Partial Satisfaction of Signal Temporal Logic Specifications for Coordination of Multi-Robot Systems

Gustavo A. Cardona[0000-0002-4257-9415] and Cristian-Ioan Vasile[0000-0002-1132-1462]

Department of Mechanical Engineering and Mechanics at Lehigh University, PA 18015, USA {gcardona,cvasile}@lehigh.edu

Abstract. This paper introduces a partial satisfaction (PS) notion for Signal Temporal Logic (STL), finding solutions for specifications that might contain unfeasible or conflicting subformulae. We formulate the planning problem for teams of agents with robust PS of STL missions as a bi-level optimization problem. The goal is to maximize the number of subformulae satisfied with preference to larger ones that have lower depth in the syntax tree of the overall specification. The second objective is to maximize the smallest STL robustness of the feasible subformulae. First, we propose three Mixed Integer Linear Programming (MILP) methods to solve the inner level of the optimization problem, two exact and a relaxation. Then, the MILP solutions are used to find approximate solutions to the outer level optimization using a linear program. Finally, we show the performance of our methods in two multi-robot case studies: motion planning in continuous spaces, and routing for heterogeneous teams over finite graph abstractions.

Keywords: Formal Methods · Signal Temporal Logic · Partial Satisfaction · Multi-robot Systems.

1 Introduction

In recent years, multi-robot systems have been widely studied due to their resiliency and robustness. Their application domains span from search-and-rescue missions and exploration of bio-hazard zones to cargo delivery and planetary exploration [3]. Solving these types of missions requires optimal task allocation over teams of robots satisfying a set of task specifications [24, 5]. Additionally, suitability, robustness, and scalability are an extension of the problem that is frequently tackled in the literature [10, 11]. However, in case of conflicting specifications or lack of robots, the planning problem becomes infeasible, whereas, in practice, at least partial execution of the task may be desired.

Our work focuses on partial satisfaction (PS) of a mission over a team of robots with conflicting or competing specifications. For example, let us consider having four robots with a camera in an environment and a task specification requiring three robots with a camera at location A and, at the same time, two

robots of the same type at location B. Although both tasks cannot be satisfied due to a lack of robots, our framework tackles this by fully satisfying one of the tasks and partially satisfying the other. Furthermore, for efficient handling and scheduling of complex tasks, we use Signal Temporal Logic (STL) specifications to enrich the specification with logical and temporal operators that allow us to describe not only where but also when and for how long robots are needed to satisfy tasks [28, 27].

There exist some methods in the literature that tackle these temporal specifications while considering PS, such as automata-based approaches [2, 1] and robustness as Mixed Integer Linear Programming (MILP) formulations [26]. In [20, 21], authors compute policies that minimize the distance to satisfaction as given by the paths in the specification automaton. On the other hand, [18] proposes a general relaxation framework for automata-based planning for various relaxation instances by using a weighted finite-state edit system. Even though these approaches take into account the partially feasible nature of specifications, they are computationally expensive as they involve computing product automata that scale poorly.

In a different direction, MILP encoding scales better, making it suitable for multi-robot setups. In [25], a limited violation in the specifications is allowed by adding more binary variables in each Boolean operation, increasing the computational cost dramatically. Authors in [4] consider PS due to time uncertainties, penalizing the model when a task is not satisfied. Modeling PS with a bilevel optimization formulation, where the lower level decides whether or not a task can be satisfied, and the higher level solves the overlapping conflicts. Although minimum violation, robustness, and completion uncertainty have been solved, the PS concept has not been tackled. Our work aims at addressing PS by maximizing the subformulae in a specification that can be satisfied. Also, our approach has a percentage metric of specification satisfaction and indicates subformulae that were partially or not satisfied.

This paper presents a robust PS problem for STL specifications with possible contradicting, competing, or infeasible subformulae. We formulate the PS planning problem as a bi-level optimization formulation where the inner level identifies satisfied subformulae. We prioritize lower-depth nodes in an Abstract Syntax Tree (AST) of the specification. We propose three MILP methods that use a novel encoding for STL based on *satisfaction fractions*, number of satisfied subformulae weighted according to depth. Our MILP encodings do not lose the qualitative semantics of STL. We use a Linear Program (LP) to maximize the robustness of all subformulae satisfied by a solution of the MILPs. The LP approximates the outer level of the robust PS problem. We show the performance of the methods in two case studies involving multi-robot teams. We consider motion planning in continuous spaces and routing for heterogeneous teams over finite graphs representing abstractions of the environment.

The main contributions of this paper are threefold. First, we introduce a PS of STL specifications that satisfies as much as possible even in the presence of conflicting or unfeasible subformulae. Second, we propose three MILP encodings for PS, describing each method's advantages and runtime performance, and an LP for approximating robust PS. We also compare PS encodings performance

with CaTL specifications [16], showing that our encoding not only can work under unfeasible specifications but also improves the runtime performance under feasible specifications. Finally, we show the versatility and performance of PS methods in two case studies, continuous state planning for robots navigating in planar environments and routing over finite graph abstractions for teams of robots with varying capabilities tasked with rich temporal logic tasks [17].

2 Preliminaries and Notation

Let $\mathbb{Z}$, $\mathbb{R}$, denote the sets of integer and real numbers. The set of integers greater than a is $\mathbb{Z}_{\geq a}$. For a set S, 2^S and $|S|$ represent its power set and cardinality. For $S \subseteq \mathbb{R}$ and $\alpha \in \mathbb{R}$, we have $\alpha + S = \{\alpha + x \mid x \in S\}$. The integer interval (range) from a to b is $[a \,..\, b]$. Let $x \in \mathbb{R}^d$ be a d-dimensional vector. The i-th component of x is given by x_i, $i \in [1 \,..\, d]$.

