Planning for Modular Aerial Robotic Tools with Temporal Logic Constraints

Gustavo A. Cardona, David Saldaña, and Cristian-Ioan Vasile

Abstract— **Modular robots are highly versatile due to their ability to reconfigure and change their mechanical properties. This ability make them optimal for scenarios that require different types of tasks. However, when the number of modules increases, the task allocation and cooperation become a challenging combinatorial problem. To tackle this problem, we propose a high-level planner for reconfigurable robots with heterogeneous capabilities, e.g., aerial motion and tool operation. Modules can attach and detach to create configurations that manipulate tools satisfying temporal and logic-constrained tasks. The mission is specified using Metric Temporal Logic (MTL) which offers the capacity to not only account for where and who needs to satisfy a task but also when and for how long. We model the problem using a Mixed Integer Linear Problem (MILP) approach, capturing cost for reconfiguration, satisfying a task, and motion in the environment in a specific configuration. Additionally, we consider that not all configurations can satisfy every task. We find trajectories for modular robots that guarantee mission satisfaction. Finally, we show performance and results by performing simulations with multiple tasks and requirements in an environment.**

I. Introduction

In recent years, the development of technology and the increase in computational capacity have made the implementation of robot swarms and multi-robot systems possible. Multi-robot systems have been widely studied for their capacity to handle multiple tasks simultaneously, robustness, and resiliency in overcoming failures or dropouts while guaranteeing task satisfaction. This type of system is a suitable platform for tasks ranging from perimeter surveillance, search and rescue missions, cargo delivery, and planetary exploration [1]–[4]. Nevertheless, for tasks that require more capabilities, such as versatile locomotion and manipulation, modular robots provide the ability to reconfigure and change both force capacity in the environment and control constraints [5].

Modular and reconfigurable robotics offers to go from one configuration to another, offering flexibility to address a broader set of tasks than a single robot of similar complexity. Here configuration is understood not only as the pose of the robot but also the module connectivity and shape [5]. Changing their capacities and the possible redundancy in the degree of freedom make modular robots versatile and robust [6]. However, there is an increase in the computational cost

Gustavo A. Cardona, David Saldaña, and Cristian-Ioan Vasile are with the Autonomous and Intelligent Robotics Laboratory –AIRLab– at Lehigh University, Bethlehem, PA, 18015, USA. {gcardona, saldana, cvasile}@lehigh.edu

The authors want to thank Diego S. D'Antonio for providing some of the illustrative figures for this paper.

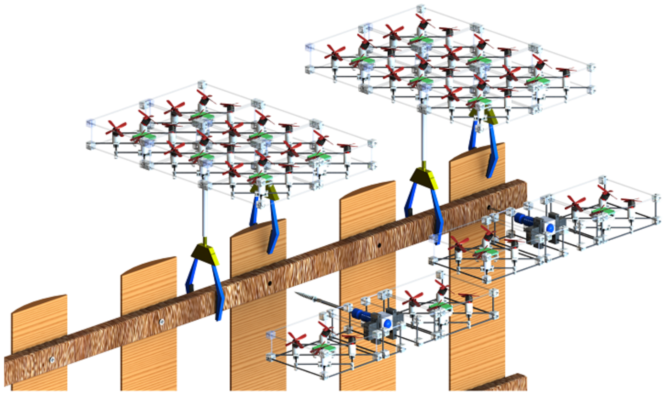

Fig. 1. Planning for multiple modular robots that can reach configurations to manipulate tools such as a gripper to grasp wood and screwdrivers to build a wooden fence.

when coordinating the robots in both low-level and high-level control. On one hand, low-level control is required to guarantee the collision-free rearrangement of modular robots to reach the desired configuration or architecture (lattices, chains, mobile swarm) and act [7]–[10]. On the other hand, the high-level control takes a set of real-world tasks and generates or matches existing configurations to satisfy the mission specification [11].

This work focuses on high-level planning coordinating multiple robots to satisfy tasks in an environment. Multiple works have tackled this problem of coordinating and allocating a set of robots to satisfy tasks from a deterministic and stochastic approach [12], [13]. However, they consider agents homogeneous and do not consider temporal logic constraints in the specifications. Several other works have used Temporal logic for heterogeneous multi-robot systems working with Linear Temporal Logic (LTL) with automata theory [14], [15] or Signal Temporal Logic [16], [17]. Nevertheless, these papers do not consider modular robots, which adds the problem of not only allocating some robots but finding proper configurations that satisfy a particular task.

Our work is focused on generating an automatic controller synthesis for planning modules trajectories that allows the creation of modular robot configurations that satisfy task specifications using formal languages, specifically Metric Temporal Logic (MTL). Few other works have addressed modular robots using Temporal Logic approaches such as [18], [19]. Nonetheless, authors consider LTL and encode the system into an automaton which might be computationally expensive when dealing with scalability. Instead, we consider MTL a rich temporal logic formalism. We propose a MILP

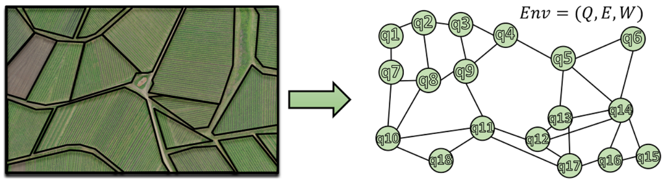

Fig. 2. Example of a tessellated agriculture environment (left) which can be abstracted into a transition system (right).

