[26] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Distributed safe planning for satisfying minimal temporal relaxations of twtl specifications," *Robotics and Autonomous Systems*, vol. 142, p. 103801, 2021.

[27] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal multi-robot path planning with temporal logic constraints," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3087–3092, IEEE, 2011.

[28] A. Nikou, J. Tumova, and D. V. Dimarogonas, "Cooperative task planning of multi-agent systems under timed temporal specifications," in *2016 American Control Conference (ACC)*, pp. 7104–7109, IEEE, 2016.

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