Planning for Heterogeneous Teams of Robots with Temporal Logic, Capability, and Resource Constraints

The International Journal of Robotics Research
XX(X):1–20

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Gustavo A. Cardona[1] and Cristian-Ioan Vasile[1]

Abstract

This paper presents a comprehensive approach for planning for teams of heterogeneous robots with different capabilities and the transportation of resources. We use Capability Temporal Logic (CaTL), a formal language that helps express tasks involving robots with multiple capabilities with spatial, temporal, and logical constraints. We extend CaTL to also capture resource constraints, where resources can be divisible and indivisible, for instance, sand and bricks, respectively. Robots transport resources using various storage types, such as uniform (shared storage among resources) and compartmental (individual storage per resource). Robots' resource transportation capacity is defined based on resource type and robot class. Robot and resource dynamics and the CaTL mission are jointly encoded in a Mixed Integer Linear Programming (MILP), which maximizes disjoint robot and resource robustness while minimizing spurious movement of both. We propose a multi-robustness approach for Multi-Class Signal Temporal Logic (mcSTL), allowing for generalized quantitative semantics across multiple predicate classes. Thus, we compute availability robustness scores for robots and resources separately. Finally, we conduct multiple experiments demonstrating functionality and time performance by varying resources and storage types.

Keywords

Formal methods, multi-robot systems, planning, transportation, capabilities.

1 Introduction

Advancements in multi-robot platforms have enabled new applications that use multiple robots with different capabilities. Such applications include aerial surveillance, disaster response, and planetary exploration, which could involve aerial and ground robots working together Dixit and Dhayagonde (2014); Tripicchio et al. (2015); Bresina et al. (2005); Cardona and Calderon (2019); Cardona et al. (2021). Having a team of heterogeneous robots can be very useful when it comes to satisfying tasks. By using their capabilities, they can explore different solutions and make the mission more resilient and robust. However, finding the best solution becomes difficult when the number of robots or classes increases. This is because considering all the different combinations and possibilities can be very complex and time-consuming. In fact, many existing planning algorithms become impractical or less effective when dealing with many robots and complex mission specifications.

Temporal logic has become increasingly popular for specifying tasks when synthesizing plans for large homogeneous and heterogeneous teams Guo and Dimarogonas (2017, 2015); Kantaros et al. (2019); Diaz-Mercado et al. (2015); Pant et al. (2018); Cardona et al. (2022, 2023c). This tool is especially useful for expressing spatial and temporal requirements for complex missions, such as the type of robots needed in a particular area at a specific time. While Linear Temporal Logic (LTL) Bhatia et al. (2010); Ulusoy et al. (2013); Karaman and Frazzoli (2011) is a commonly used temporal logic formalism in robotics, it only reasons over untimed sequences (e.g., "Visit region A before going to region B"), which may not be suitable for time-sensitive missions. To accommodate explicit timing

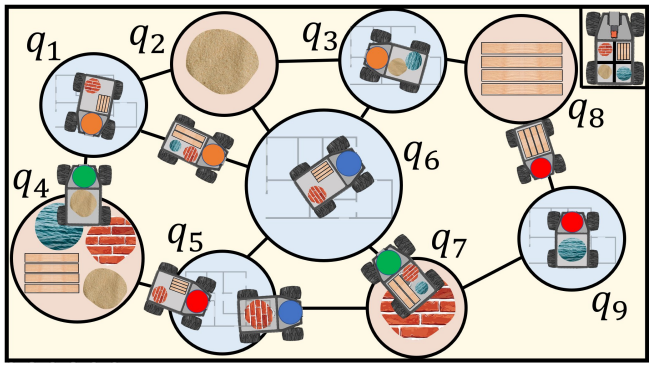

Figure 1. Schematic of a construction motion coordination problem. Connected circles correspond to construction areas (light blue) or warehouse storage (light brown). There are four resources, two indivisible (bricks and wooden beams) and two divisible (construction sand and water). The circle color on the robots indicates the agent class (set of capabilities each robot has). Uniform storage type is shown in the robots; however, a robot with compartmental storage capacities can be seen in the top-left corner.

[1]Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA.

Corresponding author:
Gustavo A. Cardona, Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA.
Email: gcardona@lehigh.edu

constraints, various other temporal logics have been proposed, such as Signal Temporal Logic (STL) Maler and Nickovic (2004), weighted-STL (wSTL) Mehdipour et al. (2020); Cardona et al. (2023a), Time Window Temporal Logic (TWTL) Vasile et al. (2017), Bounded Linear Temporal Logic (BLTL) Tkachev and Abate (2013), and Metric Temporal Logic (MTL) Brzoska (1998).

Among the formalisms that capture explicit time, STL specifications have additional advantages by considering predicates that allow reasoning about signals with relation operators instead of atomic propositions. For instance, it can express whether a car's speed limit should remain within 70 miles per hour by expressing the predicate "$speed \leq 70$". It also can be applied to discrete-valued signals, such as indicating the number of robots per class required for heterogeneous multi-robot frameworks. Additionally, unlike LTL or MTL, which only provide feedback on task satisfaction, STL offers a margin of satisfaction, also known as *robustness* Fainekos and Pappas (2009); Donzé and Maler (2010), which indicates the extent to which the mission is satisfied or violated. For example, if a car maintains a speed of 65 miles per hour when the speed limit is 70 miles per hour, the mission is considered satisfied with a robustness of 5. Nevertheless, even when considering a complex STL specification or a multi-agent approach Sun et al. (2022); Mehdipour et al. (2019); Cardona et al. (2022), robustness will be a single value computed by a recursive definition over the STL operators. The single robustness value lacks interpretability when considering an STL specification using different predicate classes, i.e., semantically different variables, e.g., reasoning about agents and resources. Instead, we propose a multi-robustness for Multi-Class Signal Temporal Logic (mcSTL), which generalizes the quantitative semantics for STL specifications with different predicate classes to compute disjoint robustness scores.

This work builds upon Jones et al. (2019); Leahy et al. (2021), which introduces Capability Temporal Logic (CaTL), an STL fragment allowing a team of robots with varying sets of capabilities, to satisfy different requests. We extend CaTL to also account for resources required to perform tasks. Tasks specify the type of robot capabilities and resources, the robot and resource quantities, the specific location in an environment, and the time robots need to be there. In the case of resources, they disappear once the satisfaction of the task starts. Satisfaction of a task is robot agnostic, i.e., there is no explicit allocation of a specific agent to the task; instead, any robot or team of robots with the required capabilities can complete the task. The resources are transported by robots with different storage types and capacities, such as uniform (i.e., regardless of the resource, all share the same space and robot's storage capacity) and compartmental (i.e., each resource has its own dedicated space and storage capacity). Agent and resource assignments and trajectories are created after computing a solution for agents' and resource flows. Additionally, multiple types of resources are considered, such as indivisible (e.g., bricks) or divisible (e.g., sand).

1.1 General Overview of the Approach

Here, we provide an overview of our approach to addressing route planning challenges in heterogeneous robots and resource transportation teams. Our method involves utilizing Capability Temporal Logic (CaTL Jones et al. (2019); Leahy et al. (2021)), a fragment of STL, to specify mission objectives. We have extended the semantics of CaTL* to capture multiple resource transportation modes and capacities.

Our model incorporates the dynamics of agents and resources and formulates an optimal route planning problem under CaTL specification using a Mixed Integer Linear Programming (MILP) approach. Fig. 2 provides a schematic overview of the problem, where only the CaTL formula and the scene description are needed as input. The CaTL formula describes the desired tasks, including their duration, the number of agent capabilities required, and the amount of resources needed to complete the task. Logical and temporal constraints can also be included. For the scene description, the user specifies the environment as a transition system, indicating the states, transitions, durations, and initial distribution of agents and resources.

The CaTL formula is parsed, and its associated Abstract Syntax Tree (AST) Hopcroft et al. (2001) is constructed using an LL(*) parser Parr (2007). We then translate the CaTL formula into an equivalent STL formula in the AST representation (refer to PyTeLo Cardona et al. (2023b) for more details). Using PyTeLo, we recursively translate the STL specification AST into a MILP, capturing the quantitative semantics of the specification. The environment, robot, and resource dynamics encoding are automatically generated as a MILP using the formulation in Sec. 6. Then, we utilize a MILP solver such as Gurobi Gurobi Optimization (2020) to solve the equivalent MILP, capturing the specification and dynamics and generating the solution as trajectories for agents and resources in the environment. The solution to the MILP provides trajectories for the flows of agents and resources in the environment. However, these trajectories do not assign specific robots to tasks or resources to be transported, as they are agent-agnostic. Instead, they rely on the number of agents per class and the amount of resources needed for satisfaction. Therefore, the final step is allocating agents and resources to flows (i.e., trips) according to the solution provided by the MILP. The overall result consists of trajectories for all agents and resources that they need to pick-up, transport, and drop-off over the mission horizon.

The main contributions of this paper are

1. Extending CaTL Leahy et al. (2021) to include tasks that require both agents and resources.
2. Defining and modeling resource transportation that can accommodate various types of resources (divisible, indivisible, packets, etc.) and considers different storage types (uniform, compartmental) and capacities for agents.
3. Defining Multi-Class Temporal Logic (mcSTL), an extension of STL that considers predicates from multiple semantic classes. We also define a

*We decided to retain the name CaTL name of the specification language that now considers resources required for tasks and not just agent capabilities as in Leahy et al. (2021)

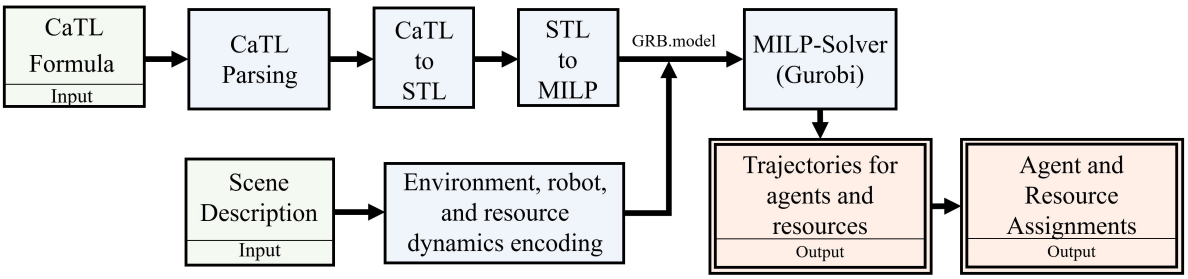

Figure 2. Schematic overview of CaTL with resource constraints.

multi-robustness score that enables quantification of satisfaction for each predicate class.

4. Applying the multi-robustness mcSTL in the context of CaTL to independently compute the robustness of robots and resources.
5. Proposing an efficient MILP-based planning approach that captures the CaTL specification, robots and resources dynamics, and transportation constraints. The MILP aims to maximize disjoint robots' and resources' robustness while minimizing spurious motion.
6. Evaluating the performance of the planning approach on construction scenarios to gain insights into the problem's characteristics and practical application.

The rest of the paper is organized as follows. First, Sec. 2, briefly reviews the related literature. Sec. 3, introduces some of the notation, STL syntax, and qualitative and quantitative semantics. Sec. 4, shows the formulation of the problem and the definitions of the environment, robots, capabilities, resources, and CaTL with resource constraints. Sec. 5, introduces the multi-robustness of mcSTL, capturing the different classes of predicates. Sec. 6, shows the mixed integer linear programming planning encoding. Sec. 7, shows some generalizations of the storage types and the use of resources. Sec. 8, shows the analysis and results of solving the MILP encoding for different case studies. Finally, Sec. 9, has the conclusions and future directions of the work.

2 Literature Review

Temporal logics have witnessed success in the domain of high-level planning for robotics, from single-agent systems Baier and Katoen (2008); Belta et al. (2017) and, increasingly, in multi-agent systems Guo and Dimarogonas (2015, 2016); Diaz-Mercado et al. (2015); Pant et al. (2018), including scenarios involving heterogeneous teams Schillinger et al. (2018). One of the most predominant temporal logic formalisms is Linear Temporal Logic which traditionally embraced automata theory as the primary paradigm Chen et al. (2011); Leahy et al. (2015); Baier and Katoen (2008); Belta et al. (2017); Finucane et al. (2010); Kamale et al. (2021); Badithela et al. (2023); Ulusoy et al. (2011); Nikou et al. (2016). However, applying automata-based methodologies in multi-robot scenarios introduces substantial challenges related to scalability and computational complexity, primarily owing to the requisite computation of automata products. Other works have considered a decomposition of tasks to allocate to single agents so that, by the composition of all agents, the satisfaction of the mission is guaranteed Tumova and Dimarogonas (2016); Yu and Dimarogonas (2021). Also, there is promising work on avoiding the need for automata products by incorporating fixed Petri nets approaches Hustiu et al. (2023); Lacerda and Lima (2019); Kloetzer and Mahulea (2014); Madridano et al. (2021). Nonetheless, handling large numbers of agents is still a challenge since the problem and its variants are NP-hard.

To alleviate these challenges, recent approaches have shifted their focus towards Mixed-Integer Linear Programs (MILP), benefitting from advancements in solution techniques and leveraging optimized off-the-shelf tools like Gurobi Gurobi Optimization (2020). This transition has proven effective, enabling MILP to handle a significantly larger number of variables and constraints with reasonable computational resources. Existing works adopting MILP strategies for multi-robot systems planning, particularly when subject to Signal Temporal Logic (STL) specifications, are evident in the literature Sun et al. (2022); Cardona and Vasile (2022); Caballero and Silano (2023); Liu et al. (2017); Buyukkocak and Aksaray (2022); Yu et al. (2023); Sewlia et al. (2023); Cardona and Vasile (2023), one of their main advantage is that they are complete, i.e., if there exists a solution it will be found. Compared to Linear Temporal Logic (LTL), Signal Temporal Logic (STL) allows for the explicit definition of time, which enhances the expressiveness of the mission by enabling the inclusion of timing constraints. However, existing STL methods focus on computing a solution by maximizing a single robustness score, even when predicates may belong to semantically different classes. In contrast, our research proposes a new approach by using a multi-robustness strategy for multi-class Signal Temporal Logic (mcSTL). This unique method allows us to formulate a multi-predicate class within the CaTL language, where the robustness of both resources and robots is maximized simultaneously but independently.

In high-level planning for multi-robot systems, besides the mission specification, the abstraction of the environment is crucial for handling the complexity of the problem. Allocating heterogeneous agents and resources to tasks with temporal logic constraints is a difficult problem known to be NP-hard. Previous works have explored the use of Petri nets and graph-based models to tackle this problem Hustiu et al. (2023); Lacerda and Lima (2019); Kloetzer and Mahulea (2014); Madridano et al. (2021). Other works have considered occupancy grids Sundram et al. (2018); Birk and Carpin (2006), cell decomposition maps Berman et al. (2009); Choset (2000), or Voronoi tessellation approaches Notomista et al. (2019); Fu et al. (2009). However, our approach incorporates dynamics efficiently by

imposing flow dynamic constraints over a transition system and strategically abstracting agents and resources. We define the joint state of the team as the count of agents with each capability and resource in each region at a given time. This abstraction model, combined with the use of the robot and resource routing flows problem approach, enhances the efficiency and scalability of our approach.