2.1 Signal Temporal Logic

Let $s : \mathbb{Z}_{\geq 0} \to \mathbb{M}$ be a discrete-time signal with values in the compact space $\mathbb{M} \subseteq \mathbb{R}^N$. Signal Temporal Logic (STL), introduced in [23], is a specification language expressing real-time properties. The syntax of STL over linear predicates is given by

$$\phi := \top \mid s_i \geq \mu \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \Diamond_I \phi \mid \Box_I \phi \mid \phi_1 \mathcal{U}_I \phi_2, \tag{1}$$

where ϕ, ϕ_1, and ϕ_2 are STL formulae, $\top$ is the logical *True* value, $s_i \geq \mu$ is a linear predicate with threshold value $\mu \in \mathbb{R}$ over the i-th component of signal s. The logical *False* value is $\bot = \neg\top$. Predicates $s \sim \mu$, $\sim \in \{>, \leq, <\}$, follow via negation and sign change. Boolean operators *negation* $\neg$, *disjunction* $\vee$ and *conjunction* $\wedge$. Timed temporal operators *eventually* $\Diamond_I$, *always* $\Box_I$ and *until* $\mathcal{U}_I$ with $I = [k_1 \,..\, k_2]$ a discrete-time interval, $k_2 \geq k_1 \geq 0$, define as in [23].

The (qualitative) semantics of STL formulae over signals s at time k is recursively defined in [23] as

$$\begin{aligned}
(s,k) \vDash (s_i \geq \mu) &\equiv s_i(k) \geq \mu, \\
(s,k) \vDash \neg\phi &\equiv (s,k) \nvDash \phi, \\
(s,k) \vDash \phi_1 \wedge \phi_2 &\equiv \big((s,k) \vDash \phi_1\big) \wedge \big((s,k) \vDash \phi_2\big), \\
(s,k) \vDash \phi_1 \vee \phi_2 &\equiv \big((s,k) \vDash \phi_1\big) \vee \big((s,k) \vDash \phi_2\big), \\
(s,k) \vDash \Diamond_I \phi &\equiv \exists k' \in I \text{ s.t. } (s,k') \vDash \phi, \\
(s,k) \vDash \Box_I \phi &\equiv \forall k' \in I \text{ s.t. } (s,k') \vDash \phi, \\
(s,k) \vDash \phi_1 \mathcal{U}_I \phi_2 &\equiv \exists k' \in k + I \text{ s.t. } (s,k') \vDash \phi_2 \wedge \forall k'' \in [k \,..\, k'] \; (s,k'') \vDash \phi_1,
\end{aligned} \tag{2}$$

where $\vDash$ and $\nvDash$ denote satisfaction and violation, respectively. A signal s satisfying ϕ, denoted as $s \vDash \phi$, is true if $(s, 0) \vDash \phi$. In addition to Boolean semantics, STL admits quantitative semantics, called *robustness*, that indicates how much

a signal satisfies or violates a specification [12, 9]. The robustness score $\rho(s, \phi, k)$ is recursively defined as

$$\begin{aligned}
\rho(s, \top, k) &= \rho_\top, \\
\rho(s, s_i \geq \mu, k) &= s_i(k) - \mu, \\
\rho(s, \neg\phi, k) &= -\rho(s, \phi, k), \\
\rho(s, \phi_1 \wedge \phi_2, k) &= \min(\rho(s, \phi_1, k), \rho(s, \phi_2, k)), \\
\rho(s, \phi_1 \vee \phi_2, k) &= \max(\rho(s, \phi_1, k), \rho(s, \phi_2, k)), \\
\rho(s, \Box_I \phi, k) &= \min_{k' \in k+I} \rho(s, \phi, k'), \\
\rho(s, \Diamond_I \phi, k) &= \max_{k' \in k+I} \rho(s, \phi, k'), \\
\rho(s, \phi_1 \mathcal{U}_I \phi_2, k) &= \max_{k' \in k+I} \Big\{ \min\{\rho(s, \phi_2, k'), \min_{k'' \in [k..k']} \rho(s, \phi_1, k'')\} \Big\},
\end{aligned} \tag{3}$$

where $\rho_\top = \sup_{s,\mu}\{s_i - \mu\}$ is the maximum robustness.

Theorem 1 (Soundness [9]). *Let s be a signal and ϕ an STL formula. It holds $\rho(s, \phi, k) > 0 \Rightarrow (s, k) \vDash \phi$ for satisfaction and $\rho(s, \phi, k) < 0 \Rightarrow (s, k) \nvDash \phi$ for violation.*

The time horizon of an STL formula [8] is defined as

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = s \geq \mu, \\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1, \\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \wedge \phi_2, \\ b + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \mathcal{U}_I \phi_2. \end{cases}$$

An STL formula is said to be in *positive normal form* (PNF) if it satisfies two conditions. First, all its predicates are of the $s_i \geq \mu$ form. Second, it does not contain the negation operator. Any STL formula can be put in PNF form [26].

3 Problem Formulation

In this work, we consider planning for teams of robots tasked with performing missions specified as STL formulae. We focus on the case where not all of the mission specification can be satisfied due to a lack of agents or competing subformulae. We define the PS problem that requires the computation of motion plans for all robots such that as much of the formula is satisfied with larger subformulae preferred over smaller ones. To this end, we introduce a partial order over subformulae that captures the preference structure. Although we focus on multi-robot problems in this paper, the PS problem and proposed approach can be used with any robot system with linear dynamics or mixed-logical linear dynamics, e.g., piecewise-affine systems, max-min-plus scaling systems [14, 26].

3.1 Team Dynamics

We consider two multi-robot planning problems. The first is planning for heterogeneous teams of robots with various capabilities over graphs and finite abstractions of the environment in which they are deployed [16]. The second is joint planning for homogeneous teams in the continuous state spaces of the robots. We assume that the robots' motion is governed by linear dynamics enforced by low-level controllers, e.g., using differential flatness [6, 31] and feedback linearization [29, 19].

The robot team dynamics is captured in both the discrete and continuous state space cases using linear difference equations

$$s(k+1) = As(k) + B\mathbf{u}(k) + D, \tag{4}$$

where $s(k) \in S$ is the team state at time $k \in \mathbb{Z}_{\geq 0}$, S is the state space of the team, and A, B, and D are the state transition, input, and drift matrices of appropriate sizes. The difference between the two cases is in the state space definition. For the heterogeneous team case over finite abstraction graphs, the team states represent the number of agents of each class at each location (node of the graph). For the homogeneous team case over continuous state spaces, the team states are composed of the states of all agents. See Sec. 5 for details and formal definitions.

