Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Hierarchical Temporal Logic Task and Motion Planning for Multi-Robot Systems

Zhongqi Wei*,[1], Xusheng Luo*,[1] and Changliu Liu[1]
*Equal contributions
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{zhongqi2, xushengl, cliu6}@andrew.cmu.edu

***Abstract*—Task and motion planning (TAMP) for multi-robot systems, which integrates discrete task planning with continuous motion planning, remains a challenging problem in robotics. Existing TAMP approaches often struggle to scale effectively for multi-robot systems with complex specifications, leading to infeasible solutions and prolonged computation times. This work addresses the TAMP problem in multi-robot settings where tasks are specified using expressive hierarchical temporal logic and task assignments are not pre-determined. Our approach leverages the efficiency of hierarchical temporal logic specifications for task-level planning and the optimization-based graph of convex sets method for motion-level planning, integrating them within a product graph framework. At the task level, we convert hierarchical temporal logic specifications into a single graph, embedding task allocation within its edges. At the motion level, we represent the feasible motions of multiple robots through convex sets in the configuration space, guided by a sampling-based motion planner. This formulation allows us to define the TAMP problem as a shortest path search within the product graph, where efficient convex optimization techniques can be applied. We prove that our approach is both sound and complete under mild assumptions. To enhance scalability, we introduce a pruning heuristic that reduces the product graph size, enabling efficient planning for high-dimensional multi-robot systems. Additionally, we extend our framework to cooperative pick-and-place tasks involving object handovers between robots. We evaluate our method across various high-dimensional multi-robot scenarios, including simulated and real-world environments with quadrupeds, robotic arms, and automated conveyor systems. Our results show that our approach outperforms existing methods in execution time and solution optimality while effectively scaling with task complexity.**

I. Introduction

Multi-robot systems often need to collaborate effectively and manage complex tasks. This has led to a growing demand for planning systems that can enable robots to efficiently execute long-horizon, intricate tasks. This challenge is commonly framed as a task and motion planning (TAMP) problem, formulated as a combination of discrete task planning and continuous motion planning [1]. TAMP is known as an NP-hard problem, particularly challenging for high-dimensional multi-robot systems tasked with complex and long-horizon operations. Traditionally, researchers in various fields have addressed these problems separately. To simplify the issues, certain assumptions are often employed, such as the existence of low-level controllers for task planning or the use of pre-defined tasks in motion planning. However, in real-world scenarios, predicting the feasibility of motion planning given a specific task specification is difficult. A feasible task description might cause an infeasible motion planning result. Therefore, traditional TAMP approaches focus on finding efficient ways to search the space of tasks [2, 3].

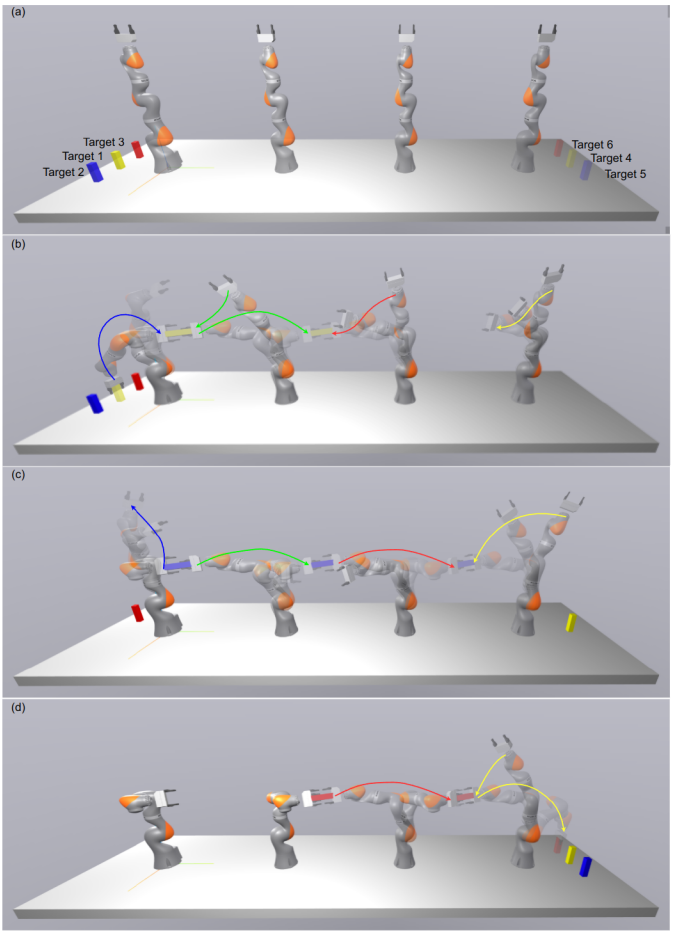


Fig. 1: Given the hierarchical temporal logic specifications (14), which specify transferring the yellow, blue, and red objects in order, our approach efficiently generates collision-free trajectories for four robotic manipulators, with a total of 28 degrees of freedom, to collaboratively complete the task in the shared workspace.

Recently, several methods have emerged that address these issues by integrating task and motion planning. The combinatorial complexity of TAMP can be partially mitigated if the problem is properly formulated [4–6]. However, those

methods either scale poorly to complex, long-horizon tasks or struggle with local minima due to the non-convex nature of the problem. A recent study [7] introduced an innovative framework that combines Linear Temporal Logic [8], as an expressive specification language for long-horizon tasks, with the Graph of Convex Sets (GCS) [9], a near-optimal motion planner. This unified approach demonstrates scalability to high-dimensional systems with up to 30 degrees of freedom. However, the method is limited to single-robot scenarios, as extending it to multi-robot systems is challenging due to the NP-hard nature of task allocation. Additionally, the LTL specifications are computationally intensive. For instance, the task of sequentially collecting five keys and opening five doors in [7] required over 40 minutes, with 32 minutes dedicated solely to handling the LTL specifications, reflecting the double-exponential complexity inherent to this process. More recently, study [10] proposed a hierarchical structure for LTL specifications, significantly reducing the computational burden of handling such specifications. While their planning algorithm claims to scale to scenarios involving up to 30 mobile robots under hierarchical LTL specifications, it does not address collision avoidance and collaboration among robots.

In this work, we address the problem of integrated task and motion planning (TAMP) for multi-robot systems under hierarchical temporal logic specifications, encompassing task allocation, task planning, and motion planning. We exploit the efficiency of hierarchical temporal logic specifications at the task level and the Graph of Convex Sets (GCS) at the motion level, integrating both into a unified framework. Building on ideas from [7], we formulate TAMP under hierarchical temporal logic specifications for multi-robot systems as a shortest-path problem within a product graph. Our proposed approach tackles several key challenges: converting hierarchical temporal logic specifications into a graph, addressing the task allocation complexities introduced by multiple robots, and efficiently connecting nodes for multiple robots in the graph of convex sets—extending beyond the existing work [9] that deals with at most two robots. The shortest-path problem in the product graph is formulated as a mixed-integer convex programming (MICP) problem, which can be efficiently solved using convex relaxation techniques [11]. We theoretically prove that our approach is both sound and complete under mild assumptions, and we empirically demonstrate its efficiency through case studies involving four robotic manipulators. To manage the complexity of the product graph, we implement a pruning strategy based on the structure of the tasks. Additionally, we extend the framework to scenarios involving multiple robotic manipulators that require handovers, where the necessity for handovers is not predetermined. The primary contributions of this work are as follows:

1) We formulate the multi-robot hierarchical temporal logic task and motion planning (TAMP) problem as a shortest-path problem in a product graph.
2) At the task level, we construct a graph for hierarchical temporal logic specifications and encompass the task allocation within the edges. To construct the GCS at the motion level, capturing the dynamics of multi-robot systems, and inspired by sampling-based motion planning, we propose an approach named *IRIS-RRT*, which efficiently connects the motion space.
3) We develop a heuristic to prune the product graph, significantly improving the computational efficiency of our method.
4) To address collaborative multi-robot pick-and-place tasks, we adapt the product graph to incorporate handover constraints and solve the shortest-path problem using mixed-integer convex programming (MICP).
5) We provide theoretical analyses to prove the soundness and completeness of our approach under mild assumptions.
6) We demonstrate the efficiency of our proposed method for long-horizon tasks, complex task specifications, and high-dimensional systems through several examples in both simulation and hardware. Additionally, we provide open-source code to reproduce our results.

The remainder of this paper is organized as follows. Section II reviews related work on task and motion planning and temporal logic specifications. Section III provides the background information on graphs of convex sets, linear temporal logic, and hierarchical specifications. The formulation of the problem and the underlying assumptions are presented in Section IV. Our main approach to the problem is described in Section V. Section VI offers proof of the soundness and completeness of the proposed approach. The multi-robot task and motion planning examples are described in Section VII. Finally, Section VIII discusses the limitations, and Section IX provides the conclusion of our approach.

II. Related Works

A. Task and Motion Planning

The task and motion planning (TAMP) aims to identify a sequence of symbolic actions and corresponding motion plans. An extensive review of TAMP can be found in [12, 13]. TAMP typically focuses on single-robot scenarios. In this work, we concentrate on multi-robot cases, as the aspect of task allocation is not applicable to a single robot. The primary focus in multi-robot TAMP is on the pick-up and placement of multiple objects by multiple manipulators, with the objective of determining which manipulator should pick up which objects and in what manner. One approach within this category employs search-based methods. This includes Conflict Based Search (CBS) [14], Monte-Carlo Tree Search (MCTS) [15], search in hyper-graphs [16], and search based on satisfiability modulo theories (SMT) solvers [17]. Another approach utilizes optimization-based methods. For example, [6] proposed the logic-geometric program (LGP), which integrates continuous motion planning and discrete task specifications into optimization problems. Similarly, [18] implicitly assigns actions based on the solution to a nonlinear optimization problem. Our work diverges from multi-robot TAMP in that most TAMP

studies do not consider logical or temporal constraints, with only a handful addressing dependency constraints that emerge from handover operations. Our approach incorporates these constraints, adding complexity to task planning and execution.

B. Control Synthesis under Temporal Logic Specifications

Temporal logic specifications play various critical roles in control synthesis, particularly within the realm of single-robot systems. Primarily, temporal logic formulas are utilized to define task specifications. For instance, [19] employs temporal logic to articulate temporally extended objectives and to respond to failures during learning from demonstrations. [20] implement skill repair mechanisms when existing skills are insufficient to satisfy LTL specifications. Similarly, [21] focuses on transferring skills across different LTL specifications. Beyond task definition, temporal logic formulas are also effective in representing constraints that dynamical systems must adhere to. For instance, [22] introduces LTLDoG, a diffusion-based policy for robot navigation that complies with LTL constraints. [23] utilizes temporal logic to define dynamic constraints, ensuring the stability of walking trajectories in bipedal robots. Additionally, [24] uses temporal logic to impose constraints on switching protocols, enabling a single agent to robustly track multiple targets.

In the realm of multi-robot systems governed by LTL task specifications or constraints, LTL formulas are generally categorized into *local* and *global* forms. One strategy, as demonstrated in studies such as [25–27], involves assigning LTL tasks locally to each individual robot within the team. Alternatively, a global LTL specification can be designated for the entire team. When global LTL specifications are employed, they may either explicitly allocate tasks to specific robots [28–35, 21, 36] or leave task assignments unspecified among the robots [37–41]. This latter approach aligns with the problem addressed in our current work.

Global specifications that do not explicitly assign tasks to robots typically require decomposition to facilitate task allocation. This decomposition can be achieved through three primary methods: (a) The most common technique, used in works such as [42–51], involves breaking down a global specification into multiple tasks by leveraging the transition relations within the automaton, which graphically represents an LTL formula. (b) As demonstrated in [38, 52], the second approach utilizes BMC techniques [53] to develop a Boolean Satisfaction or Integer Linear Programming (ILP) model. This model simultaneously handles task allocation and implicitly decomposes tasks within a unified framework. (c) The third method, proposed by [54], interacts directly with the syntax tree of LTL formulas to divide the global specification into smaller, more manageable sub-specifications.

