2022 IEEE Conference on Control Technology and Applications (CCTA)
August 22-25, 2022. Trieste, Italy

Distributed Trajectory Planning for Multi-rotor UAVs with Signal Temporal Logic Objectives

Yash Vardhan Pant[1], Houssam Abbas[2], Rahul Mangharam[3]

***Abstract*— We develop a distributed motion planner for multi-robot systems with Signal Temporal Logic (STL) objectives. Existing approaches to STL-based motion planning are limited to fragments of STL that might not capture all desired behaviors or safety requirements. Focusing on the case of a fleet of multi-rotor Unmanned Aerial Vehicles (UAVs) jointly tasked with satisfying a given STL mission, we develop a distributed method, Fly-by-Distributed-Logic (FBDL), for trajectory planning. The proposed method generates trajectories that maximize the smooth robustness of STL specifications, where the associated optimization is solved in a distributed manner. This, to the best of our knowledge, is the first distributed method that can handle the full grammar of STL (bounded time). Simulation studies show the applicability of our approach to a variety of STL missions, including those with nested STL operators and the *Until* operator. We also compare its performance to a centralized approach and show that our method is computationally faster, but generally computes trajectories with lower robustness.**

2022 IEEE Conference on Control Technology and Applications (CCTA) | 978-1-6654-7338-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCTA49430.2022.9966096

I. Introduction

Signal Temporal Logic (STL) is proving to be an effective formalism for representing multi-robot missions that not only have spatial, but also temporal and inter-robot requirements [1], [2], [3]. Most methods for motion planning for multi-agent systems with STL specifications are centralized approaches, where a single computation resource generates trajectories (or computes control signals) for each agent. In many applications however, this may be impractical, e.g. in the case of Unmanned Aerial Vehicle (UAV)s managed by different operators but flying in the same airspace, the operators would want the UAVs to cooperate for safety purposes, but are unlikely to rely on a single operator to generate trajectories for all UAVs. Such applications necessitate the development of distributed motion planning of multi-agent systems with objectives specified as Signal Temporal Logic (STL) specifications. In this paper, we present one such approach for distributed trajectory generation of UAVs. As will be seen in later sections, the approach requires each the UAVs to iteratively solve an optimization and communicate their solution with other UAVs. Building upon the optimization solver presented in [4], we show how this distributed approach results in the UAVs jointly maximizing a notion of *robustness* of satisfying a STL specification.

Contributions of this work: We present Fly-by-Distributed-Logic (FBDL), a distributed algorithm for trajectory generation, that:

1) Has convergence guarantees (section V-B).
2) In addition to making a best effort to compute UAV trajectories that satisfy the STL specification, generates UAV trajectories that respect kinematic (velocity and acceleration) constraints.
3) Is applicable to specifications generated using the full grammar of STL with bounded temporal operators and possibly non-convex predicates, i.e. leverages the full expressivity of STL.

Through three multi- UAV simulation studies, we show the applicability of our approach and compare it to the centralized method in [1].

Outline of the paper: Section II presents an overview of both centralized and distributed approaches for motion planning with STL specifications. Section III presents the notation used in this paper, and formally states the problem we aim to solve. Section IV introduces the optimization formulation for UAV trajectory planning with STL specifications. The FBDL algorithm, i.e. our approach for solving this optimization is presented in section V. Simulation results over different types of STL specifications are in Section VI. Finally, we conclude and discuss the limitations of our approach and future work in section VII.

II. Related Work

The control and trajectory planning of systems with Signal (or Metric) Temporal Logic specifications has become a well studied problem in recent times. In this section, we cover some of the relevant literature on both centralized and distributed approaches for this problem.

A. Centralized methods

Most approaches that solve the problem of control of dynamical systems with STL specfications, when applicable to the case of multi-agent systems, work in a centralized manner. The works of [5], [6], [7] build upon [8] and develops an automatic encoding of an STL specification as a set of constraints in a Mixed Integer Linear Program (MILP). To overcome some of the computational limitations of such methods, gradient-based approaches have been developed in [9], [1] and have been applied to problems in multi-agent trajectory planning. Also finding applications for multi-agent planning with specifications similar to STL, [10] presents an approach that combines Linear Programming with SAT solving. A control barrier function-based approach [11] enables the development of closed-form control laws for systems

[1]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada yash.pant@uwaterloo.ca.
[2]School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA houssam.abbas@oregonstate.edu.
[3]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA {rahulm}@seas.upenn.edu.

with STL specifications. However this approach is applicable only to a fragment of STL. Recently, [12] developed a method that trains a Neural Network-based controller for satisfying STL specifications through data generated by using gradient-based methods.

B. Distributed methods

The problem of distributed control of multi-agent systems with STL specifications is one that has only recently gained attention. An early method is that of [13], where the authors develop a receding horizon control algorithm for communication-aware distributed motion planning of multi-agents systems. This however is applicable only to *reach-avoid* STL specifications. [14] presents a distributed method based on a multi-stage control scheme, and [3], [15] further develops a control barrier function-based approach. These methods are limited to a particular fragment of STL, and are applicable only to specifications with convex atomic predicates. As a consequence, collision avoidance (inter-agent or with obstacles) cannot be a part of the requirements and is dealt with using lower-level planners. The MILP-based distributed approaches of [2], [16] allows for collision avoidance objectives to be taken into account, however they also work with limited fragments of STL. [17] presents a framework for multi-agent motion planning, in a discretized workspace, with objectives similar to STL specifications. Our approach builds upon the formulation of [1], and through a gradient-based distributed optimization, allows for planning for multi-agent system with specifications that can involve the full STL grammar.

III. Preliminaries and problem statement

A. Notation

Let $\mathcal{D} = \{1, \dots, D\}$ denote the set of all UAVs. We use $\mathbf{p}_d : [0, T] \to \mathbb{R}^3$ to represent a trajectory (in the 3D position space) over a time interval $[0, T]$ for a UAV $d \in \mathcal{D}$. $\mathbf{p}_d(t) \in \mathbb{R}^3$ refers to the position of UAV d at time t. Similarly, we use $\mathbf{v}_d$ and $\mathbf{a}_d$ to refer to the velocity and acceleration trajectories respectively. For ease of notation, we drop the subscript d when it is not required. We use the shorthand $\dot{\mathbf{p}}(t) = \frac{d\mathbf{p}(t)}{dt}$, and correspondingly $\dot{\mathbf{p}}(t) = \mathbf{v}(t)$, $\ddot{\mathbf{p}}(t) = \dot{\mathbf{v}}(t) = \mathbf{a}(t)$, and so on. Finally, the full state of a UAV at time t is denoted by $\mathbf{x}(t) = \begin{bmatrix}\mathbf{p}(t), & \mathbf{v}(t), & \mathbf{a}(t)\end{bmatrix}' \in \mathbb{R}^9$. The full state trajectory is denoted by $\mathbf{x} : [0, T] \to \mathbb{R}^9$. We use X to denote the set of all trajectories, or signals, $\mathbf{x} : [0, T] \to X$.