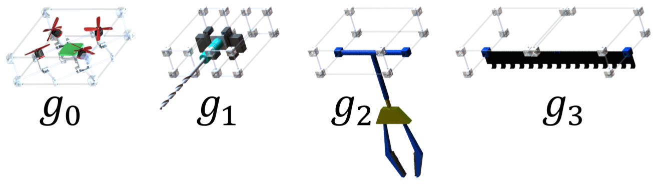

Fig. 3. Example of module capabilities, form left to right g_{fly}, $g_{screwdriver}$, $g_{grasping}$, g_{sawing}.

approach to efficiently solve the problem while accounting for the cost of reconfiguration, motion, and satisfying a task in a specific configuration.

The contributions of the paper are

1) We propose and formalize a planning problem for modular aerial robots with heterogeneous capabilities and configurations tasked with performing timed temporal logic missions. The missions involve tasks that can be performed only by a subset of robot configurations, and may require robots to reconfigure during the mission.
2) We propose an efficient Mixed Integer Linear Programming (MILP) approach that minimizes the total energy consumption while satisfying the MTL mission specification, robot motion, and reconfiguration constraints.
3) We show the performance of the proposed MILP method in two case studies.

II. Preliminaries and Notation

Let $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of integer and real numbers, respectively. The set of integers greater than a is denoted by $\mathbb{Z}_{\geq a}$. The set of binary values is $\mathbb{B} = \{0, 1\}$. For a set S, 2^S and $|S|$ denote its power set and cardinality. For $S \subseteq \mathbb{R}$ and $t \in \mathbb{R}$, we have $t + S = \{t + x \mid x \in S\}$. The integer interval (range) from a to b is $[a \,..\, b]$. The empty set is denoted by $\varnothing$, while missing or undefined values are marked with $\varnothing$. Let $x \in \mathbb{R}^d$ be a d-dimensional vector. The i-th component of x is given by x_i, for $i \in [1 \,..\, d]$.

A. Metric Temporal Logic

Metric Temporal Logic (MTL), as introduced in [20], is a specification language expressing real-time properties. The syntax of MTL is

$$\phi ::= \top \mid \neg\phi \mid \pi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Box_I \phi \mid \Diamond_I \mid \phi_1 \mathcal{U}_I \, \phi_2,$$

where ϕ, ϕ_1, and ϕ_2 are MTL formulae, $\top$ is the logical *True* value, $\pi \in \Pi$ is an atomic proposition. over the i-th component of signal s, $\neg$, $\vee$, and $\wedge$ are the Boolean negation, disjunction, and conjunction operators, and $\Box_I$, $\Diamond_I$ $\mathcal{U}_I$ are the timed *always*, *eventually*, and *until* operator with $I = [t_1 \,..\, t_2]$ a discrete-time interval, $0 \leq t_1 \leq t_2$. The logical *False* value is $\bot = \neg\top$.

The semantics of MTL formulae ϕ at time t is recursively defined over timed state sequence $w = (s, t)$ where $s = s_0, s_1 \ldots s_p$ is a sequence of atomic prepositions by [21] as

$$\begin{aligned}
w \vDash \pi &\equiv s_o \vDash \pi,\\
w \vDash \neg\phi &\equiv w \nvDash \phi,\\
w \vDash \phi_1 \wedge \phi_2 &\equiv w \vDash \phi_1 \wedge w \vDash \phi_2,\\
w \vDash \phi_1 \vee \phi_2 &\equiv w \vDash \phi_1 \vee w \vDash \phi_2,\\
w \vDash \Diamond_I \phi &\equiv \exists t \in I, (s, t) \vDash \phi,\\
w \vDash \Box_I \phi &\equiv \forall t \in I, (s, t) \vDash \phi,\\
w \vDash \phi_1 \mathcal{U}_I \, \phi_2 &\equiv \exists t' \in t + I \text{ s.t. } (s, t') \vDash \phi_2\\
&\quad \wedge \forall t'' \in [t \,..\, t'] \; (s, t'') \vDash \phi_1,
\end{aligned} \tag{1}$$

where $\vDash$ and $\nvDash$ denote satisfaction and violation, respectively. A timed state sequence $w = (s, t)$ that satisfies ϕ is denoted by $w \vDash \phi$, and it is true if $(s, 0) \vDash \phi$.

B. Time Horizon of MTL formula

The time horizon of an MTL formula [22] is defined as

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = \pi,\\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1,\\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \wedge \phi_2,\\ b + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \mathcal{U}_{[a..b]} \, \phi_2. \end{cases} \tag{2}$$

An MTL formula is said to be in *positive normal form* (PNF) if it it does not contain the negation operator.

III. Problem Formulation

In this section, we introduce the planning problem for teams of heterogeneous modular robots tasked with rich temporal logic tasks that require specialized tools such as driller, screwdriver, saw, grasping. Robots are able to reconfigure during the mission to perform the various tasks of the specification. Thus, the number and configurations of modular robots active in the mission space may be time-varying during the mission. We introduce the models for the environment, modules, configurations, robots, and tasks that define the planning problem for modular aerial robots.

In this work, we are motivated by construction problems for agriculture. For instance, consider the construction of a wooden fence as shown in Fig. 1. The task is described in the following Example 1.