3.2 Partial Satisfaction Control Synthesis Problem

This paper defines missions as STL formulae with predicates over team states that capture the temporal and logical sequencing of tasks and their timing constraints. Informally, we desire PS to: (a) prefer subformulae with lower-depth [1] (see Fig. 1), and (b) consider subformulae satisfaction at only required time points modulated by the time intevals of temporal operators $\Diamond$, $\Box$, and $\mathcal{U}$, see example 1. We capture these requirements formally using a partial order [7].

Partial order over an AST Any STL formula can be represented using an Abstract Syntax Tree (AST) in which intermediate nodes correspond to logical and temporal operators, and leaves to predicates [15]. The depth of a formula φ with respect to a formula ϕ is the path distance between the root of ϕ's AST and the node associated with φ. If φ is not a subformula of ϕ, the depth is by convention ∞. We denote the depth by $depth_\phi(\varphi)$. We use $\varphi \sqsubset \phi$ to denote that φ is a proper subformula of ϕ, and $\varphi \sqsubseteq \phi$ when they can also be equal. Let s be a state trajectory, and ϕ an STL mission specification. The set of subformulae of ϕ and depth $d \in \mathbb{Z}_{\geq 0}$ satisfied by s is $\Phi^d_\phi(s) = \{(\varphi, k) \mid \text{depth}_\phi(\varphi) = d, (s, k) \vDash \varphi\}$. We say that state trajectory s is less than s' with respect to ϕ, denoted $s \prec_\phi s'$, if there is $d \in \mathbb{Z}_{\geq 0}$ such that (a) $|\Phi^d_\phi(s)| < |\Phi^d_\phi(s')|$, and (b) $|\Phi^{d'}_\phi(s)| = |\Phi^{d'}_\phi(s')|$ for

[1] Lower-depth node satisfaction implies a higher percentage of specification satisfaction since every all node underneath are guaranteed to be satisfied when their parent nodes are satisfied.

all $d' \in [0 \,..\, d'-1]$. Equality, $s =_\phi s'$, holds iff $|\Phi^d_\phi(s)| = |\Phi^d_\phi(s')|$ for all $d \in \mathbb{Z}_{\geq 0}$. The trajectory s generated by some robotic system (linear or otherwise) is said to satisfy ϕ *as much as possible* if s is maximal under the partial order $\prec_\phi$.

Partial Satisfaction Robustness we define the PS robustness as follows

$$\varrho(s,\phi) = \min_{(\varphi,k)\in F_\phi(s)} \rho(s,\varphi,k), \tag{5}$$

where $F_\phi(s) = \{(\varphi,k) \mid \nexists(\varphi',k') \text{ s.t. } \varphi \sqsubset \varphi', (s,k) \vDash \varphi, (s,k') \vDash \varphi'\}$ is the set of lowest-depth subformulae satisfied by s.

Problem 1. Given a multi-robot system with linear team dynamics (4), and an STL specification ϕ, find team input signal $\mathbf{u}$ such that the generated state trajectory s satisfies ϕ as much as possible and maximizes the PS robustness (5). Formally, we have the bi-level optimization problem

$$\begin{aligned} \max_{\mathbf{u}} \quad & \varrho(s,\phi) \\ \text{s.t.} \quad & \mathbf{u} \text{ induces } s \\ & s \in \max^{\phi}_{\mathbf{u}'} \{s'\} \text{ s.t. } \mathbf{u}' \text{ induces } s' \end{aligned}, \tag{6}$$

where $\max^\phi$ denotes maximization with respect partial order $\prec_\phi$ defined above and meaning it finds the maximal elements in the induce lattice [7].

Problem 1, takes a specification formula ϕ, and finds a team trajectory s that satisfies the maximum number subformulae of lowest depth, and has the largest minimum robustness among them. The inner optimization guarantees that the number of low-depth formulae is maximum. Since $\mathbf{u}'$ induces multiple trajectories s' in which $\max^\phi$ looks for the one that satisfies as much as possible according to the partial order $\prec_\phi$. While the outer optimization accounts for their robustness.

Let us remark on the differences between PS and the canonical robustness in (3). The latter represents the margin of satisfaction or violation of an STL formula in the signal value space. In other words, the largest deviation to a signal changes the satisfaction with respect to the specification formula. In case of violation, it does not express which subformulae are satisfied or violated. Thus, it cannot be used to enforce satisfying as many subformulae as possible. In contrast, PS can guarantee that the specification will be satisfied as much as possible with preference to lower-depth subformulae. PS also captures the fraction of satisfaction for unsatisfied subformulae.

Example 1. Consider the following STL specification

$$\begin{aligned} \phi_{ex} &= \Box_I \left((s_1 \geq 0) \wedge (s_2 \geq 0) \right) \wedge \phi'_{ex} \\ \phi'_{ex} &= \Diamond_{I'} \left((s_3 \leq 0) \vee (s_1 \geq 2) \vee \left((s_1 \geq 3) \wedge (s_2 > 1) \right) \right), \end{aligned} \tag{7}$$

where $I = [0,1]$ and $I' = [0,2]$. The AST of ϕ_{ex} and depth of each of its subformulae are shown in Fig. 1. Consider a signal s such that $s(0) = -\mathbf{1}_3$, $s(1) = \mathbf{1}_3$, and

$s(2) = 2 \cdot \mathbf{1}_3$, where $\mathbf{1}_n$ is the vector of all ones of dimension $n \in \mathbb{Z}_{\geq 1}$. The canonical robustness of s is $\rho(s, \phi_{ex}, 0) = -1$ due to the violation of predicates $\mu_1 = s_1 \geq 0$ and $\mu_2 = s_2 \geq 0$ at time $k = 0$ in the left subformula containing the always operator. However, the predicates are satisfied at time $k = 1$ within the interval I of the always operator. Moreover, the right subformula ϕ'_{ex} corresponding to the eventually operator is satisfied (as well as all of its subformulae). PS captures their satisfaction with $\varrho(s, \phi_{ex}) = \min\{\rho(s, \phi'_{ex}, 0), \rho(s, \mu_1 \wedge \mu_2, 1)\} = \min\{1, 1\} = 1$, where $F_\phi(s) = \{(\phi'_{ex}, 0), (\mu_1 \wedge \mu_2, 1)\}$.