Some of the works closely related to our approach are Sahin et al. (2017, 2019), which have incorporated censusSTL Xu and Julius (2016) into (cLTL+) to enable agent or capability counting in abstracted time specifications. However, these works have not accounted for tasks involving robots and resource constraints. Related works considering capabilities and resource constraints in heterogeneous multi-robot systems are Schillinger et al. (2018); Guo and Zavlanos (2017). More specifically, in Guo and Zavlanos (2017), authors consider agents capable of gathering data, receiving, and uploading via buffers, defining in this way a resource transportation. Still, both use LTL language to express mission specifications, lacking the expressivity to account for time explicitly. The concept of resources is presented dually in the paper Schillinger et al. (2018). Firstly, it involves incorporating a proposition in LTL format, which is expressed as $(\varsigma > \mu)$, where ς denotes a specific resource, such as the battery power of a robot, while μ sets the threshold. Secondly, regions in the environment are labeled as either true or false, and their state is updated based on the actions taken in an automaton that captures resource constraints. In contrast, we expand on the notion of resource planning by considering divisible and indivisible types. Also, robots might have different storage types (compartmental and uniform) and adjustable storage capacity. Our proposed planning model does not include unnecessary binary variables for routing the resources while robots move in the environment.

3 Preliminaries and Notation

This section presents the notation and a glossary of symbols used throughout the paper and provides preliminaries about STL.

Let $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{B}$ denote the sets of integers, real numbers, and $\{0,1\}$, respectively. The set of integers greater than or equal to a is $\mathbb{Z}_{\geq a}$. For a set $\mathcal{S}$, $2^{\mathcal{S}}$ and $|\mathcal{S}|$ 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 $[a..b]$. For a range $I = [a..b]$, we use $\underline{I} = a$ and $\bar{I} = b$. Let $x \in \mathbb{R}^d$ be a d-dimensional vector. The i-th component of x is given by x_i, $i \in [1..d]$. The empty set is denoted by $\varnothing$. For ease of reference, a glossary of the main notation in the problem formulation Sec. 4.3 and the MILP encoding Sec. 6 is provided in Tab. 1.

3.1 Signal Temporal Logic

This section explains STL's syntax and qualitative and quantitative semantics. We also discuss its computation of time-bound and the description of its soundness property. These concepts are used later in the paper for STL extension languages mcSTL, mcSTL*, proposed in this paper, and its fragment CaTL Jones et al. (2019); Leahy et al. (2021), which is extended to accept resource predicates.

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 Maler and Nickovic (2004), is a specification language expressing real-time properties over signals.

The syntax of STL Maler and Nickovic (2004) 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, $\neg$, $\vee$, and $\wedge$ are the Boolean *negation*, *disjunction*, and *conjunction* operators, and discrete-timed temporal operators *eventually* $\Diamond_I$, *always* $\Box_I$, and $\mathcal{U}_I$ is the timed *until* operator with discrete-time interval $I \subseteq \mathbb{Z}_{\geq 0}$. The logical *False* value is $\bot = \neg\top$. Predicates $s \sim \mu$, with $\sim \in \{>, \geq, \leq, <\}$, follow via negation and sign change.

The (qualitative) semantics of STL formulae over signals s at time t is recursively defined in Maler and Nickovic (2004) as

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

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 Fainekos and Pappas (2009); Donzé and Maler (2010). The robustness score $\rho(s,\phi,t)$ is recursively defined as

$$\begin{aligned}
\rho(s,\top,t) &= \rho_\top, \\
\rho(s,s_i \geq \mu,t) &= s_i(t) - \mu, \\
\rho(s,\neg\phi,t) &= -\rho(s,\phi,t), \\
\rho(s,\phi_1 \wedge \phi_2,t) &= \min(\rho(s,\phi_1,t), \rho(s,\phi_2,t)), \\
\rho(s,\phi_1 \vee \phi_2,t) &= \max(\rho(s,\phi_1,t), \rho(s,\phi_2,t)), \\
\rho(s,\Box_I\phi,t) &= \min_{t' \in t+I} \rho(s,\phi,t'), \\
\rho(s,\Diamond_I\phi,t) &= \max_{t' \in t+I} \rho(s,\phi,t'), \\
\rho(s,\phi_1 \mathcal{U}_I \phi_2,t) &= \max_{t' \in t+I} \Big\{ \min\{\rho(s,\phi_2,t'), \\
&\qquad \min_{t'' \in [t..t']} \rho(s,\phi_1,t'')\}\Big\},
\end{aligned} \tag{3}$$

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

Theorem 1. Soundness Donzé and Maler (2010). *Let s be a signal and ϕ an STL formula. It holds $\rho(s,\phi,t) > 0 \Rightarrow (s,t) \vDash \phi$ for satisfaction and $\rho(s,\phi,t) < 0 \Rightarrow (s,t) \nvDash \phi$ for violation.*

Problem Variables Sec. 4		MILP Variables Sec. 6	
Notation	Description	Notation	Description
$\mathcal{AP}$	set of atomic propositions	K	planning time horizon
$q \in \mathcal{Q}$	state in the environment	$z_{q,g,k}$	number of agents of class $g \in \mathcal{G}$ at time $k = \Delta t$ at state $q \in \mathcal{Q}$
$e \in \mathcal{E}$	edge in the environment	$u_{e,g,k}$	number of agents of class $g \in \mathcal{G}$ at time $k = \Delta t$ at edge $e \in \mathcal{E}$
$\mathcal{W}(e)$	duration of traversing $e \in \mathcal{E}$	$y_{q,h,k}$	quantity of resource $h \in \mathcal{H}$ at time $k \in [0..K]$ at state $q \in \mathcal{Q}$
$\mathcal{J}$	index set of agents	$v_{e,h,k}$	quantity of resource $h \in \mathcal{H}$ entering edge $e \in \mathcal{E}$ at time $k \in [0..K]$
$\mathcal{C}$	set of capabilities	$C_{q,h,k}$	cross-consumption of resource $h \in \mathcal{H}$ used only once at any state $q \in \mathcal{Q}$ or time $k \in [0..K]$
$\mathcal{G}$	set of agent classes	$z_{\pi,q,c,k}$	amount of agents with capability c at time step k at every state q using label π
$\mathcal{H}$	set of resources	$y_{\pi,q,h,k}$	amount of resource h at time step k at every state q using label π
Ω_g, $\Omega_{h,g}$	agent transportation capacity (uniform, compartmental)	$z_{\varsigma(\pi,c),k}$	number of agents with capability c required at time k for satisfaction of a task
s_j, $s_{\mathcal{J}}$	agent and team trajectories	$y_{\varsigma(\pi,h),k}$	amount of resource h required at time k for satisfaction of a task
b_h, $b_{\mathcal{H}}$	resource and all resources trajectories	$\Lambda(q)$	set of all tasks satisfied at state q
$n_{q,c}(t)$	number of agents at state $q \in \mathcal{Q}$ with capability $c \in \mathcal{C}$ at time $t \in \mathbb{Z}_{\geq 0}$	$x_{\phi_T,k}$	satisfaction or violation of task T at time k
$m_j(t,h)$	assignment of resource amount $h \in \mathcal{H}$ at time $t \in \mathbb{Z}_{\geq 0}$ to agent $j \in \mathcal{J}$	τ_u	agents total travel time
T	Task	γ_u	agents travel time regularization term
ϕ	CaTL formula	τ_v	resources total travel time
ρ_a, ρ_h	agent and resource robustness	γ_v	resources travel time regularization term

Table 1. Table of symbols for variables used in the problem formulation and MILP encoding sections.

The time horizon of an STL formula Dokhanchi et al. (2014) is defined as

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = s \geq \mu, \\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1, \\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2\}, \\ \bar{I} + \|\phi\|, & \text{if } \phi = \{\Diamond_I \phi, \Box_I \phi\}, \\ \bar{I} + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \mathcal{U}_I \phi_2. \end{cases} \tag{4}$$

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.

4 Problem Formulation

In this section, we define the route planning problem for heterogeneous teams of robots performing missions specified as Capability Temporal Logic formulae (CaTL) Jones et al. (2019); Leahy et al. (2021). We extend CaTL to account for the capabilities and resources required to fulfill tasks. Additionally, we extend the agent model with various resource transportation modalities and capacities. Finally, we conclude the section with the optimal route planning problem under capability temporal logic task and resource constraints.

The following example motivates the problem class we consider in this paper, where resources are required at locations of interest in the environment to perform tasks by agents with heterogeneous task and transportation capabilities.

Example 1. *Consider an environment with areas* q_1, q_3, q_5, q_6, q_9 *under construction and warehouses at* q_2, q_4, q_7, q_8. *A fleet of ground-based robots with different capabilities, such as bricklaying, digging, sample extraction, video surveillance, and structure inspection, is deployed to assist in the construction areas. Each robot has a combination of task capabilities defining a robot class (e.g., a robot can have the capabilities of bricklaying and digging) and transportation. Each robot class has a storage capacity to transport resources such as sand, water, bricks, and wooden beams from warehouses to construction areas. As usual in a construction area, not all capabilities or resources are needed simultaneously and change daily depending on the progress of construction. An example of a list of tasks the robots need to perform during a workday is given in List 1.*

1. From deployment to the end of the day, one video surveillance is required at every construction area.
2. Within 1 to 6 hours after deployment, digging is required for 2 hours and 2 wooden beams in every construction area.
3. Within 4 and 12 hours after deployment, structure inspection is required in every construction area for 1 hour. Afterward, 10 kg of sand and 10 liters of water are required for foundation-laying tasks.
4. Within 12 hours after deployment to the end of the day, 2 bricklaying and 200 bricks are required for 6 hours at every construction area.

LIST 1: Example list of construction tasks requiring capabilities and resources.

Based on this example, we introduce the environment, capabilities, resources, and robot models.

4.1 Environment, Agent, and Resource Models

Consider a team of heterogeneous agents deployed in a bounded environment $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W})$, where $\mathcal{Q}$ is a finite set of locations of interest (states), $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set capturing the possible transitions between locations, and $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ maps each transition to its travel duration. We assume that time is discretized, and all weights $\mathcal{W}(\mathcal{E})$ are integer multiples of a discretization time $\Delta t > 0$. States are labeled with atomic propositions from a set $\mathcal{AP}$. The labeling function is denoted by $L : \mathcal{Q} \to 2^{\mathcal{AP}}$. An agent stationary at $q \in \mathcal{Q}$ is modeled as a unit-weight self-transition, i.e., $(q, q) \in \mathcal{E}$ for all $q \in \mathcal{Q}$, and $\mathcal{W}((q, q)) = 1$.

We consider heterogeneous agents with varying capabilities for performing tasks and transporting resources. The finite set of agents is denoted by $\mathcal{J}$ with tasks capabilities from the finite set $\mathcal{C}$. Continuing Example 1, the capabilities of a robot may include bricklaying, digging, sample extraction, video surveillance, and structure inspection using ultrasound sensors. The capability set of agent $j \in \mathcal{J}$ is $c_j \subseteq \mathcal{C}$, and defines the *agent's class* $g = c_j$. The set of all agent classes is $\mathcal{G} \subseteq 2^{\mathcal{C}}$.

We also consider that each agent is capable of transporting resources. Consider the finite set of resources $\mathcal{H}$. Resources can be divisible (e.g., sand, water, fuel) or indivisible (e.g., bricks, wooden beams, solar panels). We consider multiple resource storage modes for transportation, *uniform storage* and *compartmental storage*. In the uniform storage case, agents' storage can hold all resource types and is characterized by a maximum transportation capacity $\Omega_g > 0$ for each agent class $g \in \mathcal{G}$. For example, the trunk of a car can hold various resources such as bags, tools, and construction materials. In the compartmental storage case, each resource $h \in \mathcal{H}$ has its own compartment with a given maximum transportation capacity $\Omega_{h,g} \geq 0$ in the agents' storage, $g \in \mathcal{G}$. In this case, we denote the set of agent classes that can transport resource $h \in \mathcal{H}$ by $\mathcal{G}_h = \{g \in \mathcal{G} \mid \Omega_{h,g} > 0\}$. For example, the reservoir of a car can only hold fuel; Robots for science missions have limited unique storage slots for samples collected from the environment.

Remark 1. *The proposed modeling framework and solution can accommodate a mixture of storage modes. For simplicity, we only consider the extreme cases of uniform (single storage for all resources), and compartmental (specialized storage for each resource) storage in the problem formulation and solution description. In Sec. 7.1, we show how to generalize and adapt the algorithms for custom storage configurations. Moreover, we consider storage defined by agent classes for performance reasons. However, we can accommodate cases of agents with the same capability sets but distinct storage, as discussed in Sec. 7.1.*

Each agent, $j \in \mathcal{J}$, is characterized by its initial state $q_{0,j} \in \mathcal{Q}$, and capability set c_j gives its class. The trajectory of an agent j in environment Env is defined as $s_j : \mathbb{Z}_{\geq 0} \to \mathcal{Q} \cup \mathcal{E}$ such that $s_j(t)$ is the state occupied or transition traversed by agent j at time $t \in \mathbb{Z}_{\geq 0}$, and every agent starts at the initial state $s_j(0) = q_{0,j}$. The synchronous trajectory of a set of agents $\mathcal{J}$ is $s_{\mathcal{J}} : \mathbb{Z}_{\geq 0} \to (\mathcal{Q} \cup \mathcal{E})^{|\mathcal{J}|}$. The number of agents at state $q \in \mathcal{Q}$ with capability $c \in \mathcal{C}$ at time $t \in \mathbb{Z}_{\geq 0}$ is

$$n_{q,c}(t) = |\{j \in \mathcal{J} \mid q = s_j(t), c = c_j\}|.$$

Similarly, the trajectory (distribution) of resource $h \in \mathcal{H}$ over the environment is $b_h : \mathbb{Z}_{\geq 0} \times (\mathcal{Q} \cup \mathcal{E}) \to \mathbb{R}_{\geq 0}$. Note that for indivisible resources $b_h \in \mathbb{Z}_{\geq 0}$. The initial distribution of resources, $b_h(0, \mathcal{Q})$ for all $h \in \mathcal{H}$, is given, and we assume that all resources are at states, i.e., $b_h(0, e) = 0$ for all $e \in \mathcal{E}$. The trajectory of all resources is denoted by $b_{\mathcal{H}}$.

For simplicity, we enforce that agents pick up, drop off, and transfer resources between themselves only at states $q \in \mathcal{Q}$. A *transportation plan* for agents $\mathcal{J}$ with respect to their trajectory $s_{\mathcal{J}}$ is a set of assignments $m_j : \mathbb{Z}_{\geq 0} \times \mathcal{H} \to \mathbb{R}_{\geq 0}$ that define the quantity of resource $h \in \mathcal{H}$ associated with agent $j \in \mathcal{J}$ at time $t \in \mathbb{Z}_{\geq 0}$. We impose $m_j(t, h) = 0$ for all $h \in \mathcal{H}$ whenever $s_j(t) \in \mathcal{Q}$ which captures pick-up, drop-off, and transfer of resources in a unified way, and may not correspond to physical operations (e.g., a robot that passes through a state $q \in \mathcal{Q}$ does not need to drop off and pick up the resources it is carrying). A transportation plan is *feasible* if the transportation capacity of agent j is satisfied at all times,

$$m_j(t, h) \leq \Omega_{h,g}, \forall h \in \mathcal{H}, \ \forall t \geq 0 \text{ (compartmental)},$$

$$\sum_{h \in \mathcal{H}} m_j(t, h) \leq \Omega_g, \ \forall t \geq 0 \text{ (uniform)}.$$

Now that we have introduced the environment, agents, capabilities, and resources models, we are ready to introduce CaTL with task and resource constraints.