Example 1. *Fence construction on crop q_1:*

1) *Always from deployment to the end of the mission, region q_1 must be video-monitored;*
2) *Within 10 minutes after deployment, the wood must be transported in q_1;*

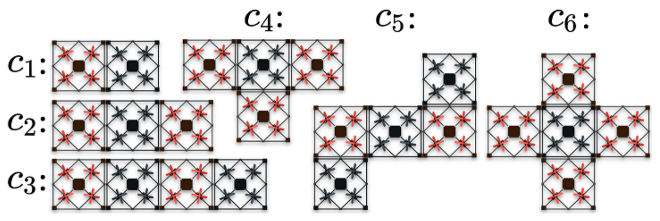

Fig. 4. Examples of configurations of modules attached in different arrangements.

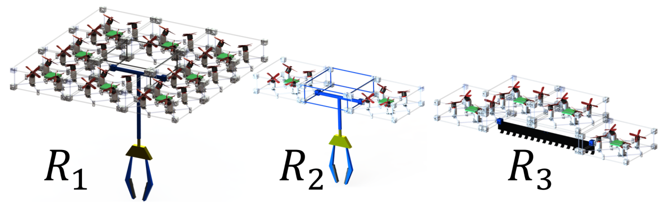

Fig. 5. Examples of modular robots are composed of a set of modules with configurations that can manipulate tools such as grippers and saws.

3) *Within 11 to 60 minutes after deployment, the wood must be cut into smaller pieces at q_1;*
4) *Always within 60 to 90 minutes the fence needs to be assembled at q_1.*

First, we define the *environment* that captures the motion of robots between locations of interest.

Definition 1 (Environment)**.** *The environment is a weighted transition system defined by the tuple $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W}, \Pi, \mathcal{L})$, 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 transitions of robots between locations in $\mathcal{Q}$, $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ maps each transition in $\mathcal{E}$ to its travel duration, Π is a set of atomic propositions that label the states $\mathcal{Q}$, and $\mathcal{L} : \mathcal{Q} \to 2^{\Pi}$ is the state-labeling function. A robot 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 $\mathcal{W}((q, q)) = 1$.*

In Fig. 2, we show an agricultural environment tessellated into different regions, which generates the abstracted environment Env describing the states $\mathcal{Q}$ and edges $\mathcal{E}$.

Extending from Example 1, we consider modular robot composed of aerial motion capability carrying and manipulating modular tools.

Definition 2 (Module)**.** *A module is a cuboid with a capability g interacting in the environment.*

The set of all capabilities is $\mathcal{G} = \{g_0, g_1, g_2, g_3, \dots g_m\}$. At least one capability is the ability to fly, i.e., generate thrust. The capability of a module also defines the module's class.

In Example 1, we have the set of module capabilities $\mathcal{G} = \{g_{fly}, g_{screwing}, g_{drilling}, g_{grasp}, g_{sawing}, g_{monitoring}\}$. Some examples of these modules are shown in Fig. 3.

Note that we do not differentiate between modules of the same class. Additionally, modules from the same or different classes can attach and detach with each other to form modular robots. Configurations describe the types, numbers and interconnections of modules that robots are composed of. Formally, we have the following definition.

Definition 3 (Configuration)**.** *A robot configuration $c = (M_c, \to_c, Cap_c)$ is a labeled graph, where the nodes M_c are modules, the $\to_c \subseteq M_c \times M_c$ represent the physical connections between modules, and $Cap_c : M_c \to \mathcal{G}$ denotes the capability of each module.*

The number of modules with capability g in configuration c is denoted by $\Lambda(c, g)$. Formally, we have $\Lambda(c, g) = |\{m \in M_c \mid Cap_c(m) = g\}|$.

Note that a configuration is defined not only by the number of modules and the class of modules but also for the arrangement. In Fig. 4, we show examples of different configurations. Configurations c_3 and c_4 and configurations c_5 and c_6 have the same number of modules. However, the arrangement of modules is different. Lastly, a single module can be regarded as a configuration, i.e., $|M_c| = 1$.

Creating different configurations is advantageous to change the robot capabilities and perform different tasks. However, considering all possible configurations given a heterogeneous group of modules with capabilities $g \in \mathcal{G}$ leads to an intractable combinatorial problem even for small number of modules. The number of all possible configurations is in the order $\mathcal{O}(|\mathcal{G}|^{2\lambda})$, where λ is the number of modules available. Instead, we consider the following assumption.

Assumption 1. *The set of feasible robot configuration $\mathcal{C} = \{c_1, c_2, c_3, \dots, c_n\}$ is given beforehand, and their number n is small.*

We are now ready to define *modular robot* which are the actual agents moving in and interacting with the environment Env to accomplish specified tasks.

Definition 4 (Modular Robot)**.** *A modular robot R_k at time $k \in \mathbb{Z}_{\geq 0}$ is defined by a configuration $c_{R_k} \in \mathcal{C}$ and a robot state $s_{R_k} \in \mathcal{Q} \cup \mathcal{E}$, where $s_{R_k} \in \mathcal{Q}$ denotes that robot R_k is at location s_{R_k} at time k, and $s_{R_k} \in \mathcal{E}$ denotes that R_k is moving along edge s_{R_k} at time k.*

Examples of modular robots are shown in Fig. 5. Robots R_1 and $R_2 \in \mathcal{R}$ use the same tool $g_{grasping} \in \mathcal{G}$, but differ in the number of quadrotors $g_{fly} \in \mathcal{G}$. The arrangement and therefore the configurations are different. Additionally, a robot having a saw tool $g_{sawing} \in \mathcal{G}$ and four quadrotors g_{fly} is shown.

Note that robots are defined as depending on time. This is due to their fundamental ability to self-reconfigure, i.e., change their configurations $c_{R_k} \in \mathcal{C}$. Moreover, the number of robots at each time k may be different. Modules from multiple robots may transform into a lesser number of robots, and vice versa. Thus, modular robots do not have a persistent identity, and are considered a unit only while maintaining their configuration $c_{R_k} \in \mathcal{C}$.

