2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

# Temporal Logic Swarm Control with Splitting and Merging

Gustavo A. Cardona[1], Kevin Leahy[2], and Cristian-Ioan Vasile[1]

***Abstract*— This paper presents an agent-agnostic framework to control swarms of robots tasked with temporal and logical missions expressed as Metric Temporal Logic (MTL) formulas. We consider agents that can receive global commands from a high-level planner, but no inter-agent communication. Moreover, agents are grouped into sub-swarms whose number can vary over the mission time horizon due to splitting and merging. However, a strict upper bound on the maximum number of sub-swarms is imposed to ensure their safe operation in the environment. We propose a two-phase approach. In the first phase, we compute the trajectories of the sub-swarms, splitting, and merging actions using a Mixed Integer Linear Programming approach that ensures the satisfaction of the MTL specification with minimal swarm division over the mission time horizon. Moreover, it enforces the upper bound on the number of sub-swarms. In the second phase, splitting fractions for sub-swarms resulting from splitting actions are computed. A distributed randomized protocol with no inter-agent communication ensures agent assignments matching the splitting fractions. Finally, we show the operation and performance of the approach in simulations with multiple tasks that require swarm splitting or merging.**

## I. Introduction

In recent years, there has been intense interest in studying robot swarms [1]–[5]. Due to their great capacity to handle multiple tasks and offer robustness and resiliency over agent failures. The emerging collective behavior from the interaction of multiple agents gives swarm robotic systems many potential applications ranging from exploration, mapping, and surveillance to search and rescue [6], [7]. Despite these advantages, controlling many agents and commanding their decisions is still challenging in both low-level and high-level control. For complex missions, temporal logics (TL) are a useful specification language for extending swarm behavior to time-varying, task-oriented goals [8]–[10].

Several works have considered high-level objectives for swarms given as temporal logic goals [11]–[15]. However, these usually consider specifications for swarms as a whole and are not concerned with agent constraints either in the control or communication [16]–[18] (e.g., if agents in a swarm are unable to communicate, methods that rely on the dissemination of information will not work). In contrast, in this work, we propose a high-level control framework for swarms of aerial robots aware of low-level control and agent communication constraints. Global plans are broadcast to all agents since it is impractical and cumbersome to specify the precise plans for each agent. Plans satisfy complex temporal logic tasks with timing constraints expressed in Metric Temporal Logic (MTL) [19], [20]. Agents are grouped into sub-swarms, and we allow them to split and merge as needed to fulfill mission tasks, e.g., when agents must be at multiple locations simultaneously.

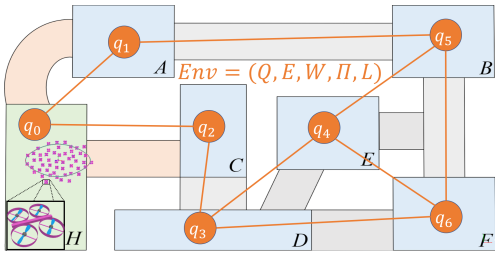


Fig. 1: Example environment. Agents begin in the region $H$ and must visit the labeled blue regions while navigating the narrower gray and orange passages.

To solve the high-level swarm planning problem, we propose a Mixed Integer Linear Programming (MILP) approach that considers the satisfaction of the MTL specification by finding a motion plan for swarms able to split and merge them as required. For the motion plan in the MILP formulation, we take inspiration from flow equality balance equations as used in [21]–[24]. However, in this work, swarms may split into multiple sub-swarms to satisfy simultaneous tasks or merge if splitting is no longer necessary and equality constraints cannot capture these flow dynamics. Instead, we propose inequality constraints that control the flow of swarms while allowing the creation and destruction of swarms, modeling splitting and merging actions.

In this work, agents cannot communicate with each other. Thus, only global commands can be used to distribute agents to sub-swarms resulting from splitting and merging actions. Therefore, we propose a distributed randomized method to split large swarms. We build a Directed Acyclic Graph (DAG) from the MILP solution that captures the division of the swarm over time. We use the DAG to compute splitting fractions of sub-swarms that balance their sizes over the mission. Agents use the randomized protocol that samples assignments based on the splitting fractions without inter-agent communication.

[1]Dept. of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA gcardona@lehigh.edu, cvasile@lehigh.edu

[2]Massachusetts Institute of Technology Lincoln Laboratory, Lexington, MA 02421, USA kevin.leahy@ll.mit.edu

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering. 

The paper's contributions include: 1) We propose an efficient MILP approach to solve the planning problem considering swarm splitting and merging behaviors for satisfying tasks while constraining the maximum number of simultaneous existing swarms and minimizing unnecessary splitting or traveling. Additionally, standard flow dynamics equations are modified to flow inequalities considering that due to the merging and splitting actions, node and edge equations could not be balanced. 2) We develop a randomized distributed method to split large swarms when communication between agents is impossible. 3) We develop a method to assign splitting fractions such that the sub-swarms are balanced, given a motion plan from the MILP. 4) We show the performance of the proposed MILP method and the balanced splitting and merging algorithm.

