Tightening Mixed-Integer Convex Relaxations for Efficient Temporal Logic Motion Planning via Logic Network Flow

Xuan Lin[1], Jiming Ren*[1], Yandong Luo*[1], Weijun Xie[2], and Ye Zhao[1]

***Abstract*— This paper proposes an optimization-based task and motion planning framework, named "Logic Network Flow," that integrates temporal logic specifications into mixed-integer programs for efficient robot planning. Inspired by the Graph-of-Convex-Sets formulation, temporal predicates are encoded as polyhedron constraints on each edge of a network flow model, instead of as constraints between nodes in traditional Logic Tree formulations. For temporal logic motion planning with piecewise-affine dynamic systems, comprehensive experiments demonstrate computational speedups of up to several orders of magnitude. Hardware demonstrations validate real-time replanning capabilities under dynamically changing environmental conditions. The project website is at `https://logicnetworkflow.github.io/`.**

I. Introduction

Task and motion planning (TAMP) with temporal logic specifications provides formal guarantees for robot motion plan safety and task completion [1]–[5]. Temporal logics, such as Linear Temporal Logic (LTL), Metric Temporal Logic (MTL), and Signal Temporal Logic (STL), offer a rich formal language for specifying complex robotic tasks, particularly when objectives extend beyond reaching predefined goal locations. However, motion planning under temporal logic constraints remains computationally intractable in theory due to its NP-hard nature. Although MILP solvers, e.g., via Branch-and-Bound (B&B), frequently solve such problems within a reasonable time in practice, they still retain worst-case exponential complexity. Solving realistically large temporal logic planning problems often requires minutes or even hours, rendering them impractical for real-time robotics applications.

In this work, we present a novel optimization formulation named "Logic Network Flow (LNF)." Compared to the conventional Logic Tree (LT) approaches [6], [7], which place temporal logic predicates at the leaf nodes of a tree structure, LNF transforms temporal logic specifications into network flow constraints to achieve tighter convex relaxations. LNF is evaluated on various temporal logic motion planning with piecewise-affine (PWA) dynamics, including Vehicle Routing Problems with Time Windows (VRPTW) for multi-robot coordination [8], optimal motion planning under temporal logic specifications using point mass [9] and linear inverted pendulum models [10], and multi-robot search-and-rescue scenarios. The formulation is further validated through hardware experiments with quadrupedal robots demonstrating real-time replanning capabilities under dynamically changing environmental conditions.

*These authors contributed equally to this work.

[1]George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA (e-mail: {xlin373, jren313, yluo471, ye.zhao}@gatech.edu).

[2]Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA (e-mail: wxie@gatech.edu).

II. Background

A. Temporal Logic Preliminaries

In this work, we focus on Metric Temporal Logic (MTL) and Signal Temporal Logic (STL), where the maximum trajectory length T to determine logic satisfiability is finite. We recursively define the syntax of temporal logic formulas as follows [9]: $\varphi := \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Diamond_{[k_1,k_2]}\ \varphi \mid \Box_{[k_1,k_2]}\ \varphi \mid \varphi_1\ \mathcal{U}_{[k_1,k_2]}\ \varphi_2$, where the semantics consists of not only boolean operations "and" ($\wedge$) and "or" ($\vee$), but also temporal operators "always" ($\Box$), "eventually" ($\Diamond$), and "until" ($\mathcal{U}$). φ, φ_1 and φ_2 in the definition are formulas, and π is an atomic predicate $\mathcal{X} \to \mathbb{B}$ whose truth value is defined by the sign of the convex function $g^\pi : \mathcal{X} \to \mathbb{R}$. We assume the convex function is a combination of linear functions, which can be expressed as $g^\pi(\boldsymbol{x}_k) = (\boldsymbol{a}^\pi)^\top \boldsymbol{x}_k + b^\pi$. Let $\Pi = \{\pi^1, \ldots, \pi^{|\Pi|}\}$ be the set of $|\Pi|$ atomic predicates. Define the binary predicate variable $z^{\pi^i_k} \in \mathbb{B}$ for each predicate π^i at time step k such that:

$$\begin{aligned}(\boldsymbol{a}^{\pi^i})^\top \boldsymbol{x}_k + b^{\pi^i} \geq 0 \Leftrightarrow z^{\pi^i_k} = 1,\\ (\boldsymbol{a}^{\pi^i})^\top \boldsymbol{x}_k + b^{\pi^i} < 0 \Leftrightarrow z^{\pi^i_k} = 0\end{aligned} \tag{1}$$

A run $\boldsymbol{\xi}$ that satisfies a temporal logic formula φ is denoted as $\boldsymbol{\xi} \models \varphi$. The satisfaction of a formula φ having a state signal $\boldsymbol{x}$ starting from time step k is defined inductively in the standard way (see [9]).