For simplicity, we allow robots to reconfigure only at states $q \in \mathcal{Q}$. We model the reconfiguration process as special self-loops $e^r = (q, q)^r$, for all $q \in \mathcal{Q}$. We abuse notation and consider $e^r \in \mathcal{E}$ in addition to normal self-loops (q,q) that capture stationarity[1]. Since we focus on high-level planning, we consider that low-level controllers for determining reconfiguration sequences of attaching and detaching of modules, and their motion during the process are available [7], [10]. Thus, we consider reconfigurations at location $q \in \mathcal{Q}$ at time $k \in \mathbb{Z}_{\geq 0}$ *feasible* if the number of modules of each class $g \in \mathcal{G}$ in all robots involved remains unchanged. We assume that the upper bounds for the duration of any reconfiguration process at a state $q \in \mathcal{Q}$ are known and denoted by $W(e^r)$, where $e^r = (q,q)^r$.

We denote the set of modular robots at time $k \in \mathbb{Z}_{\geq 0}$ by $\mathcal{R}_k$. We assume that the maximum number of robots $p \in \mathbb{Z}_{\geq 1}$ in at any moment in the environment Env is known a priori.

Each robot $R_k \in \mathcal{R}_k$ with state $s_{R_k} = q \in \mathcal{Q}$ must take a control action $u_{R_k} \in \mathcal{E}$. Specifically, it either (a) flies along a transition $(q,q') \in \mathcal{E}$, $q \neq q'$, (b) remains stationary at q via self-loop transition $(q,q) \in \mathcal{E}$, or (c) takes part in a reconfiguration process at q via self-loop transition $e^r = (q,q)^r \in \mathcal{E}$. We consider $u_{R_k} = \varnothing$ when the robots is in transition between locations, i.e., $s_{R_k} \in \mathcal{E}$.

A. Specification

The primitive units of our specifications are *tasks* that capture the required locations, duration, and robot capabilities.

Definition 5 (Task)**.** *A task is a tuple $T = (d, \pi, \mathcal{C}_T)$, where $d \in \mathbb{Z}_{\geq 0}$ is the duration of the task, $\pi \in \Pi$ is the label of all location where the task needs to be performed, and $\mathcal{C}_T \subseteq \mathcal{C}$ is the subset of configurations that can perform the desired action, e.g., transporting, sawing, drilling.*

We define the configuration-task matrix $\Gamma(c,T) \in \mathbb{B}$ such that $\Gamma(c,T) = 1$ if $c \in \mathcal{C}_T$, and $\Gamma(c,T) = 0$ otherwise. We assume that configurations are characterized with respect to tasks in the mission specification, and, thus, Γ is available for planning.

Note that to perform a task's action the robot must generate a desired wrench (forces and torques). The physical parameters, such as volume and weight, and configuration of a robot impacts its performance of tasks. In this paper, we assume that all configurations $\mathcal{C}_T$ are able to perform task T within its duration d. We will address performance differences between configurations in future work.

Duration of tasks may be expressed via the always operators in MTL. Formally, we have $T = \Box_{[0,d]} \bigwedge_{q \in \mathcal{L}(\pi)} \varpi_{q,T}$, where $\varpi_{q,T}$ denotes that there exists a robot at location $q \in \mathcal{Q}$ and configuration $c \in \mathcal{C}_T$ that can perform T.

To capture the satisfaction of robot missions, we define the *joint output word* generated by the teams of robots as $\mathbf{o} = o_0 o_1 o_2 \ldots$, where $o_k = \{T = (d, \pi, \mathcal{C}_T) \mid \forall q \in \mathcal{L}^{-1}(\pi), \exists R_k \in \mathcal{R}_k \text{ s.t. } s_{R_k} = q, c_{R_k} \in \mathcal{C}_T\}$ captures the tasks performed at time k disregarding their durations.

Lastly, mission specifications expressed as MTL formulas with tasks taking the role of atomic propositions. As noted above, tasks are MTL formulas with additional semantics regarding their satisfaction by robots with specific configurations.

[1]Formally, Env takes the structure of a multi-graph with parallel self-loops in our case. For brevity, we define Env as a directed graph, and denote reconfiguration self-loops with the the superscript $()^r$ when needed.

B. Objective function

The motion, actions, and reconfiguration of robots required energy to perform. Let $S_m = \{(e, R_k) \mid u_{R_k} = e = (q,q') \in \mathcal{R}, q \neq q'\}$ be the set of all robot motions, $S_r = \{(e, R_k) \mid u_{R_k} = e^r\}$ be the set of all robot reconfigurations, and $S_a = \{(e, R_k, T) \mid u_{R_k} = e = (q,q) \in \mathcal{R}, q \in \mathcal{L}^{-1}(\pi), c_{R_k} \in \mathcal{C}_T\}$ be the set of all robot actions. We define the cost function as

$$\begin{aligned} J &= J_m + J_r + J_a \\ J_m &= \sum_{(e,R_k) \in S_m} Y_{e, c_{R_k}} \\ J_r &= \sum_{((q,q),R_k) \in S_r} \sum_{g \in \mathcal{G}} Y_{q,g} \cdot \Lambda(c_{R_k}, g), \\ J_a &= \sum_{((q,q)^r, R_k) \in S_a} Y_{q,T,c_{R_k}} \end{aligned} \tag{3}$$

where $Y_{e,c} \in \mathbb{R}_{>0}$ is the energy required to traverse edge e with configuration c, $Y_{q,g} \in \mathbb{R}_{>0}$ is the per module energy cost for module class g at location q for reconfiguration, and $Y_{q,T,c} \in \mathbb{R}_{>0}$ is the energy required by a robot with configuration c to perform actions to satisfy task T at location q.