## II. Problem Formulation

In this section, we formulate the planning problem for swarms of aerial robots tasked with rich temporal logic specifications to visit places of interest in an environment. The initial swarm can be split into sub-swarms during the mission horizon to satisfy tasks that require multiple swarms at different locations in the specification. Also, multiple sub-swarms can merge to form a single swarm if splitting is no longer required. Thus, the number of sub-swarms active in the environment may be time-varying. We assume there is an upper bound in the number of swarms we can have simultaneously in the mission. The swarm is not infinitely divisible, and the Unmanned Aerial Vehicles (UAVs) can only fly at a limited number of altitudes for safety reasons. Moreover, splitting the swarm as few times as possible is desirable and just when the mission specification requires it. Here we introduce the models for the environment, agents, swarms, splitting and merging behaviors, and tasks that define the planning problem for the swarm of robots. First, we define the *environment* capturing the motion of swarms between locations of interest.

**Definition 1** (Environment)**.** *The environment is abstracted as a labeled transition system,* $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W}, \Pi, \mathcal{L})$*, where* $\mathcal{Q}$ *is the set of regions in the environment,* $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ *is the set of edges defining adjacent regions that swarms can travel between,* $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ *maps each transition in* $\mathcal{E}$ *to its travel duration, a stationary swarm is modeled as an unit-weight self-transition,* $\Pi$ *is a set of atomic propositions*[1]*, and* $\mathcal{L} : \mathcal{Q} \to 2^{\Pi}$ *is a labeling function.*

In Fig. 1, we show a scenario with multiple regions of interest and the corridors that connect the regions, which generates the abstracted environment $Env$ in orange.

### A. Agents' dynamics and sensing

We consider a set of agents $\mathcal{A} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$ where $|\mathcal{A}| = N$ is the total number of agents available in the environment. Each agent's state is given by the location in the environment, either a state $q \in \mathcal{Q}$ or an edge $e \in \mathcal{E}$ that $\mathbf{x}_i$ is traversing, and an altitude $h_i \in \{1, \ldots, N_s\}$, where $N_s$ is the number of discrete altitude levels at which agents are allowed to fly. The state of a single agent is given by the tuple $\mathbf{x}_i = (q/e, h_i)$. Altitude $h_i$ captures the discrete altitude level at which an agent is flying. We assume that there are a bounded number $N_s$ of altitude levels $h$ that agents can occupy. This requirement captures division or airspace for safe navigation, e.g., quadrotors need at least one-meter vertical separation to avoid downwash effects. A sub-swarm is defined by the agents $\mathbf{x}_i$ that share altitude $h_i = h$. Thus, we differentiate between sub-swarms based on their altitudes. Moreover, the maximum number of sub-swarms at any given time during the mission is bounded by $N_s$.

Although we are particularly interested in the high-level planning for swarms under temporal and logic-constrained specifications, we consider some assumptions in the lower-level control that drive modeling and solution design. Note that implementation with real robots requires low-level controllers to execute high-level plans. This can be implemented using standard formation control [25], [26] and attitude control [27], [28] methods.

**Assumption 1.** *We assume agents cannot communicate with each other, avoiding direct interaction between agents. Instead, we consider agents that transition between regions in* $\mathcal{Q}$ *using controllers that require only local information, e.g., formation control using the relative pose of neighbors by sensing them [29], [30].*

Assumption 1 is crucial for defining and modeling the splitting and merging swarm actions described in the following sub-section.

### B. Swarms' state and actions

In this work, we are interested in commanding the entire swarm (or swarms) instead of individual agents. Multiple sub-swarms may exist due to swarms' splitting or merging as demanded by the specification. For instance, to fulfill tasks that need to be satisfied in overlapping time intervals, we require multiple sub-swarms to be present in disjoint regions simultaneously. We take inspiration from recent work on controlling swarms as abstract objects in the environment that allow us to send global commands to the swarm, not to individual agents. As is shown in Fig. 1 we abstract the swarm [31]–[35] as ellipses, where the mean and covariance of the agents' position represent the center and shape of the swarm, respectively.

The center of the swarm $\mu$ and shape $\Sigma$ are required for the low-level control of the swarms' position, heading, and size using communication-free formation control techniques [36]–[38]. However, since we abstract the environment as a transition system $Env$, $\mu$, and $\Sigma$ are unnecessary for high-level planning. Instead, we use the location in the environment described by a state $q \in \mathcal{Q}$ or an edge $e \in \mathcal{E}$ to capture the sub-swarms' states and motion.

**Definition 2** (Swarm)**.** *A swarm is a tuple* $\mathbf{s} = (\mathcal{A}_s, h_s, q/e)$ *with* $\mathcal{A}_s \subseteq \mathcal{A}$ *being the set of agents that belong to the swarm* $s$ *and that share common altitude* $h_s$*, and* $q/e$ *represents that the swarm is either at state* $q \in \mathcal{Q}$ *or traversing edge* $e \in \mathcal{E}$*.*

[1]Atomic propositions are defined in Sec. II-C

The set of active sub-swarms at time $k$ is denoted by $\mathcal{S}(k) = \{\mathbf{s}_0, \ldots, \mathbf{s}_i, \ldots\}$ which is time-varying. However, we assume there is an a priori known upper-bound $N_s \geq |\mathcal{S}(k)|$, for all $k \in \mathbb{Z}_{\geq 0}$, which is the maximum number of swarms that can exist simultaneously in the environment. The number of sub-swarms changes with time, depending on the mission's requirements. Sub-swarms are created and deleted according to splitting or merging actions defined as follows.