However, most of the aforementioned methods either adopt a hierarchical framework, where task allocation is determined first followed by low-level plan synthesis—without ensuring the feasibility at the low level—or they employ a simultaneous task allocation and planning approach to guarantee completeness, but they primarily address discrete environments and action models, which may be suitable for mobile robots but are not applicable to robotic arms. In contrast, our work distinguishes itself by performing simultaneous task allocation and planning while directly considering continuous dynamics.

III. Preliminary

Notation: Let $\mathbb{R}$ denote the set of all real values, $[K] = \{1, \ldots, K\}$ represent the set of integers from 1 to K, respectively, and $|\cdot|$ denote the cardinality of a set.

A. Linear Temporal Logic

Linear Temporal Logic (LTL) [8] is a type of formal logic whose basic ingredients are a set of atomic propositions $\pi \in \mathcal{AP}$, the boolean operators, conjunction $\wedge$ and negation $\neg$, and temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over $\mathcal{AP}$ abide by the grammar

$$\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \,\mathcal{U}\, \phi_2. \quad (1)$$

For brevity, we abstain from deriving other Boolean and temporal operators, e.g., *disjunction* $\vee$, *implication* $\Rightarrow$, *always* $\Box$, *eventually* $\Diamond$, which can be found in [8].

An infinite *word* σ over the alphabet $2^{\mathcal{AP}}$ is defined as an infinite sequence $\sigma = \sigma_0\sigma_1 \ldots \in (2^{\mathcal{AP}})^\omega$, where ω denotes an infinite repetition and $\sigma_k \in 2^{\mathcal{AP}}$, $\forall k \in \mathbb{N}$. The language $\texttt{Words}(\phi) = \{\sigma | \sigma \models \phi\}$ is defined as the set of words that satisfy the LTL formula ϕ, where $\models \subseteq (2^{\mathcal{AP}})^\omega \times \phi$ is the satisfaction relation. In this work, we focus on a particular subset of LTL formulas known as syntactically co-safe LTL, or sc-LTL for short [55]. As established by [55], any LTL formula encompassing only the temporal operators $\Diamond$ and $\mathcal{U}$ and written in positive normal form (where negation is exclusively before atomic propositions) is classified under syntactically co-safe formulas. Sc-LTL formulas can be satisfied by finite sequences followed by infinite repetitions. This characteristic makes sc-LTL apt for modeling and reasoning about systems with finite durations, such as those found in the robotics field. Any sc-LTL formula can be converted into a Deterministic Finite Automaton (DFA).

Definition 3.1: (Deterministic Finite Automaton (DFA) [8]) A DFA $\mathcal{A}$ of a sc-LTL formula ϕ over $2^{\mathcal{AP}}$ is defined as a tuple $\mathcal{A}(\phi) = \left(\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{Q}_{\mathcal{B}}^F\right)$, where

- $\mathcal{Q}$ is the set of states;
- $\Sigma = 2^{\mathcal{AP}}$ is an alphabet;
- $\delta \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$ is the transition relation with $|\delta(q, \sigma)| \leq 1$ for all states $q \in \mathcal{Q}$ and all symbols $\sigma \in \Sigma$;
- $q_0 \in \mathcal{Q}$ is the unique initial state;
- $\mathcal{Q}^F \subseteq \mathcal{Q}$ is a set of accepting states.

A *finite run* ρ of $\mathcal{A}$ over a finite word $\sigma = \sigma_0\sigma_1 \ldots \sigma_h$ is a sequence $\rho = q_0q_1 \ldots q_{h+1}$ such that $(q_i, \sigma_i, q_{i+1}) \in \delta$, $\forall i = 0, \ldots, h$. A run ρ is called *accepting* if $q_{h+1} \in \mathcal{Q}^F$. The words σ that produce an accepting run of $\mathcal{A}$ constitute the accepted language of $\mathcal{A}$, denoted by $\mathcal{L}_\mathcal{A}$. Then [8] proves that the accepted language of $\mathcal{A}$ is equivalent to the words of ϕ, i.e., $\mathcal{L}_\mathcal{A} = \texttt{Words}(\phi)$.

B. Hierarchical sc-LTL

The work [10] incorporates a hierarchical structure into LTL over finite traces. In this paper, we focus on hierarchical sc-LTL, following [10], which indicates that the approach can also be applied to sc-LTL.

Definition 3.2: (Hierarchical sc-LTL [10]) Hierarchical sc-LTL is structured into K levels, labeled $L_1, \ldots, L_K$, arranged from the highest to the lowest. Each level L_k with $k \in [K]$ contains n_k sc-LTL formulas. The hierarchical sc-LTL specification is represented as $\Phi = \{\phi_k^i \mid k \in [K], i \in [n_k]\}$, where ϕ_k^i denotes the i-th sc-LTL formula at level L_k. Let Φ_k denote the set of formulas at level L_k, and let $\texttt{Prop}(\phi_k^i)$ represent the set of propositions appearing in formula ϕ_k^i. The hierarchical sc-LTL follows these rules:

1) There is exactly one formula at the highest level: $n_1 = 1$.
2) Each formula at level L_k consists either entirely of atomic propositions, i.e., $\texttt{Prop}(\phi_k^i) \subseteq \mathcal{AP}$, or entirely of formulas from the next lower level, i.e., $\texttt{Prop}(\phi_k^i) \subseteq \Phi_{k+1}$.
3) Each formula at level L_{k+1} appears in exactly one formula at the next higher level: $\phi_{k+1}^i \in \bigcup_{j \in [n_k]} \texttt{Prop}(\phi_k^j)$ and $\texttt{Prop}(\phi_k^{j_1}) \cap \texttt{Prop}(\phi_k^{j_2}) = \varnothing$, for $j_1, j_2 \in [n_k]$ and $j_1 \neq j_2$.

Example 1: (Hierarchical sc-LTL) The following hierarchical sc-LTL specifications state that completing tasks a and b and c, and c should not be the last one to be finished:

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond \phi_2^1 \wedge \neg \phi_2^1 \,\mathcal{U}\, \phi_2^2 \\ L_2: \quad & \phi_2^1 = \Diamond a \wedge \Diamond b \\ & \phi_2^2 = \Diamond c. \end{aligned} \tag{2}$$

The symbol ϕ_k^i is referred to as *composite proposition* if it is inside a formula, and is referred to as specification otherwise.

Definition 3.3: (Specification hierarchy tree [10]) The specification hierarchy tree, denoted as $\mathcal{G}_h = (\mathcal{V}_h, \mathcal{E}_h)$, is a tree where each node represents a specification within the hierarchical sc-LTL, and an edge (u, v) indicates that specification u contains specification v as a composite proposition. Any hierarchical sc-LTL specifications can be turned into a specification hierarchy tree.

A specification is termed as a leaf specification if the associated node in the graph $\mathcal{G}_h$ does not have any children. Let Φ_ℓ denote the set of leaf specifications. Note that not all leaf specifications necessarily reside at level L_K.

C. Shorest paths in Graphs of Convex Sets (GCS)

In this section, we introduce the shortest path formulation in GCS. It is introduced in [11] and applied to robot motion planning problems in [9]. The goal is to find the minimum-cost path from a start vertex to a target vertex in a graph. [11] defines a Graph of Convex Sets as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$. Each vertex $v \in \mathcal{V}$ is associated with a convex set $\mathcal{S}_v$ and a point $s_v \in \mathcal{S}_v$, each edge $e = (u, v) \in \mathcal{E}$ is associated with a non-negative and convex function $l_e(s_u, s_v)$ and a convex constraint $(s_u, s_v) \in \mathcal{S}_e$. For a fixed start vertex s and target vertex t, we are seeking a path p as a sequence of vertices that connect the start vertex s and t through the subset $\mathcal{E}_p$ of the edges $\mathcal{E}$. Denoting the set of all paths in the graph $\mathcal{G}$ as $\mathcal{P}$, the shortest path problem in GCS states as follows:

$$\begin{aligned} \min \quad & \sum_{e=(u,v) \in \mathcal{E}_p} l_e(s_u, s_v) \\ \text{s.t.} \quad & p \in \mathcal{P}, \\ & s_v \in \mathcal{S}_v, \quad \forall v \in p, \\ & (s_u, s_v) \in \mathcal{S}_e, \quad \forall e = (u, v) \in \mathcal{E}_p. \end{aligned} \tag{3}$$

Although the shortest path problem (SPP) in the GCS is NP-hard, an efficient mixed-integer convex program (MICP) formulation was proposed in [11]. This MICP has a very tight convex relaxation, meaning the optimal result can be tightly approximated by the solution of the convex optimization. The GCS has been further extended to robotic motion planning around obstacles, as detailed in [9]. The results demonstrate that GCS is a robust trajectory optimization framework, capable of encoding various costs and constraints.

D. Convex Set (CS)-based Transition System

For simplicity, we assume that the configuration space dimension for each robot is the same, denoted by d. For a multi-robot system composed of n robots, the CS-based transition system is defined as follows.

Definition 3.4: (CS-based Transition System) A CS-based multi-robot transition system (TS) is defined as $\mathcal{T} = (\mathcal{S}, \Delta, S_0, \mathcal{L})$:

- $\mathcal{S} \subset \mathbb{R}^{nd}$ represents the set of convex sets of configuration states where all robots are guaranteed to be collision-free.
- $\Delta \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, where $(S, S') \in \Delta$ if the sets S and S' are either overlapping or adjacent.
- $\mathcal{L} : \mathcal{S} \to 2^{\mathcal{AP}}$ is the labeling function, which maps any two configuration states within the same convex set to the same sets of labels, i.e., $\mathcal{L}(s) = \mathcal{L}(s') = \mathcal{L}(S)$ for all $S \in \mathcal{S}$ and for any $s, s' \in S$. With a slight abuse of notation, we apply $\mathcal{L}$ to a convex set and any state within the convex set.
- $S_0 \in \mathcal{S}$ is the convex set that includes the initial configuration state $(s_1^0, \ldots, s_n^0)$.

Given the presence of multiple specifications in hierarchical sc-LTL, a state-specification plan associates each robot state with a sc-LTL specification that a specific robot is executing.

Definition 3.5: (State-Specification Sequence [10]) A state-specification sequence with a horizon h, represented as τ, is a timed sequence $\tau = \tau_0 \tau_1 \tau_2 \ldots \tau_h$. Here, $\tau_i = ((s_1^i, \psi_1^i), (s_2^i, \psi_2^i), \ldots, (s_n^i, \psi_n^i))$ is the collective state-specification pairs of n robots at the i-th timestep, where $(s_1^i, \ldots, s_n^i)$ is the configuration state, and $\psi_r^i \in \Phi_\ell \cup \{\epsilon\}$, with ϵ indicating the system's non-involvement in any leaf specification at that time.

A state-specification sequence is considered to satisfy the given hierarchical sc-LTL specifications if the root specification ϕ_1^1 is fulfilled; we refer the reader to [10] for further details.

IV. Problem Formulation

In this section, we introduce the problem formulation for multi-robot hierarchical temporal logic task and motion planning.

Definition 4.1: (Trajectory) A trajectory for robot i defined as $\rho_i : \mathbb{R}^+ \to \mathbb{R}^d$ is a function that maps any $t \in \mathbb{R}^+$ in time to a robot configuration $s \in \mathbb{R}^d$.