B. Brief review of Signal Temporal Logic (STL)

Signal Temporal Logic (STL) [18] allows for succinct and unambiguous mathematical representation for a wide variety of desired system behaviors. Here, we present a short introduction to STL. The interested reader can refer to [18], [19] for a detailed explanation. Let $\mu : X \to \mathbb{R}$ be a real-valued function.This defines a *predicate* $q := \mu(\mathbf{x}) \geq 0$. Further, let $I \subset \mathbb{R}$ denote a non-singleton time interval, $\top$ the Boolean True value, $\neg$ and $\wedge$ the Boolean negation and AND operators, respectively, and $\mathcal{U}$ the Until temporal operator. A STL specification is recursively built from these using the following grammar:

$$\varphi := \top | q | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \mathcal{U}_I \varphi_2$$

Informally, $\varphi_1 \mathcal{U}_I \varphi_2$ means that φ_2 must hold at some point in I, and *until* then, φ_1 must hold without interruption. The disjunction ($\vee$), implication ($\implies$), Always ($\Box$) and Eventually ($\Diamond$) operators can be derived from the operators defined above.

Example 1: Consider two UAVs with position trajectories $\mathbf{p}_1$ and $\mathbf{p}_2$. Given a region of the airspace, $R \subset \mathbb{R}^3$, UAV 2 cannot enter the region until UAV 1 has visited it within the time interval $I = [0, 10]$ seconds. This can be represented using STL as: $\phi_1 = \neg(\mathbf{p}_2 \in R)\mathcal{U}_I(\mathbf{p}_1 \in R)$. Both UAVs must also respect a speed limit of $5ms^{-1}$, which corresponds to the specification $\phi_2 = \bigwedge_{d=1}^{2} \Box_I(||\mathbf{v}_d|| \leq 5)$. Since the two UAVs must respect both these requirements, we can combine them using the logical AND operator: $\varphi = \phi_1 \wedge \phi_2$.

In the example above, the specification φ has a time horizon of $H_\varphi = 10$ seconds, i.e. in order to evaluate if φ holds, we need trajectories of time duration at least 10 seconds. See [5] for details. In the rest of this paper, we only consider specifications with a bounded time horizon.

1) Robustness of STL formulae: For every STL specification φ, we can construct a *robustness function* [20] by following the grammar of STL. It returns a *robustness* value $\rho_\varphi(\mathbf{x})$ for this formula, with respect to the signal $\mathbf{x}$ that it is defined over, and has the important following property:

Theorem 3.1: [20] For any $\mathbf{x}$ and STL formula φ, if $\rho_\varphi(\mathbf{x}) < 0$ then $\mathbf{x}$ violates φ, and if $\rho_\varphi(\mathbf{x}) > 0$ then $\mathbf{x}$ satisfies φ. The case $\rho_\varphi(\mathbf{x}) = 0$ is inconclusive.

Thus, the degree of satisfaction or violation of a specification is indicated by the robustness value. There also exists a *smooth*, or continuously differentiable robustness function [9], [1].

Theorem 3.2: [9] The *smooth robustness* of a STL formula φ, $\tilde{\rho}_\varphi$ is a continuously differentiable function such that for any trajectory $\mathbf{x}$, $|\rho_\varphi(\mathbf{x}) - \tilde{\rho}_\varphi(\mathbf{x})| \leq \delta_\varphi$ where $\delta_\varphi > 0$ is a tune-able constant.

Consequently (from Theorem 3.1), $\tilde{\rho}_\varphi(\mathbf{x}) > \delta_\varphi$ implies $\mathbf{x}$ satisfies φ. Later, we will use this smooth robustness function as an objective to be maximized for generating trajectories that robustly satisfy a given STL specification.

C. Problem Statement

In this paper, we aim to develop a method for solving the following problem:

Problem 1 (Distributed trajectory planning): Develop a *distributed* algorithm that, given a multi-UAV mission φ, generates a $\mathbf{x}_d$ for every UAV $d \in \mathcal{D}$ such that:

1) *Mission satisfaction:* $(\mathbf{x}_1, \dots, \mathbf{x}_D) \models \varphi(\mathbf{x}_1, \dots, \mathbf{x}_D)$, i.e. satisfy the given specification and,
2) *Kinematic feasibility of trajectories:* $\mathbf{v}_d \in V$, $\mathbf{a}_d \in A \, \forall d \in \mathcal{D}$, i.e. resulting trajectories have velocities and accelerations within desired bounds $V \subset \mathbb{R}^3$ and $A \subset \mathbb{R}^3$ respectively[1].

[1]For simplicity, we assume all UAVs have homogeneous kinematic constraints. Our method works in the heterogeneous case as well.

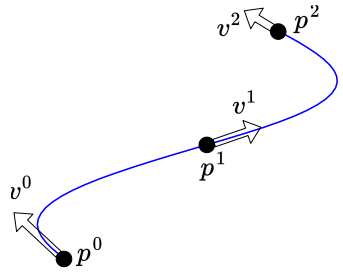

Fig. 1. A trajectory consisting of two segments of jerk minimizing splines, connecting position waypoints p^0 to p^1, and p^1 to p^2 respectively. The arrows show the velocities at these waypoints, v^i, $\forall i \in \{0, 1, 2\}$. Each segment corresponds to a duration of τ seconds, resulting a total trajectory duration of 2τ seconds.

In this *distributed* setting, each UAV can only make a decision (or generate trajectories) for itself. UAVs can communicate with each other under the following assumption:

Assumption 3.1 (Ideal inter-UAV communication): In this setting, each UAV can communicate instantaneously with each other in a lossless manner.

IV. Formulating the trajectory generation optimization, and a centralized baseline

Here, following the setup in [1], we first formulate the problem of generating UAV trajectories that satisfy a STL specification as an optimization to maximize smooth robustness over these UAV trajectories. Next, we will introduce our approach to solve this optimization in a distributed manner, as opposed to [1] which does so via a centralized approach.