**Definition 3** (Splitting action)**.** *Splitting is the action of taking a swarm $\mathbf{s}_i \to \{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$ and transforming it into $n$ sub-swarms such that $\mathcal{A}_{s_i} = \bigcup_{\ell=1}^{n} \mathcal{A}_{s_{i_\ell}}$ and $\mathcal{A}_{s_\ell} \cap \mathcal{A}_{s_{\ell'}} = \varnothing$, $\forall \ell \neq \ell' \in \{i_1, \ldots, i_n\}$. Additionally, all of the resulting sub-swarms have different altitudes, $h_{s_\ell} \neq h_{s_{\ell'}}$, $\forall \ell \neq \ell' \in \{i_1, \ldots, i_n\}$.*

Splitting action will occur when time-overlapping tasks require multiple swarms at different states $q \in \mathcal{Q}$ in the environment $Env$.

**Definition 4** (Merging action)**.** *Merging is the action of taking a set of $n$ sub-swarms $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\} \to \mathbf{s}_i$ and transforming them into a single swarm such that $\mathcal{A}_{s_i} = \bigcup_{\ell=1}^{n} \mathcal{A}_{s_{i_\ell}}$. Additionally, all of the agents converge to the same altitude $h_j = h_{s_1}$, $\forall j \in \mathcal{A}_{s_i}$, of the first sub-swarm by convention.*

Merging sub-swarms is desirable when multiple sub-swarms are not required for mission satisfaction since it may reduce the computational cost of tracking and controlling them. To simplify the problem, we assume that merging and splitting actions can only occur at states $q \in \mathcal{Q}$ and not while traversing the edges $e \in \mathcal{E}$.

### C. Mission Specification Using Metric Temporal Logic

Metric Temporal Logic (MTL), as introduced in [19], is a specification language expressing real-time properties. The syntax of MTL is given by its Backus-Naur form as

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

where $\phi$, $\phi_1$, and $\phi_2$ are MTL formulae, $\top$ is the logical *True* value, $\pi \in \Pi$ is an atomic proposition (e.g., a region in a labeled environment). $\neg$, $\vee$, and $\wedge$ are the Boolean negation, disjunction, and conjunction operators, and $\Box_I$, $\Diamond_I$, and $\mathcal{U}_I$ are the timed *always*, *eventually*, and *until* operators with $I = [k_1, k_2]$ a discrete-time interval, $0 \leq k_1 \leq k_2$. The logical *False* value is $\bot = \neg\top$. *Time horizon of MTL formula* is defined as in [39]

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = \pi, \\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1, \\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \wedge \phi_2, \\ k_2 + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi = \phi_1 \mathcal{U}_{[k_1,k_2]} \phi_2. \\ k_2 + \|\phi_1\| & \text{if } \phi \in \{\Diamond_{[k_1,k_2]}\phi_1, \Box_{[k_1,k_2]}\phi_1\} \end{cases} \tag{1}$$

We define the semantics of MTL with respect to the swarms' trajectories (plan) $\boldsymbol{\alpha}$.

**Definition 5** (Plan)**.** *A plan is the* joint state trajectory *generated by the sequence of swarms navigating in the environment $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 \boldsymbol{\alpha}_1 \boldsymbol{\alpha}_2 \ldots \boldsymbol{\alpha}_{\|\phi\|}$, where $\boldsymbol{\alpha}_k = \mathcal{S}(k)$ and each sub-swarm $s_i = (\mathcal{A}_{s_i}, h_{s_i}, q_i) \in \mathcal{S}(k)$ at time $k$ either (1) moves in the environment using edge $e = (q_i, q')$, i.e., $(\mathcal{A}_{s_i}, h_{s_i}, q') \in \mathcal{S}(k+\mathcal{W}(e))$, (2) performs a split action $\mathbf{s}_i \to \{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$, i.e., $(\mathcal{A}_{s_\ell}, h_{s_\ell}, q) \in \mathcal{S}(k+1)$, $\forall \ell \in \{i_1, \ldots i_n\}$, or (3) performs a merging action $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\} \to \mathbf{s}_j$, i.e., $(\bigcup_{\ell=i_1}^{i_n} \mathcal{A}_{s_\ell}, h_{s_1}, q) \in \mathcal{S}(k+1)$ and $i \in \{i_1, \ldots, i_n\}$.*

For proposition $\pi \in \Pi$,

$$\boldsymbol{\alpha}_k = (\mathbf{s}_i = (\mathcal{A}_s, h_s, q_i), k) \vDash \pi \iff \pi \in \mathcal{L}(q_i)\,, \tag{2}$$

meaning that an atomic proposition is satisfied if at least one swarm is in a region $q_i$ labeled with that atomic proposition $\pi$. The semantics of the remaining operators is defined as usual [40], [41].

### D. Problem

For simplicity, below, we define two problems that describe (1) how swarms split, merge, and move in the environment and (2) what fraction of agents each sub-swarm has. Note that in a plan $\boldsymbol{\alpha}$, swarms are created as required by the specification while merging is enforced via cost function $\mathcal{J} = \sum_{k=0}^{\|\phi\|} |\mathcal{S}(k)|$.