4.2 Capability Temporal Logic with Resource Constraints

We extend the Capability Temporal Logic (CaTL) [†] Specification language from Jones et al. (2019) to account for resource constraints when performing tasks. The core units of CaTL are *tasks* that capture the required number of agents with each capability required in a location of interest. We extend the task definition to specify required resource quantities as follows

Definition 1. *A* task *is a tuple* $T = (d, \pi, cp, rs)$*, where* $d \in \mathbb{Z}_{\geq 1}$ *is a discrete duration of time (i.e., multiple time steps* Δt*),* $\pi \in \mathcal{AP}$ *is an atomic proposition,* $cp : \mathcal{C} \to \mathbb{Z}_{\geq 0}$ *and* $rs : \mathcal{H} \to \mathbb{R}_{\geq 0}$ *are counting maps specifying how many agents with each capability and how many resources of each type should be in each region labeled* π*, respectively. Note that for divisible resources* $rs(h) \in \mathbb{R}_{\geq 0}$*, while for indivisible ones* $rs(h) \in \mathbb{Z}_{\geq 0}$*.*

A capability c not required to perform task T is defined by $cp(c) = 0$. Similarly, $rs(h) = 0$ indicates that h is not required to perform T. We denote the set of capabilities and resources necessary for a task T by $cp_T = \{c \in Cap \mid cp(c) > 0\}$, and $rs_T = \{h \in \mathcal{H} \mid rs(h) > 0\}$, respectively.

[†] We have decided not to introduce a new name for the extension of CaTL with resources. Instead, throughout the paper, when we refer to CaTL, we mean the version defined in this section.

Definition 2. *The* syntax *of CaTL Jones et al. (2019) is a fragment of STL described in (1). The CaTL syntax in Backus-Naur form is*

$$\phi ::= T \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \mathcal{U}_I \phi_2 \mid \Diamond_I \phi \mid \Box_I \phi$$

where ϕ is a CaTL formula, T is a task containing capability and resource constraints, *$\wedge$ and $\vee$ are the Boolean conjunction and disjunction operators, $\mathcal{U}_I$, $\Diamond_I$, and $\Box_I$ are the time-bounded until, eventually, and always operators, respectively.*

Example 2. (Continuation of Example 1). *Thus, the mission described in List 1 can now be expressed as*

$$\phi = \Box_{[0,24]} T_1 \wedge \Diamond_{[1,6]} T_2 \wedge \Diamond_{[4,12]} T_3 \wedge \Diamond_{[12,24]} T_4,$$

with the set of tasks defined as follows

$$\begin{aligned}
T_1 &= (1, \pi_{cons}, \{(Struct.Insp., 1)\}, \varnothing),\\
T_2 &= (2, \pi_{cons}, \{(Digging, 1)\}, \{(woodbeams, 2)\}),\\
T_3 &= (1, \pi_{cons}, \{(Struct.Insp., 1)\}, \{(sand, 10), (water, 10)\}),\\
T_4 &= (6, \pi_{cons}, \{(bricklaying, 2)\}, \{(bricks, 200)\}),
\end{aligned}$$

where $\pi_{cons} \in \mathcal{AP}$, captures all regions under construction $L^{-1}(\pi_{cons}) = q_1, q_3, q_5, q_6, q_9$.

Remark 2. *As in Jones et al. (2019), CaTL does not include negation since the negation of tasks does not have a well-defined meaning. This is feasible because any STL formula can be put in its positive normal form Sadraddini and Belta (2015), and CaTL is a fragment of STL.*

Definition 3. *The Boolean (qualitative) semantics of CaTL are defined over trajectories $s_{\mathcal{J}}$ of agents and $b_{\mathcal{H}}$ of resources. At time t, satisfaction of a task T is defined by*

$$\begin{aligned}
(s_{\mathcal{J}}, b_{\mathcal{H}}, t) \vDash T \Leftrightarrow \forall \tau \in [t \,..\, t+d], \forall q \in L^{-1}(\pi),\\
\forall c \in cp_T, \forall h \in rs_T,\\
n_{q,c}(\tau) \geq cp(c) \;\wedge\; b_h(t,q) \geq rs(h)\,.
\end{aligned}$$

The Boolean and temporal operators' semantics are the same as for STL, (2). A pair of team and resource trajectories satisfy a CaTL formula ϕ, denoted $(s_{\mathcal{J}}, b_{\mathcal{H}}, t) \vDash \phi$ if $(s_{\mathcal{J}}, b_{\mathcal{H}}, 0) \vDash \phi$.

Tasks require resources only at the time of satisfaction, unlike agents required throughout their duration. Resources associated with a task T are consumed when T is satisfied and disappear from the environment in the next time step in the amount $rs(h)$ required by T, for all $h \in rs_T$. However, we must ensure that concurrent tasks using the same resource type at overlapping locations do not exceed the total resources available at these locations. In such a case, not all tasks requiring the same resource can be satisfied simultaneously. We refer to this property as the *cross-consumption constraint*.

Next, we review the *agent availability robustness* from Jones et al. (2019), and propose a counterpart for resources.

Definition 4. Agent Availability Robustness Jones et al. (2019). *The agent availability robustness of a task is defined as*

$$\rho_a(s_{\mathcal{J}}, T, t) = \min_{c \in cp_T} \min_{t' \in [t..t+d]} \min_{q \in L^{-1}(\pi)} n_{q,c}(t') - cp(c). \quad (5)$$

For Boolean and temporal operators, robustness is defined recursively as for STL, (3).

The agent availability robustness of a task captures the deviation from the number of agents needed to perform the task over its duration, capabilities involved, and locations spanned. Positive values indicate the maximum number of arbitrary agents whose failure leads to failing the task. We compute ρ_a based only on the agents' trajectory $s_{\mathcal{J}}$.

We propose *resource availability* defined similarly.

Definition 5. Resource Availability Robustness. *The resource availability robustness of a task is defined as*

$$\rho_h(b_{\mathcal{H}}, T, t) = \min_{h \in rs_T} \min_{q \in L^{-1}(\pi)} b_h(t,q) - rs(h). \quad (6)$$

Again, for Boolean and temporal operators, robustness is defined recursively as for STL, (3).

Resource robustness captures the surplus and shortage of resources required to satisfy a task and is computed only from the resource trajectory $b_{\mathcal{H}}$.

4.3 Problem Statement

Before we state the problem, let us explain how the agents' trajectories with resource trajectories are linked. We say that a resource trajectory $b_{\mathcal{H}}$ is consistent with agents' trajectory $s_{\mathcal{J}}$ if a feasible transportation plan exists for all agents $j \in \mathcal{J}$ with respect to $s_{\mathcal{J}}$ that induces $b_{\mathcal{H}}$. Specifically, resource quantities at states and transition at each time step result from agents carrying them via some transportation plan. We can now formally introduce the problem as follows.

Problem 1. *Given a set of agents $\mathcal{J}$ with initial states $q_{0,j}$, capabilities $c_j \subseteq \mathcal{C}$ deployed in environment Env with initial resources distribution $b_{\mathcal{H}}(0, \mathcal{Q})$, and CaTL specification ϕ, find trajectories s_j and b_h for agents and resources such that $(s_{\mathcal{J}}, b_{\mathcal{H}}) \vDash \phi$, $b_{\mathcal{H}}$ is consistent with $s_{\mathcal{J}}$, $b_{\mathcal{H}}$ satisfies the cross-consumption constraint at all times, and the robustness score*

$$F = \rho_a(s_{\mathcal{J}}, \phi, 0) + \gamma \cdot \rho_h(b_{\mathcal{H}}, \phi, 0), \quad (7)$$

is maximized, where $\gamma \in \mathbb{R}_{>0}$ is a positive constant value.

Pb. 1 captures the goal of using a fleet of heterogeneous robots (agent classes defined by the set of capabilities) to satisfy CaTL tasks considering capabilities, resources, and logical and temporal constraints. Note that instead of routing resources and enforcing correspondence between agents and resources, we want to exploit the constraint that the transportation plan (flows of resources) has to follow agents' trajectory (flows of agents). Thus, we avoid introducing complex agent-resource assignment constraints. The robustness score F corresponds to finding a plan that maximizes the number of robots with specific capabilities and the number of resources available at locations asked in a specification.

Pb. 1 is solved by encoding the models as a Mixed Integer Linear Program (MILP). This allows us to use off-the-shelf

solvers that are orders of magnitude faster than standard automata-based approaches. We show the encodings in Sec. 6.

Note that for STL specifications, a positive robustness guarantees Boolean satisfaction (Soundness, Thm. 1). However, in this paper, we work with multiple robustness variables, such as agent and resource availability, which means that a positive result to the weighted addition of ρ_a and ρ_h in (7) does not guarantee the satisfaction of a CaTL specification formula (an example and formal results are presented in Sec. 8.3). Furthermore, in the syntax of STL presented in Sec. 3, there is no distinction when computing the robustness of the STL specification formula that uses multiple predicate classes. To address this issue, in the next section, we introduce a semantic extension of STL called Multi-class STL (mcSTL). mcSTL can handle multiple predicate classes and compute their respective robustness scores.

5 Multi-robustness for STL Specifications with Disjoint Predicate Classes

The standard robustness score (3) for STL specifications considers that all predicates ($\varsigma = s_i \sim \mu$) are the same, belong to the same class Sadraddini and Belta (2015); Kurtz and Lin (2022); Mehdipour et al. (2019); Raman et al. (2014). Thus, even when signal components have different meanings, the robustness score (3) treats them all the same and combines them into a single number. However, application settings may consider multiple robustness scores from the same specification, e.g., in Pb. 1, we consider capability and resource predicates stemming from semantically disjoint variables. To capture this, we introduce the *multi-robustness* semantics of an STL specification, which generalizes the standard quantitative semantics to multiple predicate classes. The definition follows for CaTL since it is a fragment of STL. Throughout the paper, we assume that STL formulas are in PNF form.

Definition 6. Multi-robustness for Multi-Class STL. *A Multi-Class STL (mcSTL) formula is a tuple $\psi = (\phi, \Sigma, \mathcal{L})$, where ϕ is an STL formula, Σ is a set of predicate classes, $\mathcal{L} : \mathfrak{P}(\phi) \to \Sigma$ is a function that maps each predicate $\varsigma \in \mathfrak{P}(\phi)$ to its predicate class $\sigma \in \Sigma$, and $\mathfrak{P}(\phi)$ is the set of predicates of ϕ. The multi-robustness score ρ_σ with respect to class σ of a predicate $\varsigma = s_i \geq 0$ is*

$$\rho_\sigma(s, \phi, t) ::= \begin{cases} \varsigma(s_i(t)), & \phi = \varsigma,\ \mathcal{L}(\varsigma) = \sigma, \\ \varnothing, & \phi = \varsigma,\ \mathcal{L}(\varsigma) \neq \sigma, \end{cases} \tag{8}$$

where s is a signal. The robustness of temporal and logical operators [‡] are computed in the same manner as in (3). When $\varnothing$ is encountered in a computation, the operand is ignored, i.e., $a \otimes \varnothing = a$, $\varnothing \otimes \varnothing = \varnothing$, and $-\varnothing = \varnothing$ where $a \in \mathbb{R}$ and $\otimes \in \{\min, \max\}$.

Note that $\rho_\sigma(s, \phi, 0) = \varnothing$ if and only if predicates from the class σ do not appear in ϕ, i.e., $\mathcal{L}^{-1}(\sigma) = \varnothing$.

In the following, we show that the classes' robustness are novel scores ρ_σ, $\sigma \in \Sigma$, that capture different aspects of satisfaction and violation than the overall robustness.

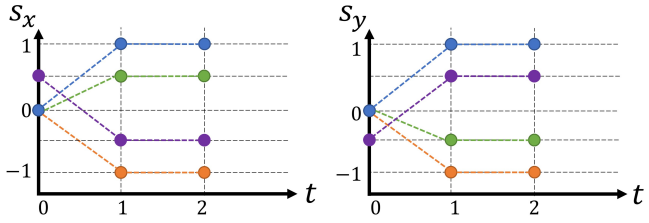

Figure 3. Ex. 3: Four different solution trajectories for ϕ_1^{ex}, ϕ_3^{ex}, ϕ_3^{ex}, and ϕ_4^{ex}.

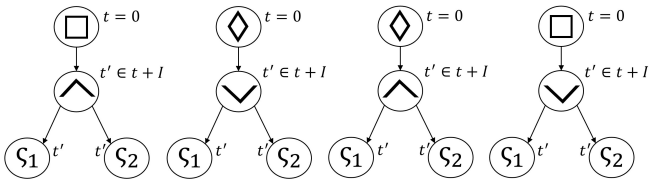

Figure 4. Ex. 3: AST for ϕ_1^{ex}, ϕ_2^{ex}, ϕ_3^{ex}, and ϕ_4^{ex}.

Example 3. *Consider the following STL specifications $\phi_1^{ex} = \Box_{[1,2]}(\varsigma_1 \wedge \varsigma_2)$, $\phi_2^{ex} = \Diamond_{[1,2]}(\varsigma_1 \vee \varsigma_2)$, $\phi_3^{ex} = \Diamond_{[0,1]}(\varsigma_1 \wedge \varsigma_2)$, and $\phi_4^{ex} = \Box_{[0,1]}(\varsigma_1 \vee \varsigma_2)$. with $\varsigma_1 = s_x > 0$, $\varsigma_2 = s_y > 0$, $\varsigma_1 \in \sigma_1$, $\varsigma_2 \in \sigma_2$ and $\Sigma = \{\sigma_1, \sigma_2\}$. The three specifications' abstract syntax trees (AST) are shown in Fig. 4. The following formulas give the robustness scores for the three specifications and a signal s:*

$$\begin{aligned}
\rho(s, \phi_1^{ex}, 0) &= \min\{\min\{s_x(1), s_y(1)\}, \min\{s_x(2), s_y(2)\}\}, \\
\rho(s, \phi_2^{ex}, 0) &= \max\{\max\{s_x(1), s_y(1)\}, \max\{s_x(2), s_y(2)\}\}, \\
\rho(s, \phi_3^{ex}, 0) &= \max\{\min\{s_x(0), s_y(0)\}, \min\{s_x(1), s_y(1)\}\}, \\
\rho(s, \phi_4^{ex}, 0) &= \min\{\max\{s_x(0), s_y(0)\}, \max\{s_x(1), s_y(1)\}\}.
\end{aligned}$$

We obtain the robustness scores for σ_1 and σ_2 by replacing $s_x(t)$ and $s_y(t)$ for $t \in 1, 2$ with $\varnothing$, respectively. For example, $\rho_{\sigma_1}(s, \phi_1^{ex}, 0) = \min\{\min\{s_x(1), \varnothing\}, \min\{s_x(2), \varnothing\}\} = \min\{s_x(1), s_x(2)\}$.

Consider the blue, green, and orange trajectories in Fig. 3. The overall and class robustness scores computed using (3) and (8), respectively, are given in Table 2.