A. UAV Trajectory Planning

Similar to [1], for each UAV, we are interested in generating trajectories that are composed of jerk-minimizing (for multi-rotor UAVs) splines [21] between *waypoints*. Figure 1 shows one such trajectory composed of two segments. Here, $p^0 \in \mathbb{R}^3$ and $v^0 \in \mathbb{R}^3$ represent the initial (at time 0) position and velocity waypoints for the UAV. The segment between one waypoint, given by position $p^0 = [p^0_x, p^0_y, p^0_z]^\top$ and velocity $v^0 = [v^0_x, v^0_y, v^0_z]^\top$, and another with desired position $p^1 = [p^1_x, p^1_y, p^1_z]^\top$, is a trajectory (see figure 1) of fixed time duration τ with ($\forall l \in \{x, y, z\}$, $\forall t \in [0, \tau]$)[2] given by [21]:

$$\begin{bmatrix} \mathbf{p}_l(t) \\ \mathbf{v}_l(t) \\ \mathbf{a}_l(t) \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + v^0_l t + p^0_l \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + v^0_l \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t \end{bmatrix} \tag{1}$$

Here, α, β and γ are linear functions of p^0, v^0 and p^1 (with parameter τ) [21]. We assume that the start and end accelerations are zero, i.e. $a^0 = a^1 = 0_{3\times 1}$ and end velocity v^1 is *free*. For brevity, we omit further details here. The interested reader can refer to [1], [21] for more details.

1) The Trajectory planning optimization:: The Fly-by-Logic method [1] formulated an optimization that generates trajectories for multiple UAVs tasked with satisfying a STL specification φ. The optimization variables here are a sequence of $N+1$ position waypoints in 3D space for each UAV ($d \in \mathcal{D}$) $p^{0:N}_d = [p^0_i, \ldots, p^N_i]$, where total flight time $N\tau \geq \text{hrz}(\varphi)$. The full trajectory for a UAV d, composed of N segments (1) connecting the $N+1$ waypoints, is given by $\mathbf{x}_d = [\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z, \mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z, \mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z]' : [0, N\tau] \rightarrow \mathbb{R}^9$. The objective is to select these waypoints such that the resulting trajectories $\mathbf{x}_i$, $\forall i \in \mathcal{D}$ maximize the smooth robustness (recall Theorem 3.2) of the specification $\tilde{\rho}_\varphi$.

$$\max_{p^{0:N}_1, \ldots, p^{0:N}_D} \quad \tilde{\rho}_\varphi(\mathbf{p}_1, \ldots, \mathbf{p}_D) \tag{2a}$$

$$\text{s.t. } \forall d = 1, \ldots, D, \forall j = 0, \ldots, N-1$$

$$\text{LB}_{\text{vel}}(v^j_d) \leq p^{j+1}_d - p^j_d \leq \text{UB}_{\text{vel}}(v^j_d) \tag{2b}$$

$$\text{LB}_{\text{acc}}(v^j_d) \leq p^{j+1}_d - p^j_d \leq \text{UB}_{\text{acc}}(v^j_d) \tag{2c}$$

This is a non-convex optimization, with linear constraints [1]. The UAV trajectories $\mathbf{p_d}$, when discretized in time[3], are linear functions of the position waypoints $p^{0:N}_d$, i.e. $\mathbf{p_d} = L(p^{0:N}_d)$ (refer to [1] for details). The constraints (2) are linear in the waypoints, and ensure kinematic feasibility of (see problem 1) the resulting trajectories. The interested reader can refer to the appendix VII-A for further details on the constraints.

2) Fly-by-Logic (FBL): Baseline centralized solution: The work in [1], solves the optimization (2) in a centralized manner, using the off-the-shelf non-convex optimization solver IPOPT [22]. We restate a key result from [1] here:

Theorem 4.1 (STL satisfaction, kinematic feasibility [1]): A feasible solution to the optimization (2) that also achieves an objective $\tilde{\rho}_\varphi \geq \delta_\phi$ (Theorem 3.2) generates trajectories $\mathbf{p_1}, \ldots, \mathbf{p_D}$ such that they:

1) Satisfy the STL specification φ
2) Have bounded velocity and acceleration(along every axis of motion $l \in \{x, y, z\}$) such that $\forall t \in [0, \text{hrz}(\varphi)]$: $\mathbf{v_d} \in V$, $\mathbf{a_d}(t) \in A$ for each UAV $d \in \mathcal{D}$.

Solving the optimization (2) approach hence serves as a basis for a solution to problem 1. Next, we present the distributed approach to do so.

V. Fly-by-Distributed-Logic (FBDL): Distributed trajectory planning with STL specifications

In order to solve problem 1, we build upon the optimization formulated in (2) and solve it in a distributed manner. The proposed approach, which we call *Fly-by-Distributed-Logic* (FBDL), builds upon the non-convex solver of [4].

A. Overview of the approach

FBDL aims to solve (2) by iteratively making UAVs solve a convex sub-problem, and then communicate these solutions to other UAVs. For compactness of representation, we introduce some new notation. Let $y_d = p^{0:N}_d$ represent the waypoints for UAV d. Also, let $r > 0$ denote the r^{th} iteration of an algorithm, and $y_d[r-1] = p^{0:N}_d[r-1]$ be the (proposed) waypoints for UAV d obtained from the previous iteration. Let $y[r-1] = [y_1[r-1], \ldots, y_D[r-1]]$ be the

[2]With some abuse of notation, here we use the subscript $l \in \{x, y, z\}$ to denote axis of motion in 3D space, and not the index for a UAV.

[3]The trajectories $\mathbf{p}_d$ here are discrete-time, with sampling time $dt \ll \tau$.

collection of waypoints for all UAVs $d \in \mathcal{D}$ at iteration $r-1$. Finally, let Y be the polyhedral feasible set for the trajectory planning optimization (2), defined by (2b) and (2c). We now define this convex objective for UAV d, with variable y_d and information (constants) $y[r-1]$ from the previous iteration:

$$f_d(y_d, y[r-1]) = -\langle \nabla_{y_d} \tilde{\rho}_\varphi(L(y[r-1])), y_d - y_d[r-1] \rangle + \frac{\alpha}{2} ||y_d - y_d[r-1]||_2^2 \quad (3)$$

Here, $\alpha > 0$ is a constant, L is a linear mapping from waypoints to discrete-time trajectories (section IV-A.1). The following Lemma states two useful properties:

Lemma 5.1: The function $f_d(y_d, y[r-1])$ is: 1) Continuously differentiable, and 2) Strongly convex[4] with respect to y_d. Also, 3) $\nabla_{y_d} \tilde{\rho}_\varphi(L(y[r-1]))$ is Lipschitz continuous on Y for all $y_d \in Y$.