**Problem 1.** *Given a team of $N$ agents, an abstracted environment $Env$, swarm splitting upper-bound $N_s$, and an MTL specification $\phi$, find plan $\boldsymbol{\alpha}$ for the creation of swarms and their trajectories $\mathbf{s}_i(k)$ such that $\phi$ is satisfied and cost $\mathcal{J}$ is minimized.*

The sequence of actions in $\boldsymbol{\alpha}$ involves not only the sequence of how to traverse the environment to satisfy the mission but also the sequence of splitting and merging actions performed by the swarm. If the number of sub-swarms at any time during the mission is close to the upper-bound $N_s$, it is necessary to ensure an appropriate number of agents for all of them. Note that we do not assign roles to agents or pre-allocate them to any swarm since we assume there will be no communication between agents as described in Assumption 1. In contrast, agents must find a way to decide which of the swarms to join. Formally, we define this problem as follows.

**Problem 2.** *Given a symbolic plan $\boldsymbol{\alpha}$ which provides the information about the number of swarms $\mathcal{S}(k)$ at all times, find the appropriate splitting fractions for all of the agents such that any swarm at any time during the mission has a sufficient quantity of agents. Additionally, find a protocol without communication between agents that enforces the splitting fractions.*

## III. Control Synthesis Solution

This section discusses the components required to solve Problem 1 and Problem 2. We first solve the problem of finding a plan $\boldsymbol{\alpha}$, the sequence of actions for visiting the regions and splittings or merging actions of the swarm that satisfies $\phi$. Then, given such a sequence of splitting or merging actions from the initial swarm throughout the mission horizon $\|\phi\|$, we propose an algorithm for assigning splitting fractions such that the sub-swarms are balanced given a motion plan.

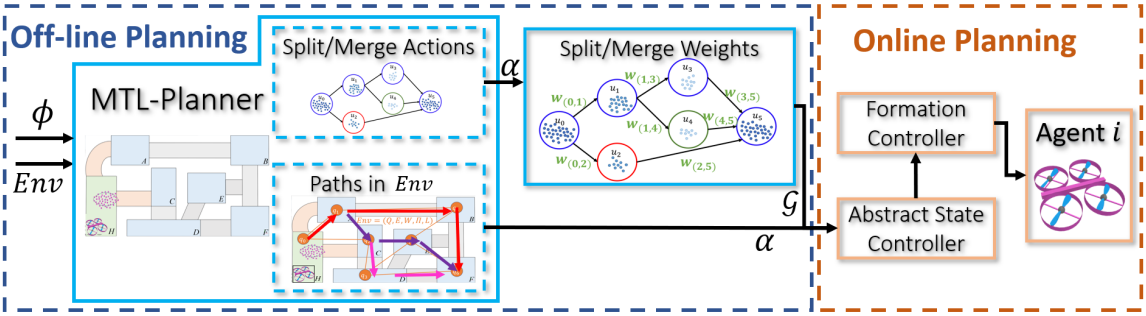


Fig. 2: Block diagram of proposed control framework.

### A. MILP Encoding for Swarm Planning

Here, we formulate a MILP encoding for capturing satisfaction of the MTL specification, which not only indicates the times when a region needs to be visited but also a sequence of splittings and merging actions of the swarm in the environment. First, we define integer variables $z_{q,k}$ and $z_{e,k} \in \mathbb{Z}$ to indicate the number of sub-swarms at state $q \in \mathcal{Q}$ or edge $e \in \mathcal{E}$ at time $k \in \{0, \dots, \|\phi\|\}$, respectively. We set the initial condition at $k = 0$ by setting $z_{q_0,0} = 1$, $z_{q,0} = 0$ for all $q \neq q_0$ which states that the entire swarm starts at $q_0$. The number of sub-swarms we have in the environment at any time is not larger than $N_s$, which induces the constraints $z_{q,k}, z_{e,k} \in [0, N_s]$, for all $q \in \mathcal{Q}$, $e \in \mathcal{E}$ and $k \leq \|\phi\|$. Taking into account the swarm variables, we capture the flow dynamic constraints in the environment as follows

$$z_{q,k} \geq z_{e,k-W(e)}, \ \forall e = (q', q) \in \mathcal{E}, \ k \leq \|\phi\|, \quad (3)$$

$$z_{e,k-W(e)} \leq z_{q,k-W(e)}, \ \forall e = (q, q') \in \mathcal{E}, \ k \leq \|\phi\|, \quad (4)$$

Notice that (3) and (4) ensure that swarms will go to required regions. However, to make sure the swarms can only travel to neighboring nodes, the creation (splitting) of swarms will occur only at states, and merging swarms are all present at the same state, we impose the following constraints

$$z_{q,k} \leq \sum_{e=(q',q)\in\mathcal{E}} z_{e,k-W(e)}, \forall q \in \mathcal{Q}, \ k \leq \|\phi\|, \quad (5)$$

$$\sum_{e=(q,q')\in\mathcal{E}} z_{e,k} \geq z_{q,k}, \ \forall q \in \mathcal{Q}, \ k \leq \|\phi\|. \quad (6)$$

The flow constraints capture how a swarm (swarms) travel through the environment $Env$. However, they do not constrain the number of sub-swarms we have at any time. For this purpose, we introduce the following split-bound constraint

$$\underbrace{\sum_{e\in\mathcal{E}} \sum_{k'\in H(e,k')} z_{e,k'}}_{\text{Number of sub-swarms at time k}} \leq N_s, \quad (7)$$

where $H(e,k) = \{\{k - W(e) + 1, ..., K\} \cap \mathbb{Z}_{\geq 0}\}$ which is the history of departures for edge $e$ at time $k$.