Signal	ϕ	ρ	ρ_{σ_1}	ρ_{σ_2}
s^{blue}	ϕ_1^{ex}	1	1	1
s^{green}	ϕ_1^{ex}	-0.5	0.5	-0.5
s^{orange}	ϕ_1^{ex}	-1	-1	-1
s^{blue}	ϕ_2^{ex}	1	1	1
s^{green}	ϕ_2^{ex}	0.5	0.5	-0.5
s^{orange}	ϕ_2^{ex}	-1	-1	-1
s^{purple}	ϕ_3^{ex}	-0.5	1	0.5
$-s^{purple}$	ϕ_4^{ex}	0.5	-1	-0.5

Table 2. STL robustness scores for the overall formula and for classes σ_1 and σ_2.

Positive overall robustness does not imply that any class robustness score needs to be positive. The counter-examples at lines 5 and 8 of Table 2 illustrate cases with 1 and 2 negative class robustness scores and positive overall robustness, respectively. Similarly, negative overall robustness provides no information on the class scores.

[‡] For brevity, we identify the mcSTL formula ψ with its STL formula ϕ when the classes and labeling functions are clear from context.

Counter-examples at lines 2, 3, and 7 of Table 2 have negative overall robustness, but 1, 2, and 0 have negative class robustness scores, respectively.

As shown in Ex. 3, there is no relation between the signs of the multi-robustness values for mcSTL and the sign of the overall robustness. Thus, there is no direct soundness property stemming from multi-robustness. The combination of min and max operators from Boolean and temporal operators hinders the consistency when predicate classes are modulated. To address this inconsistency, we introduce a fragment of STL called class-complete mcSTL (mcSTL*), defined as follows.

Definition 7. Class-complete mcSTL. *A class-complete mcSTL (mcSTL*) is an mcSTL formula $\psi = (\phi, \Sigma, \mathcal{L})$ that has the following properties. For all $\varsigma \in \mathfrak{P}(\phi)$*

$$(1)\, pa_\phi(\varsigma) = \wedge,$$
$$(2)\, \forall \sigma' \in \Sigma \setminus \{\mathcal{L}(\varsigma)\},\ \exists \varsigma' \text{ such that, } \mathcal{L}(\varsigma') = \sigma', \text{ and } \varsigma' \in ch_\phi(pa_\phi(\varsigma)),$$

where $pa_\phi(\phi')$ and $ch_\phi(\phi')$ return the parent and children formulas of ϕ' based on the AST of ϕ, respectively.

The two properties of mcSTL* mean that every predicate $\varsigma \in \mathfrak{P}(\phi)$ has a conjunction operator as its parent, and has sibling predicates from all predicate classes. When computing (8) over an mcSTL* specification ψ, a consistency property exists that relates class robustness to the overall specification robustness.

Theorem 2. mcSTL* Multi-robustness consistency. *Let s be a signal and $\psi = (\phi, \Sigma, \mathcal{L})$ an mcSTL* formula with predicate set $\mathfrak{P}(\phi) \neq \varnothing$ from classes Σ. The following property holds for all $t \geq 0$ and $\sigma \in \Sigma$*

$$\rho_\sigma(s, \phi, t) \geq \rho(s, \phi, t).$$

The proof of Thm. 2 is in Appendix A.

Corollary 1. *Under the same setting as in Thm. 2 and $\rho(s, \phi, t) > 0$, it follows that $\rho_\sigma(s, \phi, t) > 0$ for all $\sigma \in \Sigma$.*

Cor. 1 implies the necessary condition that all robustness class values $\rho_\sigma(s, \phi, 0)$ are positive for the overall robustness $\rho(s, \phi, 0)$ to be positive. However, it is not sufficient.

5.1 mcSTL and mcSTL* Discussion about Limitations and Connections

Here, we discuss the connections and limitations of all temporal logic formalisms introduced in this paper: STL, CaTL, mcSTL, and mcSTL*. STL is a powerful language that enables users to reason about continuous signals while abiding by logical and temporal constraints. It allows for incorporating multiple signal components and the computation of both qualitative and quantitative semantics. The qualitative semantics defined in (2) determine whether operators are satisfied or violated, while the quantitative semantics determine the degree of satisfaction as defined in (3). Furthermore, STL has been shown to possess soundness properties about its quantitative semantics, as demonstrated in Thm. 1. This theorem establishes a connection between an STL formula's satisfaction and its robustness score being greater than zero. However, when considering semantically unrelated predicates on a single specification, it amalgamates all predicate categories to calculate a single robustness score, which may not be ideal. For instance, consider predicates that pertain to the position and velocity of a vehicle and establish a specification that encompasses the car's safety regulations. It becomes imperative to determine if the vehicle's position is in proximity to violating safety standards and colliding with other vehicles, regardless of whether it abides by speed rules. If the velocity margin satisfaction is significantly larger, it could mask the potential danger of violating position rules when calculating the overall robustness score.

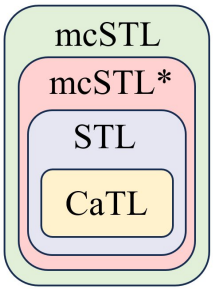

Figure 5. Set of expressivity definition of STL, CaTL, mcSTL, and mcSTL*.

Motivated by computing robustness scores over a single specification that contains multiple classes of predicates that are semantically disjoint between them, we have defined a semantic extension called multi-class Signal Temporal Logic (mcSTL), therefore, $STL \subseteq mcSTL$. Nevertheless, even though we can now define multiple types of predicates $\sigma \in \Sigma$ over the same specification, their individual robustness class $\rho_\sigma(s, \phi, t)$ and overall robustness scores $\rho(s, \phi, t)$ have not an evident correlation in their signs as shown in Table 2. The combination of $\min$ and $\max$ operators hinders the consistency of the signs when signals are modulated, making it difficult to determine the overall satisfaction of the specification. This poses a considerable limitation of this semantic extension on determining satisfaction without extensive bookkeeping of variables.

We define class-complete multi-class Signal Temporal Logic (mcSTL*) to address this inconsistency with $mcSTL^* \subseteq mcSTL$. mcSTL* requires that every predicate in the specification has as parent node a conjunction operator and all other classes existing as siblings. Consider, as an example, specifications $\phi_1^{ex} = \square_{[1,2]}(\varsigma_1 \wedge \varsigma_2)$ and $\phi_3^{ex} = \lozenge_{[0,1]}(\varsigma_1 \wedge \varsigma_2)$ in Ex. 3, where $\Sigma = \{\sigma_1, \sigma_2\}$ with $\sigma_1 = \{\varsigma_1\}$ and $\sigma_2 = \{\varsigma_2\}$. For this type of specification, we have that all robustness classes have to be positive for the overall robustness to be positive, implying that the specification will be satisfied only if all individual classes are satisfied, as shown in Th. 2 and Cor. 1. Nonetheless, mcSTL*'s most significant limitation is its definition, requiring applications involving all predicate classes in conjunction whenever we want to define predicate properties in the specification.

In contrast to all other temporal logic formalisms discussed, CaTL allows us to give meaning to predicates for the number of agents and amount of resources. Note that the resource and agent availability problem studied in this

paper is just a particular case of the multi-robustness for mcSTL*. The definition of a CaTL task in Def. 3 includes the predicate classes in conjunction, in accordance with the mcSTL* properties. We have $\Sigma = \{\sigma_a, \sigma_h\}$, where σ_a defines the agents' class of predicates and σ_h the resources class of predicates. Allowing the linear combination computation of the robustness of different predicate classes such as agents' availability $\rho_a(s_{\mathcal{J}}, \phi, 0)$ and resources availability $\rho_h(b_{\mathcal{H}}, \phi, 0)$ shown in (7).

Therefore, some of the relations and properties of CaTL, STL, mcSTL*, and mcSTL are shown in Fig. 5 and Table 3.

	CaTL	STL	mcSTL*	mcSTL
Logical operators	✓	✓	✓	✓
Temporal operators	✓	✓	✓	✓
Robustness	✓	✓	✓	✓
Multi-class	✓	✗	✓	✓
Tasks	✓	✗	✗	✗
Multi-robustness consistency	✓	NA	✓	✗

Table 3. Properties hold for every semantic extension or fragment of STL discussed in this work.

We have defined a framework that allows us to handle specifications with multiple predicate classes and have mission satisfaction guarantees on their individual robustness computation being greater than zero. In the next section, we translate the problem formulation described in Sec. 4 into its corresponding MILP encoding. This enables us to compute agent and resource trajectories that satisfy the CaTL specification and dynamics constraints.

6 Mixed Integer Linear Programming Encoding

This section describes the encoding of agents' and resource dynamics, CaTL specifications defined in Sec. 4 as a MILP. We formulate Pb. 1 using these variables and define an objective function that captures the desired behavior of the robot fleet, maximizing the robot availability and resource availability robustness while minimizing the total travel time of robots and resources.

6.1 Agents' Dynamics Encoding

We introduce agent class decision variables for all states and edges in the environment Env over the planning time horizon $K = \|\phi\|$ as in (4) to encode agent dynamics.

Let $z_{q,g,k} \in \mathbb{Z}_{\geq 0}$ represent the number of agents of class $g \in \mathcal{G}$ at time $k = \Delta t$ at state $q \in \mathcal{Q}$, and $u_{e,g,k} \in \mathbb{Z}_{\geq 0}$ the number of agents of class $g \in \mathcal{G}$ entering edge $e \in \mathcal{E}$ at time k, for all $k \in [0..K]$. Then the agents' dynamics Jones et al. (2019) for all $q \in \mathcal{Q}$, $g \in \mathcal{G}$, $k \in [0..K]$ is captured as follows

$$z_{q,g,0} = |\{j \in \mathcal{J} \mid q_{0,j} = q, c_j = g\}|, \tag{9}$$

$$z_{q,g,k} = \sum_{(q',q)\in\mathcal{E}} u_{(q',q),g,k-\mathcal{W}((q',q))}, \tag{10}$$

$$\sum_{(q,q')\in\mathcal{E}} u_{(q,q'),g,k} = \sum_{(q',q)\in\mathcal{E}} u_{(q',q),g,k-\mathcal{W}((q',q))}, \tag{11}$$

where (9) captures the initial agent distribution over the environment; specifically, the number of agents $j \in \mathcal{J}$ with a capability set c_j related to class g at $k = 0$ present at region $q \in \mathcal{Q}$. Note that the same agent cannot be at two places simultaneously and that no agent is allowed to be initially at edges $\mathcal{E}$. On the other hand, (10) and (11) capture the agent distribution at every time k such that the flow of agents is conserved over the environment Env at all times. Thus, the number of agents entering a node $(q', q) \in \mathcal{E}$ must equal the number of agents going out $(q, q') \in \mathcal{E}$. This covers the case of agents staying at state q modeled as taking the self-loop $(q, q) \in \mathcal{E}$ for a one-time unit.

6.2 Resource Dynamics Encoding

Similarly, as for agents' dynamics, we encode resource dynamics by imposing flow constraints using the following variables. Let $y_{q,h,k} \in \mathbb{H}$ represents the quantity of resource $h \in \mathcal{H}$ at time $k \in [0..K]$ at state $q \in \mathcal{Q}$, and $v_{e,h,k} \in \mathbb{H}$ represents the quantity of resource $h \in \mathcal{H}$ entering edge $e \in \mathcal{E}$ at time $k \in [0..K]$. Note that resources can be divisible $\mathbb{H} = \mathbb{R}_{\geq 0}$, indivisible $\mathbb{H} = \mathbb{Z}_{\geq 0}$, or binary $\mathbb{H} = \mathbb{B}$. Lastly, let $C_{q,h,k}$ be the cross-consumption constraint, which ensures that the same resource is not used more than once in case different tasks ask for the same resource simultaneously and at the same location. Then resource dynamics for all $q \in \mathcal{Q}$, $h \in \mathcal{H}$, and $k \in [0..K]$, is encoded as follows

$$y_{q,h,0} = b_{\mathcal{H}}(0, q), \tag{12}$$

$$y_{q,h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))}, \tag{13}$$

$$C_{q,h,k} + \sum_{(q,q')\in\mathcal{E}} v_{(q,q'),h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))}, \tag{14}$$

where (12) captures how the resources $h \in \mathcal{H}$ are distributed over the environment Env at time $k = 0$. Similarly, (13) and (14) represent the resource distribution at every time step $k \in [0..K]$, and impose the conservation of resource flows while considering cross-consumption constraint (23) defined below.

We can now impose the transportation constraints that resources must be carried by agents using the agent and resource flow variables. These constraints depend on the transportation type, either uniform or compartmental, as described in Sec. 4.1. The motion for all of the resources $h \in \mathcal{H}$, over every edge $e = (q, q') \in \mathcal{E}$ for all $k \in [0..K]$, is

$$v_{e,h,k} \leq \sum_{g\in\mathcal{G}_h} \Omega_{h,g} u_{e,g,k}, \qquad \text{(compartmental)}, \tag{15}$$

$$\sum_{h\in\mathcal{H}} v_{e,h,k} \leq \sum_{g\in\mathcal{G}} \Omega_g u_{e,g,k}, \qquad \text{(uniform)}, \tag{16}$$

where $u_{e,g,k}$ is the number of agents of class $g \in \mathcal{G}$ entering edge $e \in \mathcal{E}$ at time k defined in Sec. 6.1 and $\Omega_{h,g}$, Ωg are the transportation capacities for the compartmental and uniform cases, respectively. Thus, (15) and (16) guarantee the total amount of resource $h \in \mathcal{H}$ transported by agents do not exceed the maximum capacity that agents can transport based on storage capacity and type. Variable $v_{e,h,k}$ is bounded by

$$\overline{v_{e,h,k}} = \min\{|\mathcal{J}| \cdot \max_{g\in\mathcal{G}} \Omega_{g,h}, \sum_{q\in\mathcal{Q}} b_h(0, q)\},$$

for compartmental transportation, for all $h \in \mathcal{H}$, and

$$\overline{v_{e,h,k}} = \min\{|\mathcal{J}| \cdot \max_{g\in\mathcal{G}} \Omega_g, \sum_{q\in\mathcal{Q}} \sum_{h\in\mathcal{H}} b_h(0, q)\},$$

for uniform transportation. In the case of package transportation (binary resource), we set $y_{q,h,k}$ and $v_{e,h,k}$ as binary variables.

Remark 3. *The main insight of the encoding is that we do not need to assign resources to agents explicitly. We only need to ensure that agents can transport resources along their routes. Thus, we extract the resource trajectories from resource flows similar to the agent case, see Sec. 6.6 for details. Moreover, no binary and integer auxiliary variables are required to enforce a satisfiable solution.*

6.3 CaTL Specification Encoding

We translate the CaTL specification ϕ to be captured into the MILP in three steps.

First, we introduce the variables $z_{\pi,q,c,k} \in \mathbb{R}_{\geq 0}$, $y_{\pi,q,h,k} \in \mathbb{R}_{\geq 0}$ that capture the amount of agents with capability $c \in \mathcal{C}$ and resources $h \in \mathcal{H}$ at time step $k \in [0..K]$ at every state $q \in \mathcal{Q}$ using label $\pi \in \mathcal{AP}$. We couple them with the system variables for agents and resources, $z_{q,g,k}$ and $y_{q,h,k}$, respectively. The following constraints ensure that capabilities and resources are not counted more than once in all regions with atomic proposition π labeled with $L(q)$

$$\sum_{\pi \in L(q)} z_{\pi,q,c,k} = \sum_{g:c \in G} z_{q,g,k}, \tag{17}$$

$$\sum_{\pi \in L(q)} y_{\pi,q,h,k} = y_{q,h,k}. \tag{18}$$

Second, we introduce variables $z_{\varsigma(\pi,c),k} \in \mathbb{R}$ for agents and $y_{\varsigma(\pi,h),k} \in \mathbb{R}$ for resources, coupled with the counting proposition variables (see (21) and (22) for definitions, respectively). We use these variables to guarantee that the number of agent capabilities and resources in the regions is at least the amount requested in the counting propositions. We have

$$z_{\varsigma(\pi,c),k} \leq z_{\pi,q,c,k}, \tag{19}$$

$$y_{\varsigma(\pi,h),k} \leq y_{\pi,q,h,k}, \tag{20}$$

for all $q \in L^{-1}(\pi)$, $c \in \mathcal{C}$, $h \in \mathcal{H}$, $\pi \in \mathcal{AP}$, and $k \in [0..K]$.