B. Logic Tree

Logic Tree (LT) [6], [7], also referred to as STL Tree [11], STL Parse Tree [12], and AND-OR Tree [13], is a hierarchical data structure encapsulating temporal logic formulas to facilitate efficient optimization solve. We provide its definition:

Definition 1. *A Logic Tree T^φ constructed from a temporal logic specification φ is defined as a tuple $(\circ, \Pi, \mathcal{N}, \tau)$, where:*

- $\circ \in \{\wedge, \vee\}$ *denotes the combination type;*
- $\Pi = \{\pi^1, \ldots, \pi^{|\Pi|}\}$ *is the set of $|\Pi|$ predicates associated with each leaf node in the tree T^φ. Each leaf node is assigned a variable $z^{\pi^i_k}$ to indicate its validity.*
- $\mathcal{N} = \{T^{\varphi_0}, T^{\varphi_1}, \ldots, T^{\varphi_n}\}$ *represents the set of $n+1$ internal nodes having at least one child, where the root*

node is denoted by $T^{\varphi} = T^{\varphi_0}$. Each node is associated with a temporal logic formula φ_i and a combination type $\circ$. Similarly, each internal node is assigned a variable z^{φ_i} to indicate the formula's validity.

- *$\tau = \{t^{\varphi_0}, t^{\varphi_1}, \ldots, t^{\varphi_n}\}$ is a list of starting times corresponding to each of the temporal logic formulas at the internal nodes and predicates at the leaf nodes.*

To encode temporal logic constraints represented by an LT into an optimization formulation, the works in [6] and [7] propose an MILP where binary variables are assigned to both internal nodes $z^{\varphi_i}, \forall i \in \{0, \ldots, n\}$ and leaf nodes $z^{\pi_k^i}, \forall t, i \in \{1, \ldots, |\Pi|\}$. For each internal node with a conjunction combination type: $\varphi = \wedge_{i=1}^{p} \varphi_i$ where φ_i is the suboperand of the child nodes, the following constraints are enforced:

$$z^{\varphi} \leq z^{\varphi_i}, \quad i = 1, \ldots, p \tag{2a}$$

$$z^{\varphi} \geq 1 - p + \sum_{i=1}^{p} z^{\varphi_i} \tag{2b}$$

Similarly, for each internal node with a disjunction combination type: $\varphi = \vee_{i=1}^{q} \varphi_i$, the following constraints are applied:

$$z^{\varphi} \geq z^{\varphi_i}, \quad i = 1, \ldots, q \tag{3a}$$

$$z^{\varphi} \leq \sum_{i=1}^{q} z^{\varphi_i} \tag{3b}$$

On the root node, $z^{\varphi_0} = 1$ must hold to ensure the full temporal logic specification is satisfied.

To complete the optimization formulation, we specify the dynamics and objective. A wide range of nonlinear dynamic systems can be effectively approximated using PWA dynamics [14]. Let $\boldsymbol{x} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T\}$ and $\boldsymbol{u} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{T-1}\}$ represent the state and control trajectories. We first define a collection of polytopes (i.e., bounded polyhedra): $\mathcal{D}_i \triangleq \{(\boldsymbol{x}_k, \boldsymbol{u}_k) \mid \boldsymbol{H}_1^i \boldsymbol{x}_k + \boldsymbol{H}_2^i \boldsymbol{u}_k \leq \boldsymbol{h}^i\}$, where $i \in \mathcal{I}$ denotes the index set. The discrete-time PWA dynamics are:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}^i \boldsymbol{x}_k + \boldsymbol{B}^i \boldsymbol{u}_k \tag{4a}$$