Remark 1. Note that PS depends explicitly on the AST of a formula[2] and is not associative by design, meaning that the order of operators matters (e.g., $(\phi_1 \wedge \phi_2) \wedge \phi_3$ gives priority to ϕ_3, whereas in $\phi_1 \wedge (\phi_2 \wedge \phi_3)$ priority is in ϕ_1). This work considers that AST is always given. Thus, an operator has the freedom to specify the priorities and importance of subformulae using parenthesis that determines the ASTs. Another way to indicate the preferences over subformulae could be by adding intermediate virtual (no operation) nodes to increase the size of subformulae and prefer to satisfy those subformulae over others at the same depth but on a different branch.

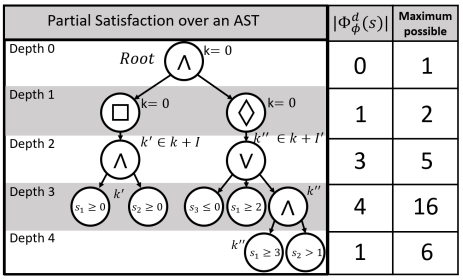

Fig. 1: Abstract syntax tree of formula ϕ_{ex} in (7). The depth of each subformula associated with the AST's nodes are shown. Moreover, for each subformula the time points that are involved in the satisfaction of ϕ_{ex} are given. The numbers of satisfied subformulae $|\Phi^d_\phi(s)|$ of each depth $d \in [0 \,..\, 4]$ are shown in the middle column. The maximum numbers of subformulae-time pairs for each depth that are required to satisfy ϕ are in the right column.

4 Partial Satisfaction Encoding

In this section, we propose to solve Problem 1 in two steps that decouple the bi-level optimization problem. First, we propose two Mixed Integer Linear Programming (MILP) formulations to find the maximum number of subformulae

[2] A formula can have many equivalent ASTs [15] determined by the parsing methods used.

of lowest depth that can be satisfied. The MILP corresponds to the inner optimization in (6). Furthermore, we consider a relaxation of the MILP formulations that reduces the number of required binary variables, and has better runtime performance. Second, we introduce a Linear Program (LP) that approximates the solution to the outer level of (6) using the solution of the inner level. The two-step approach trades off optimality with runtime performance.

4.1 MILP Encoding of Satisfaction Fractions

The foundation of our encoding of STL formulae is based on the fraction of subformulae-time pairs that signals satisfy. Instead of encoding margins that are propagated towards a formula's root to compute robustness [26, 25], we keep track of how many subformulae are satisfied weighted according to their depth.

Let ϕ be an STL formula. We represent by $\xi_{\varphi,k} \in [0,1]$ the satisfaction fraction of subformula φ of ϕ at time $k \in [0 \,..\, \|\phi\|]$. We define the following MILP to capture the satisfaction fractions generated by a signal s generated by the team dynamics (4). The recursive definition is

$$
\begin{aligned}
&\xi_{\varphi,k} = \begin{cases} s_i(k) - \mu + M(1-\xi_{\mu,k}) \geq 0 \\ s_i(k) - \mu - M\xi_{\mu,k} \leq 0 \end{cases}, && \varphi = s_i \geq \mu, \\
&m \cdot \xi_{\varphi,k} = \sum_{\ell=1}^{m} \xi_{\varphi_\ell,k}, && \varphi = \bigwedge_{\ell=1}^{m} \varphi_\ell, \\
&\xi_{\varphi,k} = \max_{\ell=1:m} \{\xi_{\varphi_\ell,k}\}, && \varphi = \bigvee_{\ell=1}^{m} \varphi_\ell, \\
&|I| \cdot \xi_{\varphi,k} = \sum_{k' \in k+I} \xi_{\varphi',k'}, && \varphi = \Box_I \varphi', \\
&\xi_{\varphi,k} = \max_{k' \in k+I} \{\xi_{\varphi',k'}\}, && \varphi = \Diamond_I \varphi', \\
&\xi_{\varphi,k} = \max_{k' \in k+I} \{\xi_{\varphi_2,k'} + \sum_{k'' \in [k..k']} \xi_{\varphi_1,k''}\}, && \varphi = \varphi_1 \mathcal{U}_I \varphi_2,
\end{aligned}
\tag{8}
$$

where $M > \rho_\top$, $\xi_{\mu,k} \in \{0,1\}$ (since satisfaction of predicates needs to be enforced) for all predicates μ and times $k \in [0 \,..\, \|\phi\|]$. We have $\xi_{\mu,k} = 1$ when $s_{i,k} \geq \mu$ is true, and $\xi_{\mu,k} = 0$ when it is false. For all other logical and temporal operators $\xi_{\varphi,k}, \in [0,1]$ at all times $k \in [0 \,..\, \|\phi\|]$. A subformula φ (associated with a temporal or logical operator node) is fully satisfied when $\xi_{\varphi,k} = 1$, and it is fully violated when $\xi_{\varphi,k} = 0$. In all other cases, φ is partially satisfied at time k.

Next, we capture the satisfaction of subformulae time pairs (φ, k) in $F_\phi(s)$ using the binary variables $\eta_{\varphi,k} \in \{0,1\}$. We enforce satisfaction and lowest depth with constraints

$$\eta_{\varphi,k} \leq \xi_{\varphi,k}, \tag{9}$$

$$\eta_{\varphi,k} \leq 1 - \eta_{\varphi',k'}, \forall \varphi \sqsubset \varphi' \sqsubseteq \phi, k' \in \mathcal{K}', \tag{10}$$

where $\mathcal{K}' = \{k' \in [0 \,..\, \|\phi\|] \mid (s,k') \vDash \varphi' \Rightarrow (s,k) \vDash \varphi, \forall s\}$ is the finite set of all times where φ' supersedes φ at k (see Sec.3.2) which is constructed in polynomial

time since the time horizon $\|\phi\|$ and the number of subformulae in ϕ are finite. Note that $\eta_{\varphi,k}$ can take value one only when φ is fully satisfy i.e., $\xi_{\varphi,k} = 1$, due to constraint (9). All ancestors must not be fully satisfied, i.e., $\eta_{\varphi',k'} = 0$, for φ to be lowest depth as captured in (10).