Again, we assume that our configurations are characterized with respect to motion, reconfiguration, and tasks, and the energy costs $Y_{e,c}$, $Y_{q,g}$, and $Y_{q,T,c}$ are available for planning.

C. Problem

Now that we have define all the components of our framework we formally describe our problem as follows.

Problem 1. *Given a MTL mission specification ϕ and a set of modules with capabilities $\mathcal{G}$ deployed in environment Env that can assemble in a maximum number p of robots at each time $k \in \mathbb{Z}_{\geq 0}$ with configurations $\mathcal{C}$, find the set of robots $\mathcal{R}_k$ and control actions u_{R_k} for all robots $R_k \in \mathcal{R}_k$ at each time $k \in [0 .. \|\phi\|]$ such that $\mathbf{o} \vDash \phi$ and the cost J in* (3) *is minimized.*

IV. Solution

In this section, we formulate Problem 1 as a Mixed Integer Linear Program (MILP).

In the following, we introduce the set of virtual robots $\mathcal{R} = \{R_1, R_2, R_3, \ldots, R_p\}$. Instead of keeping track of the set of robots $\mathcal{R}_k$ at each time $k \in \mathbb{Z}_{\geq 0}$, we consider the fixed set of virtual robots $\mathcal{R}$, where some of the robots are active and some robots are inactive (e.g., do not exist physically). We want to emphasize that it is not required to track each robot identity from when it is first assembled until it is reconfigured. The "same" virtual robot might be created, disappear, and then assembled again potentially in a different configuration.

We consider time horizon based on (2) for the team of robots $\mathcal{R}$ solving specification mission ϕ and denoted as $K = \|\phi\|$.

A. Robot and Module Dynamics

For tracking modular robots navigating in the environment Env, let us define the following binary variables $z_{q,R,c,k}$, $u_{e,R,c,k} \in \mathbb{B}$, which take value one if there is a modular robot R with configuration c, at time k, at state q or edge e, respectively. Initial position of modular robots and at an initial configuration can be encoded as the following equality constraint

$$z_{q,R,c,0} = I(q = q_0, R, c), \tag{4}$$

for all $q \in \mathcal{Q}$, $R \in \mathcal{R}$, $c \in \mathcal{C}$ and $I(\cdot)$ is an indicator function.

Then, we need to constraint that every robot has to be in a specific configuration or it does not exists as follows

$$\sum_{c \in \mathcal{C}} z_{q,R,c,k} \leq 1, \tag{5}$$

for all $q \in \mathcal{Q}$, $R \in \mathcal{R}$, and $k \in [0 \,..\, K]$, and is at single location if it exists,

$$\sum_{q \in \mathcal{Q}} z_{q,R,c,k} \leq 1, \tag{6}$$

for all $c \in \mathcal{C}$, $R \in \mathcal{R}$, and $k \in [0 \,..\, K]$.

Remark 1. *Note that the number of modular robots is not conserved in the environment since they can reconfigure (merge or split into a configuration with different number of robots). Nevertheless, the number of modules is conserved in the entire mission. We can track individual modules g from every robot R in the environment Env.*

Therefore, let us define the following variables $z_{q,g,k}$, $u_{e,g,k} \in \mathbb{Z}_{\geq 1}$ as the number of modules with capability g at time k at state q or edge e, respectively. We can recover the number of modules from the modular robot variables as follows

$$z_{q,g,k} = \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} \Lambda(c, g) \cdot z_{q,R,c,k}, \tag{7}$$

$$u_{e,g,k} = \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} \Lambda(c, g) \cdot u_{e,R,c,k}, \tag{8}$$

for all $q \in \mathcal{Q}$, $e \in \mathcal{E}$, $g \in \mathcal{G}$, where $\Lambda(c, g)$ is the number of modules with capability g used in configuration c.

Using this module capabilities variables we can define flow dynamic constraints in the environment as follows

$$z_{q,g,k} = \sum_{(q',q) \in \mathcal{E}} u_{e,g,k}, \tag{9}$$

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

The flow constraints are enforcing the conservation of modules of each type while robots fly, perform tasks, and reconfigure in the environment. Note that the sums in (9) and (10) also include the reconfiguration self-loops $e^r = (q,q)^r$.

Next, we constrain the robots to not change configurations unless a reconfiguration self-loop is used

$$z_{q,R,c,k} + z_{q',R,c,k+W(e)} \geq 2 \cdot u_{(q,q'),R,c,k}, \tag{11}$$

for all $e = (q, q') \in \mathcal{E} \setminus \{e^r = (q,q)^r \mid q \in \mathcal{Q}\}$, $R \in \mathcal{R}$, $c \in \mathcal{C}$, $k \in [0 \,..\, K - W(e)]$. We do not enforce the constraint for reconfiguration self-loops e^r, and let the variables free to change robots' configurations as required by tasks' satisfaction.

B. Task Satisfaction

For encoding task satisfaction, let us consider the following binary variables $z_k^T \in \mathbb{B}$, which takes value one if task T is satisfied. Thus, we have

$$z_k^T \leq \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} z_{q,R,c,k} \cdot \Gamma(c, T), \tag{12}$$

for all $T = (d, \pi, \mathcal{C}_T)$ with $\pi \in AP$, $q \in \mathcal{L}^{-1}(\pi)$, $k \in [0 \,..\, K]$, which ensures that modular robot R with configuration c can perform task T at state q, and at time k.