$$\boldsymbol{H}_1^i \boldsymbol{x}_k + \boldsymbol{H}_2^i \boldsymbol{u}_k \leq \boldsymbol{h}^i \quad \text{for some } i \in \mathcal{I} \tag{4b}$$

where $\boldsymbol{A}^i \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{B}^i \in \mathbb{R}^{n_x \times n_u}$ are the system matrices, and $\boldsymbol{H}_1^i \in \mathbb{R}^{n_c \times n_x}$, $\boldsymbol{H}_2^i \in \mathbb{R}^{n_c \times n_u}$, and $\boldsymbol{h}^i \in \mathbb{R}^{n_c}$ are the constraint matrices and vector, respectively.

We also structure the objective function for the control problem in the form of $f_{\text{obj}}(\boldsymbol{\xi}, \boldsymbol{z}^{\pi}) = f_{\text{obj}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}^{\pi}) = \boldsymbol{x}^{\top} \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u} + \boldsymbol{\Theta}^{\top} \boldsymbol{z}^{\pi}$, where the quadratic terms $\boldsymbol{x}^{\top} \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u}$ can represent energy consumption, control effort, or maximizing robustness measures. The linear term $\boldsymbol{\Theta}^{\top} \boldsymbol{z}^{\pi}$ rewards the satisfaction of specific predicates.

Finally, we present the following formulation to solve the motion planning problem with PWA dynamics under LT constraints:

Formulation 1. *(Optimization Formulation for Temporal Logic Motion Planning with PWA Dynamics using Logic Tree)*

$$\begin{aligned}
&\underset{\boldsymbol{\xi}, \boldsymbol{z}^{\varphi}, \boldsymbol{z}^{\pi}}{\textit{minimize}} \quad f_{obj}(\boldsymbol{\xi}, \boldsymbol{z}^{\pi}) \\
&\textit{s.t.} \quad \textit{Eqn. (4)}, \quad \boldsymbol{x}_0 \in \mathcal{X}_0 && \textit{(PWA dynamics)} \\
&\qquad \textit{Eqn. (1)}, \quad \forall\, k, \forall\, \pi \in \Pi && \textit{(Atomic predicates)} \\
&\qquad \left.\begin{array}{l} \textit{Eqn. (2) (3)}, \quad \forall\, T^{\varphi_i} \in \mathcal{N} \\ z^{\varphi_0} = 1 \end{array}\right\} && \textit{(LT constraints)}
\end{aligned}$$

where $\boldsymbol{z}^{\varphi}$ which is defined as:

$$\boldsymbol{z}^{\varphi} = [z^{\varphi_0}, z^{\varphi_1}, \ldots, z^{\varphi_n}]^{\top} \in \mathbb{B}^{n+1} \tag{5}$$

Constraints (2) and (3) guarantee that if all child variables z^{φ_i} claim binary values, then the parent variable z^{φ} will also take a binary value. By induction, this implies that as long as the leaf variables $\boldsymbol{z}^{\pi}$ are binary, all internal variables z^{φ_i} will be binary as well. Nevertheless, we explicitly impose binary constraints on all variables to ensure mathematical completeness of the formulation. If f_{obj} is linear in its variables, Formulation 1 is a Mixed-Integer Linear Program (MILP); with a quadratic objective, it becomes a Mixed-Integer Quadratic Program (MIQP).

III. The Logic Network Flow

A. Logic Network Flow

In this section, we present LNF, a new formulation for encoding temporal logic specifications. The key advantage of LNF over LT is that flow conservation constraints couple predicates across different branches of a disjunction, yielding tighter convex relaxations that significantly speed up the computation. We give the definition of LNF:

Definition 2. *A Logic Network Flow $\mathcal{F}^{\varphi}$ from the specification φ is defined as a tuple $(\mathcal{G}, \mathcal{P}, \Pi)$, where:*