4.2 Objective Functions for Partial Satisfaction

In this section, we formulate three MILP problems that solve the inner level of (6). The first two are exact solutions, while the third is a relaxation.

Let $\gamma_{\phi,d} = \sum_{(\varphi,k)\in\Upsilon^d(\phi)} \eta_{\varphi,k}$, for all $d \in [0 \,..\, d_{\max}]$, where $d_{\max}$ is the maximum depth of ϕ, and $\Upsilon^d_\phi = \{(\varphi,k) \mid \text{depth}_\phi(\varphi) = d, (s,0) \vDash \phi \Rightarrow (s,k) \vDash \varphi, \forall s\}$ is the set of all pairs of subformulae φ of depth d and time points k required for satisfaction of formula ϕ.

Hierarchical Optimization (HO) In this approach, we define the multi-objective function $R_{HO} = (\gamma_{\phi,0}, \gamma_{\phi,1}, \ldots, \gamma_{\phi,d_{\max}})$, and require optimization of the objectives in order [30], also known as lexicographical optimization [13]. This leads to the hierarchical MILP problem

$$\max_{s,u,\xi,\eta} R_{HO} \text{ (optimized in order) s.t. } (4),(8),(9),(10). \tag{11}$$

Lowest Depth First (LDF) We convert the multi-objective function R_{HO} into a scalar function R_{LDF} using a method similar to the big-M trick. The objective is

$$R_{LDF} = \sum_{(\varphi,k)} \eta_{\varphi,k} P^{-\text{depth}_\phi(\varphi)} = \sum_{d=0}^{d_{\max}} \gamma_{\phi,d} P^{-d},$$

that leads to the MILP problem

$$\max_{s,u,\xi,\eta} R_{LDF} \text{ s.t. } (4),(8),(9),(10), \tag{12}$$

where P is a large constant, greater than $\|\phi\| \cdot |\phi|$, and $|\phi|$ is the length of ϕ, i.e., the number of Boolean and temporal operators and predicates.

Weighted Largest Number (WLN) In this approach, we relax the strict requirement of finding lowest depth formulae, and instead maximize the satisfaction fraction of ϕ. Taking $R_{WLN} = \xi_{\phi,0}$ is a sensible choice, because subformulae are still weighted according to their depth due to the recursive constraints (8). Consequently, we obtain the MILP

$$\max_{s,u,\xi} R_{WLN} \text{ s.t. } (4),(8), \tag{13}$$

that does not require the additional binary variables $\eta_{\varphi,k}$ and associated constraints.

Remark 2. It is important to note that HO (11) approach can be solved directly using Gurobi [13]. However, any MILP solver can be used via iterative methods [30]. Formulation (12) avoids defining multiple objectives, but can lead to numerical issues and slower convergence for large depth formulae due to the exponentially decreasing weights. Finally, WLN (13) leads to better runtime performance at the expense of minimal depth of satisfied subformulae.

4.3 Partial Satisfaction Robustness LP

In this section, we propose to approximate the robust solution of Problem 1 using an LP based on solutions to one of the MILPs from Sec. 4.2. Let $\{\xi_{\varphi,k}\}_{\varphi \sqsubseteq \phi, k \in [0..\|\phi\|}$ be the set of decision variables for satisfaction fractions obtained from solving (11), (12), or (13). The following LP,

$$\max_{s,u} \rho \text{ s.t. } (4),\ \rho \leq s_i(k) - \mu, \forall \mu \text{ with } \xi_{\mu,k} = 1 \tag{14}$$

computes the signal s and control u that maximize the robustness of all predicates μ at all times k that are satisfied in the reference solution encoded by $\xi_{\varphi,k}$.

4.4 Analysis

First, show that all three MILP formulations find trajectories satisfying ϕ if they exist. Let ϕ be an STL formula, and $\{\xi_{\varphi,k}\}_{\varphi \sqsubseteq \phi, k \in [0..\|\phi\|}$ generated by solving (11), (12), or (13).

Theorem 2. *There exists a control u that generates trajectory s according to* (4) *that satisfies ϕ if and only if $\xi_{\phi,0} = 1$.*

Proof (Sketch). First, if $\xi_{\phi,0} = 1$, it follows by structural induction that all required subformulae, including ϕ, are satisfied by the computed trajectory s. Conversely, if there exists s that satisfies ϕ, then it follows that s satisfies all required subformulae. Thus, the satisfaction fractions $\xi_{\varphi,k}$ must be 1, including $\xi_{\phi,0}$.

Let s^* and $\{\eta_{\varphi,k}\}_{\varphi \sqsubseteq \phi, k \in [0..\|\phi\|}$ be the optimal solutions of (11) or (12). The following intermediate result is given without proof due to brevity.

Proposition 1. *If $\eta_{\varphi,k} = 1$, then $(\varphi, k) \in F_\phi(s^*)$.*

Next, we show that (11) and (12) satisfy the PS requirement of lowest-depth formulae.

Theorem 3. *There exists no signal s such that $s^* \prec_\phi s$.*

Proof (Sketch). The proof follows from Prop. 1, which states that the $\eta_{\varphi,k}$ correctly encode lowest-depth subformulae, and the objectives R_{HO} and R_{LDF}. For HO, the optimization considers solutions that maximize the number of formulae of lower-depth first by construction. For LDF, it is easy to show that any solution that satisfies a lower-depth formula leads to a larger objective value than any other solution with any number of higher-depth satisfied subformulae.

We give the result on the correctness of LP without proof.

Theorem 4. *The solution of LP* (14) *has maximum PS robustness* $\varrho(s,\phi)$ *over all trajectories that satisfy the same set of subformulae* (φ,k) *as* s^*.