Problem 1: Consider a n-robot system with initial configuration $s_0 = (s_1^0, \ldots, s_n^0) \in \mathbb{R}^{nd}$, and hierarchical sc-LTL specifications Φ, the hierarchical temporal logic task and motion planning problem requires finding the collision-free robot trajectories $\rho = [\rho_1, \ldots, \rho_n]$ with minimum-cost that satisfy the hierarchical sc-LTL specifications. The planning problem is shown as follows:

$$\min_{\rho=[\rho_1,\ldots,\rho_n]} \quad \mathtt{J}(\rho) \tag{4a}$$

$$\text{s.t.} \quad \mathtt{Trace}(\rho) \models \Phi, \tag{4b}$$

$$\rho_i(t) \cap \rho_j(t) = \emptyset, \quad \forall i, j \in [n], t \in \mathbb{R}^+, \tag{4c}$$

$$\rho_i(t) \cap \mathcal{O} = \emptyset, \quad \forall i \in [n], t \in \mathbb{R}^+, \tag{4d}$$

$$\rho_i(0) = s_i^0, \quad \forall i \in [n]. \tag{4e}$$

where $\mathcal{O}$ represents obstacles, and $\mathtt{Trace}$ returns the trace of trajectories by applying labeling function $\mathcal{L}$ to each state in the trajectories ρ.

We assume the cost $\mathtt{J}$ in (4a) is smooth and strictly convex. It can be any convex function of trajectory ρ. For example, the cost can be the path length and include derivatives of states. The first constraint (4b) ensures the trajectories satisfy the task specification expressed as hierarchical sc-LTL specifications. The second constraint (4c) and third constraint (4d) ensure non-convex collision-avoidance constraints, requiring the robot to avoid collisions with itself and the surrounding environment. To deal with those non-convex collision-avoidance constraints, inspired by trajectory optimization [9], we mitigate those collision avoidance constraints by requiring the robot to move through a collection of safe convex sets $S_1, S_2, \ldots \subset \mathbb{R}^{nd}$ that do not collide with obstacles. The last constraint (4e) imposes initial conditions for each robot.

V. Approach

In this section, we present our approach to Problem 1. The basic idea of our method involves several key steps: First, we construct the labeled convex set regions in configuration space to apply hierarchical sc-LTL specification to multi-robot motion planning. To ensure the existence of a feasible path between these convex set regions, we proposed a rapidly exploring random tree (RRT)-guided method to construct the connected convex set regions to connect those labeled convex set regions. Using these convex sets and their labels,

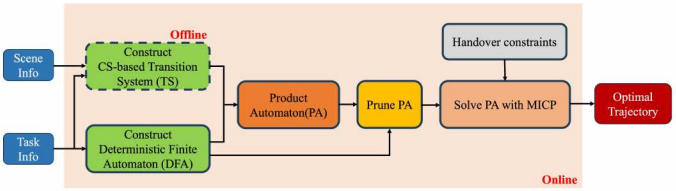


Fig. 2: An architecture for hierarchical temporal logic task and motion planning, where the transition system can be precomputed offline.

Algorithm 1: Construct a labeled convex set

Input: Multi-robot system plant, atomic proposition π

Output: Labeled joint configuration s_{label}, labeled convex set S_{label}

1 $s_{\text{label}} \leftarrow \mathtt{CalLabeledConfiguration}(\pi)$;

2 $S_{\text{label}} \leftarrow \mathtt{IRIS\text{-}NP}(\text{plant}, s_{\text{label}})$;

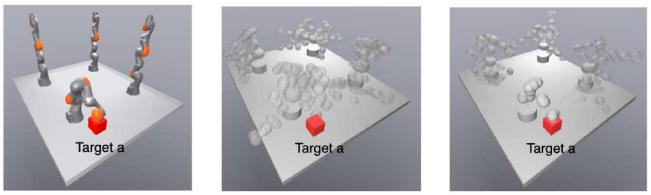
3 **return** $s_{\text{label}}, S_{\text{label}}$;

we then construct the CS-based transition system for multi-robots, as introduced in Section V-A. Second, we convert each hierarchical sc-LTL formula into a Deterministic Finite Automaton (DFA) and create a product graph by taking the product of the CS-based transition system and the DFAs, as described in Section V-B. Note that while our approach can model robot collaboration and collision avoidance, these aspects are not considered in [10]. Next, to mitigate the computational complexity associated with the potentially large product graph, we implement a graph pruning technique to simplify the problem based on task specifications, as outlined in Section V-C. Finally, in Section V-D, we solve the pruned product graph using the mixed integer convex program (MICP) and extend our optimization framework to handle multi-robot handover tasks. The overall architecture of our multi-robots task and motion planning algorithm is illustrated in Fig. 2. For a static environment, the CS-based transition system can be precomputed offline, while the remaining modules are computed online.

A. Construct CS-based Transition System for Multi-robots

To build a CS-based transition system defined in Def. 3.4, we begin by constructing labeled convex sets, represented as S_{label}, for each atomic proposition $\pi \in \mathcal{AP}$, ensuring that $\mathcal{L}(S_{\text{label}}) = \pi$. Subsequently, given any two labeled convex sets S^i_{label} and S^j_{label}, we create a collection of *connected* convex sets, denoted as $\mathcal{S}^{i,j}_{\text{connect}}$ to establish a feasible pathway between S^i_{label} and S^j_{label}, if possible.

The process of constructing labeled convex sets, as outlined in Alg. 1, proceeds as follows: Initially, for a multi-robot system and a given atomic proposition π, we compute a configuration s_{label} that satisfies the specified atomic proposition π [line 1]. This configuration is typically determined through robot inverse kinematics. Following this, we generate a labeled convex set S_{label} starting with s_{label} as the seed point, meaning that this configuration is contained within the convex set.



(a) (b) (c)

Fig. 3: An example of generating the labeled convex region for a multi-robot system. The atomic proposition is target a. In Fig. 3(a), the atomic proposition π is used to compute the labeled joint configuration s_{label} through robot inverse kinematics. In this configuration, the bottom robot reaches the position labeled as target a, while the configurations of the remaining robots are unconstrained. Using s_{label} as a seed point, the IRIS-NP algorithm can generate a convex set region that contains this seed point. The sampled configurations inside the convex region are shown in Fig. 3(b). Note that not all configurations of the bottom robot ensure reaching target a. Fig. 3(c) illustrates the labeled convex region S_{label} generated by adding the bottom robot's end-effector position constraints in IRIS-NP algorithm, ensuring that all configurations in the labeled convex region satisfy the atomic proposition target a.

Algorithm 2: IRIS-RRT

Input: Multi-robot system plant, start configuration s^i_{label} and target configuration s^j_{label}
Output: Sets of connected convex sets $\mathcal{S}^{i,j}_{\text{connect}}$

1 path $\leftarrow$ `RRT`$(s^i_{\text{label}}, s^j_{\text{label}})$;
2 $\mathcal{S}^{i,j}_{\text{connect}} \leftarrow$ `IRIS-NP`(plant, s^i_{label}) ;
3 $s^{\text{old}}_{\text{seed}} \leftarrow s^i_{\text{label}}$;
4 **while** s^j_{label} **is not in convex sets** $\mathcal{S}^{i,j}_{\text{connect}}$ **do**
5 **maximize** $_{s^{\text{new}}_{\text{seed}}}$ `distance_along_path`$(s^{\text{old}}_{\text{seed}}, s^{\text{new}}_{\text{seed}})$;
6 **subject to** $s^{\text{new}}_{\text{seed}} \in \mathcal{S}^{i,j}_{\text{connect}}$ and $s^{\text{new}}_{\text{seed}} \in$ path ;
7 $S_{\text{iris}} \leftarrow$ `IRIS-NP`(plant, $s^{\text{new}}_{\text{seed}}$) ;
8 $\mathcal{S}^{i,j}_{\text{connect}} \leftarrow \mathcal{S}^{i,j}_{\text{connect}} \cup \{S_{\text{iris}}\}$;
9 $s^{\text{old}}_{\text{seed}} \leftarrow s^{\text{new}}_{\text{seed}}$;
10 **return** $\mathcal{S}^{i,j}_{\text{connect}}$;

In this work, convex sets are constructed using the IRIS-NP algorithm [56], which ensures that any configuration within the convex set is free from collisions. However, IRIS-NP does not guarantee that every configuration in the convex set satisfies the atomic proposition π. To address this limitation, we incorporate additional configuration constraints in the IRIS-NP algorithm to ensure that all configurations within S_{label} satisfy the atomic proposition π [line 2]. An example of this algorithm applied to a four-robot manipulator system is depicted in Fig. 3. One such additional constraint is requiring a specific robot's end-effector to reach a designated position.

After generating the labeled convex sets $\mathcal{S}$, we construct a sequence of connected convex sets $\mathcal{S}^{i,j}_{\text{connect}}$ to ensure connectivity between any two labeled convex sets, if possible. For high-dimensional degree-of-freedom (DoF) multi-robot systems, generating these connected convex regions randomly in the robot configuration space might not successfully establish connections between the labeled convex sets, and the optimal solution might not traverse these convex sets. To address this challenge, we introduce the IRIS-RRT algorithm, outlined in Alg. 2. Initially, Alg. 2 uses Rapidly-exploring Random Tree (RRT) method [57] to find a feasible path between the two labeled configurations [line 1]. Subsequently, it constructs a set of connected convex sets along this path, spacing the seed configuration states at maximum intervals along the feasible path to minimize the number of connected convex sets [lines 4-9]. To this end, starting with the initial configuration state s^i_{label}, its convex set S^i_{label} is used to initialize $\mathcal{S}^{i,j}_{\text{connect}}$. The next step involves identifying another configuration state along the feasible path that is the farthest from s^i_{label} yet still within $\mathcal{S}^{i,j}_{\text{connect}}$. This configuration state then serves as a new seed, and the process is iteratively continued to extend $\mathcal{S}^{i,j}_{\text{connect}}$ until it contains the goal configuration s^j_{label}. Note that the RRT method in Alg. 2 could be replaced by any motion planning algorithm. A 2D example of the construction of convex sets by IRIS-RRT is shown in Fig. 4.

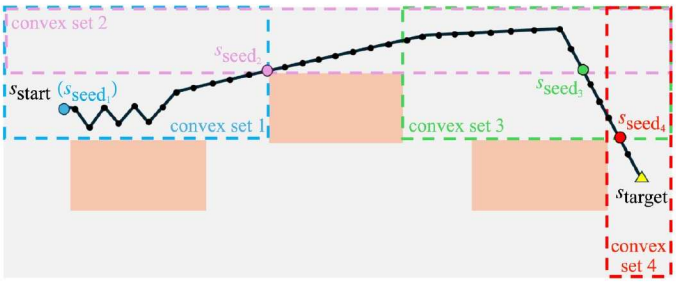


Fig. 4: A 2D example for collision-free convex set construction. The orange blocks represent obstacles. Given the robot's start configuration s_{start} and goal configuration s_{target}, IRIS-RRT algorithm uses RRT path (black line) as a guide to approximate the connected convex sets to connect the start and goal configurations. In the graph, s_{seed_i} represents the seed configuration anchoring the i-th convex set, which is depicted as a rectangle surrounded by dashed lines. Each pair of seed configuration and its corresponding convex set is highlighted in the same color. Note that the seed configurations typically lie at the intersections of the path generated by RRT and the previous convex sets. Moreover, the path generated by RRT does not need to be smooth or optimal, as it primarily serves to guide the construction of convex sets.

B. Construct Product Automaton

In what follows, let $\delta(q, q')$ denote the propositional logic formula that enables the transition from q to q' in DFA. We begin by addressing the challenge of constructing a graph, referred to as the *total product DFA*, for hierarchical sc-LTL specifications. First, for a set of specifications that have the same level, we construct their product DFA.