- *$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph with a vertex $v_s \in \mathcal{V}$ as the source vertex and a vertex $v_t \in \mathcal{V}$ as the target vertex.*
- *$\Pi = \{\pi^1, \ldots, \pi^{|\Pi|}\}$ is the set of $|\Pi|$ predicates associated with each leaf node in the tree T^{φ} (as in Def. 1). The binary vector $\boldsymbol{z}^{\pi}$ represents the truth values of all predicates in Π at given times.*
- *$\mathcal{P}$ is a collection of sets of n_e ($n_e \leq |\Pi|$) possibly negated predicates that must hold true (or false if negated) to pass through each edge $e \in \mathcal{E}$, defined as $P_e := \{\pi^i | \pi^i \in \Pi$ s.t. $z^{\pi^i} = 1$ (or $z^{\pi^i} = 0$ if π^i is negated)$\} \in \mathcal{P}$.*

Let $e \in \mathcal{E}$ denote any edge in the LNF. For each vertex $v \in \mathcal{V}$, $\mathcal{E}_v^{\text{in}}$ is defined as the set of incoming edges to v and $\mathcal{E}_v^{\text{out}}$ as the set of outgoing edges from v. We present an algorithm in [15] that translates an LT into an LNF. Fig. 1 illustrates the conversion of LTs to LNFs for conjunction and disjunction nodes.

Given a $\mathcal{F}^{\varphi}$, we propose an optimization formulation with a tighter convex relaxation than LT. For each edge $e \in \mathcal{E}$ in the LNF, let $y_e \in \mathbb{B}$ specify if the edge is traversed by the flow, with a total of one unit of flow entering the network

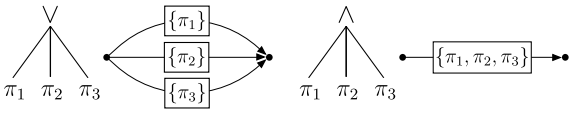

Fig. 1: An illustration of the strategy to translate conjunction and disjunction combination types from an LT to an LNF.

(i.e., $\sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} y_e = 1$). In addition, we associate a multi-dimensional continuous flow variable $\boldsymbol{\omega}_e \in [0,1]^{|\Pi|}$ to each edge, with its in-flow at v_s being $\boldsymbol{z}^\pi$.

We devise the convex set constraint to instantiate the logical requirements, such that if the in-flow passes through an edge e, we require $\boldsymbol{\omega}_e[i] = 1$ for all non-negated predicate $\pi_i \in P_e$, where π_i is the i^{th} predicate in Π; for negated predicates $\neg\pi_i \in P_e$, we require $\boldsymbol{\omega}_e[i] = 0$; and for predicates not contained in P_e, no constraint is imposed on $\boldsymbol{\omega}_e[i]$. These requirements can be expressed as two convex set constraints that need to be enforced concurrently:

$$\boldsymbol{\omega}_e \geq y_e \boldsymbol{v}_e^+, \quad \boldsymbol{\omega}_e \leq \mathbf{1}_{|\Pi|} - y_e \boldsymbol{v}_e^- \tag{6}$$

where $\boldsymbol{v}_e^+ \in \mathbb{B}^{|\Pi|}$ and $\boldsymbol{v}_e^- \in \mathbb{B}^{|\Pi|}$ are artificially designed column vectors to help encode the logical constraints. Specifically, $\boldsymbol{v}_e^+[i] = 1$ if $\pi_i \in P_e$, and 0 otherwise; and $\boldsymbol{v}_e^-[i] = 1$ if $\neg\pi_i \in P_e$, and 0 otherwise.

For each vertex $v \in \mathcal{V}$ with the input edges $\mathcal{E}_v^{\text{in}} \subset \mathcal{E}$ and the output edges $\mathcal{E}_v^{\text{out}} \subset \mathcal{E}$, flow conservation constraints are enforced on both y_e and $\boldsymbol{\omega}_e$:

$$\sum_{e\in\mathcal{E}_v^{\text{in}}} y_e = \sum_{e\in\mathcal{E}_v^{\text{out}}} y_e \tag{7a}$$

$$\sum_{e\in\mathcal{E}_v^{\text{in}}} \boldsymbol{\omega}_e = \sum_{e\in\mathcal{E}_v^{\text{out}}} \boldsymbol{\omega}_e \tag{7b}$$

Flow conservation constraints are also applied to source vertex v_s, where one unit of flow is injected:

$$1 = \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} y_e \tag{8a}$$

$$\boldsymbol{z}^\pi = \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} \boldsymbol{\omega}_e \tag{8b}$$