Then, satisfaction of an MTL specification is captured via a recursive encoding using binary variables as $z_k^\varphi \in \mathbb{B}$ for a subformula φ at k. The variable z_k^φ takes value one if subformula φ holds at time k, and is zero otherwise. The full encoding is similar to the one in [16], [23], and we omit it for brevity.

C. Objective Function

Finally, we define the objective functions to capture the desired optimal behavior of the modular robots.

1) Reconfiguration cost function: We define a cost to penalize reconfiguration of the modular robots into new configurations. The cost becomes

$$J_r = \sum_{k \in [0..K]} \sum_{q \in \mathcal{Q}} \sum_{g \in \mathcal{G}} Y_{q,g} \cdot u_{e^r,g,k}, \tag{13}$$

where $Y_{q,g}$ is the energy for reconfiguration per module of type g at state q, $e^r = (q,q)^r$.

2) Action cost function: We define a cost for satisfying a task T at state q using a modular robot in a configuration c as follows

$$J_a = \sum_{k \in [0,..K]} \sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}} \sum_{R \in \mathcal{R}} Y_{q,T,c} \cdot z_{aux} \cdot \Gamma(c, T), \tag{14}$$

where $z_{aux} = z_k^T \cdot z_{q,R,c,k}$, and $Y_{q,T,c}$ is the per time unit cost of using configuration c at state q to satisfy task T. Note that since both variables are binary the product can be reduce to $z_{aux} = \min\{z_k^T, z_{q,R,c,k}\}$, which can be encoded as mixed integer linear constraints.

3) Motion cost function: We capture the cost for traversing the edges e with configuration c

$$J_m = \sum_{k \in [0,..K]} \sum_{e \in \mathcal{E}'} \sum_{c \in \mathcal{C}} \sum_{R \in \mathcal{R}} Y_{e,c} \cdot u_{e,R,c,k}, \tag{15}$$

where $\mathcal{E}' = \mathcal{E} \setminus \{(q,q), (q,q)^r \mid q \in \mathcal{Q}\}$, and $Y_{e,c}$ is the per time unit cost of using configuration c while traversing edge e.

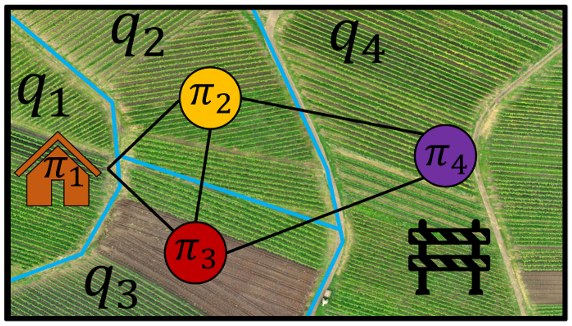

Fig. 6. Case study 1: Environment, transition system, atomic propositions, head-quarters at q_1, and task request.

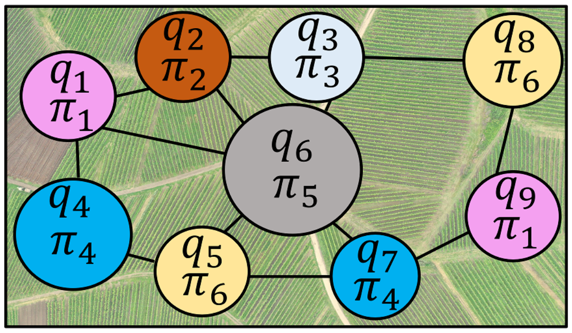

Fig. 7. Case study 2: Environment, transition system, atomic propositions, and head-quarters at q_6.

4) Optimization problem: Then, we can formulate Problem 1 as the following MILP optimization problem

$$\min_{u_{e,R,c,k}} J_r + J_a + J_m,$$
$$\text{subject to}$$
$$\text{Module and robot dynamics (4) – (11),}$$
$$\text{Task satisfaction (12),}$$
$$\text{Mission satisfaction: } z_0^{\phi} = 1.$$

V. Analysis

correctness and complexity. the latter is number of binary variables to provide an idea on how complex it is complemented by run time performance in the results

Proposition 1. *number of modules that enter $e^r = (q,q)^r$ = number of modules that exit $e^r = (q,q)^r$*

VI. Case Studies

In this section, we perform multiple simulations to demonstrate the performance of the MILP formulation in terms of scalability and correctness. We are mainly motivated by case studies where we need to specify time and logically constrained tasks that describe a construction mission in agriculture environments.

All computation of the following case studies were performed on a PC with 4 cores at 2.7 GHz with 32 GB of RAM. We used Gurobi [24] as the MILP solver.