Note that an optimal solution to Problem 3 might satisfy other subformulae than s^* even if both achieve the same number at each depth. As such, the LP (14) may lead to suboptimal solutions.

5 Case Studies

We describe two different study cases that show how PS works and its performance over the three encodings. First, we consider a multi-robot motion planning problem in a planar, continuous environment. Second, we show routing for heterogeneous teams of robots with sets of capabilities over graphs that represent discrete finite abstractions. More details can be found in [17]. All computation was performed on a PC with 20 cores at 3.7GHz with 64 GB of RAM. We used Gurobi [13] as MILP solver, which ensures that as long as the encoding is correctly defined, corner cases such as fully satisfiable and non-satisfiable specifications are cover

5.1 Continuous Space Multi-robot Planning

Let us consider first a single robot navigating in a planar environment $\mathcal{M} \subset \mathbb{R}^2$. Thus, $s(k) = [s_x, s_y]^\top \in \mathbb{R}^2$. We define regions of interest $\mathcal{A} = [-9.5,-5.5]^2$, $\mathcal{B} = [5.5,9.5] \times [-9.5,-5.5]$, $\mathcal{C} = [5.5,9.5]^2$, $\mathcal{D} = [-9.5,-5.5] \times [5.5,9.5]$ in $\mathcal{M}$, and region $\mathcal{E} = [-2.5,2.5]^2 \subseteq \mathcal{M}$ that robot $s(k)$ needs to avoid. We arbitrarily choose $A = B = I_{2\times 2}$, and $D = 0_{2\times 2}$ in (4). Throughout this case study, we consider methods discussed in Sec. 4. All three methods HO, LDF, and WLN, are able to find trajectories that satisfy the given specification when they exist. For example, consider the following STL formula $\phi_{FS} = ((\Box_{[0,3]}\mathcal{A}) \wedge (\Diamond_{[7,14]}\mathcal{B}) \wedge (\Box_{[15,20]}\mathcal{D}) \wedge (\sim \Box_{[0,20]}\mathcal{E}))$. In plain English, it required that "within time interval $[0,3]$ stay at region $\mathcal{A}$, eventually within $[0,14]$ visit $\mathcal{B}$, stay at $\mathcal{D}$ within $[15,20]$ and always avoid region $\mathcal{E}$". In Fig. 2 red lines show the solution for this specification using the three methods. All of three solution trajectories start at the blue star in $\mathcal{A}$, then they visit $\mathcal{B}$ and end at $\mathcal{D}$, where squares denote the final position of the robot.

Next, consider the STL formula $\phi_{PS} = (\Box_{[0,3]}\mathcal{A}) \wedge (\Box_{[10,21]}\mathcal{B}) \wedge (\Box_{[10,21]}\mathcal{D}) \wedge (\sim \Box_{[0,20]}\mathcal{E})$ that can not be fully satisfied due to the competing subformulae that require staying at $\mathcal{B}$ and $\mathcal{D}$ in the same time interval $[10,21]$. The solutions trajectories are shown in Fig. 2 with blue lines. Methods HO and LDF partially satisfy the specification by choosing to visit region $\mathcal{D}$, and, thus, satisfying the lowest-depth subformulae of ϕ_{PS}. In contrast, WLN splits the satisfaction between the predicates defining the two regions $\mathcal{B}$ and $\mathcal{D}$. In other words, the specification requires either $(s_x \geq 5.5) \wedge (s_x \leq 9.5) \wedge (s_y \geq -9.5) \wedge (s_y \leq -5.5)$ or $(s_x \geq -9.5) \wedge (s_x \leq -5.5) \wedge (s_y \geq 5.5) \wedge (s_y \leq 9.5)$ to hold, and WLN is satisfying half from the former, $(s_x \geq -9.5) \wedge (s_x \leq -5.5)$, and half of the latter,

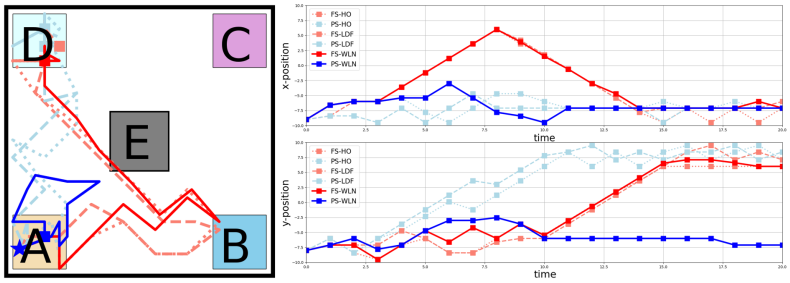

Fig. 2: State space and timeline in x-position and y-position of ϕ_{FS} in red and ϕ_{PS} in blue using HO, LDF, and WLN encoding methods.

$(s_y \geq -9.5) \wedge (s_y \leq -5.5)$, in the time window $[10, 21]$. The WLN trajectory satisfies the same maximum satisfaction fraction as HO and LDF, but does not fully satisfy any of the two tasks. However, the case was hand-crafted to show that WLN does not always return the optimal solution. In particular, we made the tasks symmetric. If the time intervals for visiting $\mathcal{B}$ and $\mathcal{D}$ are different, then the weight of their associated predicates differs and leads to a preference for one of the two regions.

In a multi-robot setting, unsatisfiability can result due to robot dropout as opposed to competing tasks. For instance, in Fig. 3 we consider two scenarios satisfying the following specification $\phi_{mr} = ((\square_{[15,30]} s_1 \in \mathcal{C}) \wedge (\square_{[15,30]} s_2 \in \mathcal{D}) \wedge (\square_{[15,30]} s_3 \in \mathcal{C}) \wedge (\square_{[15,30]} s_4 \in \mathcal{B}) \wedge (\square_{[15,30]} s_5 \in \mathcal{C}) \wedge (\square_{[15,30]} s_6 \in \mathcal{C}))$ with six robots. First, we solve the problem with HO approach shown in Fig. 3 with dashed lines. Then, we consider the same specification, but we make robot s_1 remain stuck in region $\mathcal{E}$. Even though it is just one agent specification ϕ_{mr} is only partially satisfied. The PS solution computed using LDF is shown in solid lines.