Lastly, we encode the CaTL specification for the satisfaction of tasks by translating it into an STL specification. This is possible since CaTL is a fragment of STL. CaTL allows intuitive, compact encodings of tasks, and like STL, it can be efficiently encoded as a MILP Leahy et al. (2021); Cai et al. (2021); Liu et al. (2023); Cardona and Vasile (2022) by recursively encoding the specification based on the robustness definition (3). A task $T = (d, \pi, cp, rs)$ defined as in Def. 1 is semantically equivalent to the following STL specification formula

$$\phi_T = \bigwedge_{h \in \mathcal{H}} \varsigma(\pi, h) \wedge \Box_{[0,d]} \bigwedge_{c \in \mathcal{C}} \varsigma(\pi, c),$$

where $\varsigma(\pi, c)$ and $\varsigma(\pi, h)$ are

$$\varsigma(\pi, c) = \min_{q \in L^{-1}(\pi)} \{n_{q,c}\} \geq cp(c), \tag{21}$$

$$\varsigma(\pi, h) = \min_{q \in L^{-1}(\pi)} \{b_h(\cdot, q)\} \geq rs(h). \tag{22}$$

Therefore, CaTL specification can be fully translated into an STL specification and further added into the MILP by encoding the operators using (8). Satisfaction can be guaranteed via the binary variable associated with the satisfaction of the overall STL specification Raman et al. (2014); Sadraddini and Belta (2015); Mahajan (2010).

6.4 Resources Cross-consumption Encoding

Before introducing the objective function, we define the cross-consumption constraint which ensures that a resource is used once and immediately consumed in the required amount at the start of a task's satisfaction. Let $\Lambda(q) = \{T = (d, \pi, cp, rs) \mid \pi \in L(q)\}$ be the set that captures all the tasks that are going to be satisfied at the same location. Resource cross-consumption is given by

$$C_{q,h,k} = \sum_{T \in \Lambda(q)} rs(h) \cdot x_{\phi_T,k}, \tag{23}$$

where $x_{\phi_T,k} \in \mathbb{B}$, capture whether or not the task T in the set $\Lambda(q)$ is satisfied. Thus, the cross-consumption constraint is

$$C_{q,h,k} \leq y_{q,h,k}, \tag{24}$$

for all $q \in \mathcal{Q}$, $h \in \mathcal{H}$, $k \in [0..K]$. In other words, (24) ensures that the total amount of resources needed to satisfy multiple tasks in overlapping regions is less or equal to the number of resources at that location.

6.5 Objective Definition

Here, we formulate Pb. 1 as an optimization problem (MILP), which can be solved using any off-the-shelf software tool. In addition to the objective in Pb. 1, we want to account for agents' and resources' travel time and eliminate inefficient behavior, i.e., spurious motion. For instance, agents taking longer paths or transporting resources even when they are not required to satisfy tasks wastes time and energy. We use regularization terms based on the weighted total travel times for agents and resources

$$\tau_u = \sum_{k=0}^{K} \sum_{g \in \mathcal{G}} \sum_{e=(q,q') \in \mathcal{E}, q \neq q'} u_{e,g,k}, \tag{25}$$

$$\gamma_u = \frac{\alpha_u}{K \cdot |\mathcal{J}|}, \tag{26}$$

$$\tau_v = \sum_{k=0}^{K} \sum_{h \in \mathcal{H}} \sum_{e=(q,q') \in \mathcal{E}, q \neq q'} v_{e,h,k}, \tag{27}$$

$$\gamma_v = \frac{\alpha_v}{K \cdot \max_h \sum_{q \in \mathcal{Q}} b_h(0, q)}, \tag{28}$$

where $\alpha_v, \alpha_u \in [0, 1]$ and τ_u, τ_v are the total travel times of the agents and resources moving along edges in the environment. The weights γ_u, γ_v scale the regularization terms to ensure that agents and resources robustness have higher priority; agent and resource regularization does not come at the expense of maximizing either agents or resources robustness.

Finally, the overall optimization problem of maximizing agents and resources' robustness while minimizing agent and

resource traveling time is

$$\begin{aligned}\max_{z,u,y,v}\ & \rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v,\\ \text{s.t. } & (s_{\mathcal{J}}, b_{\mathcal{H}}) \vDash \phi,\\ & (9), (10), (11), \quad \text{(Agent dynamics)},\\ & (12), (13), (14), \quad \text{(Resource dynamics)},\\ & (15) \text{ or } (16), \quad \text{(Agent resource capacity)},\\ & (17) - (20), \quad \text{(Task satisfaction)},\\ & (23) \text{ and } (24), \quad \text{(Resources cross-consumption)}.\end{aligned} \tag{29}$$

Note that the solution to the problem provides us with the counts of agents and amounts of resources required to meet the mission specification and dynamics constraints. However, it does not explicitly assign agents for transporting and fulfilling the tasks. To solve this issue, the next section introduces an algorithm that can be used for agent and resource assignments.

6.6 Agent and Resource Assignments

The outcome of solving (29) are $z_{q,g,k}$, $y_{q,h,k}$, $u_{(q,q'),g,k}$, and $v_{(q,q'),h,k}$ capturing the number of robots of each class $g \in \mathcal{G}$ and the amount of resource $h \in \mathcal{H}$ at time $k \in [0..K]$ at either state $q \in \mathcal{Q}$ or traversing edge $(q, q') \in \mathcal{E}$, respectively. However, these values do not provide explicit information about the trajectories of individual robots or the assignment of resources to robots. Therefore, we extend the approach presented in Leahy et al. (2021) for extracting individual robot trajectories and assigning resource transportation to robots. The method is described in the Alg. 1.

The sets $\mathcal{U}_{\mathcal{E},\mathcal{G},K} = \{u_{e,g,k} \mid e \in \mathcal{E}, g \in \mathcal{G}, k \in [0..K]\}$, and $\mathcal{V}_{\mathcal{E},\mathcal{H},K} = \{v_{e,h,k} \mid e \in \mathcal{E}, h \in \mathcal{H}, k \in [0..K]\}$ representing the MILP solution are the inputs of the method. Alg. 1 is executed for each time step, generating a trajectory for each agent and assigning resources to agents that fulfill the mission specification ϕ. The trajectory of a robot j is represented as a sequence of states and edges $s_j(k) = q_{0,j} \dots (q, q') \dots q_{K,j}$, where $k \in [0..k]$ starting at the initial position $s_j(0) = q_{0,j}$ of agent $j \in \mathcal{J}$, see Sec. 4. Let $\mathcal{J}_{q,g,k}$ be the set of agents of class $g \in \mathcal{G}$, at state $q \in \mathcal{Q}$, at time $k \in [0..K]$. The transportation assignment of resource h to agent j at time k is denoted by $\mathcal{B}_{\mathcal{H}}(j, h, k)$. The procedure initializes each agent j's trajectory with its initial state $q_{0,j}$, and the sets of agents at each state q of class g at time 0, lines 1-2. For each time step, the algorithm propagates agents from states onto outgoing transitions (line 4), iteratively computes the agents trajectories (lines 5-7), recomputes agents at states (line 8), and allocates resources to agents for transportation (line 9-10). In the first step of the loop, the set of agents at each state q from each class g is partitioned over the outgoing transitions $\mathcal{E}^+(q) = \{(q, q') \in \mathcal{E}\}$ using function $Part()$ based on the agent numbers from the solution $\mathcal{U}_{\mathcal{E}^+(q),g,k}$, line 4. The trajectories of agents $\mathcal{J}_{e,g,k}$ departing on each outgoing transition $e = (q, q')$ are extended over the transition's duration $\mathcal{W}(e)$, lines 5-7. The third step, computes the agents $\mathcal{J}_{q,g,k+1}$ from each class g arriving at each state q at time $k+1$ as the union over incoming transitions $\mathcal{E}^-(q) = \{(q', q) \in \mathcal{E}\}$, line 8.

Finally, we use the $Alloc(\bigcup_{g\in\mathcal{G}} \mathcal{J}_{e,g,k}, (v_{e,h,k})_{h\in\mathcal{H}})$ function to assign resources that must be transported by agents traveling along edges $e \in \mathcal{E}$ according to their corresponding storage type and capacity, lines 9-10, described in Alg. 2. It formulates a Linear Programming (LP) problem to allocate amount $\xi_{j,h} \in \mathbb{R}_{\geq 0}$ of resources h to be transported by agent $j \in J$ according to its storage type and capacity. The LP's variables are $\xi_{j,h}$ and are created for all agents in the input set J, line 1. Next, we create the constraints set $\mathcal{CS}$ for resource allocation. The first constraint requires the exact allocation of the amount $v(h)$ for each resource $h \in \mathcal{H}$ among agents in j, line 2. If the storage type is uniform, we add the total transportation capacity bound Ω_{c_j} for each agent j, lines 3-4. Conversely, if the storage type is compartmental, we add the capacity bounds Ω_{h,c_j} for each resource h and each agent j, lines 5-6. Finally, we solve the LP and allocate the amounts of resource $\xi^*_{j,h}$ to agent j, line 7.

Algorithm 1: Robot trajectories and resource transportation assignments.

Input: $\mathcal{U}_{\mathcal{E},\mathcal{G},K}, \mathcal{V}_{\mathcal{E},\mathcal{H},K}$
Output: $\mathcal{S}_{\mathcal{J}}, \mathcal{B}_{\mathcal{H}}$

1 $s_j(0) \leftarrow q_{0,j}, \forall j \in \mathcal{J}$
2 $\mathcal{J}_{q,g,0} \leftarrow \{j \in \mathcal{J} \mid q = q_{0,j}, g = c_j\}$
3 **for** $k \in [0..K-1]$ **do**
  `// Propagate - departing robots`
4 $\mathcal{J}_{\mathcal{E}^+(q),g,k} \leftarrow Part(\mathcal{J}_{q,g,k}, \mathcal{U}_{\mathcal{E}^+(q),g,k})\ \forall q \in \mathcal{Q}, \forall g \in \mathcal{G}$
  `// Extend trajectories`
5 **for** $j \in \mathcal{J}_{e,g,k}$, $\forall q \in \mathcal{Q}, \forall g \in \mathcal{G}$, *where* $e = (q, q')$ **do**
6 $s_j((k+1)..(k+\mathcal{W}(e)-1)) \leftarrow e$
7 $s_j(k+\mathcal{W}(e)) \leftarrow q'$
  `// Accumulate - arriving robots`
8 $\mathcal{J}_{q,g,k+1} \leftarrow \bigcup_{e\in\mathcal{E}^-(q)} \mathcal{J}_{e,g,k+1-\mathcal{W}(e)}, \forall q \in \mathcal{Q}, \forall g \in \mathcal{G}$
9 **for** $e \in \mathcal{E}$, $k' \in [k..k+W(e)]$ **do**
10 $\mathcal{B}_{\mathcal{H}}(\cdot, \cdot, k') \leftarrow Alloc(\bigcup_{g\in\mathcal{G}} \mathcal{J}_{e,g,k}, (v_{e,h,k})_{h\in\mathcal{H}})$
11 **return** $\mathcal{S}_{\mathcal{J}}, \mathcal{B}_{\mathcal{H}}$

Algorithm 2: Resource Assignments – $Alloc()$

Input: $J \subseteq \mathcal{J}, V : \mathcal{H} \to \mathbb{R}_{\geq 0}$
Param: Ω_g or $\Omega_{h,g}$, $\forall g \in \mathcal{G}, h \in \mathcal{H}$, type
Output: $(\xi^*_{j,h})_{j\in J, h\in\mathcal{H}}$ – amount of h transported by agent j

1 Create $\xi_{j,h} \in \mathbb{R}_{\geq 0}$, $\forall j \in J, h \in \mathcal{H}$
2 $\mathcal{CS} = \{\sum_{j\in J} \xi_{j,h} = V(h) \mid h \in \mathcal{H}\}$
3 **if** *type = 'uniform'* **then**
4 $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{\sum_{h\in\mathcal{H}} \xi_{j,h} \leq \Omega_{c_j} \mid \forall j \in J\}$
5 **else if** *type = 'compartmental'* **then**
6 $\mathcal{CS} \leftarrow \mathcal{CS} \cup \{\xi_{j,h} \leq \Omega_{h,c_j} \mid \forall j \in J, h \in \mathcal{H}\}$
7 Solve LP $\xi^*_{j,h} = \arg\min_{\xi_{j,h}} 1$ s.t. $\mathcal{CS}$
8 **return** $(\xi^*_{j,h})_{j\in J, h\in\mathcal{H}}$

7 Encoding Generalization

In this section, we show extensions to the solution presented in Sec. 6, that handle generalized storage configurations and external resource dynamics, i.e., resource creation and destruction during deployment. Moreover, we present an encoding for gradual resource creation during a task's satisfaction.

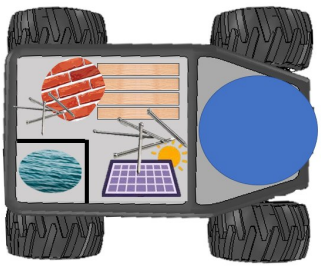

Figure 6. Example of generalization of storage type: a combination of compartmental and uniform storage.

7.1 Generalized storage configurations

In Sec. 6, we showed uniform and compartmental storage setups encodings. However, robots may have a complex combination of these storage types. See the example in Fig. 6.

Let $\mathcal{H}_a \subseteq \mathcal{H}$ define the storage compartment $a \in \mathcal{A}$ as the set of resources that it can store. All resources must be handled, i.e., $\mathcal{H} = \bigcup_{a\in\mathcal{A}} \mathcal{H}_a$. Compartmental storage requires $|\mathcal{H}_a| = 1$ for all $a \in \mathcal{A}$, while uniform storage corresponds to $|\mathcal{A}| = 1$. Let $\Omega_{a,g}$ be the total transportation capacity of compartment a on an agent from class $g \in \mathcal{G}$ for all resources $\mathcal{H}_a$. Next, we consider first the case where resources are stored only in one compartment and then generalize it to the case of multiple compartments.

7.1.1 Case 1 (Mutually Exclusive): In this case, each resource can be stored only in a single compartment. Thus, the compartments are mutually exclusive and partition the set of resources. Formally, $\mathcal{H}_a \cap \mathcal{H}_{a'} = \varnothing$ for all $a \neq a'$.

The resource transportation capacity constraints become

$$\sum_{h\in\mathcal{H}_a} v_{e,h,k} \leq \sum_{g\in\mathcal{G}_a} \Omega_{a,g} u_{e,g,k}, \tag{30}$$

for all $e \in \mathcal{E}$, $k \in [0..K]$, $a \in \mathcal{A}$, where $\mathcal{G}_a = \{g \mid \Omega_{a,g} > 0\}$ is the set of agent classes that have compartment $a \in \mathcal{A}$. The constraints in (15) and (16) are special cases of (30).