Definition 5.1: (Product DFA (PDFA)) Consider a DFA $\mathcal{A}^i_k = (\mathcal{Q}^i_k, \Sigma^i_k, \delta^i_k, q^i_{0,k}, \mathcal{Q}^{F,i}_k)$ of the i-th specification ϕ^i_k

at level k. The PDFA for level k, denoted as $\mathcal{A}_k = (\mathcal{Q}_k, \Sigma_k, \delta_k, q_{0,k}, \mathcal{Q}_k^F)$, is defined as follows:

- $\mathcal{Q}_k = \mathcal{Q}_k^1 \times \ldots \times \mathcal{Q}_k^{n_k}$ is the Cartesian product of automaton states across the specifications at level L_k;
- $\Sigma_k = \Sigma_k^1 \times \ldots \times \Sigma_k^{n_k}$ is the combined set of symbols from all DFAs at this level, where $\Sigma_k^i = 2^{\texttt{Prop}(\phi_k^i)}$ if ϕ_k^i is a non-leaf specification, representing the child specifications of ϕ_k^i at the immediate lower level $k+1$; otherwise, $\Sigma_k^i = 2^{\mathcal{AP}}$;
- $\delta_k \subseteq \mathcal{Q}_k \times \Sigma_k \times \mathcal{Q}_k$ is the transition relation, where a transition $\left((q_k^1, \ldots, q_k^{n_k}), (\sigma_k^1, \ldots, \sigma_k^{n_k}), (q_k^{1'}, \ldots, q_k^{n_k'})\right) \in \delta_k$ exists if $(q_k^i, \sigma_k^i, q_k^{i'}) \in \delta_k^i$ for all $i \in [n_k]$;
- $q_{0,k} = (q_{0,k}^1, \ldots, q_{0,k}^{n_k})$ is the initial state of the product automaton;
- $\mathcal{Q}_k^F = \mathcal{Q}_k^{F,1} \times \ldots \times \mathcal{Q}_k^{F,n_k}$ is the set of accepting states.

Note that, by designing Σ_k to be the product of individual symbols, each DFA is determined separately as to whether a transition occurs.

Definition 5.2: (Total PDFA (TPDFA)) The TPDFA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{Q}^F)$ for hierarchical sc-LTL specifications is detailed as follows:

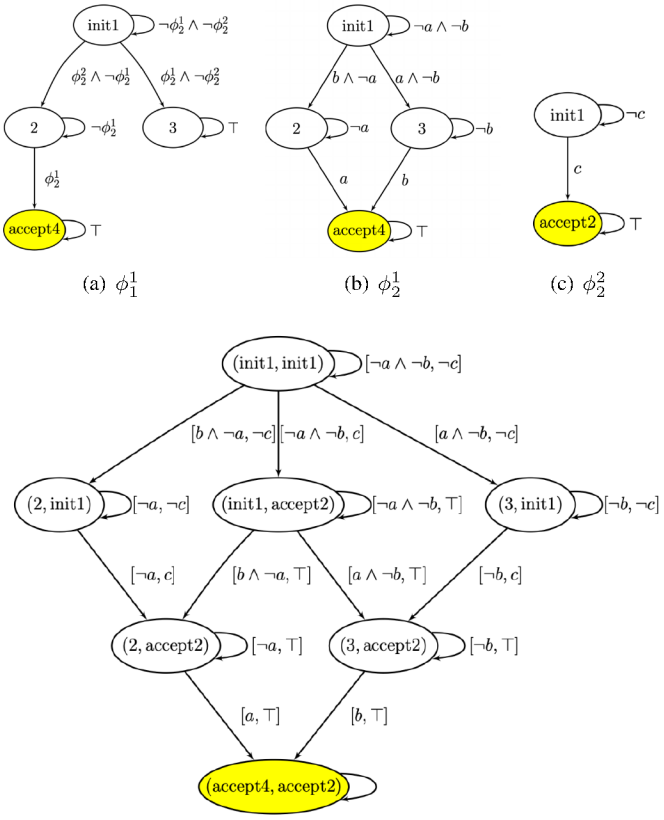
- $\mathcal{Q} = \mathcal{Q}_K \times \ldots \times \mathcal{Q}_1$ represents the set of product states across all levels;
- $\Sigma = \Sigma_K \times \ldots \times \Sigma_1$ denotes the set of symbols;
- $\delta \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$ defines the transition relation, where a transition $((q_K, \ldots, q_1), (\sigma_K, \ldots, \sigma_1), (q'_K, \ldots, q'_1)) \in \delta$ is valid if:
  - $(q_k, \sigma_k, q'_k) \in \delta_k$ for each $k \in [K]$;
  - $\sigma_k = (\sigma_k^1, \ldots, \sigma_k^{n_k})$ for all $k \in [K]$, with $\sigma_k^i = \{\phi_{k+1}^j \in \texttt{Prop}(\phi_k^i) \,|\, q_{k+1}^j \in \mathcal{Q}_{k+1}^{F,j}\}$ if ϕ_k^i is a non-leaf specification, representing the child specifications of ϕ_k^i at the immediate lower level $k+1$ that are fulfilled given q_{k+1}; otherwise, $\sigma_k^i \in \Sigma_k^i = 2^{\mathcal{AP}}$.
- $q_0 = (q_{0,K}, \ldots, q_{0,1})$ is the initial product state;
- $\mathcal{Q}^F = \{q \in \mathcal{Q} \,|\, q_1^1 \in \mathcal{Q}_1^{F,1}\}$ is the set of accepting product states where the root specification ϕ_1^1 is satisfied.

The transition relation in Def. 5.2 is constructed iteratively, starting from the bottom level upwards. For leaf specifications, transitions are determined based on atomic propositions, whereas for non-leaf specifications, transitions are defined by the truth of composite propositions from the immediately lower level.

Example 1: continued (TPDFA) The corresponding DFAs for each specification and the TPDFA for the hierarchical sc-LTL specified in (2) are depicted in Fig. 5.

Definition 5.3: (Product Automaton (PA)) The product automaton combining TPDFA and TS is denoted as $\mathcal{P} = (\mathcal{Q}_\mathcal{P}, \Sigma_\mathcal{P}, \delta_\mathcal{P}, q_{\mathcal{P},0}, \mathcal{Q}_\mathcal{P}^F)$, where:

- $\mathcal{Q}_\mathcal{P} = \mathcal{S} \times \mathcal{Q}$ represents the set of product states, combining the states of TS and TPDFA;
- $\Sigma_\mathcal{P} = \Sigma$ is the set of symbols used in the transitions;
- $\delta_\mathcal{P} \subseteq \mathcal{Q}_\mathcal{P} \times \Sigma_\mathcal{P} \times \mathcal{Q}_\mathcal{P}$ is the transition relation, as defined in Def. 5.5;



(a) ϕ_1^1 (b) ϕ_2^1 (c) ϕ_2^2

(d) TPDFA

Fig. 5: The DFAs corresponding to specifications have their accepting states highlighted in yellow. In Fig. 5(d), only the automaton states for leaf specifications are displayed since the states of non-leaf specifications can be deduced from those of the leaf specifications in a bottom-up manner. Within the diagram, the labels inside the nodes and along the edges are derived from two parts: the first component is from ϕ_2^1 and the second from ϕ_2^2. There are four paths leading from the initial state $(\text{init1}, \text{init1})$ to the accepting state $(\text{accept4}, \text{accept2})$. Notably, in all these paths, the symbol c is not the last one to be fulfilled.

- $\mathcal{L}_\mathcal{P} = \mathcal{L}$ is the labeling function that maps states to the set of satisfied propositions;
- $q_{\mathcal{P},0} = (S_0, q_0)$ is the initial product state, combining the initial state of TS with that of TPDFA;
- $\mathcal{Q}_\mathcal{P}^F = \mathcal{S} \times \mathcal{Q}^F$ is the set of accepting product states.

Before we detail the transition relation, we present how to determine whether the observations $\mathcal{L}_\mathcal{P}(S)$, produced by n robots, can enable transitions $q_\ell \to q'_\ell = (q_\ell^1, \ldots, q_\ell^m) \to (q_\ell^{1\prime}, \ldots, q_\ell^{m\prime})$ within m leaf specifications, where q_ℓ^i denotes the automaton state of the i-th leaf specification. This incorporates the task allocation into the edges of the product automaton. The key idea is to verify whether it is possible to construct $\sigma_\ell = (\sigma_\ell^1, \ldots, \sigma_\ell^m)$ from $\mathcal{L}_\mathcal{P}(S)$ in a manner that allows $(q_\ell, \sigma_\ell, q'_\ell) \in \delta_\ell$ as outlined in Def. 5.2.

Definition 5.4: (Model) Given the set of atomic propositions $\mathcal{L}_{\mathcal{P}}(S)$ generated by n robots and the transition $q_\ell \to q'_\ell$ within m leaf specifications, we deem $\mathcal{L}_{\mathcal{P}}(S)$ to be a *model* of the propositional logic formulas $\delta_\ell(q_\ell, q'_\ell) = \big(\delta^1_\ell(q^1_\ell, q^{1\prime}_\ell), \ldots, \delta^m_\ell(q^m_\ell, q^{m\prime}_\ell)\big)$ if:

1) $\mathcal{L}_{\mathcal{P}}(S)$ does not falsify any propositional logic formula $\delta^i_\ell(q^i_\ell, q^{i\prime}_\ell)$, for $i \in [m]$.
2) The leaf specifications are divided into two groups Φ_{ℓ_v} and Φ_{ℓ_ϵ} such that the total number of specifications $|\Phi_{\ell_v}| + |\Phi_{\ell_\epsilon}| = m$, with $|\Phi_{\ell_v}| = v$. Each group may contain any number of specifications, including none.
3) The set of robots is partitioned into several groups $\mathcal{R}_1, \ldots, \mathcal{R}_v$ and $\mathcal{R}_\epsilon$ such that the sum of robots in these groups equals the total number of robots, $\sum_{i=1}^{v} |\mathcal{R}_i| + |\mathcal{R}_\epsilon| = n$, and each group $\mathcal{R}_i$ and $\mathcal{R}_\epsilon$ can include zero or multiple robots.
4) For each $i \in [v]$, there is a one-to-one correspondence between a leaf specification $\phi^i_\ell \in \Phi_{\ell_v}$ and a group of robots $\mathcal{R}_{i^*}$, where the atomic propositions generated by robots in $\mathcal{R}_{i^*}$ meet the propositional logic requirement of ϕ^i_ℓ, expressed as $\mathcal{L}_{\mathcal{R}}(S) \models \delta^i_\ell(q^i_\ell, q^{i\prime}_\ell)$. Here, $\mathcal{L}_{\mathcal{R}}(S)$ represents the set of atomic propositions related to $\mathcal{R}_{i^*}$. In this case, $\sigma^i_\ell = \mathcal{L}_{\mathcal{R}}(S)$.
5) Robots in $\mathcal{R}_\epsilon$ are not assigned any leaf specifications, indicating they are idle.
6) There is no correspondence between any leaf specification in Φ_{ℓ_ϵ} and any robots, meaning that currently, no robots are engaged with the specifications in Φ_{ℓ_ϵ}, but the propositional logic for each specification in Φ_{ℓ_ϵ} can be trivially fulfilled by $\varnothing$. In this case, $\sigma^i_\ell = \varnothing$.

Condition 1) requires that no leaf specification is violated by the joint robot configuration. Condition 2) categorizes robots into those actively executing tasks and those not assigned to any task. Similarly, Condition 3) classifies tasks into those currently being performed by robots and those temporarily on hold. Condition 4) connects robots actively executing tasks and tasks currently being performed, allowing for scenarios where multiple robots collaborate on a single task, such as jointly carrying a heavy load. Conditions 5) and 6) pertain to situations involving idle robots and tasks that are not currently assigned to any robot, respectively.

Definition 5.5: (Transition Relation) A transition from one product state $q_{\mathcal{P}} = (S, q)$ to another $q'_{\mathcal{P}} = (S', q')$ occurs if the following conditions are satisfied:

- $(S, S') \in \Delta$, as specified in Def. 3.4;
- The set of propositions $\mathcal{L}_{\mathcal{P}}(S)$ is a model of the transition $\delta_\ell(q_\ell, q'_\ell)$, as outlined in Def. 5.4, indicating that the observed propositions at S satisfy the transitions at the leaf specifications.
- For each level k from 1 to K, the transition $(q_k, \sigma_k, q'_k) \in \delta_k$, as defined in Def. 5.2.

