Accelerating Large-scale Temporal-Logic-based Task and Motion Planning with Logic Network Flow

Xuan Lin, Jiming Ren, Samuel Coogan, and Ye Zhao

***Abstract*— We present "Logic Network Flow" (LNF), a novel optimization framework that encodes temporal logic specifications as network flow constraints. Unlike traditional formulations that recursively add constraints for logical operators, LNF encodes temporal predicates as polyhedron constraints on network flow edges, yielding provably tighter convex relaxations. When synthesized with Dynamic Network Flow (DNF) for system dynamics, our approach demonstrates significant computational advantages in multiple robot motion planning case studies. Empirical results show that LNF achieves tighter bounds while exploring fewer nodes during branch-and-bound, delivering computational speedups of up to several orders of magnitude over the traditional formulations.**

I. Introduction

Task and motion planning (TAMP) with temporal logic specifications has been widely accepted in the robotics community for two reasons. First, it provides safety and task completion guarantees for planning [1]–[5]. Second, temporal logics offer a rich formal language for specifying complex robotic tasks that goes beyond simple goal-reaching objectives. Despite these advantages, solving such TAMP problems remains challenging due to their NP-hard complexity, often requiring minutes or hours to reach a solution. As a result, current methods are often impractical for real-time robotics applications.

In this abstract, we build upon the LNF formulation, first introduced in [6], which encodes temporal logic specifications as network flow constraints to achieve tighter convex relaxations, by introducing a network-flow-based Fourier–Motzkin elimination method. This approach reduces the number of variables in the original formulation while preserving the tight relaxations critical for efficient optimization.

We evaluate our reduced formulation through extensive simulations on large-scale problems that surpass the typical complexity demonstrated in prior temporal logic planning literature [7], [8], such as planning bipedal locomotion for large-scale search and rescue. By achieving tighter convex relaxations without introducing additional variables, our formulation finds better incumbent solutions faster with computational speedups of up to several orders of magnitude compared to traditional formulations in [9], [10].

II. Logic Network Flow Formulation

A. Logic Network Flow

Several examples of LNFs converted from traditional structure representing temporal logic specifications are shown in Fig. 1. For each edge $e \in \mathcal{E}$ in the LNF, let $y_e \in \mathbb{B}$ indicate if the edge is traversed by the flow, with a total of one unit of flow entering the network (i.e., $\sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} y_e = 1$). Let $\Pi = \{\pi_1, \dots, \pi_{|\Pi|}\}$ be the set of $|\Pi|$ atomic predicates, we also 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 define the convex set constraint to represent the logical requirements, such that if the in-flow passes through an edge e, we require $\boldsymbol{\omega}_e[i] = 1$ for each 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 $\boldsymbol{\omega}_e[i] = 0$. For predicate π_i's that do not appear in P_e, we do not constrain $\boldsymbol{\omega}_e[i]$. This requirement can be expressed as two convex set constraints that need to be enforced simultaneously:

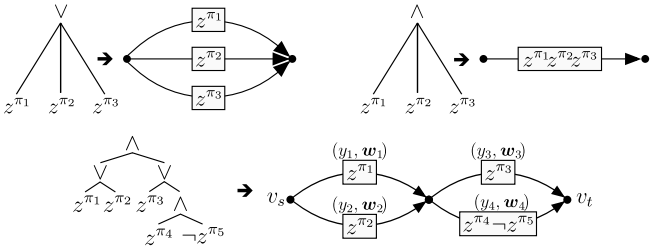

Fig. 1: Examples of Logic Network Flows (LNFs) converted from traditional "tree" structures. Each edge in the LNF possesses a binary variable y_i and a vector variable $\boldsymbol{\omega}_i$.