Then, we formulate Problem 1 as the following MILP optimization problem

$$\begin{aligned} &\min \sum_{k=0}^{\|\phi\|} \left( \sum_{q\in Q} z_{q,k} + \sum_{e\in\mathcal{E}} z_{e,k} \right), \\ &\text{subject to} \\ &\text{Swarm flow dynamics (3) – (6),} \\ &\text{Split bound (7),} \\ &\text{Mission satisfaction: } \{z_{q,k}\} \vDash \phi. \end{aligned} \quad (8)$$

Note that (8) discourages the unnecessary swarm splitting and traveling in the environment over time horizon $\|\phi\|$. Mission satisfaction $\{z_{q,k}\} \vDash \phi$ is encoded similarly to [21], [42], and we omit it for brevity.

### B. Swarm Splitting and Merging Actions Weights

From the swarm planning stage, we obtain sequences of swarm splitting and merging actions of sub-swarms necessary to satisfy the mission specification. The sequences are captured as a Directed Acyclic Graph (DAG) $G = (V, E, w)$, where $V$ is the set of nodes, and each node is defined by a tuple $u = (q, k)$ meaning that there is a swarm at location $q$ in the environment at time $k$. Edges $E \subseteq V \times V$ capture splitting (i.e., multiple outgoing edges from a state $u$) and merging (i.e., multiple incoming edges to a state $u$). Let $\mathcal{N}_u^+ = \{v \mid (u, v) \in E\}$ and $\mathcal{N}_u^- = \{v \mid (v, u) \in E\}$ be the sets of outgoing and incoming edges of state $u$. A leaf node $u$ has no outgoing neighbors, $\mathcal{N}_u^+ = \varnothing$. The weights $w : E \to \mathbb{R}_{>0}$ represent splitting fractions such that the unit-sum constraint $\sum_{v\in\mathcal{N}_u^+} w((u, v)) = 1$ holds for all non-leaf $u \in V$ including the starting node $u_0 = (q_0, 0)$. Thus, $w(u) \in (0, 1]$ for all $u \in V$.

We compute splitting fractions proportional to the requirements on parallel paths in the DAG $G$ to balance swarm sizes over locations and time. Formally, we capture this using a partial order [43]. We define the partial order $\preccurlyeq$ over $U$ such that $u \preccurlyeq v$ if a path exists from $u$ to $v$, i.e., if $u$ is an ancestor of $v$ or $v$ is a descendent of $u$. The sets of ancestors and descendants are $\mathcal{H}_u = \{v \mid v \preccurlyeq u\}$ and $\mathcal{F}_u = \{v \mid u \preccurlyeq v\}$, respectively. Let $p : V \to (0, 1]$ function such that $p(u)$ is the fraction of the swarm agents at node $u \in V$.

Algorithm 1 computes proportions $p$ backward from leaf nodes in $G$, and then computes the splitting fraction weights $w$. First, the method computes the number of ancestors and descendants of nodes for all nodes $u \in V$, lines 3-8. Moreover, we compute the total number of ancestors of all

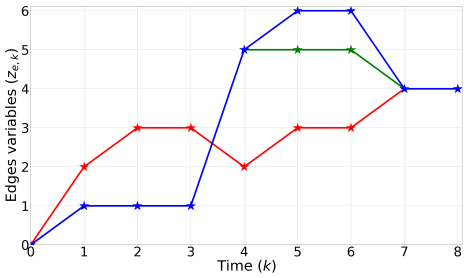


Fig. 3: Edges variable solution from case study 1.

leaf nodes $\mathcal{H}_{max}$ at line 6. For all leaf nodes $u$ with $\mathcal{N}_u^+ = \varnothing$, the proportion $p(u) = \frac{|\mathcal{H}_u|}{\mathcal{H}_{max}}$, line 9. For all other nodes, we proceed in inverse topological order (i.e., from leaves towards $u_0$) and compute $p(u) = \sum_{v \in \mathcal{N}_u^+} \frac{p(v)}{|\mathcal{N}_u^-|}$, line 10. Finally, the weights are computed as $w((v,u)) = \frac{p(u)}{p(v)}$ for all edges $(v,u) \in E$, line 11. It follows from Algorithm 1 that the returned weights $w$ satisfy the unit-sum constraints.

**Algorithm 1** Splitting and Merging Weights over DAG

1: **input:** $G = (V, E)$ ▷ DAG from MILP solution.
2: **output:** $G = (V, E, w)$ ▷ DAG with splitting fraction weights.
3: **for** $u \in V$ **do**
4: $|\mathcal{H}_u| = ancestors(\mathcal{G}, u)$
5: $|\mathcal{F}_u| = descendants(\mathcal{G}, u)$
6: **if** $|\mathcal{F}_u| = \varnothing$ **then** $\mathcal{H}_{max} = \mathcal{H}_{max} + |\mathcal{H}_u|$
7: **end if**
8: **end for**
9: For all $u \in leaves$, $p_u = |\mathcal{H}_i|/\mathcal{H}_{max}$
10: For all $u \in V \setminus leaves$, compute $p(u) = \sum_{v \in \mathcal{N}_u^+} \frac{p(v)}{|\mathcal{N}_u^-|}$ in inverse topological order.
11: For all $(u,v) \in E$ compute weights $w((v,u)) = \frac{p_u}{p_v}$, $\forall (v,u) \in E$ in a topological order.