C. Prune PA

To manage the large size of the PA and facilitate the optimization process, we implement pruning techniques to reduce its complexity.

Algorithm 3: Construct essential PA

Input: PA $\mathcal{P} = (\mathcal{Q}_{\mathcal{P}}, \Sigma_{\mathcal{P}}, \delta_{\mathcal{P}}, q_{\mathcal{P},0}, \mathcal{Q}^F_{\mathcal{P}})$
Output: Essential PA $\mathcal{P}_e = (\mathcal{Q}^e_{\mathcal{P}}, \Sigma_{\mathcal{P}}, \delta^e_{\mathcal{P}}, q_{\mathcal{P},0}, \mathcal{Q}^F_{\mathcal{P}})$
1 $\mathcal{Q}^*_{\mathcal{P}} \leftarrow$ `GetEssentialStates`$(\mathcal{Q}_{\mathcal{P}}, \Sigma_{\mathcal{P}})$;
2 $\mathcal{Q}^*_{\mathcal{P}} \leftarrow \mathcal{Q}_{\mathcal{P}} \cup \{q_{\mathcal{P},0}\} \cup \mathcal{Q}^F_{\mathcal{P}}$;
3 $\mathcal{Q}^e_{\mathcal{P}} \leftarrow \mathcal{Q}^*_{\mathcal{P}}$;
4 **for** $q_{\mathcal{P}} = (S, q) \in \mathcal{Q}^*_{\mathcal{P}}$ **do**
5 **for** $q'_{\mathcal{P}} = (S', q') \in \mathcal{Q}^*_{\mathcal{P}}$ **do**
6 **if** $q \to q'$ **then**
7 $\mathcal{S}_{\text{connect}} \leftarrow$ `IRIS_RRT`(S, S') ;
8 $\mathcal{Q}^e_{\mathcal{P}} \leftarrow \mathcal{Q}^e_{\mathcal{P}} \cup \{q''_{\mathcal{P}} = (S'', q'') \in \mathcal{Q}_{\mathcal{P}} \,|\, S'' \in \mathcal{S}_{\text{connect}}, q'' = q'\}$;
9 $\delta^e_{\mathcal{P}} =$ `GetTransitions`$(\delta_{\mathcal{P}}, \mathcal{Q}^e_{\mathcal{P}})$;
10 **return** $\mathcal{P}_e = (\mathcal{Q}^e_{\mathcal{P}}, \Sigma_{\mathcal{P}}, \delta^e_{\mathcal{P}}, q_{\mathcal{P},0}, \mathcal{Q}^F_{\mathcal{P}})$;

Definition 5.6: (Essential State) Given a pair of transitions $q_{\mathcal{P}} \to q'_{\mathcal{P}}$ where $q_{\mathcal{P}} = (S, q)$ and $q'_{\mathcal{P}} = (S', q')$, the product state $q_{\mathcal{P}}$ is defined as an essential state if the automaton states differ, that is, if $q \neq q'$.

An essential state marks that there has been progress at the task level. Based on this concept, the specifics of the pruning process are detailed in Alg. 3. The procedure starts by expanding the set of essential states to include both the initial and accepting states [lines 1-2]. Using these essential states as the product space skeleton, we then establish connections between each pair of essential states where possible, utilizing intermediate states that navigate the robots through their configuration states according to a path determined by the RRT [lines 7-8].

D. Optimization Formulation and Extension for Multi-robot Handover

Upon building the product automaton $\mathcal{P}$, we define a target product state $q^{\text{target}}_{\mathcal{P}}$, which serves as the endpoint for all accepting product states within $\mathcal{Q}^F_{\mathcal{P}}$. The graph of the product automaton, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is used to structure the optimization problem by focusing exclusively on the configuration aspect of the product states, as the automaton aspect of the product states shapes the graph structure. This approach effectively transforms the problem into motion planning in the configuration space [9]. The initial configuration state is derived from the initial product state, and the target configuration state corresponds to $q^{\text{target}}_{\mathcal{P}}$. This setup is tackled through an optimization problem formulated with convex programming as shown in equation (3), aiming to establish a viable path from the initial to the target configuration states. In what follows, we extend this framework to accommodate multi-robot pick-and-place tasks, which include handover interactions, by introducing relevant constraints.

Consider a scenario with l objects. We define binary decision variables $b^{i,j}$ to indicate whether robot i holds object j, with each vertex v in $\mathcal{V}$ having an associated variable $b^{i,j}_v$.

When $b_v^{i,j} = 1$, it means that robot i is actively transporting object j. In contrast, $b_v^{i,j} = 0$ indicates that the robot i is not engaged in transporting the object j. The handover constraints governing the transfer of objects between robots are categorized into three types: incoming constraints, conflict constraints, and labeled convex set constraints.

1) Incoming Constraints: Incoming constraints depict scenarios in which robots transport objects to various (intermediate) locations. For any given vertex $v \in \mathcal{V}$, these constraints depend on whether the robot i, for each $i \in [n]$, is engaged in a handover with another robot. There are two possible cases for incoming constraints:

(a) **Robot i is not conducting the handover**. This scenario is governed by an equality constraint that ensures the continuity of possession, meaning object j remains with robot i during transit when no handover occurs:

$$b_{v'}^{i,j} = b_v^{i,j}, \quad \forall v' \in \mathcal{V}'. \tag{5}$$

Here, v' refers to vertices in the set $\mathcal{V}'$, which are predecessors leading into the vertex v.

It should be noted that if the optimal path in the product graph does not traverse certain vertices, the binary decision variables associated with those vertices, $b_v^{i,j}$, must be set to 0, and the incoming constraint (5) becomes irrelevant. To ensure that the incoming constraint is only applied along the optimal path, let the binary variable y_e indicate whether the optimal path includes the edge from v to v' [11]. We adopt the Big-M method to establish a connection between the optimal path and the binary decision variable $b_v^{i,j}$. The revised form of the incoming constraint (5), when robot i is not engaged in handover within the vertex v, is expressed as follows:

$$M(y_e - 1) \leq (b_{v'}^{i,j} - b_v^{i,j}) \leq M(1 - y_e), \quad \forall v' \in \mathcal{V}', \tag{6}$$

where M is typically a large positive integer. We set $M = 2$ to ensure a tighter formulation.

(b) **Robot i is conducting the handover**. Assume i', where $i' \in [n]$, is the robot with which the handover is being conducted. The following constraint ensures that the two robots successfully transfer object $j, \forall j \in [l]$, upon reaching the location by toggling the decision variables associated with each robot at the respective vertices:

$$b_{v'}^{i',j} = 1 - b_v^{i,j}, \quad \forall v' \in \mathcal{V}'. \tag{7}$$

To ensure that the incoming constraint (7) for robot handovers is applied strictly along the optimal path, we utilize the Big-M method to reformulate the handover constraint for robot i in the vertex v as follows:

$$M(y_e - 1) \leq (b_{v'}^{i',j} - b_v^{i,j}) \leq M(1 - y_e), \quad \forall v' \in \mathcal{V}'. \tag{8}$$

The constraint (8) effectively prevents a handover from occurring unless the path is optimal.

2) Conflict Constraints: To ensure that each robot handles no more than one object at a time, and each object is managed by only one robot simultaneously, the following constraints apply:

$$\sum_{j \in [l]} b_v^{i,j} \leq 1, \quad \forall v \in \mathcal{V}, \forall i \in [n], \tag{9a}$$

$$\sum_{i \in [n]} b_v^{i,j} \leq 1, \quad \forall v \in \mathcal{V}, \forall j \in [l]. \tag{9b}$$

3) Labeled Convex Set Constraints: For a product automaton $\mathcal{P}$ associated with labeled convex set that is labeled with a robot, denote by i, whose state aligns with the location of object j and the vertex $v \in \mathcal{V}'$:

$$b_v^{i,j} = 1. \tag{10}$$

Those handover constraints ensure an orderly and conflict-free transfer of objects among the robots. Due to the involvement of binary decision variables in the handover constraints, we solve the optimization Problem 1 using mixed-integer convex programming (MICP).

VI. Theoretical Analysis

Theorem 6.1: (Soundness) The returned path p satisfies the hierarchical sc-LTL specifications Φ.

Proof: The proof consists of two steps. In the first step, we construct a state-specification sequence τ as in Def. 3.5 from the path p. In the second step, we prove that the state-specification sequence τ satisfies the hierarchical sc-LTL Φ.

To obtain the state-specification sequence τ, the goal is to pair each robot with a leaf specification that it is undertaking, if any. Note that each point in the path p is a product state q composed of a configuration state $s = (s_1, \ldots, s_n)$ and a product DFA state $q_{\mathcal{P}} = (q_K, \ldots, q_1)$ with $q_\ell = (q_\ell^1, \ldots, q_\ell^m)$ for leaf specifications. Let $q'_{\mathcal{P}} = (q'_K, \ldots, q'_1)$ with $q'_\ell = (q_\ell^{1'}, \ldots, q_\ell^{m'})$ denote the next state of $q_{\mathcal{P}}$ in the path p. As stated in Def. 5.5, $\mathcal{L}_{\mathcal{P}}(S)$ is a model of $\delta_\ell(q_\ell, q'_\ell)$. Next, we analyze depending on conditions in Def. 5.4. Specifically, condition 1) ensures that no leaf specification is violated. Furthermore, we pair the leaf specification $\phi_\ell^i \in \Phi_{\ell_v}$ with every robot in the corresponding set $\mathcal{R}_{i^*}$, according to condition 4), otherwise, we pair robot $i \in \mathcal{R}_\epsilon$ with null specification ϵ according to condition 5).

Given the state-specification sequence τ, the labels generated by the sequence of configuration states satisfy not only the leaf specifications, but also the non-leaf specifications, according to the transition relation in Def. 5.5,. Moreover, it reaches an accepting product state where the top-most specification ϕ_1^1 is satisfied, implying that the hierarchical sc-LTL is satisfied according to semantics in [10]. ■

Definition 6.2: (Incompatible specifications) Two sc-LTL specifications are considered incompatible if there exists a path that satisfies one but inevitably violates the other, regardless of how the path is extended.

For instance, $\phi_1 = \Diamond a$ and $\phi_2 = \neg a \; \mathcal{U} \; b$ are incompatible, as a path that produces label a but not label b satisfies ϕ_1 while

violating ϕ_2. This path cannot be extended to satisfy ϕ_2. We say that a hierarchical sc-LTL specification Φ is considered compatible if it does not include any pair of leaf specifications that are mutually incompatible.

Theorem 6.3: (Completeness) Assuming the hierarchical sc-LTL Φ is compatible, up to the space decomposition and trajectory parameterization, our approach returns a path that satisfies Φ.

Proof: The CS-based product transition system (Def. 3.4) encompasses all possible behaviors of the robot system, while the total product of DFAs encompasses all solutions to fulfill the hierarchical sc-LTL (Def. 5.2). The construction of PA (Def. 5.3) is based on the concept of a model (Def. 5.4). In particular, condition 1) excludes propositional logic formulas that contradict each other. Since we consider only compatible hierarchical sc-LTL specifications, no contradictory propositional logic formulas arise. Conditions 2)-6) ensure the existence of a feasible task allocation. Consequently, the product system (Def. 5.3) includes all behaviors of the robot system that conform to the hierarchical sc-LTL. According to [9], by increasing the number of convex sets and the degree of Bézier curves to enhance the approximation, the MICP is guaranteed to find a path. ■

VII. Experiments

We evaluate our approach across a variety of multi-robot task scenarios, including planar robot motion planning, coordination among multiple robotic manipulators with handovers, quadrupedal mobile robots performing manipulator handovers, and a structured industrial environment featuring several robots and a conveyor system. All experiments were conducted using the Drake [58], and executed on a desktop computer equipped with an Intel i9 processor and 32GB of RAM. To solve Mixed-Integer Convex Programming (MICP), we use the MOSEK solver [59] via the Drake interface. Our open-source code can be accessed at https://github.com/intelligent-control-lab/Task_Motion_Planning_with_HLTL_and_GCS.git. The running times for the algorithm of all scenarios are detailed in Table I. We precompute the CS-based transition system offline and report computation times for online modules, including PA construction, PA pruning, and MICP solving. The demonstration video is available at this link.