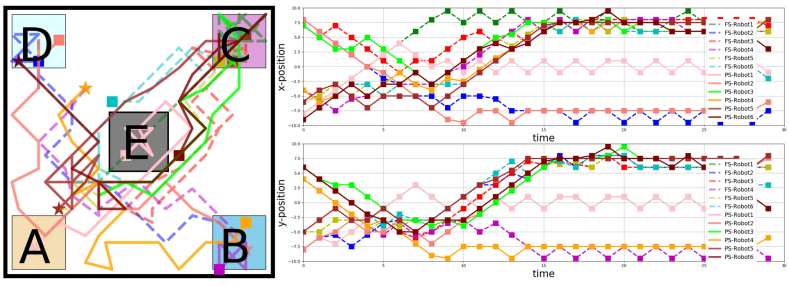

Fig. 3: State space and timeline in x-position and y-position of ϕ_{mr} fully satisfied with HO approach in dashed lines. Then, solution using LDF for robot one getting stuck in solid.

Finally, we compare the runtime performance between methods HO, LDF, and WLN by increasing the number of robots from one to six. In Fig. 4, we see that when there are one or two robots, LDF and WLN are faster than HO. Nevertheless, when the number of robots increases, so does the depth in the specification, and LDF becomes the worst performing method. The specification we used for this comparison is $\phi_n = \bigwedge_{j=1}^{n}(\bigwedge_{\ell=1}^{3}(\Box_{I_{\ell,j}}(s_n \in \mathcal{X}_{\ell,j})))$, with $n \in [1\,..\,6]$ the number of robots, $\mathcal{X}_{\ell,j}$ any arbitrary region in $\{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}\}$, and $I_{\ell,j}$ arbitrary time interval, respectively. It is also relevant to highlight that even when WLN is the fastest, it can drive to undesired PS if the specification contains symmetric competing subformulae.

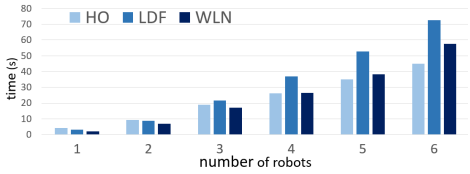

Fig. 4: Runtime performance comparison between HO, LDF, and WLN varying the number of robots from one to six, satisfying specification ϕ_n.

5.2 Route Planning with Capability Temporal Logic (CaTL)

We consider the precision agriculture application shown in Fig. 5. The environment is abstracted as a set of states $\mathcal{Q} = \{q_1, q_2, ..., q_9\}$ corresponding to regions of interest. Edges $\mathcal{E}$, shown as black lines, between states represent feasibility of transition between regions with positive integer durations $\mathcal{W}$. Self-loops (not shown) capture staying at states and have duration 1. States are labeled with atomic propositions from a set AP. Agents have capabilities $Cap = \{Vis, IR, UV, Mo\}$. The capability set of an agent determines its class. Fig. 5 shows the initial states of each agent and their class.

We use CaTL, a fragment of STL, to specify the mission for the multi-robot team. The core units of CaTL are tasks $T = (d, \pi, cp)$ defined by a duration d, a label π of regions where it takes place, and the required capabilities cp. The map $cp : Cap \to \mathbb{Z}_{\geq 0}$ indicates the minimum number of agents with each capability; zero means that capability is not needed. Tasks are combined using Boolean and temporal operators the same as STL (excluding negation). We use the encoding of team dynamics in [17], where robots are bundled together based on class. Thus, the team state is the number of robots of each class at each state of the environment. The team dynamics are encoded as flows on the environment graph which gives rise to a time-delayed linear system (slightly more involved than (4)). The MILP constraints and further details can be found in [17]. Consider tasks $T_1 = (3, q_2, (c_2, 3))$, $T_2 = (3, q_4, (c_1, 4))$, $T_3 = (6, q_9, (c_3, 3))$, $T_4 = (5, q_1, (c_4, 4))$, $T_5 = (3, q_6, (c_2, 1), (c_4, 1))$, and $T_6 = (2, q_8, (c_4, 4))$, and three CaTL specifications $\phi_1, \phi_2 = \bigwedge_i \Box_{[0,20]} T_i$, with $i \in \{1, 3, 5\}$ and $i \in \{2, 4, 6\}$ respectively, and $\phi_3 =$

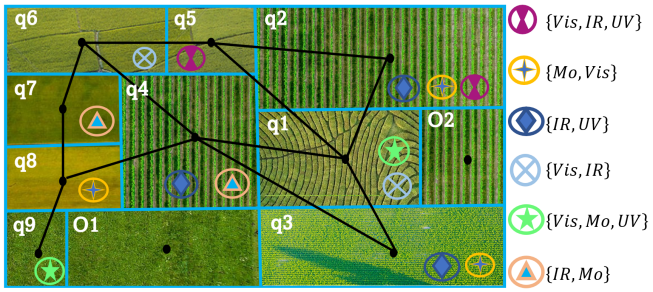

Fig. 5: Schematic of the precision agriculture problem. Regions correspond to crop types. Symbols represent robots with their associated capabilities. Capabilities include ultraviolet sensing (UV), moisture sensing (Mo), infrared sensing (IR), and vision (Vis). Black lines denote transition system between regions.

$\bigwedge_{i=1}^{6} \Diamond_{[0,20]} T_i$. Partial satisfaction of ϕ_ℓ can arise due to lack of robots with required capabilities, transition system constraints, and competing subformulae. The following table shows the performance of the three methods for the three specifications.

Table 1: Comparison on time performance and satisfaction percentage of specifications ϕ_1, ϕ_2, and ϕ_3 by methods HO, LDF, WLN, and baseline CaTL [22].

spec	**HO**		**LDF**		**WLN**		**CaTL**	
	Satisfaction	**t**(s)	**Satisfaction**	**t**(s)	**Satisfaction**	**t**(s)	ρ	**t**(s)
ϕ_1	0.929	1.884	0.943	1.732	0.972	1.498	-6	2.075
ϕ_2	0.982	1.617	0.990	1.859	0.991	1.635	-4	2.341
ϕ_3	1	1.773	1	2.009	1	1.940	1	2.32

Remark 3. For the baseline CaTL encoding [22], negative robustness indicates that ϕ_1 and ϕ_2 are violated. However, it does not indicate specification violation percentage, i.e., which and how many subformulae are satisfied.

