Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Probabilistic Swarm Guidance Subject to Graph Temporal Logic Specifications

Franck Djeumou*, Zhe Xu†, and Ufuk Topcu‡
*Electrical and Computer Engineering, †Oden Institute for Computational Engineering and Sciences, ‡Aerospace Engineering
University of Texas at Austin, Austin, Texas 78712, USA
{fdjeumou, zhexu, utopcu}@utexas.edu

***Abstract*—As the number of agents comprising a swarm increases, individual-agent-based control techniques for collective task completion become computationally intractable. We study a setting in which the agents move along the nodes of a graph, and the high-level task specifications for the swarm are expressed in a recently proposed language called graph temporal logic (GTL). By constraining the distribution of the swarm over the nodes of the graph, GTL specifies a wide range of properties, including safety, progress, and response. In contrast to the individual-agent-based control techniques, we develop an algorithm to control, in a decentralized and probabilistic manner, a collective property of the swarm: its density distribution. The algorithm, agnostic to the number of agents in the swarm, synthesizes a time-varying Markov chain modeling the time evolution of the density distribution of a swarm subject to GTL. We first formulate the synthesis of such a Markov chain as a mixed-integer nonlinear program (MINLP). Then, to address the intractability of MINLPs, we present an iterative scheme alternating between two relaxations of the MINLP: a linear program and a mixed-integer linear program. We evaluate the algorithm in several scenarios, including a rescue mission in a high-fidelity ROS-Gazebo simulation[1].**

I. Introduction

Large numbers, or *swarms*, of autonomous agents have been widely studied because they can collaboratively complete complex tasks that would be difficult or impossible for a single agent. Applications of the swarm concept include the construction of a complex formation shape [21, 27], task allocations [5, 22], surveillance, and search or rescue missions with ground or aerial vehicle swarms [10, 16, 25, 30]. However, how to control such a swarm to achieve global task requirements remains a challenging problem as the number of agents in the swarm increases.

Individual-agent-based control techniques are centered on the idea of generating the trajectory of each agent in the swarm separately. Therefore, as the number of agents comprising the swarm increases, the computational cost for assigning the targets of each agent and generating all the optimal trajectories one by one becomes prohibitively high. We study an alternative setting in which the agents in a swarm move along the nodes of a graph [6]. We sometimes refer to this graph as the *configuration space*. In this scenario, instead of controlling each agent individually, we propose to control over time a collective property of the swarm, more specifically the density distribution of the swarm over the nodes of the graph.

In the aforementioned scenario, constraining the evolution of the density distribution of the swarm expresses a set of collective behaviors. We consider graph temporal logic (GTL) [38] as the formal language to specify such constraints on the swarm. GTL, an extension of linear temporal logic (LTL) [26], is an expressive language for task specifications that focuses on the spatial-temporal properties of the labels of a graph. Specifically, GTL expresses more concisely spatial-temporal properties on a graph structure than other logics such as alternating-time temporal logic (ATL) [2], and LTL. As an example, GTL concisely expresses properties such as "whenever the density of the swarm in a node is less than 0.3, eventually in the next 3 time steps, at least two of its neighbor nodes have their density above 0.6". This property will result in a lengthier formula if expressed in either ATL or LTL.

In this paper, we synthesize controllers for swarms of autonomous agents subject to high-level task specifications expressed in GTL. Specifically, we are interested in developing control algorithms with the following properties: (a) correctness, i.e., the algorithm should enable the satisfaction of GTL specifications; (b) scalability, i.e., the algorithm should scale with the size of the swarm and the size of the configuration space; and (c) distributed, i.e., the algorithm should return decentralized control laws to be executed by each agent [1].

More specifically, we develop an algorithm to control, in a probabilistic manner, the time evolution of the density distribution of the swarm. The algorithm synthesizes a time-varying Markov chain [1, 9], which models the time evolution of the density distribution in the configuration space. From the perspective of an agent, the transition probability between two nodes is specified by the transition probability between the matching states of the synthesized Markov chain.