Proof sketch: Note that f_d is convex: the first term is linear in the variable y_d since the gradient term $\nabla_{y_d} \tilde{\rho}_\varphi(L(y[r-1]))$ is a constant (vector), and the second is quadratic in y_d. We can set the constant $\alpha > 0$ to a large enough value such that the function is also strongly convex [4]. 1) and 3) above follow from the definition of smooth robustness, which is continuously differentiable [9], and from that fact that the constraint set Y is convex (in fact linear) and compact.

B. Algorithm for distributed trajectory planning

Intuitively, minimizing f_d (3) at iteration r corresponds to taking a step along the gradient of $\tilde{\rho}_\varphi$ (in the space of variables for UAV d). Note the $-$ sign in (3). This is since we aim to *maximize* the smooth robustness in (2) in order to generate trajectories that satisfy the STL specification φ (Theorem 3.2). To do so across all UAVs in a distributed and cooperative manner, taking inspiration from [4] we introduce the following trajectory generation algorithm:

Algorithm 1 is an iterative algorithm that outputs a sequence of waypoints $p_d[r^*]$ for each UAV d involved in the STL specification φ. Each UAV starts with an initial solution $y[0]$, obtained by solving a linear feasibility problem with equations (2b) and (2c) as constraints. Note, unlike [2], [13] where each UAV solves an optimization one after the other, here multiple UAVs (in the set $S[r]$) solve their (convex) optimizations in parallel at each iteration (see line 6). In line 7, the UAVs that solved an optimization at this iteration update their variables. Here, $\gamma[r]$ is an adaptive step-size that meets the following conditions: 1) $\gamma[r] \in (0, 1]$, 2) $\sum_{r=0}^{\infty} \gamma[r] = +\infty$, 3) $\limsup_{r\to\infty} \gamma[r] < C$ where C is a positive constant. UAVs ($\mathcal{D} \setminus S[r]$) that did not solve an optimization at iteration r simply re-use their variables from the previous iteration. The UAVs then broadcast their updated variables to each other (line 9). The UAVs keep track of the best solution (lines 10-12), and the trajectories can be flown if they satisfy the specification φ, i.e. when $\rho_\varphi(L(y[r*])) > 0$. We first present a result about the convergence properties of the algorithm:

[4]For $c > 0$, $f(x)$ is strongly convex if $f(x) - (c/2)||x||^2$ is convex [23](chapter 9).

Algorithm 1 FBDL: Distributed trajectory planning with STL specifications

Data: Initial positions of UAVs, $p_d^0 \, \forall d \in \mathcal{D}$, Integer $r_{\max} \gg 0$

Result: Waypoints $y_d[r^*]$ for each UAV $d \in \mathcal{D}$ and trajectories $\mathbf{x}_d = L(y_d[r^*])$

1: Initialization: $y[0]$, set $r = 0$, $y_d[r^*] = y_d[0] \, \forall d \in \mathcal{D}$
2: **while** $r < r_{\max}$ **and** $|\rho_\varphi[r] - \rho_\varphi[r-1]| \geq$ Threshold **do**
3: Increment r, $r \leftarrow r + 1$
4: For each UAV $d \in \mathcal{D}$, compute the gradient $\nabla_{y_d} \tilde{\rho}_\varphi(L(y[r-1]))$
5: Choose subset of UAVs $S[r] \subseteq \mathcal{D}$
6: Compute $y_d^* = \text{argmin}_{y_d \in Y} f_d(y_d, y_d[r-1]) \, \forall d \in S[r]$
7: Update $y_d[r] = y_d[r-1] + \gamma[r](y_d^* - y_d[r-1]) \, \forall d \in S[r]$
8: Update $y_d[r] = y_d[r-1] \, \forall d \notin S[r]$
9: Broadcast all updated solutions to form $y[r]$
10: **if** $\rho_\varphi(L(y[r])) > \rho_\varphi(L(y[r-1]))$ **then**
11: $y_d[r^*] = y_d[r]$ {Store best solution}
12: **end if**
13: **end while**
14: **return** $p_d[r^*] \, \forall d \in \mathcal{D}$

Theorem 5.1 (Convergence of FBDL): Let $S_1, \ldots, S_m$ be non-overlapping partitions of the set (of UAVs) $\mathcal{D} = \{1, \ldots, D\}$. If: 1) $\gamma[r]$ meets the conditions above, and 2) $S[r] = S_i$, where i is picked in a cyclic (one-after-other) manner from $1, \ldots, m$, then algorithm 1 converges to a stationary point of $\tilde{\rho}_\varphi$.

This follows directly from the convergence properties of the non-convex optimization algorithm in [4] (see Theorem 1) when the conditions defined here and in Lemma 5.1 are satisfied. Next, we discuss the properties of the trajectories generated via algorithm 1.

Theorem 5.2 (STL satisfaction and kinematic feasibility): If the resulting trajectories (from computed waypoints $p_d[r^*]$) have a robustness value $\rho_\varphi(L(y[r^*])) > 0$, then the trajectories satisfy the STL specification φ and are also kinematically feasible (see problem 1).

This is a consequence of Theorem 3.1 and the second point in Theorem 4.1, and shows that algorithm 1 is a best-effort solution to problem 1.

Remark: In this work, we only consider the problem of UAV trajectory generation with STL specifications. However, since we base algorithm 1 on the non-convex optimization solver in [4], it is (with minor modifications) applicable to linear time invariant (LTI) systems of the form $x[k+1] = Ax[k] + Bu[k]$. We leave a detailed exploration of this to future work.

VI. Simulation studies

We carried out three simulation studies, where we compare our distributed approach, FBDL, to the centralized approach, FBL [1]. Video playbacks for these simulations can be found at: `https://bit.ly/3nQNmnu` (for FBDL) and `https://bit.ly/3xMroXo` (for FBL).