**Protocol:** Finally, we define the communication-free protocol that agents implement to determine what sub-swarms they belong to when receiving split and merge actions. Communication is restricted between agents, but they receive global commands from the MTL-Planner, see Fig. 2. For *splitting*, all agents receive the command $\mathbf{s}_i \to \{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$ containing the altitude $h_{s_i}$ of the sub-swarm $\mathbf{s}_i$ to be split, heights $h_\ell$ for all resulting sub-swarms $\mathbf{s}_\ell$, $\ell \in \{i_1, \ldots, i_n\}$ and the splitting fraction weights $w((u,v))$, $v \in \mathcal{N}_u^+$, where node $u$ in DAG $G$ corresponds to the splitting action. Note that $n = |\mathcal{N}_u^+|$. If an agent is at altitude $h_{s_i}$, i.e., belongs to $\mathbf{s}_i$, then it samples the set of sub-swarms $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$ with probabilities $w((u,v))$ and moves to the altitude $h_{\bar{s}}$, where each neighbor node $v$ corresponds to a sub-swarm $\mathbf{s}_\ell$, $\ell \in \{i_1, \ldots, i_n\}$, and $\bar{s}$ is the randomly sampled sub-swarm. For *merging*, all agents receive the command $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\} \to \mathbf{s}_i$ containing the altitudes $h_{s_\ell}$ of sub-swarms $\mathbf{s}_\ell$ to be merged, $\ell \in \{i_1, \ldots, i_n\}$. In this case, all agents with altitudes $h_{s_\ell}$ move to altitude $h_{s_{i_1}}$ of resulting sub-swarm $\mathbf{s}_i$. Note that the protocol guarantees the provided swarm splitting fractions only in expectation. However, it does not require any communication between agents.

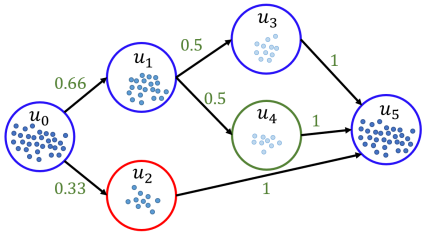


Fig. 4: Split and merge generated DAG for case study 1.

## IV. Case Studies

This section describes three case studies showing how swarm planning and control work and their performance. First, we consider a small mission specification to showcase the functionality of the high-level planner and then the low-level control. Second, we specify a more complex specification that requires the swarm to split and merge multiple times to show the correctness of the merging and splitting algorithm described in Sec. III-B. Finally, we test scalability and run-time performance by increasing the mission specification size gradually. All computation was performed on a PC with 20 cores at 3.7GHz with 64 GB of RAM. We used Gurobi [44] as MILP solver and CoppeliaSim [45] as simulator.

***Case Study 1:*** *(functionality showcase):* Here we consider the $Env$ in Fig. 1, the number of robots in the swarm $N = 80$. The swarm is tasked to satisfy the following mission specification $\phi = (\square_{[1,4]} q_1 \wedge \square_{[2,4]} q_3) \wedge \square_{[6,7]}(q_3 \wedge q_5 \wedge q_6) \wedge \square_{[8,9]} q_4$, with initial location of the swarm $q_0$, and maximum simultaneously existing swarms $N_s = 3$. Under these conditions, we get the solution plan shown in Fig. 3. Initially, the entire swarm is in $q_0$ at time $k = 0$. Next, it splits in two, represented here as red and blue lines traversing to states $q_1$ and $q_2$ at $k = 1$. Both swarms satisfy the first part of the specification, and then the blue swarm splits again at $k = 4$, where the green line represents the formation of a new swarm to satisfy the specification requesting sub-swarms in three different locations simultaneously. Lastly, the three swarms merge in $q_4$. The MILP solution computed a solution with an objective cost value of 39, and it took 0.01 seconds to compute, and 228 and 17 integer and binary variables, respectively.

From the MILP solution, we can compute the DAG $\mathcal{G} = (V, E, w)$ that describes the swarm splitting and merging actions in Sec. III-B and shown in Fig. 4. Note that the nodes follow the same color code as in Fig. 3. The agents in the initial swarm (blue) split into sub-swarm $u_1$ (blue) and $u_2$ (red). Then $u_1$ splits into two more sub-swarms $u_3$ (blue) and $u_4$ (green). Lastly, all of them merge in $u_5$, the same as in Fig. 3. We show the splitting fraction (probabilities) of the agents going into the different swarms on the edges. The weights are computed to balance the number of agents in each sub-swarm by taking into account the subsequent merging and splitting actions. An extended example is shown in Case study 2.

Finally, we compute the abstract state commands, as

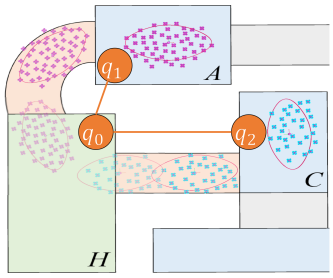


Fig. 5: Solution performed in CoppeliaSim with actions by the swarms when traversing corridors connecting two states.