The proposed formalism builds on the notion of transition probabilities between nodes of the configuration space, which is agnostic to the low-level individual dynamics or local interactions between agents as long as the transitions imposed by the synthesized Markov chain can be achieved.

We formulate the synthesis of a Markov chain for the control of a swarm subject to GTL specifications as a mixed-integer nonlinear programming (MINLP) [31] feasibility problem. MINLPs are NP-hard problems. Thus, we seek for algorithms to efficiently compute solutions of the resulting MINLP. In the particular case where the agents move along the nodes of a complete graph [6], we prove an equivalence between the feasibility of the MINLP and the feasibility of a mixed-integer linear program (MILP) [34]. In the general case, based on an idea similar to the coordinate descent approach [32], we present a method to efficiently solve the MINLP by iteratively solving two of its relaxations: a linear program [7] and an MILP. We demonstrate on a set of problems how the developed method improves scalability over off-the-shelf MINLPs solvers [4, 12]. Specifically, we show that even on relatively small problems, the developed method is three orders of magnitude faster than these MINLPs solvers. In a gridworld setting and a high-fidelity ROS-Gazebo simulation, with agents of the swarm being able to achieve collision avoidance based on *ORCA* [33], we demonstrate the scalability and correctness of the developed algorithm.

The algorithm developed in this paper is scalable as it does not depend on the number of agents in the swarm. Furthermore, the algorithm is correct since, by construction, the resulting Markov chain enables the satisfaction of the GTL specifications. Assuming that the transition time between two nodes is synchronized, each agent individually chooses the node to transit solely based on the transition probabilities of the synthesized Markov chain. Henceforth, the algorithm returns a decentralized control law for each agent.

Contributions. We make the following contributions: (a) we present a novel, correct-by-construction, scalable, and distributed algorithm for the problem of controlling a swarm of autonomous agents subject to GTL specifications; (b) we formulate the problem as an MINLP, identify a particular case where the MINLP can be recast as an MILP, and develop a method based on relaxations of the MINLP to efficiently solve the general case; (c) we evaluate the developed algorithm on simulation examples involving a large number of agents.

Related work. Existing techniques for probabilistic density control [3, 11] of swarms based on the synthesis of a Markov chain do not consider complex behaviors of the swarm, such as the ones induced by temporal logic specifications. To the best of our knowledge, this is the first paper to investigate probabilistic density control with temporal logic specifications.

The problem of synthesizing a controller for systems with multiple agents from a high-level temporal logic specification is considered in [18, 19, 37, 39]. These papers define the specifications on the agent level and use an automata-based approach [17, 35] to compute a discrete controller satisfying the specifications over a finite abstraction of the system. However, it is expensive to compute such a finite abstraction, and the size of the automaton may be exponential in the length of the specifications while the synthesis of a controller can be double exponential in the length of the specifications. Moreover, the length of the specifications depends on the size of the configuration space and may also grow exponentially with the number of controllable agents. Instead, this paper presents a synthesis algorithm with a worst-case time complexity that is only exponential in the size of the configuration space.

The synthesis of control algorithms for swarms subject to spatial and temporal logic specifications has also been considered in recent work [28, 29, 36, 15, 14, 23]. When considering spatial-temporal properties on a graph, GTL is more expressive than the spatial-temporal logics such as counting LTL [28, 29] or SpaTeL [14, 15]. Besides, the approaches based on these logics are significantly less scalable than the proposed approach, and most of them require a central unit to assign targets to individual agents. Specifically, the number of integer variables in the optimization problems resulting from counting LTL-based, GR(1)-based [23], and SpaTeL-based approaches depends on the size of the specifications. Besides, it exhibits quadratic dependency on the size of the considered abstraction. In contrast, the number of integer variables in the proposed approach depends only on the size of the specifications.