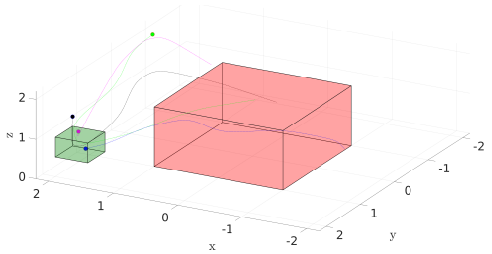

Fig. 2. Workspace for the multi-UAV timed reach-avoid study, and trajectories for $D = 4$ UAVs. The goal set G is in green, and the unsafe set U is in red. A (3D) playback of these trajectories can be viewed at `https://youtu.be/c5w-avQKc7M`, and with a top-down view at `https://youtu.be/HQyhVLEklxQ`.

A. Simulation setup

All simulations were performed on a laptop with a i7-9750 (2.6GHz) processor and 16 GB RAM, running Ubuntu 18.04. Algorithm 1 was implemented in MATLAB R2019b, with qpOASES [24] as the solver for the convex optimization associated with minimizing (3) (see line 6 of algorithm 1). The gradient computation (line 4 of algorithm 1) was done using Casadi [25]. For all problems, the UAV kinematic bounds were $V = [-0.75, 0.75] \times [-0.75, 0.75] \times [-0.75, 0.75]$ and $A = [-1, 1] \times [-1, 1] \times [-1, 1]$. The constant α in (3) was set through trial and error to different values (order of 10^{-3}) for each of the three studies that follow. The time varying step size was set to $\gamma[r] = 0.99^r + 0.001$, which meets the conditions in Theorem 5.2. For the optimization formulation in (2), the jerk minimizing spline segments have a duration of $\tau = 1s$, and the trajectories are discretized with a discretization time of $dt = 0.05s$.

B. Multi-UAV timed reach-avoid

In order to benchmark the performance of our approach, we first compare it to Fly-by-Logic, the centralized method in [1]. Here, D UAVs (where $D \in \{2, 4, 8\}$) are tasked with *reaching* a goal set $G = [1.5, 2] \times [1.5, 2] \times [0.5, 1]$ within 8 seconds, while also *avoiding* an unsafe set $U = [-1, 1] \times [-1, 1] \times [0, 1.5]$ throughout this time. Additionally, each UAV should also be at least 0.2m away from each other, in the inf-norm sense. The workspace is shown in figure 2.

These requirements can be encoded in STL as:

$$\varphi_{RA} = \bigwedge_{d=1}^{D} (\Box_{[0,8]} \neg(p_d \in \mathrm{U}) \wedge \Diamond_{[0,8]} (p_d \in \mathrm{G})) \\ \wedge \bigwedge_{d,d',d \neq d'} \Box_{[0,8]} (||p_d - p_{d'}|| \geq 0.2) \tag{4}$$

To solve this problem using algorithm 1, we divide the D UAVs into two groups S_1 and S_2 such that $S_1 \cup S_2 = \{1, \ldots, D\}$. Here S_1 contains the first $D/2$ UAVs, and S_2 contains the remaining $D/2$, and at each iteration, the algorithm selects group S_1 or S_2 (in an alternating manner) to solve their optimizations (see line 5, algorithm 1 and Theorem 5.1).

1) Simulation results: For each value of $D \in \{2, 4, 8\}$, we randomly generate 20 initial positions for the UAVs and generate trajectories to maximize the robustness of φ_{RA}.

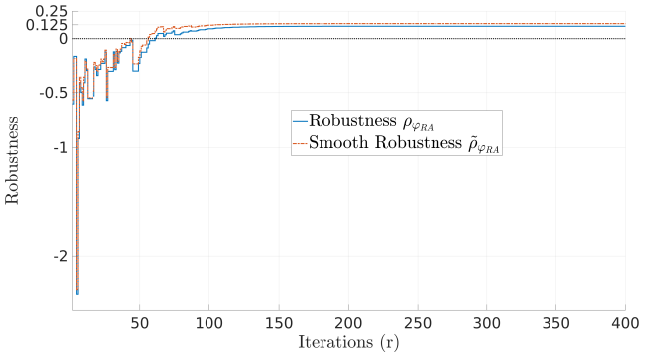

Fig. 3. Evolution of robustness (and smooth robustness) versus iteration for algorithm 1 generating trajectories to satisfy φ_{RA} 4 with $D = 4$ UAVs. At iteration $r = 61$, each UAV generates trajectories that together satisfy φ_{RA}. Also seen is the algorithm converging to a robustness value of nearly 0.125, which for this instance of the simulation is the global maxima of robustness (limited by the initial positions of the UAVs).

Table I shows a comparison of the centralized method (FBL) with our proposed distributed approach (FBDL) on:

1) Computation time[5] to find a satisfying solution (T_{sat}), or the first instance where the solver finds trajectories $\rho_{\varphi_{RA}} > 0$. As seen in the table, this computation time increases with the number of UAVs D. The distributed method is on average faster than the centralized approach.
2) Computation time for the solver to converge to the maximum robustness solution and terminate (T_{term}). A trend similar to T_{sat} is seen here.
3) Robustness of computed trajectories after convergence ($\rho_{\varphi_{RA}}$). Here, the centralized method outperforms FBDL in each case and generates trajectories with a higher robustness for each setting of D.

Figure 3 shows the evolution of the robustness (and smooth robustness) of computed trajectories after each iteration of algorithm 1.

2) Discussion: The simulation results for this benchmark show that the distributed method developed here has performance that compares well with the centralized approach. For all 20 initial positions, and for each setting of D, FBDL generates trajectories that satisfy the specification $\rho_{\varphi_{RA}}$ in (4) (as does the centralized approach). FBDL also takes less time than the centralized approach to find these satisfying trajectories[6]. However, the centralized approach outperforms FBDL in terms of the robustness value. This is to be expected as in general centralized optimization approaches outperform distributed ones. In particular, the centralized approach (FBL) guarantees convergence to a local optimum, while our distributed method only guarantees convergence to a stationary point (Theorem 5.1). A comparison of the

[5]These are the (accumulated) computation times for the optimization step (line 6) in algorithm 1. Computation times for computing the gradient (line 4) are negligible.

[6]The computation performance of FBL can be improved upon by using the HSL linear solvers [26] (with IPOPT), however due to compatibility issues, we could not use them in this study and uses the default linear solver instead.

centralized approach to a MILP-based method on this benchmark can be found in [1].