Given a constructed LNF $\mathcal{F}^\varphi$, we present the following formulation to solve the motion planning problem with PWA dynamics under LNF constraints:

Formulation 2. *(Optimization Formulation for Temporal Logic Motion Planning with PWA Dynamics using Logic Network Flow)*

$$\begin{aligned}
&\underset{\boldsymbol{\xi},\boldsymbol{y},\boldsymbol{\omega},\boldsymbol{z}^\pi}{\textit{minimize}} \quad f_{obj}(\boldsymbol{\xi}, \boldsymbol{z}^\pi) \\
&\textit{s.t.} \quad \textit{Eqn. (4)}, \quad \boldsymbol{x}_0 \in \mathcal{X}_0 \qquad \textit{(PWA dynamics)} \\
&\qquad \textit{Eqn. (1)}, \quad \forall\, k, \forall\, \pi \in \Pi \quad \textit{(Atomic predicates)} \\
&\qquad \left.\begin{array}{ll} \textit{Eqn. (6)}, & \forall e \in \mathcal{E} \\ \textit{Eqn. (7), (8)}, & \forall v \in \mathcal{V} \end{array}\right\} \quad \textit{(LNF constraints)}
\end{aligned}$$

where $\boldsymbol{y}$ and $\boldsymbol{\omega}$ are defined as:

$$\boldsymbol{y} = [y_1, y_2, \ldots, y_{|\mathcal{E}|}]^\top \in \mathbb{B}^{|\mathcal{E}|} \tag{9a}$$

$$\boldsymbol{\omega} = [\boldsymbol{\omega}_1^\top, \boldsymbol{\omega}_2^\top, \ldots, \boldsymbol{\omega}_{|\mathcal{E}|}^\top]^\top \in [0,1]^{|\mathcal{E}|\cdot|\Pi|} \tag{9b}$$

The convex relaxation of Formulation 2 is provably tighter than that of Formulation 1, which is formalized with the following theorem. The proof is provided in [15] and is omitted here for brevity.

Theorem 1. *For any given $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{\Theta}$, and $\boldsymbol{x}_0$, if the convex relaxation problem LNF-relax is feasible with an optimal cost $f^*_{\text{LNF-r}}$, then the convex relaxation problem LT-relax is also feasible with an optimal cost $f^*_{\text{LT-r}} \leq f^*_{\text{LNF-r}}$.*

In [15], we present a network-flow-based Fourier-Motzkin (F-M) elimination procedure that removes the continuous flow variables $\boldsymbol{\omega}_e$ while preserving convex relaxation tightness. This procedure replaces the flow variables $\boldsymbol{\omega}_e$ and their associated constraints in (6), (7b), and (8b) with sets of inequalities involving only variables $\boldsymbol{z}^\pi$ and y_e, such as (see [15] for the complete derivation):

$$\boldsymbol{z}^\pi \geq \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} y_e \boldsymbol{v}_e^+, \quad \boldsymbol{z}^\pi \leq \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} \mathbf{1}_{|\Pi|} - y_e \boldsymbol{v}_e^- \tag{10}$$

This transformation produces a formulation with significantly fewer continuous variables while retaining the same tightness of convex relaxation as the original formulation.

IV. Experiments

We conduct comprehensive experiments to evaluate the computational advantages of the LNF formulation compared to the LT formulation. In our full paper [15], we present four case studies: (i) vehicle routing problems with time windows using DNF models, (ii) multi-robot coordination using pre-computed trajectory libraries, (iii) optimal motion planning with PWA dynamics using big-M formulation, and (iv) hardware experiments demonstrating real-time replanning capabilities with quadrupedal robots in dynamic environments. Due to space limitations, we present one representative simulation experiment and one hardware demonstration below.