II. Preliminaries

Notation. $\mathbf{0}$ is the zero matrix or vector of appropriate dimensions. $\mathbf{1}$ denotes a vector with all elements equal to 1 of appropriate dimensions. e_i is a vector of appropriate dimensions with its i-th entry 1 and its other entries 0. A^{T} denotes the transpose of a matrix A. $A_{i,j} = A[i,j] = e_i^{\mathrm{T}} A e_j$ for a matrix A. $x_i = x[i] = e_i^{\mathrm{T}} x$ for a vector x. Comparisons (e.g., $\geq$) between matrices or vectors are conducted element-wise. The operator $\odot$ represents the element-wise product. $|V|$ denotes the number of elements in the set V.

A. Markov Chain-based Control Approach

We present the definitions and assumptions used in the Markov chain approach to control swarms of autonomous agents. Note that most of the definitions in this section can be found in the existing literature [1, 3].

Definition 1 (Bins): The configuration space over which the state of an agent is distributed is denoted as $\mathcal{R}$. It is assumed that $\mathcal{R}$ is partitioned into n_{r} disjoint subspaces called *bins*.

$$\mathcal{R} = \cup_{i=1}^{n_{\mathrm{r}}} \mathcal{R}_i, \text{ s.t. } \mathcal{R}_i \cap \mathcal{R}_j = \emptyset, \text{ for all } i \neq j.$$

Each bin $\mathcal{R}_i$ (also referred to as bin i) represents a predefined range of the state of an agent, e.g., position, behavior, etc.

Definition 2 (State of an agent): We denote by N_{a} the number of agents in the swarm. We define $r^m(t) \in \{0,1\}^{n_{\mathrm{r}}}$ as the state of agent m at time t. If $r^m(t)$ belongs to the bin $\mathcal{R}_i$, for some $i \in \{1,\ldots,n_{\mathrm{r}}\}$, then $r^m(t) = e_i$.

Definition 3 (Motion constraints): The state of each agent can transition, between two consecutive time steps, from a bin to only certain bins because of the dynamics or the environment. These motion constraints are specified by the fixed matrix $A_{\text{adj}} \in \{0,1\}^{n_{\mathrm{r}} \times n_{\mathrm{r}}}$, called an *adjacency matrix*. Each component of A_{adj} is given by

$$A_{\text{adj}}[i,j] = \begin{cases} 1 & \text{if the transition from bin } \mathcal{R}_i \\ & \text{to bin } \mathcal{R}_j \text{ is allowed,} \\ 0 & \text{if this transition is not allowed.} \end{cases}$$

Equivalently, the topology of the bins can be modeled as a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_{n_{\mathrm{r}}}\}$ represents the set of bins, and $E \subseteq V \times V$ represents the set of edges such that $(v_i, v_j) \in E$ if and only if $A_{\text{adj}}[i,j] = 1$, $\forall i,j \in \{1,\ldots,n_{\mathrm{r}}\}$.

In the rest of the paper, when we refer to an agent belonging to a bin, we mean that its state belongs to that bin. Similarly, when we refer to an agent transiting between bins, we mean that its state transits between these bins.

Example 1: Consider a swarm scenario where the state of an agent is its position, and the physical configuration space is partitioned into $n_\mathrm{r} = 3$ bins. Consider that $A_\mathrm{adj} = [[1,1,0]^\mathrm{T}, [1,1,1]^\mathrm{T}, [0,1,1]^\mathrm{T}]$. Having that $A_\mathrm{adj}[1,1] = A_\mathrm{adj}[1,2] = 1$, and $A_\mathrm{adj}[1,3] = 0$ enforces agents in bin $\mathcal{R}_1$ to either stay in $\mathcal{R}_1$ or transit to $\mathcal{R}_2$ between two consecutive time steps. The corresponding graph $G = (V, E)$ is given by $V = \{v_1, v_2, v_3\}$, where the nodes v_1, v_2, and v_3 represent respectively the bins $\mathcal{R}_1, \mathcal{R}_2$, and $\mathcal{R}_3$. The set of edges is given by $E = \{(v_1,v_1), (v_1,v_2), (v_2,v_1), (v_2,v_2), (v_2,v_3), (v_3,v_2), (v_3,v_3)\}$.