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

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 if otherwise, and $\boldsymbol{v}_e^-[i] = 1$ if $\neg\pi_i \in P_e$ and 0 if 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 constrains are enforced for both y_e and $\boldsymbol{\omega}_e$:

$$\sum_{e\in\mathcal{E}_v^{\text{in}}} y_e = \sum_{e\in\mathcal{E}_v^{\text{out}}} y_e, \quad \sum_{e\in\mathcal{E}_v^{\text{in}}} \boldsymbol{\omega}_e = \sum_{e\in\mathcal{E}_v^{\text{out}}} \boldsymbol{\omega}_e \tag{2}$$

In addition, flow conservation constraints are imposed on the source vertex v_s, where one unit of flow is injected:

$$\sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} y_e = 1, \quad \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} \boldsymbol{\omega}_e = \boldsymbol{z}^\pi \tag{3}$$

B. Dynamic Network Flow

Dynamics systems are abstracted into DNF [11]. To build a DNF, a set of discrete points $\mathcal{S} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_m\}$ is selected to represent locations robots must visit based on STL specifications. A DNF $\mathcal{G}_D = (\mathcal{E}_D, \mathcal{V}_D)$ contains $N \times |\mathcal{S}|$ vertices where each vertex $v_{p_i}^t$ corresponds to point $\boldsymbol{p}_i$ at timestep t. Edges connect vertices $v_{p_i}^t$ to $v_{p_j}^{t+K}$ if traveling

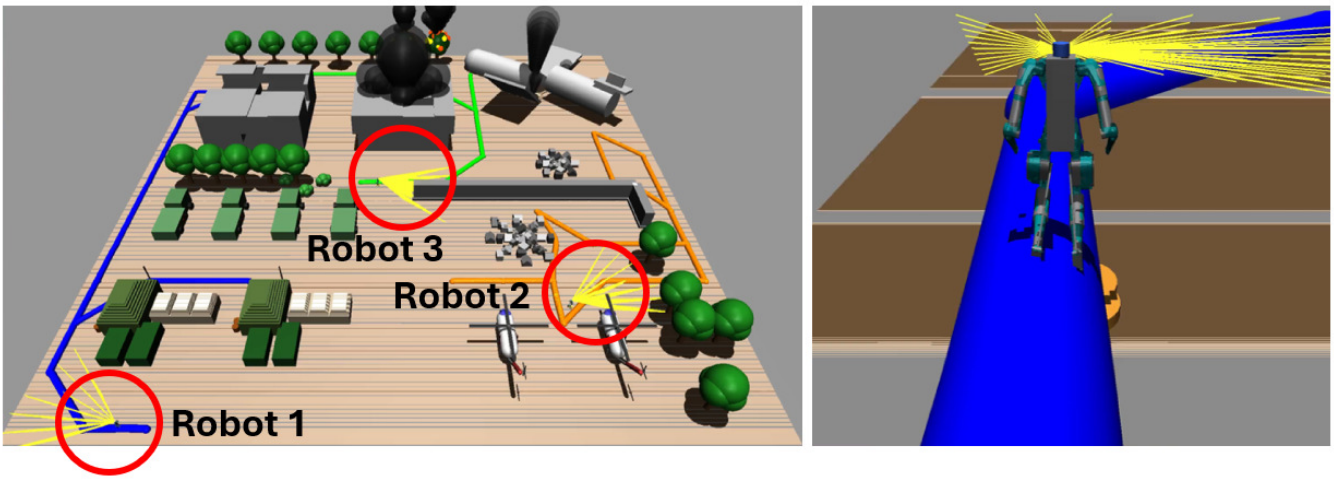

Fig. 2: Multi-robot search and rescue scenario using bipedal locomotion. **Left**: Three bipedal robots collaboratively search for subjects among disaster sites including crashed aircraft, burning houses, and forest fires. The robots follow planned paths (represented by blue, orange, and green paths) that satisfy temporal logic constraints. **Right**: An amplified view of a bipedal robot executing the planned trajectory.

from $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ takes K timesteps, with additional edges connecting consecutive timesteps at the same location. Each edge e carries flow $r_e \in [0,1]$ with associated cost $c_e r_e$. Flow conservation constraints are imposed at all vertices, with unit flow injected at the source vertex:

$$\sum_{e\in\mathcal{E}_v^{\text{in}}} r_e = \sum_{e\in\mathcal{E}_v^{\text{out}}} r_e, \forall v \in \mathcal{V}_D, \sum_{e\in\mathcal{E}_{v_s}^{\text{out}}} r_e = 1 \quad (4)$$

III. Experiments

Our approach is evaluated in a virtual scenario where multiple bipedal robots cooperatively execute a search and rescue mission in a 100×100 unit diaster area. The robots must reach several critical locations (aircraft crash sites, building fires, and forest fires) within strict time constraints to rescue survivors and extinguish fires, as shown in Fig. 2.