A. Case Study 1

In this case study, we describe a small construction mission for showcasing how the MILP planning algorithm encoded in Sec. IV works. We consider an agricultural environment Env shown in Fig. 6 with four different states $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$ and atomic proposition set $\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$. The mission considers that at some time between deployment wood must be cut at state q_4 and then wood needs to be used to assemble a fence. In MTL, the mission specification is $\phi = \Diamond_{[0,3]}T(1, \pi_{purple}, g_3) \wedge \Diamond_{[4,6]}(T(1, \pi_{purple}, g_2) \wedge T(1, \pi_{purple}, g_1))$. We consider the following initial conditions, module capabilities set $\mathcal{G} = \{g_0, g_1, g_2, g_3\}$ whose meaning is shown in Fig. 3. We have $|\mathcal{C}| = 10$ predefined configurations with the matrix $\Lambda(c,g) = \{c_0 : \{g_0 = 4, g_1 = 1, g_2 = 1\}, c_1 : \{g_0 = 8, g_1 = 1, g_2 = 1\}, c_2 : \{g_0 = 3, g_2 = 1\}, c_3 : \{g_0 = 9, g_2 = 1\}, c_4 : \{g_0 = 4, g_1 = 1\}, c_5 : \{g_0 = 7, g_2 = 1, g_3 = 1\}, c_6 : \{g_2 = 1\}, c_7 : \{g_1 = 1\}, c_8 : \{g_3 = 1\}, c_9 : \{g_0 = 1\}\}$.

We consider that there are twelve g_0 modules with flying capability and two of each g_1, g_2, g_3 screwdriver, gripper, and saw, respectively, at headquarters q_1. We constraint the maximum number of robots we can generate to be $|\mathcal{R}| = 13$. The binary matrix indicating whether or not a configuration can manipulate a tool $\Gamma(c,T)$ has value one if c contains $g_\mathcal{T}$ (capability needed in the task T) and at least three g_0 and is zero otherwise. Motion, reconfiguration, and task energy consumption for using a specific configuration are generated so that $Y_{q,g}, Y_{qtc}$, and $Y_{e,c} \in [1\,..\,10]$. Finally, the reconfiguration time $W(e^r)$ for going from any configuration to other feasible configuration is first considered as one. We use these initial parameters for four different cases described bellow.

1) Feasible Case: The generated solution shows that there are six initial robots at time $k = 0$ at state q_1, robots are in the following initial configurations $R_0 : c_8$, $R_1 : c_4$, $R_2 : c_8$, $R_3 : c_1$, $R_4 : c_6$. Note, that the rest of the robots do not appear since not all the robots must exists at every time; they may reconfigure, merging and splitting as required. Satisfaction of the subformula $\varphi_1 = \Diamond_{[0,3]}T(1, \pi_{purple}, g_3)$ is achieved at time $k = 2$ where the robots have reconfigured and traveled as follows at state q_1. We have robots with the following configurations $R_5 : c_9$, $R_6 : c_6$, $R_9 : c_9$, $R_{10} : c_7$, $R_{11} : c_8$, $R_{12} : c_9$, at state q_4 time $k = 2$. There is one robot with configuration $R_4 : c_5$. The type of modules in configuration c_5 show that the subformula φ_1 has been satisfied. Satisfaction of subformula $\varphi_2 = \Diamond_{[4,6]}(T(1, \pi_{purple}, g_2) \wedge T(1, \pi_{purple}, g_1))$ is achieved at time $k = 6$. The robots at state q_1 are the following $R_0 : c_4$, $R_7 : c_8$, $R_{11} : c_6$, at state q_4 robots $R_0 : c_9$, $R_1 : c_2$, $R_5 : c_8$, $R_{12} : c_4$. Here we can see that robots $R_1 : c_2$ and $R_{12} : c_4$ satisfy the subformula.

The solution for this specification generates 4050 integer variable and 3036 binary variables in Gurobi. The computation time for solving this case study is 1.30 seconds. The objective values are reconfiguration cost $J_r = 4$, action cost $J_a = 6$, motion cost $J_m = 45$.

2) Infeasible Case: For this simulation, we consider the same initial parameters as the one above; however, changing the reconfiguration time $W(e^r = (q, q')^r) = 3$, the model becomes infeasible since robots need to enter into reconfiguration to satisfy tasks, but time for reconfiguring causes a violation in the deadlines of the specification.

3) Feasible case with additional modules, configuration, and possible robots: In this simulation, we consider the same initial parameters, but we add twelve additional quadrotors and increase the set of configurations $|\mathcal{C}| = 20$, and possible maximum number of robots to $|\mathcal{R}| = 23$. Note that initially, there are so many robots and configurations that reconfigurations are not necessary, and the chosen robots are the ones that reduce the motion and task cost. All changes are at the expense of increasing the computation since every time we increase the set of possible configurations, $\mathcal{C}$, and the set of possible robots $\mathcal{R}$, we include additional binary variables. The solution took 5.86 seconds and specification satisfaction $z_0^\phi = 1$ is achieve while the other objective values are reconfiguration cost $J_r = 0$, action cost $J_a = 4$, and motion cost $J_m = 29$. There is an increase in the number of variables generated in Gurobi, having 70618 integer variables and 70042 binary variables.

4) Feasible with an increase in the number of states and edges: Lastly, we use the same initial parameters, but augment the environment by adding six states $\mathcal{Q}$ and generating edges $\mathcal{E}$ according to proximity. The transition system is shown in Fig. 7. Here the computation time increase to 27.06 seconds The specification is satisfied, $z_0^\phi = 1$, with reconfiguration cost $J_r = 0$, action cost $J_a = 4$, and motion cost $J_m = 33$. Additionally, there a drastic increase of number of generated variables; 187062 integer variables and 185542 binary variables which shows that the algorithm is susceptible to the number of states and edges in terms of time and number of variables. This is not surprising, since for every state and edge in the environment Env, all the variables for modules and robots have to be created.