7.1.2 Case 2 (Overlapping): When resources can be stored in multiple compartments, we must track the amount of each resource in each compartment. Let $\mathcal{A}(h) = \{a \in \mathcal{A} \mid h \in \mathcal{H}_a\} \neq \varnothing$ be the set of compartments that can store resource $h \in \mathcal{H}$.

We introduce variables $v_{e,h'_a,k}$ that indicate the amount of resource h in compartment a, and the constraints

$$v_{e,h,k} = \sum_{a\in\mathcal{A}(h)} v_{e,h'_a,k} \tag{31}$$

$$\sum_{h\in\mathcal{H}_a} v_{e,h'_a,k} \leq \sum_{g\in\mathcal{G}_a} \Omega_{a,g} u_{e,g,k} \tag{32}$$

for all $e \in \mathcal{E}$, $k \in [0..K]$, $a \in \mathcal{A}$. When $|\mathcal{A}(h)| = 1$, we have the previous case. Specifically, (31) and (32) become $v_{e,h,k} = v_{e,h'_a,k}$ and (30), respectively.

7.2 Generalized creation, destruction, and gradual consumption of resources

The encoding discussed in Sec. 6 assumes predetermined total amounts for each resource. However, resources may be added or removed from the environment by external processes (e.g., delivery and material loss) or by agents' actions (e.g., mining). Additionally, when the resources are used to satisfy a task, we consider these to disappear from the environment at the beginning of a task and are no longer available. However, commonly resources are consumed gradually by tasks.

7.2.1 Creation and destruction of resources: We consider two cases of creation and destruction of resources: (a) robot independent that captures external processes, and (b) robot dependent that requires agent action.

Robot independent: Let us introduce the rates $C^+_{q,h} \geq 0$ and $C^-_{q,h} \geq 0$ of creation and destruction for all resources $h \in \mathcal{H}$ at each state $q \in \mathcal{Q}$, respectively. These constants define external, independent processes that govern resources in the environment. We adjust the resource flow dynamics in (13) to account for the net resource change $C^+_{q,h} - C^-_{q,h}$ per time step

$$y_{q,h,k} = \max\{\sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))} + C^+_{q,h} - C^-_{q,h}, 0\},$$

where the max ensures resource amounts remain non-negative. The bound on variables $v_{e,h,k}$ becomes $\overline{v_{e,h,k}}^* = \overline{v_{e,h,k}} + k \cdot \sum_{q\in\mathcal{Q}} C^+_{q,h}$. Lastly, the RHS of (14) is set to the new $y_{q,g,k}$.

Robot dependent: Each robot class g has a rate of creation $C^+_{h,g} > 0$ and destruction $C^-_{h,g} > 0$ of resources $h \in \mathcal{H}$. We denote the sets of classes that can create and destroy resource h by $\mathcal{G}^+_h$ and $\mathcal{G}^-_h$, respectively. We modify the resource flow constraints (13) to account for agents' resource creation and destruction work while stationary at states q

$$y_{q,h,k} = \max\{\sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}} + \sum_{g\in\mathcal{G}^+_h} C^+_{h,g} u_{(q,q),g,k} - \sum_{g\in\mathcal{G}^-_h} C^-_{h,g} u_{(q,q),g,k}, 0\},$$

where, again, we use the max operator to disallow negative resource amounts. The bound of $v_{e,h,k}$ becomes $\overline{v_{e,h,k}}^* = \overline{v_{e,h,k}} + k \cdot |\mathcal{J}| \cdot \sum_{g\in\mathcal{G}^+_h} C^+_{h,g}$. As before, the RHS of (14) is set to the new expression of $y_{q,g,k}$.

7.2.2 Creation of resources on demand: Previously, we have explored situations where resources were either produced or depleted based on the environment or robot. However, resources may be generated only when and as much as needed for specific tasks. Consider $\iota^+_{q,k} \in \mathbb{H}^+$ as a resource creation on demand variable at state $q \in \mathcal{Q}$ and at time $k \in [0..K]$, where $\mathbb{H}^+ = \mathbb{B}$ if the resource is binary, $\mathbb{H}^+ = [0..C^+_{q,h,k}]$ if it is indivisible, and $\mathbb{H}^+ = [0, C^+_{q,h,k}]$ if it is divisible. The updated resource flow constraint is

$$y_{q,h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}} + \iota^+_{(q,q),h,k},$$

To avoid the unrestricted creation of resources that may lead to trivial solutions and numerical issues, we incorporate a penalty for creating on-demand resources in the objective function. The modified optimization function is

$$J = \rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v - \gamma_c \sum_q \sum_k P_{q,k} \cdot \iota^+_{q,k},$$

where $P_{q,k} > 0$ is the cost for creating a resource at state $q \in \mathcal{Q}$ at time $k \in [0..K]$, and $\gamma_c > 0$ is a scaling weight.

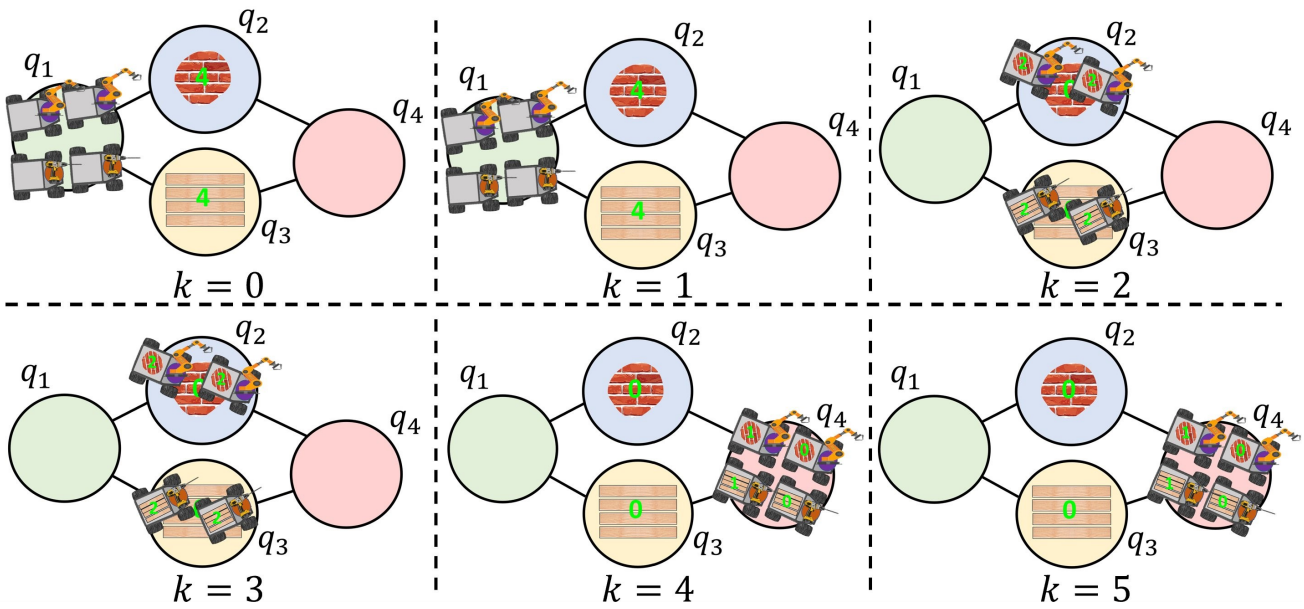

Figure 7. Solution computed for basic example in Sec. 8.1.

7.2.3 Gradual consumption of resources: In Sec. 6, we assumed that resources are consumed when the task starts. However, in some applications, this assumption may not hold. For example, resources may be consumed as they are created, which can not be accommodated with advanced reservation of resources at the beginning of tasks' satisfaction. To address this, we modify the cross-consumption constraint to enable gradual resource consumption during tasks

$$C_{q,h,k} = \sum_{T \in \Lambda(q)} \sum_{\kappa=0}^{d_T} x_{\phi_T,(k-\kappa)} \frac{rs(h)}{d_T},$$

$$\sum_{\kappa=0}^{d_T} x_{\phi_T,(k-\kappa)} \leq 1, \ \forall k \in [0, K - d_T],$$

where d_T is the task's duration, and κ is an iterator index going from the beginning of the task to its end.

8 Analysis and Results

In this section, we examine several case studies that showcase the functionality and performance of the framework. First, a simple example illustrates the fundamental functionality of the MILP encoding for CaTL planning with resources. Following this, we explore four case studies highlighting performance differences between various resource types (divisible and indivisible) and transportation storage types (compartmental and uniform) under the same specification. Additionally, we provide examples that underscore how positive values of the objective function do not guarantee specification satisfaction and showcase some robustness properties for CaTL. Finally, we analyze runtime performance for an increasing number of agents and growing specification size.

All computations for the case studies were conducted on a computer featuring 20 cores at 3.7 GHz, with 64 GB of RAM. To solve the MILP, we utilized Gurobi Gurobi Optimization (2020). For the MILP encoding of CaTL specifications we used PyTeLo Cardona et al. (2023b) and ANTLRv4 Parr (2007), LOMAP Vasile and Ulusoy (2024) and networkx Hagberg et al. (2008) for transition system models of environments.

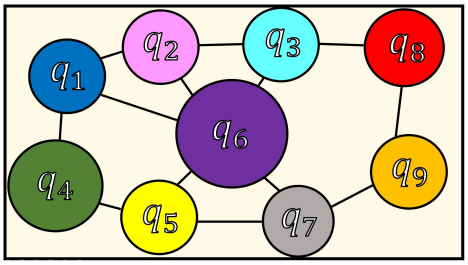

Figure 8. Abstracted labeled transition system from environment used for case studies.

8.1 Basic Example

Consider the small case study in Fig. 7. The environment has four regions with transitions between them of duration one. Two robots from each of the classes $c_1 = drilling$ and $c_2 = arm$ start at the state q_1. The scenario also includes two resources, $r_1 = bricks$ and $r_2 = wooden\,beams$, which are indivisible and initially distributed as $b_{\mathcal{H}}(0, \mathcal{Q}) = \{r_1 : \{q_2 : 4\}, r_2 : \{q_3 : 4\}\}$. The storage type is uniform, and the storage capacity $\Omega_g = 2, \ \forall g \in \mathcal{G}$. The mission specification is

$$\phi = \Diamond_{[3,4]}(T(1, \pi_{red}, (arm, 1), (brick, 3)) \wedge T(1, \pi_{red}, (drill, 1), (wooden\,beam, 3))).$$

In the given scenario, four robots travel to pick up resources from states q_2 and q_3 and then move towards q_4 to satisfy the specification. The trajectories of these robots are computed and displayed in Fig. 7. The resources are consumed upon the start of the tasks $k = 4$, and the task is fully satisfied after a one-time unit at $k = 5$. Both agents and resources have a robustness of one in this solution, as there is one additional agent and resource for each agent and resource class, respectively.

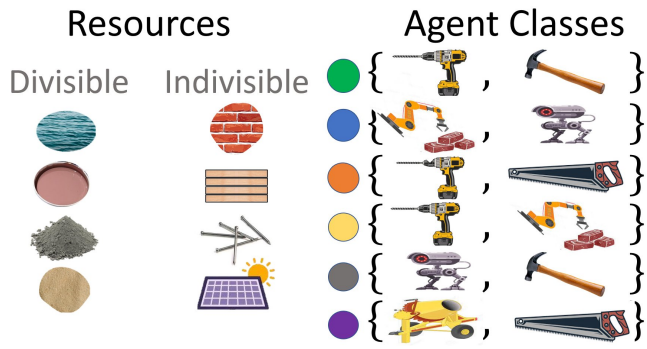

Figure 9. Resource types: divisible (water, paint, cement, sand) and indivisible (bricks, wooden beams, steel nails, solar panels). Colored circles on the right indicate a robot class with its corresponding set of capabilities.

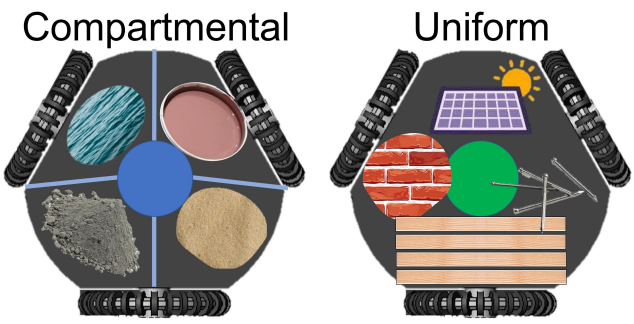

Figure 10. The transportation type examined in case studies can be compartmental (left) or uniform (right). In the former, each resource has its own compartment, while in the latter, all resources share the same storage capacity.

8.2 Resource storage and types

In this section, we show the functionality and performance of various resources and storage types described in Sec. 4. In all cases, we consider a construction environment $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W})$ illustrated in Fig. 1. The environment is abstracted as the transition system shown in Fig. 8. The set $\mathcal{H}$ contains divisible resources such as water, paint, cement, and sand and indivisible resources such as bricks, wooden beams, steel nails, and solar panels. Fig. 9 shows agent classes with their corresponding set of capabilities, such as drilling, hammering, bricklaying, sawing, monitoring, and cement mixing. Lastly, transportation storage types (i.e., compartmental and uniform) are shown in Fig. 10. We design the construction mission to be feasible and given as a CaTL specification

$$\phi = \Diamond_{I_1} T_1 \wedge \Box_{I_2} T_2 \wedge \Diamond_{I_3} T_3 \wedge \Diamond_{I_4} T_4 \wedge \Box_{I_5} T_5 \wedge \Box_{I_6} T_6, \tag{33}$$

where $I_1 = [0, 5]$, $I_2 = [10, 12]$, $I_3 = [10, 14]$, $I_4 = [10, 14]$, $I_5 = [20, 22]$, $I_6 = [20, 22]$, tasks T_i with $i \in [1..5]$ are described in Table 4. The counting resource $rs(h)$ column and its amount are not explicitly defined since they vary depending on whether the resource is divisible or indivisible, so they are described later in each case study. We consider $|\mathcal{J}| = 18$ for the mission, with agent classes shown in Fig. 9 and the following quantities per class ● = 2, ● = 4, ● = 2, ● = 2, ● = 3, ● = 5, all with initial location q_1. With all this information, we define the initial agent distribution $s_{\mathcal{J}}(0, \mathcal{Q})$.