A. Planar Motion Planning Case

The planar motion planning scenario depicted in Fig. 6 involves a robot tasked with collecting five keys to navigate through the corresponding doors. This benchmark example, noted for its complex specifications as proposed in [7], originally required transforming the sc-LTL formula into DFA. The conversion process from sc-LTL to DFA is known to exhibit double-exponential complexity [60], leading to a prolonged conversion time. By employing hierarchical sc-LTL, our approach significantly reduces this complexity. The original standard sc-LTL formula is

$$\phi = \bigwedge_{i=1}^{5} \neg \mathsf{door}_i \, \mathcal{U} \, \mathsf{key}_i \wedge \Diamond \mathsf{goal}. \tag{11}$$

The hierarchical sc-LTL specifications are represented as

$$\begin{aligned} L_1: \quad & \phi_1^1 = \bigwedge_{i=1}^{5} (\Diamond \phi_2^i) \wedge \Diamond \phi_2^6 \\ L_2: \quad & \phi_2^i = \neg \mathsf{door}_i \, \mathcal{U} \, \mathsf{key}_i, \quad i = 1, \ldots, 5. \\ & \phi_2^6 = \Diamond \mathsf{goal}. \end{aligned} \tag{12}$$

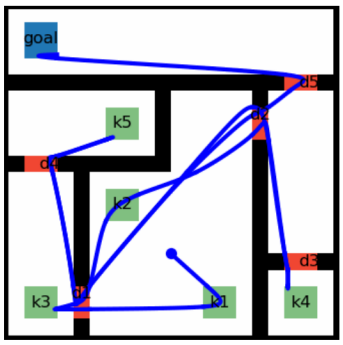


Fig. 6: The door puzzle problem, where the blue dot represents the initial robot location. [7, 61].

Hierarchical sc-LTL enhances the efficiency of representing temporal logic specifications, resulting in quicker conversion times and faster motion planning. This improved efficiency is evident in the performance comparison shown in Table II. Our approach yields a solution with a cost almost identical to that of the method described in [7], yet it achieves this result approximately 40 times faster.

B. Multi-robot Motion Planning and Handover Case

To illustrate the scalability of our method in high-dimensional spaces, we evaluated it in four systems with different tasks: a system involving two robotic manipulators (Fig. 7), a system with four robotic manipulators with multiple objects(Figs. 8-10), a system with robotic manipulators and mobile robots (Fig. 11), and a structured industrial environments with robotic manipulators and conveyor (Fig. 12).

1) Two robotic manipulators: In the initial example, two robotic manipulators are tasked to pick up an object from target 1 and place it on target 2. The hierarchical sc-LTL specifications for this scenario are articulated as follows:

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond \phi_2^1 \\ L_2: \quad & \phi_2^1 = \Diamond(\mathsf{target1} \wedge \Diamond \mathsf{target2}). \end{aligned} \tag{13}$$

Our planning algorithm uses Bézier splines to navigate a valid path, ensuring C^2 continuity for smooth trajectories. An L2 norm for the length of the joint path is also integrated to optimize for the shortest possible route. We evaluate our planner in two distinct pick-and-place scenarios. In the first scenario, depicted in Fig. 7(a), where targets 1 and 2 are equidistant from both robots, our planner decides only the

TABLE I: Algorithm Running Time.

Tasks	Figure	Construct PA (s)	Prune PA (s)	Solver Time (s)
two-robot motion planning	7(a)	0.106	0.002	5.531
two-robot handover	7(b)	0.061	0.001	0.277
four-robot handover (scenario 1)	1	12.711	0.007	11.601
four-robot handover (scenario 2)	8	7.262	0.003	7.691
four-robot handover with obstacle (scenario 3)	9	8.355	0.003	6.291
four-robot handover (scenario 4)	10	187.18	0.083	19.10
Spot-robot handover	11	0.074	0.001	0.378
two-robots with conveyor	12	5.738	0.004	1.019

Method	Time (s)				Cost
	Sc-LTL to DFA	Construct PA	Solver	Total	
[7]	401.6	94.4	2.6	498.7	774.7
Ours	**8.0**	**4.0**	**0.9**	**13.0**	774.2

TABLE II: Performance comparison.

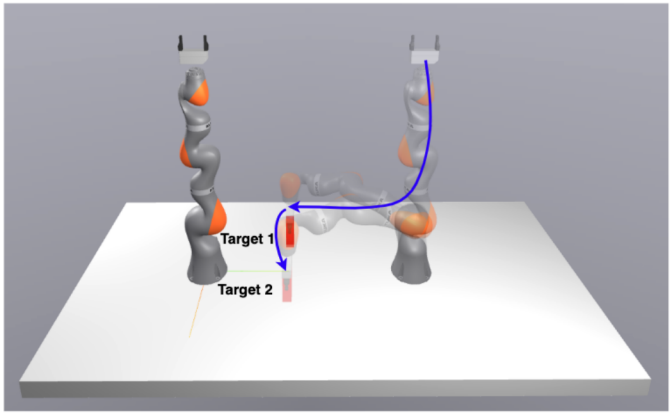
robot with the shortest path to execute the task, leaving the other robot stationary to minimize the total path length. In the second scenario, illustrated in Fig. 7(b), each robot exclusively accesses one of the targets, necessitating a handover to complete the task. Consequently, one robot picks up the object and passes it to the other, which then places it at target 2. These tests confirm that our planner adeptly identifies the most efficient strategy autonomously, eliminating the need for pre-defined orders on robot movement or handover timing.

2) Four robotic manipulators with multiple objects: The second example shows our planner in a more complex scenario and task specification that involves four KUKA iiwa robotic manipulations, with a 28 degree of freedom (DOF). We evaluated our algorithm in several distinct scenarios to assess its performance under different task specifications.

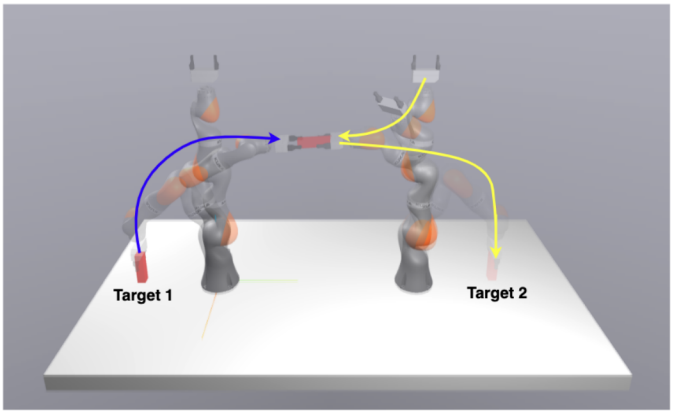
a) Scenario 1: In the first scenario, the robot was aligned in a linear arrangement, illustrated in Fig. 1. The robots are assigned to pick up three objects from specifically labeled pickup positions (targets 1, 2, and 3) and place them in corresponding placement positions (targets 4, 5, and 6). The hierarchical sc-LTL specification for this task is given by:

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond(\phi_2^1 \wedge \Diamond(\phi_2^2 \wedge \Diamond\phi_2^3)) \\ L_2: \quad & \phi_2^1 = \Diamond(\mathsf{target1} \wedge \Diamond\mathsf{target4}) \\ & \phi_2^2 = \Diamond(\mathsf{target2} \wedge \Diamond\mathsf{target5}) \\ & \phi_2^3 = \Diamond(\mathsf{target3} \wedge \Diamond\mathsf{target6}), \end{aligned} \tag{14}$$

which states that the robot should pick up movable objects from targets regions 1, 2, and 3, and place them in regions 4, 5, and 6 in order. In this example, the planner does not take into account the interactions between the gripper and objects, nor does it consider the dynamics constraints of the robot trajectory. Instead, the focus of our planner is on identifying a collision-free path in the configuration space that satisfies the hierarchical sc-LTL specifications within a 28-DOF system. This is a benchmark example proposed in [4]. The problem was solved by nonlinear trajectory optimization with smoothed discrete variables, solving multi-robot handover



(a) Both robots can reach the target positions 1 and 2.



(b) The left robot can only reach target 1, and the right robot can only reach target 2. The handover is necessary to complete the task.

Fig. 7: Example setup with two robotic manipulators with one movable object.

with one object in 36.5 seconds. However, this gradient-based method relies on local information and often faces challenges in finding feasible trajectories that deviate significantly from the initial guess, leading to suboptimal solutions with unnecessary joint movements. In contrast, the considerably more complex specification with three objects handover is solved by our proposed method in approximately 11.6 seconds. In addition, our method provides a sound, complete, and lower-cost solution without relying on an initial guess of the solver.

b) Scenario 2: In the second scenario, we designed a different layout from Scenario 1, with the robots arranged

in a narrow rectangular formation. The hierarchical sc-LTL specifications are still (14). This case is more complex than the previous one due to the narrow robot workspace, requiring the planner to ensure collision avoidance and determine the necessity of handovers between robots. The result is illustrated in Fig. 8. Note that in Fig. 8(b) and (c), while two robotic manipulators are transferring the yellow and red objects, the bottom robot simultaneously moves to transfer the blue object, effectively minimizing the overall completion time.

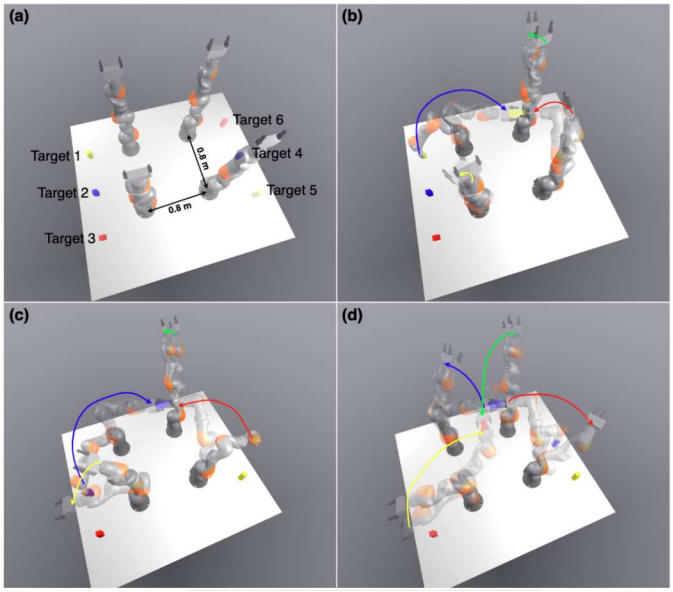


Fig. 8: Setup involving four robotic manipulators operating in a rectangular formation. The robots are tasked with handling three movable objects.

c) Scenario 3: In this scenario, the layout is the same as Scenario 2, with the robots arranged in a narrow rectangular formation. We introduce a conditional passable region in the middle, as illustrated in Fig. 9. The task involves picking up two objects from designated pickup positions (targets 1 and 2), placing them at the corresponding placement positions (targets 3 and 4), and returning the object from target 3 to target 1 while avoiding the conditional passable region. The hierarchical sc-LTL specifications are written as:

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond(\phi_2^1 \wedge \Diamond(\phi_2^2 \wedge \Diamond\phi_2^3)) \\ L_2: \quad & \phi_2^1 = \Diamond(\mathsf{target1} \wedge \Diamond\mathsf{target3}) \\ & \phi_2^2 = \Diamond(\mathsf{target2} \wedge \Diamond\mathsf{target4}) \\ & \phi_2^3 = \Diamond(\mathsf{target3} \wedge \neg\mathsf{obstacle}\ \mathcal{U}\ \mathsf{target1}). \end{aligned} \tag{15}$$