We showcase a practical application of our framework with a multi-robot search and rescue scenario, where three bipedal robots coordinate to complete multiple missions simultaneously in a disaster environment. The experiment runs on a 100 m $\times$ 100 m disaster zone covering multiple mission sites including crashed airplanes, factory plants and woods on fire, a search crew base camp, a helicopter landing zone, and multiple obstacles. The mission objective is to coordinate three robots to execute: (1) search and rescue at two aircraft crash sites with transport to the helicopter, (2) equipment retrieval from two damaged factories back to base camp, and (3) wildfire extinguishing, all within specified time windows. Let $\pi_k^{i,j}$ denote the predicate that robot $i \in \{1,2,3\}$ is at location j at time step k. The overall mission specification takes the form $\phi_{\text{mission}} = \bigwedge_{j=1}^{5} \phi_{\text{mission(j)}}$, where each sub-mission involves visiting task locations, staying for a specified duration, and returning to the designated base. The objective function incorporates two risk-aware locomotion costs modeled as negative log-probabilities of successful traversal: terrain roughness risk (incurred during movement) and time-varying overheating risk from fire intensity (incurred regardless of motion status).

We compare LNF and LT formulations solved by Gurobi 12.0 on two problem instances with different terrain roughness patterns. As expected, both formulations find the same globally optimal solution, with their key distinction in computational efficiency. For the first terrain configuration, LNF achieves $4\times$ speedups in both global optimality (165 s vs 632 s) and 10%-optimal solutions (39 s vs 165 s). For the second terrain configuration, LNF finds the global optimum in 69 seconds while LT exceeds 3000 seconds (over $40\times$ faster), and achieves $5\times$ speedup for 10%-optimal solutions (44 s vs 213 s). These results highlight that LNF's tighter convex relaxations (58% and 68%, versus 90% and 91% for LT) enabling the solver to find lower-risk robot paths within a shorter time.

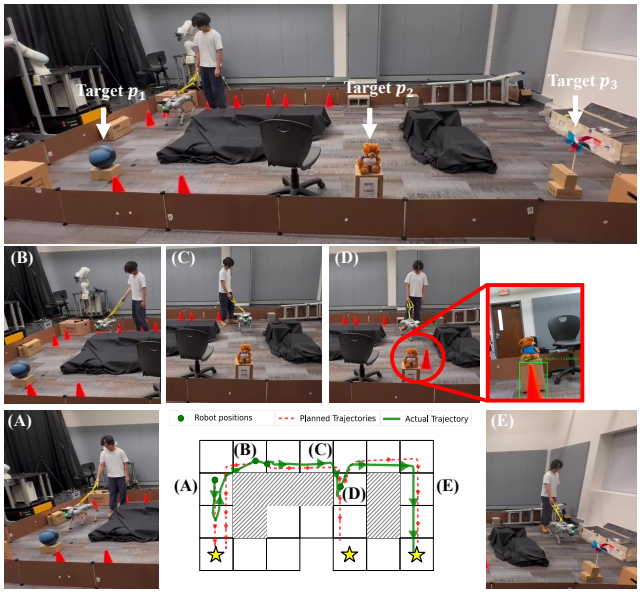

Fig. 2: A quadrupedal robot navigates toward three potential target locations (p_1, p_2, p_3) with continuous replanning under STL specifications. The central trajectory plot shows the robot's planned paths (red dotted lines) at several key instants with green dots indicating robot positions, and actual executed trajectory (green solid line). (A) The robot initially plans a path toward target p_1. (B) A human operator physically drags the robot backward, and the replanning drives the robot toward target p_2 (C) as a more optimal choice given the new position. (D) A cone obstacle is placed in the robot's path, which is recognized by the onboard RGB-D camera (highlighted in red box), making target p_2 infeasible. (E) The robot automatically replans and successfully reaches the new optimal goal, i.e., target p_3.

For the second experiment, we implement LNF in continuous configuration space for real-time temporal logic replanning with a Unitree Go2 quadrupedal robot. A single quadrupedal robot must select from 3 disjunctive goals. We use Bézier curves to model robot motion, with collision avoidance ensured through their convex hull property [16]. Integration with LNF is achieved using binary edge variables y_{ij}^k as predicate variables. The formulation includes initial position and velocity constraints enforced through the 0^{th} and 1^{st} order derivatives of the position Bézier curve, which also enforces the robot's initial orientation by assuming it aligns with the velocity direction. To enable this mapping, we add constraints ensuring that traversing each region takes a fixed duration Δt, so that the discrete time steps of y_{ij}^k correspond to actual time intervals $[k\Delta t, (k+1)\Delta t)$. We designate 3 target regions p_1, p_2, p_3, and the STL specification requires the robot to visit at least one of them with a minimum dwell time. The temporal logic specification is $\phi_{\text{replanning}} = \bigvee_{i\in\{p_1,p_2,p_3\}} \Diamond_{[0,T]}\Box_{[0,1]}\pi_k^i$.