Definition 4 (Density distribution of the swarm): The density distribution $x(t) \in \mathbb{R}^{n_\mathrm{r}}$ of a swarm is a column-stochastic vector, i.e. $x(t) \geq \mathbf{0}$ and $\mathbf{1}^\mathrm{T}x(t) = 1$, such that a component $x_i(t)$ is the proportion of agents in bin $\mathcal{R}_i$ at time t:

$$x_i(t) := \frac{1}{N_\mathrm{a}} \sum_{m=1}^{N_\mathrm{a}} r_i^m(t).$$

Definition 5 (Transition policy of an agent): At time t, the agent m transits from bin $\mathcal{R}_j$ to bin $\mathcal{R}_i$ with probability

$$M_{i,j}^m(t) = Pr(r_i^m(t+1) = 1 | r_j^m(t) = 1),$$

where $M^m(t) \in \mathbb{R}^{n_\mathrm{r} \times n_\mathrm{r}}$ is a column-stochastic matrix, i.e. $\mathbf{1}^\mathrm{T}M^m(t) = \mathbf{1}^\mathrm{T}, M^m(t) \geq \mathbf{0}$. We refer to $M^m(t)$ as the time-varying Markov matrix of agent m at time t.

Remark 1: Under the motion constraints given by A_adj, the transitions between some bins may not be allowed. For agent m, $M_{i,j}^m(t)$ is the probability of transition from bin $\mathcal{R}_j$ to bin $\mathcal{R}_i$. Hence, $M_{i,j}^m(t) = 0$ if $A_\mathrm{adj}[j,i] = 0$.

In Example 1, if $M^m(t)$ is the time-varying Markov matrix of agent m at time t, then $M_{2,1}^m(t)$ gives the probability of agent m to transit from bin $\mathcal{R}_1$ to bin $\mathcal{R}_2$ in one time step. Moreover, having $A_\mathrm{adj}[1,3] = 0$ enforces that $M_{3,1}^m(t) = 0$.

We focus on methods that ensure that each agent has the same time-varying Markov matrix at any given time t, i.e. $M^1(t) = \cdots = M^{N_\mathrm{a}}(t) = M(t)$. When the agents independently choose their transitions between bins using $M(t)$, two mathematical interpretations are given for $x(t)$ [1]: (a) $x(t)$ is the vector of expected ratio of the number of agents in each bin; (b) the ensemble of agent state, $\{r^k(t)\}_{k=1}^{N_\mathrm{a}}$, has a distribution that approaches $x(t)$ with probability one as N_a increases towards infinity (due to the law of large numbers). As a consequence, the dynamics of the density distribution of the swarm can be modeled by [1, 11]

$$x(t+1) = M(t)x(t), \tag{1}$$

as N_a increases towards infinity. The Markov chain approach for the control of swarms relies on the synthesis of a time-varying Markov matrix $M(t)$ such that the time evolution of the density distribution of the swarm is given by (1).

B. Graph Temporal Logic

Let $G = (V, E)$ be a graph, where V is a finite set of nodes and E is a finite set of edges. We use $\mathcal{X}$ to denote a (possibly infinite) set of node labels. $\mathbb{T} = \{0, 1, \dots\}$ is a discrete set of time indices. A graph with node labels is also called a *labeled graph.* A trajectory $g : V \times \mathbb{T} \to \mathcal{X}$ on the graph G denotes the time evolution of the node labels.

Consider the graph G in Example 1. We label each node of G with the density of the swarm in the corresponding bin. For example, the label of node v_1 at time t, denoted by $x_1(t)$, is the density of the swarm in bin $\mathcal{R}_1$. As a consequence, the graph trajectory g at node v_i and time t is given by $g(v_i, t) = x_i(t)$.

An *atomic node proposition* is a predicate on $\mathcal{X}$, i.e. a Boolean valued map from $\mathcal{X}$. We use π to denote an atomic node proposition, and $\mathcal{O}(\pi)$ to denote the subset of $\mathcal{X}$ for which π is true.