Note that in Figs. 9(b) and (c), two robots perform the handover of the yellow and blue objects within the passable regions to minimize the trajectory length. However, robots avoid the passable region when transferring the yellow object back, as shown in Fig. 9(d).

d) Scenario 4: In the scenario, as illustrated in Fig. 10, we consider more complex tasks with 3 levels of hierarchical sc-LTL. The robots are assigned two tasks involving picking up objects from one location to another. For those tasks, the robot is allowed to execute either one of them. The hierarchical sc-LTL specifications for this task are as follows:

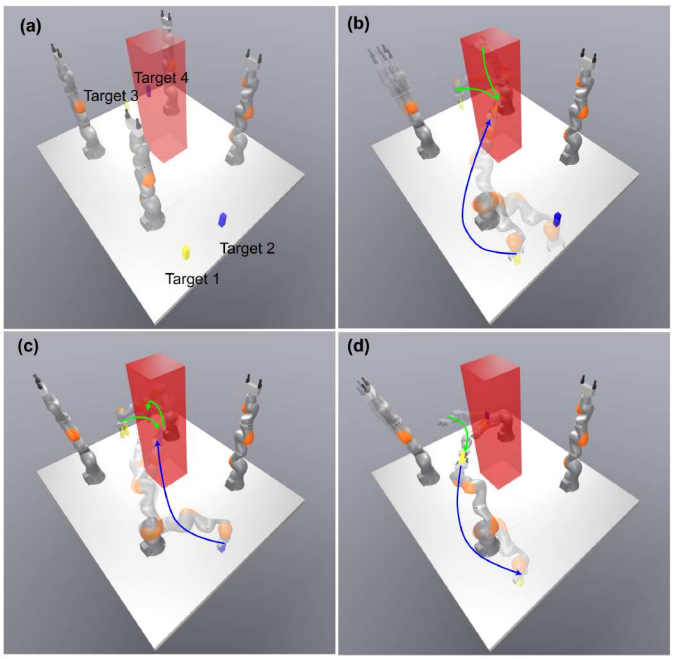


Fig. 9: Setup involving four robotic manipulators operating in a rectangular formation with a passable region (red).

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond(\phi_2^1 \vee \phi_2^2) \\ L_2: \quad & \phi_2^1 = \Diamond\phi_3^1 \wedge \Diamond\phi_3^2 \\ & \phi_2^2 = \Diamond(\phi_3^3 \wedge \Diamond\phi_3^4) \\ L_3: \quad & \phi_3^1 = \Diamond(\mathsf{target1} \wedge \Diamond\mathsf{target5}) \\ & \phi_3^2 = \Diamond(\mathsf{target2} \wedge \Diamond(\mathsf{target4} \vee \mathsf{target6})) \\ & \phi_3^3 = \Diamond(\mathsf{target1} \wedge \neg\mathsf{obstacle}\ \mathcal{U}\ \mathsf{target5}) \\ & \phi_3^4 = \Diamond(\mathsf{target2} \wedge \Diamond(\mathsf{target1} \vee \mathsf{target5})) \end{aligned} \tag{16}$$

The solution prioritizes completing task ϕ_2^2, as it has a lower trajectory length cost compared to task ϕ_2^1. In addition, in Fig. 10(c), the robot places the blue object from target 2 to target 1 instead of target 5 to minimize the trajectory length.

3) Mobile robots and robotic manipulators: In prior experiments, our focus was on stationary handovers between manipulators. In this example, we explore handovers involving mobile robots and robotic manipulators, using a Boston Dynamics Spot robot and a Kuka iiwa robot. The task is to move the objects from target 1 to target 2, as depicted in Fig. 11. The hierarchical sc-LTL specification are defined as follows:

$$\begin{aligned} L_1: \quad & \phi_1^1 = \Diamond\phi_2^1 \\ L_2: \quad & \phi_2^1 = \Diamond(\mathsf{target1} \wedge \Diamond\mathsf{target2}). \end{aligned} \tag{17}$$

Given that the iiwa robot can access target 1 but not target 2, a handover to the Spot robot is necessary. We assume that the Spot robot has a floating base with 3 DoF for movement and a 6-DoF manipulator for handling tasks. The goal of planning is to identify a collision-free trajectory that minimizes both the path length (L2 norm) and the total time. Our proposed method

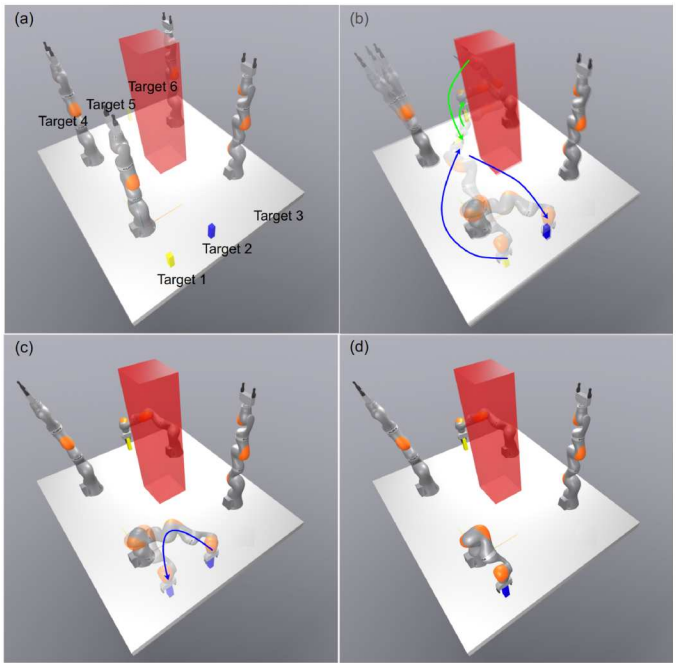


Fig. 10: Setup involving four robotic manipulators operating in a rectangular formation with a passable region (red).

generated an optimal trajectory in just 0.378 seconds. In this trajectory, the Spot robot operates at maximum speed and performs a continuous handover with the iiwa robot, ensuring a transition of the object without any stop.

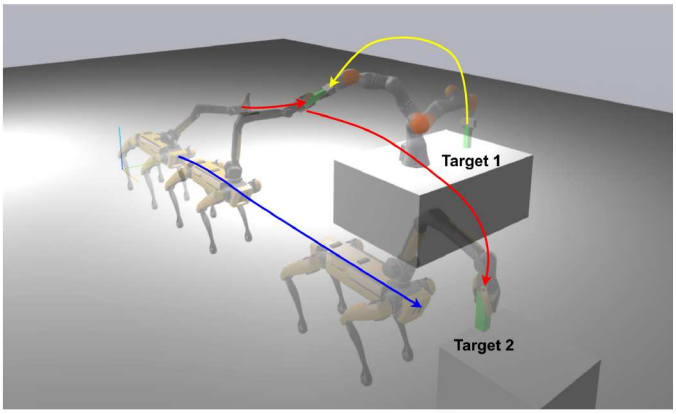


Fig. 11: Example setup featuring a mobile robot paired with manipulators tasked with handling a single movable object.

4) Structured industrial environments with robotic manipulators and conveyor: Lastly, we construct a factory setting where two robotic manipulators are integrated with a conveyor system, as illustrated in Fig. 12. This example is designed to demonstrate the capability of our planner in structured industrial environments. The task involves transporting three objects using robotic manipulators and the conveyor system, moving them from targets 1, 2, and 3 to targets 4, 5, and 6. The hierarchical LTL specifications follow the formulation in (14). Each robot has 7 DoF, and the conveyor has one DoF. The objective of our planner is to generate a collision-free trajectory in 15 DOF that minimizes the L2 norm of the path

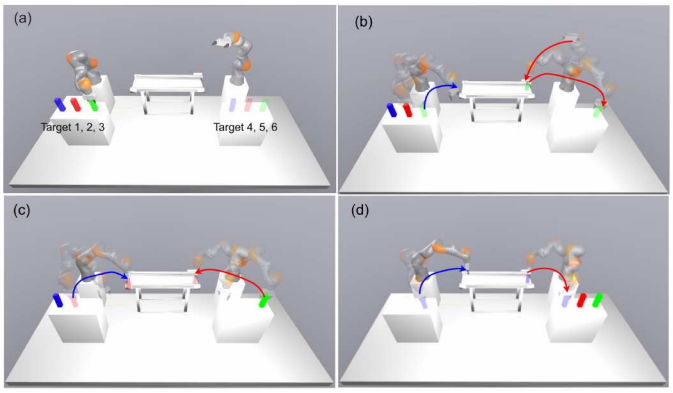


Fig. 12: Example setup featuring two robotic manipulators and one conveyor with three movable objects.

length and the cycle times across all robots and conveyor. Our planner successfully finds a time-optimal path in just 1.019 seconds.

C. Robot Experiments

We demonstrate the deployment of our planner on real robot hardware, using four WidowX 200 robots equipped with two-finger grippers. The hierarchical sc-LTL specifications follow the formulation (14). Our planner successfully generates collision-free trajectories, which are then executed via the ROS-based position controller provided by Tossen Robotics [62]. The experimental result is illustrated in Fig. 13.

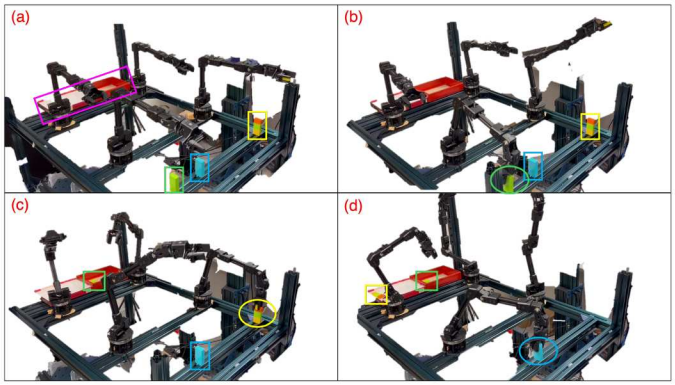


Fig. 13: A setup with four manipulators designed for pick-and-place tasks.

D. Scalability

We assess the scalability of our proposed method by analyzing solve times as the number of tasks increases, using the four-robot manipulator setup described in Scenario 2. Each task involves picking and placing a single object. The source and target locations of objects are carefully chosen to avoid the need for handovers. Tasks are evaluated in two settings: with and without handover constraints. Since handover constraints introduce binary decision variables, the problem is solved using mixed-integer convex optimization. In the absence of handover constraints, standard convex optimization is sufficient. The results in Fig. 14 indicate that solve times

increase approximately linearly with the number of tasks when handover constraints are enforced. In contrast, without handover constraints, solve times scale more efficiently, exhibiting sublinear performance.

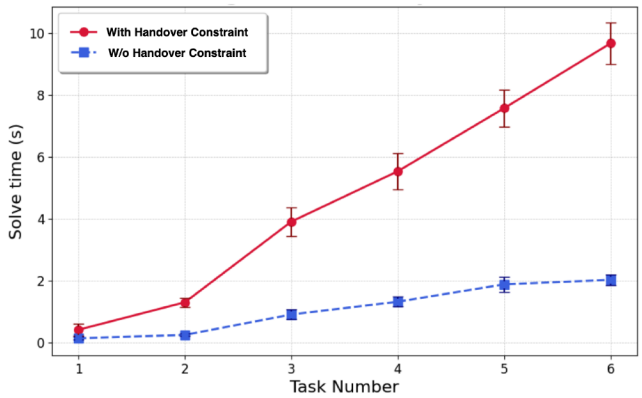


Fig. 14: Scalability results of runtimes w.r.t number of tasks.