8.2.1 Case study 1 – compartmental storage with divisible resources We consider the abstracted Env shown in Fig. 8, mission specification (33), and initial agent distribution $s_{\mathcal{J}}(0, \mathcal{Q})$ described before. The initial resource distribution is

$$b_{\mathcal{H}}(0, \mathcal{Q}) = \{r_1 : \{q_1 : 10, q_5 : 10\},\ r_2 : \{q_2 : 10, q_6 : 10\},\ r_3 : \{q_3 : 10, q_7 : 10\},\ r_4 : \{q_4 : 10, q_8 : 10\}\}, \tag{34}$$

where r_1, r_2, r_3, and r_4 correspond to *water, paint, cement,* and *sand*, respectively. The amount of resources required are $x = 1.4$ and $y = 0.7$ as indicated in Table 4. The storage capacities of each resource per agent class are

$$\Omega_{h,g} = \{ ● : \{r_1 : 4.2, r_2 : 4.3, r_3 : 4.1, r_4 : 3.2\},\ ● : \{r_1 : 2.1, r_2 : 2.2, r_3 : 2.3, r_4 : 2.4\},\ ● : \{r_1 : 2.2, r_2 : 2.3, r_3 : 2.4, r_4 : 2.1\},\ ● : \{r_1 : 2.1, r_2 : 2.2, r_3 : 2.3, r_4 : 2.2\},\ ● : \{r_1 : 2.2, r_2 : 2.1, r_3 : 2.1, r_4 : 2.2\},\ ● : \{r_1 : 2.3, r_2 : 2.2, r_3 : 2.1, r_4 : 2.3\}\}.$$

The weights in (26), (28), and (29) are $\alpha_u = 0.2$, $\alpha_v = 0.2$, and $\gamma = 0.7$, respectively. With these weights, the objective prioritizes agent robustness, followed by resource robustness. Although the normalized travel time for agents and resources are also objectives, they have less priority. The time horizon of the given specification is $\|\phi\| = K = 22$(s).

The mission specification is feasible; see Sec. 8.3 for an infeasible case. Upon computing the solution to the planning problem, the availability robustness for agents and resources are $\rho_a = 3$ and $\rho_h = 5.89$. The normalized travel time for agents and resources of $\tau_u = 0.18$, and $\tau_v = 0.25$, indicating there was more flow of resources in the environment than agents. Note that both agent and resource robustness classes are positive. Thus, the satisfaction of the mission is guaranteed by Thm. 2. Moreover, the overall optimization value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5.89 - 0.2 \cdot 0.18 - 0.2 \cdot 0.25 = 7.03$.

8.2.2 Case study 2 – compartmental storage with indivisible resources Since we consider indivisible resources, see Fig. 9, resource decision variables are integer, rather than real as in the previous case. The mission specification is defined by equation (33), where r_1 = bricks, r_2 = wooden beams, r_3 = nails, and r_4 = solar panels. For tasks in Table 4, we set $x = 1$ and $y = 2$. The initial resource distribution $b_{\mathcal{H}}(0, \mathcal{Q})$ is the same (34). The resource capacities per agent class are

Table 4. List of tasks considered for case studies comparing resources and storage types.

Name	Duration (d)	Region ($\pi \in \mathcal{AP}$)	Counting proposition ($cp(c)$)	Counting resource ($rs(h)$)
T_1	1	π_{Cyan}	$\{(drill, 2), (hammer, 2)\}$	$\{(r_1, x), (r_2, x)\}$
T_2	1	π_{Yellow}	$\{(mixer, 2), (sawing, 1)\}$	$\{(r_3, y), (r_2, x)\}$
T_3	1	π_{Purple}	$\{(drill, 2), (sawing, 1)\}$	$\{(r_2, x), (r_3, x)\}$
T_4	1	π_{Orange}	$\{(monitor, 3), (hammer, 2)\}$	$\{(r_4, y), (r_3, x)\}$
T_5	1	π_{Gray}	$\{(bricklaying, 1), (monitor, 2)\}$	$\{(r_1, y), (r_3, y)\}$
T_6	1	π_{Pink}	$\{(mixer, 2), (sawing, 1)\}$	$\{(r_1, y), (r_2, y)\}$

$$\Omega_{h,g} = \{ : \{r_1 : 4, r_2 : 4, r_3 : 4, r_4 : 3\},$$
$$: \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$: \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$: \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$: \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$: \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\}\}.$$

As both robustness classes consider integer predicates, we have integer scores of availability for agents $\rho_a = 3$ and resources $\rho_h = 5$. Thm. 2 guarantees the mission's satisfaction, as both robustness classes are positive. Thus, the overall robustness is positive. The normalized travel time for agents and resources are $\tau_u = 0.18$ and $\tau_v = 0.26$, respectively, indicating a greater resource flow in the environment. Interestingly, the values for both cases are close, which suggests that the trajectories taken by agents to satisfy the specification are similar or identical. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5 - 0.2 \cdot 0.18 - 0.2 \cdot 0.26 = 6.41$.

8.2.3 Case study 3 – uniform resource storage with divisible resource type For this case study, we consider $s_{\mathcal{J}}(0, \mathcal{Q})$, $b_{\mathcal{H}}(0, \mathcal{Q})$, resources ($r_1$, r_2, r_3, and r_4), environment Env, and mission specification ϕ as in case study 1. However, the storage type for agents is uniform. The storage capacity for each agent class is

$$\Omega_g = \{ : 8.2, : 6.1, : 6.3, : 5.4, : 4.5, : 5.6\}.$$

The robustness values for agents' and resource availability are $\rho_a = 3$ and $\rho_h = 5.89$, respectively. The latter value is not an integer, as resources are considered divisible. Since both are positive, the mission is satisfied via Thm. 1 The normalized agent and resource total travel times are $\tau_u = 0.17$ and $\tau_v = 0.24$, respectively. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5.89 - 0.2 \cdot 0.17 - 0.2 \cdot 0.24 = 7.04$.

8.2.4 Case study 4 – uniform resource storage with indivisible resource type Let us consider $s_{\mathcal{J}}(0, \mathcal{Q})$, $b_{\mathcal{H}}(0, \mathcal{Q})$, resources ($r_1$, r_2, r_3, and r_4), environment Env, and mission specification ϕ identical to case study 2. The transportation storage type is uniform (right side of Fig. 10). The storage class capacity per agent class is

$$\Omega_g = \{ : 8, : 6, : 6, : 5, : 4, : 5\}.$$

The robustness values for the availability of agents and resources are $\rho_a = 3$ and $\rho_h = 5$. The normalized travel time of agents and resources are $\tau_u = 0.14$ and $\tau_v = 0.25$. Note that the travel of agents score is slightly lower than in the previous case since using a uniform storage type provides a higher capacity to transport a specific resource, requiring fewer agents to move in the environment. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 1 + 0.7 \cdot 8 + 0.3 \cdot 0.918 + 0.3 \cdot 3.301 = 7.8657$ Similar to previous cases, both robustness class scores are positive, ensuring the satisfaction of the mission specification.

Time performance comparison: Table 5 contains the information on the time performance, objective value, and continuous, integer, and binary variables of the four case studies. All of the objective values are similar. However, their time performance differs due to the varying number of variables required to capture the problem in a MILP. Although continuous variables can affect the time performance, it is usually mainly influenced by the integer and binary variables. Notably, solving the fourth case study with more integer and binary variables takes the longest. As four cases work with the same specification, number of agents, and resources, it is plausible to conclude that a problem with the compartmental storage type is faster than the uniform storage type and that divisible resources yield better runtimes than indivisible resources. However, the quantity of integer and binary variables depends on the specifications, environment size, number of agent classes, number of resource types, and storage type. For further examples of increasing specification complexity and the number of agents, see Sec. 8.4.

8.3 Robustness and satisfaction comparison for CaTL

Here, we delve into a discussion on how robustness scores of both the agents and resources impact the fulfillment of the mission specification as a whole. To this end, we use the setup from case study 1, wherein we modify task T_6 so that instead of requesting two mixers, we now require six units. The solution for this case study differs slightly from the one in case study 1. The main difference is that in this case, the agent availability robustness score is $\rho_a = -1$, indicating that we do not have enough agents with the necessary capabilities to complete the mission. However, the overall objective function takes a value of $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3.037$, which does not guarantee the satisfaction of the mission even though it is positive. Instead, to guarantee satisfaction, we make use of Cor. 1, which implies that if $\rho_a(s_{\mathcal{J}}, \phi, 0) > 0 \wedge \rho_h(b_{\mathcal{H}}, \phi, 0) > 0 \Rightarrow \rho((s_{\mathcal{J}}, b_{\mathcal{H}}), \phi, 0) > 0$. Hence, using Thm. 1, we can infer that mission satisfaction is guaranteed only if both the availability robustness of agents and resources are positive.

Table 5. Number of continuous, integer, and binary variables, runtime, and objective values for the four case studies.

Case study	Time	Objective Values	Continuous Vars.	Integer Vars.	Binary Vars.
1	12.05(s)	7.03	2859	5281	115
2	26.39(s)	6.41	64	8083	119
3	28.86(s)	7.04	2859	5274	115
4	45.18(s)	7.86	68	8375	248

Although we may encounter situations where the specifications suggest an absence of agents, our MILP encoding still produces a solution that violates the specification to a minimum extent. In the case of the resources, this is impossible since, by definition, the resource variables in Sec. 6.2 are defined as positive variables. The rationale behind this approach is to simplify the establishment of cross-consumption and resource dynamics constraints. In situations with a shortage of resources, the problem becomes infeasible by definition, which could be addressed by introducing a slack variable. However, this would necessitate more complex bookkeeping. All things considered, satisfaction with the CaTL mission relies on the availability of agents to fulfill specifications and feasible trajectories for agents to pick and drop resources.

8.4 Runtime performance comparison

This section presents a runtime performance comparison of four different types of problems, namely, compartmental storage with divisible and indivisible resources, uniform storage with divisible and indivisible resources, and the CaTL baseline Leahy et al. (2021). The evaluation consists of two experiments: one increases the number of agents, and the other grows the specification size. For the first case, we consider the Env shown in Fig. 1. We use the same number of robot classes and divisible resource types shown in Fig. 9, compartmental resource storage type, and storage capacities defined for case study 1. The mission specification considered is $\phi = \Diamond_{[0,4]} T_1 \wedge \Box_{[18,22]}(T_3 \wedge T_4)$, where T_1, T_3 and T_4 are given in Table 4. Resource distribution is changed such that we have 2 units of each resource in q_2 and q_4. Five agents are added per iteration with random robot classes, and all start at state q_1. The results are presented in Table 6. Note that increasing the number of agents reduces the time for the four types of problems. Most of them compute a solution in around 0.7 seconds, which is still slightly longer than the CaTL baseline, which is expected since the latter does not plan for resources.

In the second scenario, we use the same initial conditions. However, rather than increasing the number of agents, we spawn 18 agents that are randomly generated and proceed to augment the specification in the following manner

$$\phi = \bigwedge_{n=1}^{N} \mathcal{X}_{I_n} \mathfrak{T},$$

where N is increased by two in every step. The temporal operators $\mathcal{X} \in \{\Box, \Diamond\}$ are chosen randomly with time intervals $I_n = [5(n-1), 5(n-1) + \text{rand}(1,5)]$. Tasks are randomly taken from Table 4 such that $\mathfrak{T} \in \{T_1, \ldots, T_6\}$. Table 7 shows the computation time for a solution when growing the specification size for the four cases. Note that the case of uniform storage and indivisible resources type takes the longest. This is mainly because finding the optimal combination of resources that each robot should carry to maximize the robustness scores becomes a combinatorial problem when considering integer values of resources. In general, computing a solution for compartmental is faster than uniform storage types, and divisible resources are faster than indivisible.

9 Conclusions and Future Work

This paper presents an extension of CaTL that considers agent capabilities and resource constraints. The extension is achieved by enhancing the MILP encoding to capture different types of resources, including divisible and indivisible, and accommodating different agent storage types for resources, such as compartmental and uniform. Moreover, an encoding is developed to extend STL to accept multiple predicate classes and compute robustness separately. The paper demonstrates the effectiveness of computing the multi-robustness score for an STL specification with multiple predicate classes and explores how satisfaction is related to the robustness scores. The proposed framework is applied to CaTL to capture agent and resource robustness scores. When these scores are positive, they guarantee satisfaction with the mission specification. Furthermore, the paper comprehensively evaluates the framework's runtime performance when considering uniform or compartmental storage types with divisible or indivisible resources. In future work, we will consider performing mission specifications on real robots and explore uncertainty in the availability of resources and agents in the environment.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by MIT Lincoln Laboratory under Inter and Intra Team Coordination from High Level Specifications program.

A Proof of Theorem 2

Before we start the main argument of the proof, we show a general lemma on the monotonicity of evaluating mcSTL* formulas. For this, we define an *evaluation tree* based on the AST. Formally, an evaluation tree (ET) is a tuple $\mathcal{T} = (op, Ch = \{\mathcal{T}_{ch}\}_{ch})$, where $op \in \{\min, \max, \text{nop}\}$, nop denotes no operation, and Ch is a list of children's evaluation trees. A leaf node is one where $Ch = \varnothing$ and $op = \text{nop}$, while

Table 6. Runtime performance comparison for an incrementing number of agents.

Agents	Compartmental		Uniform		CaTL Baseline Jones et al. (2019)
	Divisible	Indivisible	Divisible	Indivisible	(Resources not considered)
10	1.32 (s)	2.73 (s)	0.96 (s)	1.12 (s)	0.22 (s)
15	2.18 (s)	2.01 (s)	1.90 (s)	3.09 (s)	0.35 (s)
20	0.96 (s)	2.26 (s)	1.18 (s)	0.68 (s)	0.33 (s)
25	0.76 (s)	1.87 (s)	0.71 (s)	0.76 (s)	0.33 (s)
30	0.79 (s)	0.84 (s)	0.73 (s)	0.70 (s)	0.32 (s)

Table 7. Runtime performance comparison for an incrementing specification ϕ.

ϕ	Compartmental		Uniform		CaTL Baseline Jones et al. (2019)
	Divisible	Indivisible	Divisible	Indivisible	(Resources not considered)
2	0.90 (s)	0.79 (s)	1.02 (s)	0.70 (s)	0.26 (s)
4	1.87 (s)	1.75 (s)	3.79 (s)	1.50 (s)	0.35 (s)
6	3.13 (s)	7.06 (s)	2.47 (s)	8.35 (s)	0.63 (s)
8	4.12 (s)	9.23 (s)	17.50 (s)	12.38 (s)	0.68 (s)
10	12.32 (s)	9.39 (s)	17.96 (s)	61.11 (s)	1.17 (s)

an intermediate node has children ETs ($Ch \neq \varnothing$) and $op \in \{\min, \max\}$. Let $\mathcal{S}(\mathcal{T})$ denote the list of leafs of $\mathcal{T}$.

Given an AST of a mcSTL* formula ψ, we construct the ET $\mathcal{T}$ as follows: (a) a terminal conjunction node, which has only predicates as operands, is translated to a leaf node, (b) non-terminal conjunction and disjunction nodes are translated to intermediate nodes with children ETs corresponding to its operands and $op = \min$ and $op = \max$, respectively, (c) always and eventually, nodes are translated to nodes with children for each time point in their associated time window and $op = \min$ and $op = \max$, respectively, and (d) until operators are translated as a combination of the previous two rules. Essentially, ETs capture the recursive evaluation process of the robustness scores in (3) and Def. 6.