We define that a graph trajectory g satisfies the atomic node proposition π at a node v at time index k, denoted as $(g, v, k) \models \pi$, if and only if $g(v, k) \in \mathcal{O}(\pi)$. In Example 1, if $x(0) = [0.3, 0.3, 0.4]^\mathrm{T}$ and $\pi = (x \leq 0.3)$, then π is satisfied by g at time index 0 at nodes v_1 and v_2.

Definition 6 (Neighbor operator): Given a graph G, the *neighbor operation* $\bigcirc : 2^V \to 2^V$ is defined as

$$\bigcirc(V') = \{v \in V | \exists v' \in V' \text{ s.t. } (v', v) \in E\}.$$

Intuitively, $\bigcirc(V')$ consists of nodes that can be reached from V'. Note that neighbor operations can be applied successively. In Example 1, we have $\bigcirc(\{v_1\}) = \{v_1, v_2\}$.

We refer to a graph trajectory of finite length as a trajectory $g : V \times \{0, \dots, T_\mathrm{f}\} \to \mathcal{X}$, where $T_\mathrm{f} \in \mathbb{T}$. Graph trajectories of finite time length are sufficient to satisfy (resp. violate) *finite-horizon* GTL formulas. We define the syntax of a finite-horizon GTL formula φ recursively as

$$\varphi := \pi \mid \neg\varphi_1 \mid X\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{\leq i} \varphi_2 \mid \exists^N(\bigcirc \cdots \bigcirc)\varphi_1,$$

where π is an atomic node proposition, $\exists^N(\bigcirc \cdots \bigcirc)\varphi$ reads as "there exist at least N nodes under the neighbor operation $\bigcirc \cdots \bigcirc$ that satisfy φ ", $\neg$ and $\wedge$ stand for negation and conjunction respectively, X is the temporal operator "next", and $\mathcal{U}_{\leq i}$ is the temporal operator "parametrized until" where $i \in \mathbb{T}$ is a parameter. We can also derive $\vee$ (disjunction), $\Rightarrow$ (implication), $\Diamond_{[i_1,i_2]}$ (parametrized eventually), $\Box_{[i_1,i_2]}$ (parametrized always) from the above-mentioned operators [8], e.g.

$$\Diamond_{[i_1,i_2]}\varphi = \bigvee_{k=i_1}^{i_2} \overbrace{X \cdots X}^{k} \varphi,\ \Box_{[i_1,i_2]}\varphi = \bigwedge_{k=i_1}^{i_2} \overbrace{X \cdots X}^{k} \varphi.$$

The satisfaction relation $(g, v, k) \models \varphi$ for a graph trajectory g at node v at time index k with respect to a finite-horizon GTL formula φ is defined recursively as

$$\begin{aligned}
&(g,v,k) \models \pi && \text{iff } g(v,k) \in \mathcal{O}(\pi),\\
&(g,v,k) \models \neg\varphi && \text{iff } (g,v,k) \not\models \varphi,\\
&(g,v,k) \models X\varphi && \text{iff } (g,v,k+1) \models \varphi,\\
&(g,v,k) \models \varphi_1 \wedge \varphi_2 && \text{iff } (g,v,k) \models \varphi_1 \text{ and } (g,v,k) \models \varphi_2,\\
&(g,v,k) \models \varphi_1 \mathcal{U}_{\leq i} \varphi_2 && \text{iff } \exists k' \in [k, k+i], \text{s.t. } (g,v,k') \models \varphi_2,\\
& && \quad (g,v,k'') \models \varphi_1, \forall k'' \in [k, k'),
\end{aligned}$$

$$(g,v,k) \models \exists^N(\bigcirc \cdots \bigcirc)\varphi \text{ iff } \exists v_1, \dots, v_N \ (v_i \neq v_j \text{ for } i \neq j), \text{ s.t.}, \forall i, v_i \in \bigcirc \cdots \bigcirc(\{v\}), \text{ and } (g, v_i, k) \models \varphi.$$

Intuitively, a graph trajectory g satisfies $\exists^N(\bigcirc\cdots\bigcirc)\varphi$ at a node $v \in V$ at time index k, if there exist at least N nodes in $\bigcirc\cdots\bigcirc(\{v\})$ where φ is satisfied by g at time index k. Note that, by definition, if $\bigcirc\cdots\bigcirc(\{v\})$ consists of fewer than N nodes, then $\exists^N(\bigcirc\cdots\bigcirc)\varphi$ is false. In Example 1, if $x(0) = [0.3, 0.3, 0.4]^{\mathrm{T}}$, then the nodes that satisfy $\exists^2 \bigcirc (x \leq 0.3)$ at time index 0 are v_1 and v_2.