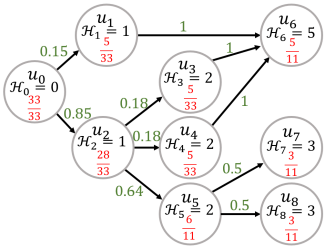


Fig. 6: Split and merge generated DAG for case study 2.

shown in Fig. 5, where two swarms are traveling from $q_0$ to $q_1$ and $q_2$ respectively, representing the first command from the solution of the MILP. Note that one of the corridors is curved, requiring the swarm to rotate and scale along it. For the second swarm, it is asked to rotate once it reaches the state to be able to traverse towards $q_3$. Simulation is performed in CoppeliaSim.

***Case Study 2**:* In this case study, we consider an extended example that showcases the functioning of the swarm splitting and merging algorithm. Consider the following mission specification $\phi = \square_{[1,2]}(q_1 \wedge q_2) \wedge \square_{[2,3]}(q_1 \wedge q_3 \wedge q_4 \wedge q_6) \wedge \square_{[3,4]}(q_5 \wedge q_3) \wedge \square_{[4,5]}(q_5 \wedge q_0 \wedge q_2)$, with maximum number of swarms $N_s = 4$ and initial state $q_0$. After computing the MILP solution, we build the merging and splitting actions DAG shown in Fig. 6. Following Algorithm 1. the number of ancestors $|\mathcal{H}_i|$ for every node is computed. We sum the number of the ancestor over all leaf nodes $\mathcal{H}_{max} = \sum_{u_i} |\mathcal{H}_i| = 5+3+3 = 11$, and then the expected proportion of the leaves is obtained $p_u = \mathcal{H}_i/\mathcal{H}_{max}$ as $5/11$, $3/11$, $3/11$, respectively. Next, from the leaves, we sum the proportions backward from the leaves to the root (shown as red values in the nodes). Finally, we compute each edge's splitting fractions (probabilities) from the root by considering the expected proportions in every node (green values on the edges). In Fig. 7, we show the agent distribution among the states. The horizontal axis represents the state and the vertical axis the number of agents in the swarm. We ran the same DAG 1000 times and considered two different initial number of agents $|\mathcal{A}_s| = 50$ (blue) and $|\mathcal{A}_s| = 150$ (red). Note that the smaller the swarm, the more difficult it becomes to split it in balanced proportions. Moreover, in both cases, node $u_2$ has the most significant fraction of agents, which makes sense since this node requires the swarm to split again into three new sub-swarms, and one of them will be split into the other two. In contrast, $u_1$ does not require a higher fraction since later it will only merge with other sub-swarms.

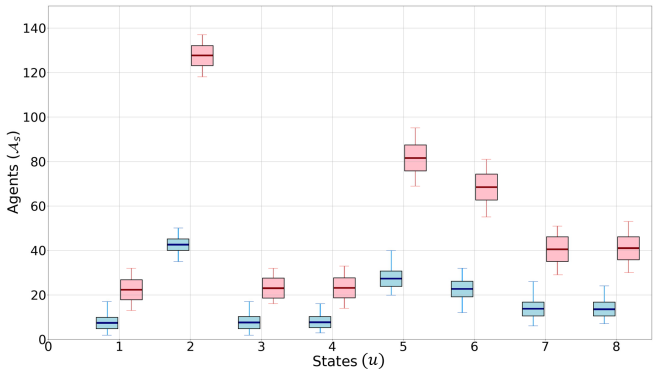


Fig. 7: Agents distribution statistics through the states when run 1000 times for initial number of agents of 150 and 50.

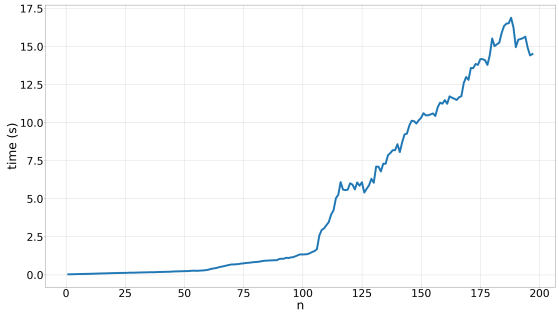


Fig. 8: Time performance by increasing mission $\phi$ gradually.

***Case Study 3:** (scalability and time performance):* Here we gradually increase the size of the mission specification to show the run time performance. The mission specification we use is $\phi = \bigwedge_1^n \mathcal{T}_I(\mathcal{X} \otimes \mathcal{X})$, where $\mathcal{T} \in \{\square, \Diamond\}$, $\otimes \in \{\wedge, \vee\}$, $\mathcal{X} \in \mathcal{Q}$ are variables randomly chosen, $n$ is an iterator that grows from 1 to 200, and the time interval of the temporal operator is defined randomly as $I = [n+4, \ldots, n + \text{rand}(1,5)]$. The initial state is $q_0$, and the split bound $N_s = 10$. The results are shown in Fig. 8, where we can see that time grows linearly for a small value of $n$ and then exponentially as $n$ becomes larger. Note that our MILP implementation is agent agnostic, and the only variables that affect the performance are the size of the transition system and the mission specification.