The planner runs in a model predictive control fashion, continuously replanning as initial conditions and the environment change. We introduce two types of disturbances to demonstrate adaptive goal selection based on optimization feasibility and cost: manual perturbations where a person drags the robot to different locations, and traffic control cone obstacles suddenly showing up during robot motion. The robot is equipped with an Intel RealSense RGB-D camera, which detects cone positions in the 3D space. The cone positions are transformed into global coordinates using the robot's own pose, providing the absolute location of cones in the environment. The replanning algorithm then adds constraints $y_{ij}^k = 0$, $\forall k$ to any edge entering cells within a 1-meter blocking radius of the cone. The objective function minimizes the path length while applying an exponential time penalty α^k ($\alpha > 1$) for delayed goal completion. Specifically, when the robot reaches the goal region and remains there at time step k, a penalty of α^k is added to the objective, discouraging late arrivals at target locations.

We conduct multiple trials, where the robot initially plans to pursue a goal and automatically switches targets due to human interference and dynamic obstacles. Throughout the process, the planner continuously operates and generates a series of trajectories in real-time. Fig. 2 shows one representative behavior. Initially, the planner drives the robot toward target 1 (subfigure (A)). When a person drags the robot back significantly, moving it farther from target 1 (subfigure (B)), the robot automatically switches to pursue target 2 as that new target becomes more optimal (subfigure (C)). During the approach to target 2, a person places a cone obstacle in the robot's path, making target 2 infeasible (subfigure (D)). The robot quickly responds and redirects itself to target 3 instead (subfigure (E)). Overall, the system maintains a $2-10$ Hz replanning frequency.

V. Conclusion

We present Logic Network Flow, a novel optimization formulation that encodes temporal logic specifications as network flow constraints, achieving provably tighter convex relaxations compared to the standard Logic Tree approach. A network-flow-based Fourier-Motzkin elimination procedure further reduces the number of continuous variables while preserving relaxation tightness. Experimental results demonstrate computational speedups ranging from several times for complex dynamics to orders of magnitude for graph-based planning problems, and hardware experiments validate real-time replanning capabilities with quadrupedal robots.

Future work includes integrating LNF with decomposition approaches such as Benders Decomposition [17], [18], where LNF handles temporal logic and simplified dynamics in the master problem while complex nonlinear dynamics are delegated to subproblems.

REFERENCES

[1] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[2] Z. Zhao, S. Chen, Y. Ding, Z. Zhou, S. Zhang, D. Xu, and Y. Zhao, "A survey of optimization-based task and motion planning: From classical to learning approaches," *IEEE/ASME Transactions on Mechatronics*, 2024.

[3] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 618–12 624.

[4] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 346–352.

[5] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Transactions on Robotics*, 2023.

[6] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 5319–5325.

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

[8] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *2008 47th IEEE conference on decision and control*. IEEE, 2008, pp. 3953–3958.

[9] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 115–140, 2019.

[10] Z. Gu, R. Guo, W. Yates, Y. Chen, Y. Zhao, and Y. Zhao, "Walking-by-logic: Signal temporal logic-guided model predictive control for bipedal locomotion resilient to external perturbations," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 1121–1127.

[11] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

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

[13] 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.

[14] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Transactions on automatic control*, vol. 26, no. 2, pp. 346–358, 1981.

[15] X. Lin, J. Ren, Y. Luo, W. Xie, and Y. Zhao, "Towards tighter convex relaxation of mixed-integer programs: Leveraging logic network flow for task and motion planning," *arXiv preprint arXiv:2509.24235*, 2025.

[16] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[17] J. Ren, X. Lin, R. Mineyev, K. M. Feigh, S. Coogan, and Y. Zhao, "Accelerating signal-temporal-logic-based task and motion planning of bipedal navigation using benders decomposition," *arXiv preprint arXiv:2508.13407*, 2025.

[18] X. Lin, "Accelerating hybrid model predictive control using warm-started generalized benders decomposition," *arXiv preprint arXiv:2406.00780*, 2024.