VIII. Limitations

In this section, we briefly outline the limitations of our proposed approach. A major drawback of our method is that the complexity of MICP scales exponentially with the number of binary variables [63]. The MICP solution for the original shortest path problem in a graph of convex sets has a very tight convex relaxation. Therefore, this problem can often be solved globally optimally with convex optimization and rounding. However, to handle handover scenarios, we introduced additional integer variables, which loosen the convex relaxation. As a result, our proposed method uses MICP to solve the hierarchical temporal logic TAMP problem with handover constraints. With the increasing complexity of the hierarchical sc-LTL specification and the number of convex sets in the transition system, the MICP scales poorly. However, in practice, this does not appear to be a significant concern. After simplifying the product graph with our graph prune method, the optimal trajectory for a 28-DOF, four-robot system, involving 719 integer decision variables, can be computed in under 10 seconds.

Another limitation is that we use the IRIS-NP algorithm [56] to generate collision-free space convex sets. The algorithm provides iterative procedures for inflating convex regions of free space. However, the IRIS-NP algorithm only has probabilistically certified that the convex region is collision-free, and it may take a long time to generate the region. In addition, the generated convex region will change when the objects transfer from one robot to another. Efficiently generating certified collision-free convex regions in the configuration space and quickly adapting the region to changes in collision geometry are important areas for future research.

IX. Conclusions

We address the multi-robot task and motion planning (TAMP) problem subject to hierarchical temporal logic specifications. The key idea is to convert the optimal planning problem into a shortest path problem in a product graph, which is then solved using MICP. Unlike methods based on nonlinear trajectory optimization, sampling-based search, or learning-based approaches, our approach can handle non-convex multi-robot motion planning challenges, delivering solutions with sound and complete results within reasonable solve times.

Acknowledgments

The authors would like to thank Prof. Oliver Kroemer for providing Trossen robots, as well as to anonymous reviewers for their insightful feedback.

References

[1] Zhigen Zhao, Shuo Cheng, Yan Ding, Ziyi Zhou, Shiqi Zhang, Danfei Xu, and Ye Zhao. A survey of optimization-based task and motion planning: from classical to learning approaches. IEEE/ASME Transactions on Mechatronics, 2024.

[2] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. The International Journal of Robotics Research, 33(14):1726–1747, 2014.

[3] Marc Toussaint and Manuel Lopes. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 4044–4051. IEEE, 2017.

[4] Jimmy Envall, Roi Poranne, and Stelian Coros. Differentiable task assignment and motion planning. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2049–2056. IEEE, 2023.

[5] Rin Takano, Hiroyuki Oyama, and Masaki Yamakita. Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation. In 2021 IEEE international conference on robotics and automation (ICRA), pages 8409–8415. IEEE, 2021.

[6] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In IJCAI, pages 1930–1936, 2015.

[7] Vince Kurtz and Hai Lin. Temporal logic motion planning with convex optimization via graphs of convex sets. IEEE Transactions on Robotics, 2023.

[8] Christel Baier and Joost-Pieter Katoen. Principles of model checking. MIT press Cambridge, 2008.

[9] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. Science robotics, 8(84):eadf7843, 2023.

[10] Xusheng Luo and Changliu Liu. Simultaneous task allocation and planning for multi-robots under hierarchical temporal logic specifications. arXiv preprint arXiv:2401.04003, 2024.

[11] Tobia Marcucci, Jack Umenberger, Pablo Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. SIAM Journal on Optimization, 34(1):507–532, 2024.

[12] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. Annual review of control, robotics, and autonomous systems, 4:265–293, 2021.

[13] Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. Recent trends in task and motion planning for robotics: A survey. ACM Computing Surveys, 2023.

[14] James Motes, Read Sandström, Hannah Lee, Shawna Thomas, and Nancy M Amato. Multi-robot task and motion planning with subtask dependencies. IEEE Robotics and Automation Letters, 5(2):3338–3345, 2020.

[15] Hejia Zhang, Shao-Hung Chan, Jie Zhong, Jiaoyang Li, Sven Koenig, and Stefanos Nikolaidis. A mip-based approach for multi-robot geometric task-and-motion planning. In 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), pages 2102–2109. IEEE, 2022.

[16] James Motes, Tan Chen, Timothy Bretl, Marco Morales Aguirre, and Nancy M Amato. Hypergraph-based multi-robot task and motion planning. IEEE Transactions on Robotics, 2023.

[17] Tianyang Pan, Andrew M Wells, Rahul Shome, and Lydia E Kavraki. A general task and motion planning framework for multiple manipulators. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3168–3174. IEEE, 2021.

[18] Jimmy Envall, Roi Poranne, and Stelian Coros. Differentiable task assignment and motion planning.

[19] Yanwei Wang, Nadia Figueroa, Shen Li, Ankit Shah, and Julie Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. In Conference on Robot Learning, pages 94–105. PMLR, 2023.

[20] Adam Pacheck and Hadas Kress-Gazit. Physically feasible repair of reactive, linear temporal logic-based, high-level tasks. IEEE Transactions on Robotics, 2023.

[21] Xusheng Luo and Michael M Zavlanos. Transfer planning for temporal logic tasks. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 5306–5311. IEEE, 2019.

[22] Zeyu Feng, Hao Luan, Pranav Goyal, and Harold Soh. Ltldog: Satisfying temporally-extended symbolic constraints for safe diffusion-based planning. arXiv preprint arXiv:2405.04235, 2024.

[23] Zhaoyuan Gu, Rongming Guo, William Yates, Yipu Chen, Yuntian Zhao, and Ye 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), pages 1121–1127. IEEE, 2024.

[24] Duc M Le, Xusheng Luo, Leila J Bridgeman, Michael M Zavlanos, and Warren E Dixon. Single-agent indirect herding of multiple targets using metric temporal logic switching. In 2020 59th IEEE Conference on Decision and Control (CDC), pages 1398–1403. IEEE, 2020.

[25] Meng Guo and Dimos V Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. The International Journal of Robotics Research, 34(2):218–235, 2015.

[26] Jana Tumova and Dimos V Dimarogonas. Multi-agent planning under local LTL specifications and event-based synchronization. Automatica, 70:239–248, 2016.

[27] Pian Yu and Dimos V Dimarogonas. Distributed motion coordination for multirobot systems under ltl specifications. IEEE Transactions on Robotics, 38(2):1047–1062, 2021.

[28] Savvas G Loizou and Kostas J Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In 43rd IEEE Conference on Decision and Control (CDC), volume 1, pages 153–158, The Bahamas, December 2004.

[29] Stephen L Smith, Jana Tůmová, Calin Belta, and Daniela Rus. Optimal path planning for surveillance with temporal-logic constraints. The International Journal of Robotics Research, 30(14):1695–1708, 2011.

[30] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1525–1532. IEEE, 2014.

[31] Yiannis Kantaros and Michael M Zavlanos. Sampling-based control synthesis for multi-robot systems under global temporal specifications. In 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS), pages 3–14. IEEE, 2017.

[32] Yiannis Kantaros and Michael M Zavlanos. Distributed optimal control synthesis for multi-robot systems under global temporal tasks. In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, pages 162–173. IEEE Press, 2018.

[33] Yiannis Kantaros and Michael M Zavlanos. Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. IEEE Transactions on Automatic Control, 64(5):1916–1931, 2018.

[34] Yiannis Kantaros and Michael M Zavlanos. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. The International Journal of Robotics Research, 39(7):812–836, 2020.

[35] Yiannis Kantaros, Samarth Kalluraya, Qi Jin, and George J Pappas. Perception-based temporal logic planning in uncertain semantic maps. IEEE Transactions on Robotics, 38(4):2536–2556, 2022.

[36] Xusheng Luo, Yiannis Kantaros, and Michael M Zavlanos. An abstraction-free method for multirobot temporal logic optimal control synthesis. IEEE Transactions on Robotics, 37(5):1487–1507, 2021.

[37] Marius Kloetzer, Xu Chu Ding, and Calin Belta. Multi-robot deployment from LTL specifications with reduced communication. In 2011 50th IEEE Conference on Decision and Control and European Control Conference, pages 4867–4872. IEEE, 2011.

[38] Yasser Shoukry, Pierluigi Nuzzo, Ayca Balkan, Indranil Saha, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming. In 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pages 1132–1137. IEEE, 2017.

[39] Salar Moarref and Hadas Kress-Gazit. Decentralized control of robotic swarms from high-level temporal logic specifications. In 2017 International Symposium on Multi-robot and Multi-agent Systems (MRS), pages 17–23. IEEE, 2017.

[40] Bruno Lacerda and Pedro U Lima. Petri net based multi-robot task coordination from temporal logic specifications. Robotics and Autonomous Systems, 122:103289, 2019.

[41] Ziyang Chen and Zhen Kan. Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications. The International Journal of Robotics Research, page 02783649241278372, 2024.

[42] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. The International Journal of Robotics Research, 37(7):818–838, 2018.

[43] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. Decomposition of finite LTL specifications for efficient multi-agent planning. In Distributed Autonomous Robotic Systems, pages 253–267. Springer, 2018.

[44] Fatma Faruq, David Parker, Bruno Laccrda, and Nick Hawes. Simultaneous task allocation and planning under uncertainty. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3559–3564. IEEE, 2018.

[45] Thomas Robinson, Guoxin Su, and Minjie Zhang. Multiagent task allocation and planning with multi-objective requirements. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pages 1628–1630, 2021.

[46] Xusheng Luo, Shaojun Xu, Ruixuan Liu, and Changliu Liu. Decomposition-based hierarchical task allocation and planning for multi-robots under hierarchical temporal logic specifications. IEEE Robotics and Automation Letters, 2024.

[47] Alberto Camacho, Eleni Triantafillou, Christian J Muise, Jorge A Baier, and Sheila A McIlraith. Non-deterministic planning with temporally extended goals: Ltl over finite and infinite traces. In AAAI, pages 3716–3724, 2017.

[48] Alberto Camacho, R Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), pages 6065–6073, 2019.

[49] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. Hierarchical LTL-task mdps for multi-agent coordination through auctioning and learning. The International Journal of Robotics Research, 2019.

[50] Xusheng Luo and Michael M Zavlanos. Temporal logic task allocation in heterogeneous multirobot systems. IEEE Transactions on Robotics, 38(6):3602–3621, 2022.

[51] Zesen Liu, Meng Guo, and Zhongkui Li. Time minimization and online synchronization for multi-agent systems under collaborative temporal logic tasks. Automatica, 159:111377, 2024.

[52] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Multirobot coordination with counting temporal logics. IEEE Transactions on Robotics, 2019.

[53] Armin Biere, Keijo Heljanko, Tommi Junttila, Timo Latvala, and Viktor Schuppan. Linear encodings of bounded LTL model checking. Logical Methods in Computer Science, 2(5:5):1–64, 2006.

[54] Kevin Leahy, Austin Jones, and Cristian-Ioan Vasile. Fast decomposition of temporal logic specifications for heterogeneous teams. IEEE Robotics and Automation Letters, 7(2):2297–2304, 2022.

[55] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. Formal Methods in System Design, 19(3):291–314, 2001.

[56] Mark Petersen and Russ Tedrake. Growing convex collision-free regions in configuration space using nonlinear programming. arXiv preprint arXiv:2303.14737, 2023.

[57] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. Algorithmic and computational robotics, pages 303–307, 2001.

[58] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[59] The mosek optimization toolbox. version 9.0, mosek aps, copenhagen, denmark., 2022. URL https://docs.mosek.com/latest/toolbox/intro_info.html#.

[60] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. Formal methods for discrete-time dynamical systems, volume 89. Springer, 2017.

[61] William Vega-Brown and Nicholas Roy. Admissible abstractions for near-optimal task and motion planning. arXiv preprint arXiv:1806.00805, 2018.

[62] https://github.com/Interbotix/interbotix_ros_manipulators.

[63] Vince Kurtz and Hai Lin. A more scalable mixed-integer encoding for metric temporal logic. IEEE Control Systems Letters, 6:1718–1723, 2022.