Table 1 shows that all of the partial satisfaction approaches outperform the baseline encoding for CaTL [22]. Due to partial satisfaction, transform the specification's robustness as a linear problem approach of all predicates and time instances known to be satisfiable. We avoid adding binary variables for disjunctions and eventually operators since the inner optimization level indicates what needs to be satisfied. Furthermore, for small problems, WLN is faster than other two. However, as the number of variables of the problem increases, the HO method using Gurobi becomes faster and suitable to use.

6 Conclusions

In this paper, we introduced the partial satisfaction problem for STL specifications that requires the satisfaction of as many large formulae as possible. We formalize the partial satisfaction synthesis problem as a bi-level optimization that maximizes the robustness of lowest-depth formulae. We propose three MILP-based methods to solve the inner level of the optimization that find trajectories satisfying the maximum number of subformulae with preference to lower-depth ones. We used their solutions in a LP formulation that also maximizes the subformulae' robustness as an approximation for the solution to the outer level of the robust PS problem. We show the performance of the proposed methods in two multi-robot case studies involving motion planning in continuous spaces and routing over finite abstractions.

References

1. Cai, M., Peng, H., Li, Z., Gao, H., Kan, Z.: Receding horizon control-based motion planning with partially infeasible ltl constraints. IEEE Control Systems Letters **5**(4), 1279–1284 (2020)
2. Cai, M., Peng, H., Li, Z., Kan, Z.: Learning-based probabilistic ltl motion planning with environment and motion uncertainties. IEEE Transactions on Automatic Control **66**(5), 2386–2392 (2020)
3. Cardona, G.A., Calderon, J.M.: Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations. Applied Sciences **9**(8), 1702 (2019)
4. Choudhury, S., Gupta, J.K., Kochenderfer, M.J., Sadigh, D., Bohg, J.: Dynamic multi-robot task allocation under uncertainty and temporal constraints. Robotics: Science and Systems (2020)
5. Cortés, J., Egerstedt, M.: Coordinated control of multi-robot systems: A survey. SICE Journal of Control, Measurement, and System Integration **10**(6), 495–503 (2017)
6. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on robotics and Automation **20**(2), 243–255 (2004)
7. Davey, B.A., Priestley, H.A.: Introduction to lattices and order. Cambridge university press (2002)
8. Dokhanchi, A., Hoxha, B., Fainekos, G.: 5th International Conference on Runtime Verification, Toronto, ON, Canada., chap. On-Line Monitoring for Temporal Logic Robustness, pp. 231–246. Springer (2014)
9. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 92–106. Springer (2010)
10. Emam, Y., Mayya, S., Notomista, G., Bohannon, A., Egerstedt, M.: Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities. arXiv preprint arXiv:2003.03344 (2020)
11. Emam, Y., Wilson, S., Hakenberg, M., Munz, U., Egerstedt, M.: A receding horizon scheduling approach for search & rescue scenarios. arXiv pp. arXiv–2004 (2020)
12. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science **410**(42), 4262–4291 (2009)
13. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2021), https://www.gurobi.com

14. Heemels, W., De Schutter, B., Bemporad, A.: Equivalence of hybrid dynamical models. Automatica **37**(7), 1085–1091 (2001). https://doi.org/https://doi.org/10.1016/S0005-1098(01)00059-0
15. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. Acm Sigact News **32**(1), 60–65 (2001)
16. Jones, A.M., Leahy, K., Vasile, C., Sadraddini, S., Serlin, Z., Tron, R., Belta, C.: Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications
17. Jones, A.M., Leahy, K., Vasile, C., Sadraddini, S., Serlin, Z., Tron, R., Belta, C.: Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications. In International Symposium of Robotics Research. International Federation of Robotics Research, (2019)
18. Kamale, D., Karyofylli, E., Vasile, C.I.: Automata-based optimal planning with relaxed specifications. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2021)
19. Khalil, H.K.: Nonlinear systems third edition. Patience Hall (2002)
20. Lacerda, B., Parker, D., Hawes, N.: Optimal policy generation for partially satisfiable co-safe ltl specifications. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
21. Lahijanian, M., Almagor, S., Fried, D., Kavraki, L.E., Vardi, M.Y.: This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
22. Leahy, K., Jones, A., Vasile, C.I.: Fast decomposition of temporal logic specifications for heterogeneous teams. IEEE Robotics and Automation Letters (2022)
23. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004)
24. Notomista, G., Mayya, S., Hutchinson, S., Egerstedt, M.: An optimal task allocation strategy for heterogeneous multi-robot systems. In: 2019 18th European Control Conference (ECC). pp. 2071–2076. IEEE (2019)
25. Raman, V., Donzé, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., Seshia, S.A.: Model predictive control with signal temporal logic specifications. In: 53rd IEEE Conference on Decision and Control. pp. 81–87. IEEE (2014)
26. Sadraddini, S., Belta, C.: Robust temporal logic model predictive control. In: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 772–779. IEEE (2015)
27. Schillinger, P., Bürger, M., Dimarogonas, D.V.: 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)
28. Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C., Rus, D.: Optimality and robustness in multi-robot path planning with temporal logic constraints. The International Journal of Robotics Research **32**(8), 889–911 (2013)
29. Voos, H.: Nonlinear control of a quadrotor micro-uav using feedback-linearization. In: 2009 IEEE International Conference on Mechatronics. pp. 1–6. IEEE (2009)
30. Wilamowsky, Y., Epstein, S., Dickman, B.: Optimization in multiple-objective linear programming problems with pre-emptive priorities. Journal of the Operational Research Society **41**(4), 351–356 (1990)
31. Zhou, D., Schwager, M.: Vector field following for quadrotors using differential flatness. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 6567–6572 (2014). https://doi.org/10.1109/ICRA.2014.6907828