C. Multi-UAV timed Reach-avoid with the Until operator

Building on the timed reach-avoid simulation study, we test our method on a multi-UAV specification involving the *Until* operator to demonstrate applicability of our approach to the full grammar of STL. Here, we consider the case of $D = 2$ UAVs tasked with the goal set G (see figure 2) within $6s$, while also avoiding the unsafe set U and each other throughout. Additionally, UAV 2 which starts closer to G should not enter it before UAV 1. This final requirement is encoded using the *until* operator. The full mission specification is:

$$\varphi_{RA \wedge \mathcal{U}} = \bigwedge_{d=1}^{2} (\Box_{[0,6]} \neg (p_d \in U) \wedge \Diamond_{[0,6]} (p_d \in G)) \\ \wedge \Box_{[0,6]} (||p_1 - p_2|| \geq 0.2) \wedge (\neg (p_2 \in G) \mathcal{U}_{[0,6]} (p_1 \in G)) \tag{5}$$

Here, the $D = 2$ UAVs are divided into two groups $S_1 = \{1\}$ and $S_2 = \{2\}$ for using algorithm 1 (see line 5).

1) Simulation results and discussion: Across 20 pairs of random initial positions for the UAVs, FBDL generates trajectories that satisfy the specification (5). Averaged over these 20 runs, the computation times are $T_{\text{sat}} = 0.35s$ and $T_{\text{term}} = 0.96s$. The centralized approach (FBL) on the other hand has computation times of $T_{\text{sat}} = 1.83s$ and $T_{\text{term}} = 19.8s$. The maximum achieved robustness (averaged over all runs) FBDL is 0.045 and 0.149 for the FBL. This is similar to the trend observed in the timed reach-avoid simulation study, where the distributed approach is faster than the centralized approach, but generates trajectories with a lower robustness. A 3D visualization of the generated trajectories for the 2 UAVs for one pair of initial positions is at `https://youtu.be/cboVcUGFGdM` and a top-down view is at `https://youtu.be/CD1WGqidX8w`.

D. Case study: Multi-UAV persistent surveillance

Finally, similar to the case study in [2][7], we consider a mission that is expressed using a STL specification involving nesting of temporal operators. For $D = 3$ UAVs, we consider the requirements where the UAVs must:

1) *Persistent surveillance:* Visit sets G_j, $j = 1, 2, 3$ (in any order) within $16s$ and remain inside each set for at least $2s$. For each UAV, with position p_d, $d = 1, 2, 3$, this requirement is specified as follows

$$\varphi_{PS} = \bigwedge_{d,j} (\Diamond_{[0,16]} \Box_{[0,2]} (p_d \in G_j))$$

2) *Maintain coherence:* The UAVs must pairwise be within a distance of $1.2m$ to each other at all times. This is referred to as *persistence* in [2], and could for example to allow the UAVs to communicate with each other in real time.

[7]We consider a 3D workspace as opposed to the 2D workspace in [2]

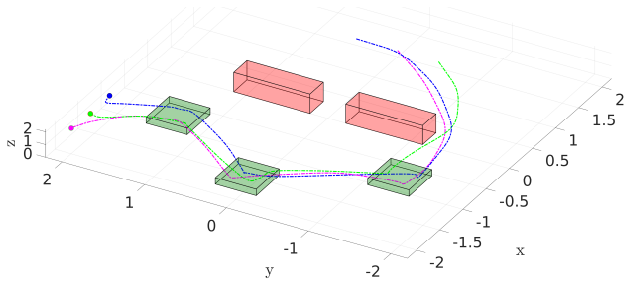

Fig. 4. Workspace and generated trajectories for the multi-UAV persistent surveillance problem. The sets to visit, G_j, $j \in \{1, 2, 3\}$ are in green, and the sets to avoid U_j, $j \in \{1, 2\}$ are in red. Playback of these trajectories is at `https://youtu.be/MlnSNciAFEs`.

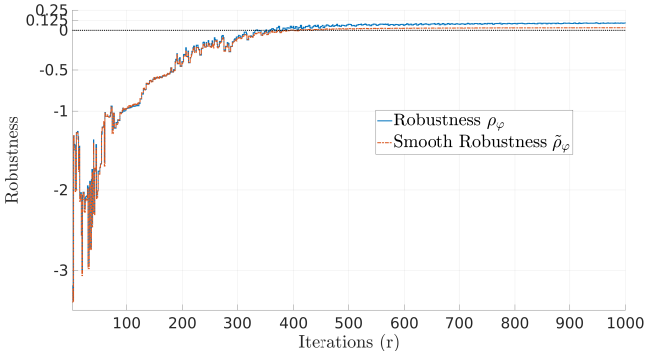

Fig. 5. Evolution of robustness (and smooth robustness) versus iteration for algorithm 1 generating trajectories to satisfy the specification in 6 with $D = 3$ UAVs. At iteration $r = 333$, each UAV generates trajectories that together satisfy $\varphi_{mission}$.

$$\varphi_{COH} = \bigwedge_{d,d',d \neq d'} (\Box_{[0,18]} (||p_d - p_{d'}|| \leq 1.2))$$

3) *Collision avoidance:* For safety, the UAVs must be pairwise no closer than $0.2m$. They must also avoid 3 unsafe sets U_j, $j = 1, 2$.

$$\varphi_{CA} = \bigwedge_{d,d',d \neq d'} (\Box_{[0,18]} (||p_d - p_{d'}|| \geq 0.2)) \\ \wedge \bigwedge_{d,j} (\Box_{[0,18} \neg (p_d \in U_j))$$

The workspace and the sets for this mission are shown in figure 4. The overall mission specification is:

$$\varphi_{mission} = \varphi_{PS} \wedge \varphi_{COH} \wedge \varphi_{CA} \tag{6}$$

This mission has a specification horizon of $hrz(\varphi) = 18s$. For generating trajectories via algorithm 1, we divide the $D = 3$ UAVs into three groups $S_1 = \{1\}$, $S_2 = \{2\}$ and $S_3 = \{3\}$.

Figure 5 shows the evolution of robustness with iterations for algorithm 1 generating trajectories to satisfy the multi-UAV persistent surveillance mission in section VI-D.