The environment is converted into a DNF using a library of precomputed trajectories based on linear inverted pendulum dynamics [12]. Each trajectory in this library defines an edge in the DNF with associated traversal time and energy cost. Rough terrain and high-temperature fire zones introduce risk factors, which are reflected in the edge costs. The robots must decide whether to avoid high-risk areas or pass through them to comply with timing constraints. Examples of increasing difficulties are evaluated: (1) small-scale: 3 robots and 9 sites, and (2) medium- and large-scale: 6 robots and 18 sites. The specifications for small and medium problems require each site to be visited at least once by any robot:

$$\varphi_{\text{search}} = \wedge_{j=1}^{n_s}(\Diamond_{[t_{j,1},t_{j,2}]}(\vee_{i=1}^{n_r}(\Box_{[0,1]} z_t^{i,p_j}))) \quad (5)$$

where n_r is the number of robots and n_s is the number of sites. For the large-scale problem, we add sequential dependencies between pairs of sites, expressed as:

$$\varphi_{\text{sequential}} = \bigwedge_{(i,j)\in\mathcal{P}} \left((\neg\pi_i \,\mathcal{U}\, \pi_j) \wedge \Diamond_{[0,T]}(\Box_{[0,1]}\pi_i)\right) \quad (6)$$

where $\mathcal{P}$ is the set of ordered pairs representing dependencies (e.g., site i cannot be visited until site j is visited). We choose $\Delta T = 8$ sec and planning horizon $N = 50$, planning 400 seconds ahead. All experiments were conducted on a computer with 12th Gen Intel Core i7-12800H CPU and 16GB memory, using Gurobi 12.0 as the MIP solver.

TABLE I: Comparison of Logic Network Flow (LNF) and Logic Tree (LT) formulations across different problem sizes. Times are in seconds.

	Small size		Medium size		Large size	
	LNF	LT	LNF	LT	LNF	LT
Relax. Gap (%)	**4.9**	56.1	**5.7**	65.8	**2.9**	59.7
# Cont. vars	**9656**	7498	**27443**	16203	**56772**	45211
# Bin. vars	**359**	359	**624**	624	**1098**	1098
# Constraints	**8125**	5862	**26604**	8765	**46622**	25495
T-Find (2% gap)	**1**	3	**9**	608	**27**	91
T-Find (optimal)	**2**	10	**93**	967	**1461**	>4000
T-Prove (10% gap)	**1**	10	**8**	3500	**16**	288
T-Prove (optimal)	**2**	10	**286**	>60000	**1916**	>4000

An example of the map with planned paths is shown in Fig. 2. For all three problem sizes, the planning results are shown in Table I. Major findings include the dramatically tighter relaxation gap achieved by LNF (2.9-5.7%) compared to the traditional formulations name Logic Tree (LT) [9], [10] (56.1-65.8%), which directly affects computational efficiency. While LNF encompasses more continuous variables and constraints, it finds global optimal solutions 5-10 times faster for small and medium problems, and significantly outperforms LT for large problems where LT cannot prove optimality within reasonable time limits. Furthermore, LNF can quickly find good suboptimal solutions (within 10% of optimal), showing speedups of up to 400 times for medium-sized problems. This demonstrates that the LNF formulation is more suitable for real-time applications than the baseline formulation.

To verify the physical feasibility of our planned motions, we simulate the search and rescue scenario in a MuJoCo environment. The bipedal robots use controllers with A-LIP model to execute the planned paths, as shown in Fig. 2. This implementation showcases that the optimized trajectories are executable on realistic bipedal systems navigating clustered environment while maintaining balance and satisfying all temporal specifications.

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] X. Lin, J. Ren, S. Coogan, and Y. Zhao, "Optimization-based task and motion planning under signal temporal logic specifications using logic network flow," *arXiv preprint arXiv:2409.19168*, 2024.

[7] V. Kurtz and H. Lin, "A more scalable mixed-integer encoding for metric temporal logic," *IEEE Control Systems Letters*, vol. 6, pp. 1718–1723, 2021.

[8] 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.

[9] 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.

[10] 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.

[11] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X: Proceedings of Workshop on the Algorithmic Foundations of Robotics*. Springer, 2013, pp. 157–173.

[12] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.