B. Case Study 2

For this case study we consider the environment shown in Fig. 7. We consider predefined configurations and number of robots to be $|\mathcal{C}| = 20$ and $|\mathcal{R}| = 20$. We use the following mission specification,

$$\phi_n = \wedge_{i=1}^{n} \big(\wedge_{j=1}^{3} \Diamond_{[0,20]} T(2, \pi_j, g_{\tau,j}) \big), \tag{16}$$

where π_j is a randomly picked state in the environment and $g_{\tau,j}$ the first three capabilities added every time that n increase. Table VI-B summarizes the information at every iteration on how the runtime and number of variables increase when adding more modules and module capabilities.

Finally, we keep all other parameters constant and gradually increase the number of possible robots and configurations. All initial parameters are the same as the one considered for VI-A.4. In Fig. 8, we can see that at first, increasing the number of variables does not affect the time performance dramatically. However, after fifteen configurations and fifteen possible robots, the combinatorial problem makes the runtime grow almost exponentially.

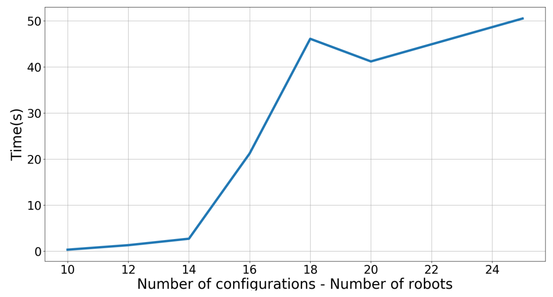

Fig. 8. Runtime performance while increasing the number of possible configurations and robots.

VII. Conclusions and Future Work

This paper proposes a MILP formulation for planning for heterogeneous modular robots that can form configurations to manipulate tools and satisfy tasks in an environment. We consider that not all configurations can satisfy every task, which creates the necessity for modular robots to reconfigure into appropriate configurations. We formulate tasks to account for logical and timing constraints using Metric Temporal Logic, where atomic propositions capture the location and the tool that the task requires. We formulate costs for motion, reconfiguration, and actions for satisfaction in a specific configuration. We can conclude that the time performance depends directly on the number of binary variables, which depend on all the possible configurations and robots that the system can generate, the size of the environment, and the specification size. We consider adding uncertainty on task satisfaction and performance differences between configurations for future work.

References

[1] D. S. K. Dixit and M. S. Dhayagonde, "Design and implementation of e-surveillance robot for video monitoring and living body detection," *International Journal of Scientific and Research Publications*, vol. 4, no. 4, pp. 2250–3153, 2014.

[2] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *2015 International Conference on Intelligent Environments*, pp. 140–143, IEEE, 2015.

[3] J. L. Bresina, A. K. Jónsson, P. H. Morris, and K. Rajan, "Activity planning for the mars exploration rovers.," in *ICAPS*, pp. 40–49, 2005.

[4] G. A. Cardona and J. M. Calderon, "Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations," *Applied Sciences*, vol. 9, no. 8, p. 1702, 2019.

[5] J. Seo, J. Paik, and M. Yim, "Modular Reconfigurable Robotics," *The Annual Review of Control, Robotics, and Annu. Rev. Control Robot. Auton. Syst*, vol. 2, pp. 63–88, 2019.

[6] H. Ahmadzadeh, E. Masehian, and M. Asadpour, "Modular robotic systems: Characteristics and applications," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 3, pp. 317–357, 2016.

[7] D. Saldana, B. Gabrich, G. Li, M. Yim, and V. Kumar, "ModQuad: The flying modular structure that self-assembles in Midair," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 691–698, 2018.

[8] C. Liu, M. Whitzer, and M. Yim, "A distributed reconfiguration planning algorithm for modular robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4231–4238, 2019.

TABLE I: Case Study 2: Runtime and complexity performance, integer and binary variables while increasing number of modules and module capabilities.

Index (n)	Number Modules	Number Capabilities	Time (s)	Integer Variables	Binary Variables
1	30	5	46.26	23799	18088
2	45	10	69.93	25234	19391
3	60	15	168.52	29321	20892
4	80	20	255.63	32321	21865
5	100	25	451.31	33321	25324

[9] M. Park, S. Chitta, A. Teichman, and M. Yim, "Automatic configuration recognition methods in modular robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 403–421, 2008.

[10] Y. Litman, N. Gandhi, L. T. X. Phan, and D. Saldaña, "Vision-based self-assembly for modular multirotor structures," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2202–2208, 2021.

[11] V. Vonásek, M. Saska, L. Winkler, and L. Přeučil, "High-level motion planning for cpg-driven modular robots," *Robotics and Autonomous Systems*, vol. 68, pp. 116–128, 2015.

[12] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[13] S. Berman, Á. Halász, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[14] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *International Journal of Robotics Research*, vol. 37, no. 7, pp. 818–838, 2018.

[15] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

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

[17] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *arXiv preprint arXiv:2201.05247*, 2022.

[18] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, "An end-to-end system for accomplishing tasks with modular robots," *Robotics: Science and Systems*, vol. 12, 2016.

[19] S. Castro, S. Koehler, and H. Kress-Gazit, "High-level control of modular robots," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3120–3125, IEEE, 2011.

[20] C. Brzoska, "Programming in metric temporal logic," *Theoretical computer science*, vol. 202, no. 1-2, pp. 55–125, 1998.

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

[22] A. Dokhanchi, B. Hoxha, and G. Fainekos, *5th International Conference on Runtime Verification, Toronto, ON, Canada.*, ch. On-Line Monitoring for Temporal Logic Robustness, pp. 231–246. Springer, 2014.

[23] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 772–779, IEEE, 2015.

[24] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020.