1) Simulation results and discussion: We solve for this specification for 5 randomly generated initial positions, generated uniformly centered around $[1.5, 1, 0.5]'$, $[1.5, 0, 0.5]'$ and $[1.5, -1, 0.5]'$ with intervals of width $[-0.5, 0.5]$ along each axis. Algorithm 1 generates trajectories that on average

TABLE I

SUMMARY OF SIMULATION RESULTS, MEAN $\pm$ VARIANCE OVER 20 RUNS WITH RANDOM INITIAL POSITIONS, FOR THE MULTI-UAV TIMED REACH-AVOID PROBLEM WITH NUMBER OF UAVS D RANGING FROM 2 TO 8. T_{SAT} IS THE COMPUTATION TIME (SECONDS), TAKEN BY PROPOSED DISTRIBUTED ALGORITHM (FBDL) AND THE BASELINE CENTRALIZED ALGORITHM (FBL), TO GENERATE TRAJECTORIES THAT SATISFY φ_{RA}. T_{TERM} IS THE TIME FOR THE ALGORITHMS TO TERMINATE, AND $\rho_{\varphi_{RA}}$ IS THE ROBUSTNESS VALUE ACHIEVED.

D	$FBL : T_{\text{sat}}(s)$	$FBDL : T_{\text{sat}}(s)$	$FBL : T_{\text{term}}(s)$	$FBDL : T_{\text{term}}(s)$	$FBL : \rho_{\varphi_{RA}}$	$FBDL : \rho_{\varphi_{RA}}$
2	0.135 ± 0.038	$\mathbf{0.084 \pm 0.048}$	2.763 ± 2.2024	$\mathbf{0.931 \pm 0.029}$	$\mathbf{0.209 \pm 0.052}$	0.188 ± 0.059
4	0.609 ± 0.333	$\mathbf{0.276 \pm 0.074}$	6.994 ± 4.323	$\mathbf{1.9141 \pm 0.084}$	$\mathbf{0.121 \pm 0.059}$	0.059 ± 0.036
8	3.033 ± 0.697	$\mathbf{1.081 \pm 0.408}$	23.468 ± 11.709	$\mathbf{6.672 \pm 4.049}$	$\mathbf{0.064 \pm 0.023}$	0.023 ± 0.020

satisfy the specification in $T_{\text{sat}} = 14.67s$. With an average termination time of $T_{\text{term}} = 44.05s$, the maximum value of robustness achieved (averaged over the 5 runs) via FBDL is 0.094. On average, the centralized approach takes $T_{\text{sat}} = 33.0s$, $T_{\text{term}} = 64.01s$ and achieves a maximum robustness value of 0.096. These trends are similar to those observed in the other simulation studies. This example was inspired from [2], where a planar ($2D$) workspace is considered, but with a specification horizon that is twice as long $40s$. Since there approach solves a feasibility problem, they generate with a robustness value of 0. While computation times across these slightly different problems (and on different computation platforms) are incomparable, the computation times reported in [2] of $59.4s$ for their distributed method and $341.83s$ for a centralized (MILP-based) approach respectively show that the computational burden of FDBL is similar to [2].

VII. CONCLUSION

Summary. We developed Fly-by-Distributed-Logic, an convex optimization-based, distributed approach for UAV trajectory planning with Signal Temporal Logic (STL) specification. To the best of our knowledge, this is the first such distributed method that can handle specifications leveraging the full grammar of STL. We showed the convergence guarantees, and the ability of the method to generate kinematically feasible trajectories. Extensive simulations showed the performance of our approach, demonstrating that it can successfully generate trajectories that satisfy various types of STL specifications, and do so while taking less computation time than a state-of-the-art centralized approach.

Limitations and future work. Here, we address some of the main limitations of our method and possible ways to improve upon these:

1) While FBDL successfully generated trajectories that satisfied the STL specifications in all 45 simulation runs, over the 3 simulation studies, the algorithm is not *complete*, i.e., even if a satisfying solution exists, there is no guarantee than FBDL will find it. In practice, this can be mitigated (but not entirely overcome) by multi-starting the optimization [9].
2) Compared to the centralized approach, FBL (also not a complete algorithm), FBDL on average returned trajectories with a lower robustness value. Based on a distributed optimization algorithm [4], FBDL only guarantees convergence to a stationary point of the *smooth* robustness function, which is not necessarily a satisfying solution (even if it was observed for the simulations).
3) A practical limitation of FBDL is in the communication requirements that it imposes, namely nearly instantaneous and lossless communication in a fully-connected network. Future work will aim to address these by relaxing the *broadcast* communication requirements based on distances/coupling between agents in the STL specification of interest. Another possible method is to use a *block-cyclic* scheme instead of a cyclic scheme for selecting which groups of UAVs solve the optimization at a given iteration. This would reduce the across-group communication. Initial results for this show promise, and a brief overview is presented in appendix VII-B.

Future work will also include extending our approach to work with planar (as opposed to $3D$) workspaces, and hence allow for a comparison to existing approaches like [2].

Conclusion. The method developed here shows promise, and in its current form we envision it as a *one-shot* trajectory generation approach. Future improvements will allow us to use it as an *online*, predictive trajectory planner.

REFERENCES

[1] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: control of multi-drone fleets with temporal logic objectives," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE Press, 2018, pp. 186–197.

[2] A. T. Büyükkoçak, D. Aksaray, and Y. Yazicioglu, "Distributed planning of multi-agent systems with coupled temporal logic specifications," in *AIAA Scitech 2021 Forum*, 2021, p. 1123.

[3] L. Lindemann and D. V. Dimarogonas, "Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 89–94.

[4] M. Razaviyayn, M. Hong, Z.-Q. Luo, and J.-S. Pang, "Parallel successive convex approximation for nonsmooth nonconvex optimization," *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[5] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conf. on Decision and Control*, Dec 2014, pp. 81–87.

[6] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Allerton conference*, September 2015.

[7] S. Saha and A. A. Julius, "An milp approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proceedings of the 2016 American Control Conference (ACC)*, 2016.

[8] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[9] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1235–1240.