Let $\nu : \mathcal{S}(\mathcal{T}) \to \mathbb{R}$ be an valuation for the leafs of ET $\mathcal{T}$. We define the tree evaluation $\eta(\mathcal{T}, \nu)$ by

$$\eta(\mathcal{T}, \nu) = \begin{cases} op(\mathcal{T}_1, \dots, \mathcal{T}_{|Ch|}), & \mathcal{T}_\ell \in Ch \neq \varnothing, \\ \nu(\mathcal{T}), & \text{otherwise.} \end{cases} \tag{35}$$

Lemma 1. *Let $\mathcal{T}$ be an ET and ν and ν' two leaf valuations. If $\nu'(\varsigma) \geq \nu(\varsigma)$ for all $\varsigma \in \mathcal{S}(\mathcal{T})$, then $\eta(\mathcal{T}, \nu') \geq \eta(\mathcal{T}, \nu)$.*

Proof. The proof follows by structural induction over ET $\mathcal{T}$.
Base case: Let $\mathcal{T} = (\text{nop}, \varnothing)$ be a leaf node. We have $\eta(\mathcal{T}, \nu') = \nu'(\mathcal{T}) \geq \nu(\mathcal{T}) = \eta(\mathcal{T}, \nu)$.
Induction step: Let $\mathcal{T} = (op, Ch)$ be a tree with children, i.e., $Ch \neq \varnothing$. In case $op = \min$, we have that $\eta(\mathcal{T}, \nu) = \min_{\mathcal{T}_{ch} \in Ch} \{\eta(\mathcal{T}_{ch}, \nu)\}$. By the induction hypothesis, it follows that $\eta(\mathcal{T}_{ch}, \nu') \geq \eta(\mathcal{T}_{ch}, \nu)$ for all $\mathcal{T}_{ch} \in Ch$. Finally, we have $\eta(\mathcal{T}, \nu') = \min_{\mathcal{T}_{ch} \in Ch} \{\eta(\mathcal{T}_{ch}, \nu')\} \geq \min_{\mathcal{T}_{ch} \in Ch} \{\eta(\mathcal{T}_{ch}, \nu)\} = \eta(\mathcal{T}, \nu)$. The case $op = \max$ follows similarly.

Proof of Theorem 2.

Proof. Let $\mathcal{T}_\phi$ be the ET of mcSTL* formula ϕ, and s be a signal such that $\rho(s, \phi, t) > 0$.

We define ET valuation $\nu_\phi : \mathcal{S}(\mathcal{T}_\phi) \to \mathbb{R}$ such that $\nu_\phi(\mathcal{T}) = \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T})$, where $(t_\mathcal{T}, \phi_\mathcal{T})$ is the time-formula pair associated with ET leaf node $\mathcal{T}$. By construction, each $\mathcal{T} \in \mathcal{S}(\phi)$ corresponds to a terminal conjunction whose operands are predicates covering all predicate classes Σ. Thus, the tree evaluation captures the robustness computation for signal s with respect to ϕ at time t. Formally $\eta(\mathcal{T}_\phi, \nu) = \rho(s, \phi, t)$. Similarly, we define ET valuation $\nu_{\phi,\sigma} : \mathcal{S}(\mathcal{T}_\phi) \to \mathbb{R}$ such that $\nu_\phi(\mathcal{T}) = \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho_\sigma(s, \varsigma, t_\mathcal{T})$. The tree evaluation using $\nu_{\phi,\sigma}$ captures the robustness score with respect to predicate class $\sigma \in \Sigma$. Formally, $\eta(\mathcal{T}_\phi, \nu_{\phi,\sigma}) = \rho_\sigma(s, \phi, t), \forall \sigma \in \Sigma$.

Since all terminal conjunctions have predicates from all classes as operands, their robustness with respect to any class is defined, i.e., not equal to $\varnothing$. It follows that $\nu_{\phi,\sigma}(\mathcal{T}) = \min_{\varsigma \in ch^\sigma_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T}) \geq \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T}) = \nu_\phi(\mathcal{T})$ since $ch^\sigma_\phi(\mathcal{T}) \subset ch_\phi(\mathcal{T})$ for all $\mathcal{T} \in \mathcal{S}(\mathcal{T}_\phi)$ and $\sigma \in \Sigma$, where $ch^\sigma_\phi(\mathcal{T}) = \{\varsigma \in ch_\phi(\mathcal{T}) \mid \mathcal{L}(\varsigma) = \sigma\}$. Thus, by Lemma 1 we have $\rho_\sigma(s, \phi, t) = \eta(\mathcal{T}_\phi, \nu_{\phi,\sigma}) \geq \eta(\mathcal{T}_\phi, \nu_\phi) = \rho(s, \phi, t)$ which concludes the proof.

References

Badithela A, Graebener JB, Ubellacker W, Mazumdar EV, Ames AD and Murray RM (2023) Synthesizing reactive test environments for autonomous systems: testing reach-avoid specifications with multi-commodity flows. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 12430–12436.

Baier C and Katoen JP (2008) *Principles of model checking*. MIT press.

Belta C, Yordanov B and Gol EA (2017) *Formal methods for discrete-time dynamical systems*, volume 89. Springer.

Berman S, Halász A, Hsieh MA and Kumar V (2009) Optimized stochastic policies for task allocation in swarms of robots. *IEEE transactions on robotics* 25(4): 927–937.

Bhatia A, Kavraki LE and Vardi MY (2010) Sampling-based motion planning with temporal goals. In: *2010 International Conference on Robotics and Automation*. IEEE, pp. 2689–2696.

Birk A and Carpin S (2006) Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE* 94(7): 1384–1397.

Bresina JL, Jónsson AK, Morris PH and Rajan K (2005) Activity planning for the mars exploration rovers. In: *International Conference on International Conference on Automated Planning and Scheduling*. pp. 40–49.

Brzoska C (1998) Programming in metric temporal logic. *Theoretical computer science* 202(1-2): 55–125.

Buyukkocak AT and Aksaray D (2022) Temporal relaxation of signal temporal logic specifications for resilient control synthesis. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, pp. 2890–2896.

Caballero A and Silano G (2023) A signal temporal logic motion planner for bird diverter installation tasks with multi-robot aerial systems. *IEEE Access* .

Cai M, Leahy K, Serlin Z and Vasile CI (2021) Probabilistic coordination of heterogeneous teams from capability temporal logic specifications. *IEEE Robotics and Automation Letters* 7(2): 1190–1197.

Cardona GA and Calderon JM (2019) Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations. *Applied Sciences* 9(8): 1702.

Cardona GA, Kamale D and Vasile CI (2023a) Mixed integer linear programming approach for control synthesis with weighted signal temporal logic. In: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*. pp. 1–12.

Cardona GA, Leahy K, Mann M and Vasile CI (2023b) A flexible and efficient temporal logic tool for python: Pytelo. *arXiv preprint arXiv:2310.08714* .

Cardona GA, Leahy K and Vasile CI (2023c) Temporal logic swarm control with splitting and merging. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 12423–12429.

Cardona GA, Ramirez-Rugeles J, Mojica-Nava E and Calderon JM (2021) Visual victim detection and quadrotor-swarm coordination control in search and rescue environment. *International Journal of Electrical and Computer Engineering* 11(3): 2079.

Cardona GA, Saldaña D and Vasile CI (2022) Planning for modular aerial robotic tools with temporal logic constraints. In: *2022 Conference on Decision and Control (CDC)*. IEEE, pp. 2878–2883.

Cardona GA and Vasile CI (2022) Partial satisfaction of signal temporal logic specifications for coordination of multi-robot systems. In: *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, pp. 223–238.

Cardona GA and Vasile CI (2023) Preferences on partial satisfaction using weighted signal temporal logic specifications. In: *2023 European Control Conference (ECC)*. IEEE, pp. 1–6.

Chen Y, Ding XC, Stefanescu A and Belta C (2011) Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics* 28(1): 158–171.

Choset H (2000) Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots* 9: 247–253.

Diaz-Mercado Y, Jones A, Belta C and Egerstedt M (2015) Correct-by-construction control synthesis for multi-robot mixing. In: *Conference on Decision and Control*. IEEE, pp. 221–226.

Dixit DSK and Dhayagonde MS (2014) Design and implementation of e-surveillance robot for video monitoring and living body detection. *International Journal of Scientific and Research Publications* 4(4): 2250–3153.

Dokhanchi A, Hoxha B and Fainekos G (2014) *International Conference on Runtime Verification, Toronto, ON, Canada.*, chapter On-Line Monitoring for Temporal Logic Robustness. Springer. ISBN 978-3-319-11164-3, pp. 231–246.

Donzé A and Maler O (2010) Robust satisfaction of temporal logic over real-valued signals. In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, pp. 92–106.

Fainekos GE and Pappas GJ (2009) Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 410(42): 4262–4291.

Finucane C, Jing G and Kress-Gazit H (2010) Ltlmop: Experimenting with language, temporal logic and robot control. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1988–1993.

Fu JGM, Bandyopadhyay T and Ang MH (2009) Local voronoi decomposition for multi-agent task allocation. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1935–1940.

Guo M and Dimarogonas D (2015) Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research* 34(2): 218–235.

Guo M and Dimarogonas DV (2016) Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Transactions on Automation Science and Engineering* 14(2): 797–808.

Guo M and Dimarogonas DV (2017) Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Transactions on Automation Science and Engineering* 14(2): 797–808.

Guo M and Zavlanos MM (2017) Distributed data gathering with buffer constraints and intermittent communication. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 279–284.

Gurobi Optimization L (2020) Gurobi optimizer reference manual. URL http://www.gurobi.com.

Hagberg AA, Schult DA and Swart PJ (2008) Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T and Millman J (eds.) *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, pp. 11 – 15.

Hopcroft JE, Motwani R and Ullman JD (2001) Introduction to automata theory, languages, and computation. *Acm Sigact News* 32(1): 60–65.

Hustiu S, Dimarogonas DV, Mahulea C and Kloetzer M (2023) Multi-robot motion planning under mitl specifications based on time petri nets. In: *2023 European Control Conference (ECC)*. IEEE, pp. 1–8.

Jones A, Leahy K, Vasile C, Sadraddini S, Serlin Z, Tron R and Belta C (2019) Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications. In: *International Symposium of Robotics Research*. pp. 224–241.

Kamale D, Karyofylli E and Vasile CI (2021) Automata-based optimal planning with relaxed specifications. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6525–6530.

Kantaros Y, Guo M and Zavlanos M (2019) Temporal logic task planning and intermittent connectivity control of mobile robot networks. *IEEE Transactions on Automatic Control* 64(10): 4105–4120.

Karaman S and Frazzoli E (2011) Linear temporal logic vehicle routing with applications to multi-uav mission planning. *International Journal of Robust and Nonlinear Control* 21(12): 1372–1395.

Kloetzer M and Mahulea C (2014) A petri net based approach for multi-robot path planning. *Discrete Event Dynamic Systems* 24: 417–445.

Kurtz V and Lin H (2022) Mixed-integer programming for signal temporal logic with fewer binary variables. *IEEE Control Systems Letters* 6: 2635–2640.

Lacerda B and Lima PU (2019) Petri net based multi-robot task coordination from temporal logic specifications. *Robotics and Autonomous Systems* 122: 103289.

Leahy K, Jones A, Schwager M and Belta C (2015) Distributed information gathering policies under temporal logic constraints. In: *2015 IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 6803–6808.

Leahy K, Serlin Z, Vasile CI, Schoer A, Jones AM, Tron R and Belta C (2021) Scalable and robust algorithms for task-based coordination from high-level specifications (scratches). *IEEE Transactions on Robotics* .

Liu W, Leahy K, Serlin Z and Belta C (2023) Robust multi-agent coordination from catl+ specifications. In: *2023 American Control Conference (ACC)*. IEEE, pp. 3529–3534.

Liu Z, Dai J, Wu B and Lin H (2017) Communication-aware motion planning for multi-agent systems from signal temporal logic specifications. In: *2017 American Control Conference (ACC)*. IEEE, pp. 2516–2521.

Madridano A, Al-Kaff A, Martín D and De La Escalera A (2021) Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications* 173: 114660.

Mahajan A (2010) Presolving mixed-integer linear programs. *Wiley Encyclopedia of Operations Research and Management Science* : 4141–4149.

Maler O and Nickovic D (2004) Monitoring temporal properties of continuous signals. In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, pp. 152–166.

Mehdipour N, Vasile CI and Belta C (2019) Arithmetic-geometric mean robustness for control from signal temporal logic specifications. In: *2019 American Control Conference (ACC)*. IEEE, pp. 1690–1695.

Mehdipour N, Vasile CI and Belta C (2020) Specifying user preferences using weighted signal temporal logic. *IEEE Control Systems Letters* 5(6): 2006–2011.

Nikou A, Tumova J and Dimarogonas DV (2016) Cooperative task planning of multi-agent systems under timed temporal specifications. In: *2016 American Control Conference (ACC)*. IEEE, pp. 7104–7109.

Notomista G, Mayya S, Hutchinson S and Egerstedt M (2019) An optimal task allocation strategy for heterogeneous multi-robot systems. In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 2071–2076.

Pant YV, Abbas H, Quaye RA and Mangharam R (2018) Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In: *International Conference on Cyber-Physical Systems*. IEEE, pp. 186–197.

Parr T (2007) *The definitive ANTLR reference: building domain-specific languages*. Pragmatic Bookshelf.

Raman V, Donzé A, Maasoumy M, Murray RM, Sangiovanni-Vincentelli A and Seshia SA (2014) Model predictive control with signal temporal logic specifications. In: *2014 Annual Conference on Decision and Control (CDC)*. IEEE, pp. 81–87.

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

Sahin YE, Nilsson P and Ozay N (2017) Provably-correct coordination of large collections of agents with counting temporal logic constraints. In: *2017 International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, pp. 249–258.

Sahin YE, Nilsson P and Ozay N (2019) Multirobot coordination with counting temporal logics. *IEEE Transactions on Robotics* 36(4): 1189–1206.

Schillinger P, Bürger M and Dimarogonas DV (2018) Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The international journal of robotics research* 37(7): 818–838.

Sewlia M, Verginis CK and Dimarogonas DV (2023) Maps2: Multi-robot anytime motion planning under signal temporal logic specifications. *arXiv preprint arXiv:2309.05632* .

Sun D, Chen J, Mitra S and Fan C (2022) Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters* 7(2): 3451–3458.

Sundram J, Van Nguyen D, Soh GS, Bouffanais R and Wood K (2018) Development of a miniature robot for multi-robot occupancy grid mapping. In: *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, pp. 414–419.

Tkachev I and Abate A (2013) Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems. In: *Proceedings of the international conference on Hybrid systems: computation and control*. pp. 283–292.

Tripicchio P, Satler M, Dabisias G, Ruffaldi E and Avizzano CA (2015) Towards smart farming and sustainable agriculture with drones. In: *International Conference on Intelligent Environments*. IEEE, pp. 140–143.

Tumova J and Dimarogonas DV (2016) Multi-agent planning under local ltl specifications and event-based synchronization. *Automatica* 70: 239–248.

Ulusoy A, Smith SL, Ding XC, Belta C and Rus D (2011) Optimal multi-robot path planning with temporal logic constraints. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, pp. 3087–3092.

Ulusoy A, Smith SL, Ding XC, Belta C and Rus D (2013) Optimality and robustness in multi-robot path planning with temporal logic constraints. *The International Journal of Robotics Research* 32(8): 889–911.

Vasile CI, Aksaray D and Belta C (2017) Time window temporal logic. *Theoretical Computer Science* 691: 27–54.

Vasile CI and Ulusoy A (2024) Ltl optimal multi-agent planner (lomap). https://github.com/wasserfeder/lomap.

Xu Z and Julius AA (2016) Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Transactions on Automation Science and Engineering* 15(1): 264–277.

Yu P and Dimarogonas DV (2021) Distributed motion coordination for multirobot systems under ltl specifications. *IEEE Transactions on Robotics* 38(2): 1047–1062.

Yu X, Yin X and Lindemann L (2023) Efficient stl control synthesis under asynchronous temporal robustness constraints. *arXiv preprint arXiv:2307.12855* .