## V. Conclusions

We present a framework for controlling swarms subject to Metric Temporal Logic (MTL) constraints. We propose a MILP approach for computing the trajectories of sub-swarms in the environment, the splitting and merging actions if required, such that the MTL mission is satisfied with minimal division of the swarm over the mission horizon. We modify standard flow dynamics constraints using inequalities to capture the swarm motion in the environment. The solution of the MILP is used to build a DAG that captures the splitting and merging actions and is used to compute splitting fractions for resulting sub-swarms. A distributed randomized protocol ensures that the agents choose sub-swarms close to the computed splitting fraction with no inter-agent communication.

## REFERENCES


[1] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm robotics: Past, present, and future [point of view]," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.

[2] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial swarms: Recent applications and challenges," *Current Robotics Reports*, pp. 1–12, 2021.

[3] M. Majid, M. Arshad, and R. Mokhtar, "Swarm robotics behaviors and tasks: A technical review," *Control Engineering in Robotics and Industrial Automation*, pp. 99–167, 2022.

[4] M. S. Talamali, A. Saha, J. A. Marshall, and A. Reina, "When less is more: Robot swarms adapt better to changes with constrained communication," *Science Robotics*, vol. 6, no. 56, p. eabf1416, 2021.

[5] Y. Ozkan-Aydin and D. I. Goldman, "Self-reconfigurable multilegged robot swarms collectively accomplish challenging terradynamic tasks," *Science Robotics*, vol. 6, no. 56, p. eabf1628, 2021.

[6] 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.

[7] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science Robotics*, vol. 5, no. 49, p. eabe4385, 2020.

[8] I. Haghighi, S. Sadraddini, and C. Belta, "Robotic swarm control from spatio-temporal specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 5708–5713.

[9] F. Djeumou, Z. Xu, and U. Topcu, "Probabilistic swarm guidance subject to graph temporal logic specifications," in *Robotics: Science and Systems (RSS)*, 2020.

[10] S. Moarref and H. Kress-Gazit, "Decentralized control of robotic swarms from high-level temporal logic specifications," in *2017 international symposium on multi-robot and multi-agent systems (MRS)*. IEEE, 2017, pp. 17–23.

[11] R. Yan, Z. Xu, and A. Julius, "Swarm signal temporal logic inference for swarm behavior analysis," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3021–3028, 2019.

[12] R. Yan and A. Julius, "Distributed consensus-based online monitoring of robot swarms with temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9413–9420, 2022.

[13] J. Chen, H. Wang, M. Rubenstein, and H. Kress-Gazit, "Automatic control synthesis for swarm robots from formation and location-based high-level specifications," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8027–8034.

[14] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 320–330, 2007.

[15] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1189–1206, 2019.

[16] J. Chen, S. Moarref, and H. Kress-Gazit, "Verifiable control of robotic swarm from high-level specifications," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 568–576.

[17] S. Moarref and H. Kress-Gazit, "Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications," *Autonomous Robots*, vol. 44, no. 3, pp. 585–600, 2020.

[18] R. Yan and A. Julius, "A decentralized B & B algorithm for motion planning of robot swarms with temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7389–7396, 2021.

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

[20] G. A. Cardona, D. Saldaña, and C.-I. Vasile, "Planning for modular aerial robotic tools with temporal logic constraints," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2878–2883.

[21] 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.

[22] A. M. Jones, K. Leahy, C. Vasile, S. Sadraddini, Z. Serlin, R. Tron, and C. Belta, "Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications," *In International Symposium of Robotics Research. International Federation of Robotics Research,*, 2019.

[23] D. S. Hochbaum, "The pseudoflow algorithm: A new algorithm for the maximum-flow problem," *Operations research*, vol. 56, no. 4, pp. 992–1009, 2008.

[24] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 157–173.

[25] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.

[26] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[27] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[28] D. Saldana, B. Gabrich, G. Li, M. Yim, and V. Kumar, "Modquad: The flying modular structure that self-assembles in midair," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 691–698.

[29] S. Shahrokhi and A. T. Becker, "Stochastic swarm control with global inputs," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 421–427.

[30] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[31] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on robotics*, vol. 20, no. 5, pp. 865–875, 2004.

[32] N. Michael, C. Belta, and V. Kumar, "Controlling three dimensional swarms of robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 964–969.

[33] T. Setter, A. Fouraker, M. Egerstedt, and H. Kawashima, "Haptic interactions with multi-robot swarms using manipulability," *Journal of Human-Robot Interaction*, vol. 4, no. 1, pp. 60–74, 2015.

[34] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 916–923, 2018.

[35] I. Buckley and M. Egerstedt, "Infinitesimal shape-similarity for characterization and control of bearing-only multirobot formations," *IEEE Transactions on Robotics*, 2021.

[36] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 495–500, 2002.

[37] J. Cortés, "Global and robust formation-shape stabilization of relative sensing networks," *Automatica*, vol. 45, no. 12, pp. 2754 – 2762, 2009.

[38] A. Franchi, C. Masone, V. Grabe, M. Ryll, H. H. Bülthoff, and P. R. Giordano, "Modeling and control of uav bearing formations with bilateral high-level steering," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1504–1525, 2012.

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

[40] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

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

[42] G. A. Cardona and C.-I. Vasile, "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, 2022, pp. 223–238.

[43] B. A. Davey and H. A. Priestley, *Introduction to lattices and order*. Cambridge university press, 2002.

[44] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[45] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. [Online]. Available: www.coppeliarobotics.com"