[10] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Smc: Satisfiability modulo convex optimization," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 19–28.
[11] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, 2019.
[12] W. Liu, N. Mehdipour, and C. Belta, "Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints," *IEEE Control Systems Letters*, 2021.
[13] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for multi-agent systems from stl and spatel specifications," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 4452–4457.
[14] L. Lindemann and D. V. Dimarogonas, "Decentralized robust control of coupled multi-agent systems under local signal temporal logic tasks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1567–1573.
[15] Lindemann, Lars and Dimarogonas, Dimos V, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1916–1928, 2020.
[16] Yan, Ruixuan and Julius, Agung, "A Decentralized B&B Algorithm for Motion Planning of Robot Swarms With Temporal Logic Specifications," *IEEE Robotics and Automation Letters*, 2021.
[17] A. Desai, I. Saha, J. Yang, S. Qadeer, and S. A. Seshia, "Drona: A framework for safe distributed mobile robotics," in *Proceedings of the 8th International Conference on Cyber-Physical Systems*, 2017, pp. 239–248.
[18] O. Maler and D. Nickovic, *Monitoring Temporal Properties of Continuous Signals*. Springer Berlin Heidelberg, 2004.
[19] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systems*, 2010.
[20] G. Fainekos and G. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, 2009.
[21] M. W. Mueller, M. Hehn, and R. DÁndrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," in *IEEE Transactions on Robotics*, 2015.
[22] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, 2006.
[23] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[24] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.
[25] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, 2013.
[26] "Hsl. a collection of fortran codes for large scale scientific computation." http://www.hsl.rl.ac.uk, accessed: 2021-05-11.

APPENDIX

A. Constraints for the trajectory planning optimization

For the constraints of the trajectory planning optimization (2), consider the min-jerk trajectory segment [21], of time duration τ, between a waypoint $p^j = [p_x^j, p_y^j, p_z^j]^\top$ with velocity $v^j = [v_x^j, v_y^j, v_z^j]^\top$, and the waypoint with desired position $p^{j+1} = [p_x^{j+1}, p_y^{j+1}, p_z^{j+1}]^\top$. The kinematic bounds be hyper-rectangles of the form $V = [v_{\min}, v_{\max}]^3 \subset \mathbb{R}^3$ and $A = [a_{\min}, a_{\max}]^3 \subset \mathbb{R}^3$. We now define [1] for $t \in [0, \tau]$:

$$\begin{aligned} K_3(t) &= (90t^4)/(48\tau^5) - (90t^3)/(12\tau^4) + (30t^2)/(4\tau^3) \\ K_4(t) &= (90t^3)/(12\tau^5) - (90t^2)/(4\tau^4) + (30t)/(2\tau^3) \end{aligned} \tag{7}$$

Let $t' = \text{argmax}_{t\in[0,\tau]} K_t(t)$. We can now define the constraints (for each UAV) that ensure velocity and acceleration are within bounds $[v_{\min}, v_{\max}]$ and $[a_{\min}, a_{\max}]$ respectively, for each axis of motion l:

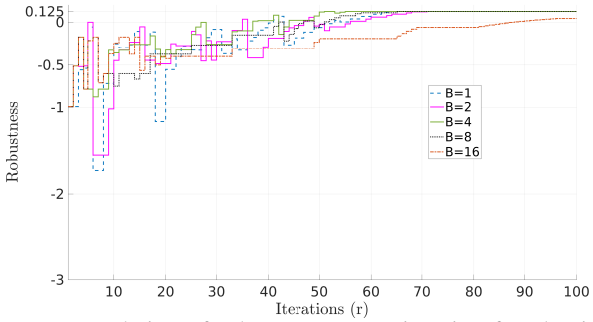

Fig. 6. Evolution of robustness versus iteration for algorithm 1 generating trajectories to satisfy the specification in (4) with $D = 4$ UAVs. Here, we consider different block-cyclic schedules for which subset of UAVs solve their optimization at each iteration.

$$\begin{aligned} \text{LB}_{\text{vel}}(v_l^j) &= (v_{\min} - (1 - \tau K_3(\tau))v_l^j)/K_3(\tau) \\ \text{UB}_{\text{vel}}(v_l^j) &= (v_{\max} - (1 - \tau K_3(\tau))v_l^j)/K_3(\tau) \\ \text{LB}_{\text{acc}}(v_l^j) &= \tau v_l^j + a_{\min}/K_4(t') \\ \text{UB}_{\text{acc}}(v_l^j) &= \tau v_l^j + a_{\max}/K_4(t') \end{aligned} \tag{8}$$

Combining these constraints for all axis of motion gives x, y, z the velocity and acceleration constraints in the optimization (2) of the form $\text{LB}_{\text{vel}}(v^j) = [\text{LB}_{\text{vel}}(v_x^j), \text{LB}_{\text{vel}}(v_y^j), \text{LB}_{\text{vel}}(v_z^j)]$ and similarly for the upper bound for velocities and upper/lower bounds for acceleration. These constraints are such that:

$$\begin{aligned} &\text{LB}_{\text{vel}}(v^j) \leq p^{j+1} - p^j \leq \text{UB}_{\text{vel}}(v^j) \\ &\Rightarrow v_l^j \in [v_{\min}, v_{\max}]\, \forall t \in [0, \tau],\ \forall l \in \{x, y, z\}, \text{ and,} \\ &\text{LB}_{\text{acc}}(v^j) \leq p^{j+1} - p^j \leq \text{UB}_{\text{acc}}(v^j) \\ &\Rightarrow a_l^j \in [a_{\min}, a_{\max}]\, \forall t \in [0, \tau],\ \forall l \in \{x, y, z\} \end{aligned} \tag{9}$$

B. Block-cyclic UAV group selection for optimization

In order to leverage the convergence guarantees of theorem 5.1, algorithm 1 requires broadcasting each UAVs solution to other UAVs at every iteration r (see line 9). This is since the algorithm requires selecting a subset $S[r] = S_i$ of UAVs (in a cyclic manner) from the set of all UAVs $\mathcal{D}$ (such that $\cup_i S_i = \mathcal{D}$) to solve their optimization (see line 5 in algorithm 1). In order to reduce the frequency of inter-group communication, we can instead use a *block-cyclic* schedule for which subset of UAVs solve their optimization at an iteration. Here, for B consecutive iterations, we select the same subset S_i of UAVs to solve the optimization in line 5 and 6 of the algorithm 1. Through this, while each UAV in the subset S_i needs to communicate at each iteration, they only need to communicate with the other UAVs in $\mathcal{D} \setminus S_i$ every B iterations. While this would lower the communication burden, it no longer satisfies the criteria for convergence of the algorithm (theorem 5.1). Figure 6 shows the evolution of robustness for different settings of B (default is 1) for algorithm 1 generating trajectories to satisfy (4). The convergence to satisfying trajectories for all settings of B shows that the convergence guarantees of the algorithm are applicable even outside of the conditions in theorem 5.1. Future work will aim to relax these necessary conditions.