We also define that a graph trajectory g satisfies φ at node v, denoted as $(g, v) \models \varphi$, if g satisfies φ at node v at time 0.

Definition 7 (Time horizon of finite-horizon GTL formulas): We define the time horizon $T_\varphi \in \mathbb{T}$ of the formula φ as the minimum time length of a graph trajectory to evaluate both its satisfaction and its violation with respect to φ.

As examples, $\varphi_1 = \lozenge_{[0,5]}(x \geq 0.2)$ has a time horizon $T_{\varphi_1} = 5$, and $\varphi_2 = X((x \leq 0.1)\mathcal{U}_{\leq 8}(\exists^2 \bigcirc (x \geq 0.3)))$ has a time horizon $T_{\varphi_1} = 9$. In the remainder of the paper, we will use GTL instead of finite-horizon GTL for brevity.

III. Problem Formulation

In this section, we first specify the link between a graph trajectory satisfying a graph temporal logic (GTL) formula and the time evolution of the density distribution of a swarm. Then, we formulate the problem of controlling, in a probabilistic manner, the density distribution of a swarm subject to GTL, as the problem of synthesizing a time-varying Markov matrix.

Definition 8 (GTL specifications): Let $G = (V, E)$ be the graph induced by the topology of the bins (Definition 3). By labeling each node $v_i \in V$ with the time-varying density $x_i(t)$ of the swarm (Definition 4 in bin $\mathcal{R}_i$, we define GTL specifications on the swarm as GTL formulas on G.

Definition 8 specifies that a graph trajectory g on the labeled graph G, at node v_i and time index $t \in \mathbb{T}$, is given by $g(v_i, t) = x_i(t)$. In the remainder of the paper, we write $(x, v_i) \models \varphi$ to denote the fact that the graph trajectory g, with $g(v_i, t) = x_i(t)$, satisfies φ at node v_i.

Assumption 1: For every atomic node proposition π of a given GTL formula, we assume that $\mathcal{O}(\pi) \subseteq \mathcal{X}$ is a convex polytope set [7].

Problem 1: Given the adjacency matrix A_{adj} and its induced graph $G = (V, E)$, a set $V' \subseteq V$, and a GTL formula φ on the induced labeled graph (Definition 8), compute a time-varying Markov matrix $M(t)$ such that the following are true:

1) The motion constraints are satisfied.
2) $(x, v) \models \varphi$ for all $v \in V'$.

Remark 2: $M(t)$ dictates the evolution (1) of the density distribution of the swarm $x(t)$. Thus, $M(t)$ specifies the graph trajectory and its design is crucial in the satisfaction of φ.

Consider Example 1 with $x(0) = [0.3, 0.3, 0.4]^T$, and the GTL formula $\varphi_1 = (\square_{[1,10]}(x = 0))$ is specified for node v_2 (bin $\mathcal{R}_2$). Intuitively, φ_1 means that from time index 1 to time index 10, there should always be no agents in bin $\mathcal{R}_2$. A time-varying Markov matrix solution to Problem 1 is

$$M(0) = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 1.0 \end{bmatrix}, M(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, t = 1, \ldots, 9.$$

Intuitively, at time 0, an agent in $\mathcal{R}_2$ must move to $\mathcal{R}_1$ or $\mathcal{R}_3$ with probability 0.5 and remains there for $t \geq 1$. The reader can check that with $x(t) = M(t)x(t-1)$ for $t \geq 1$, we have $x_2(t) = 0$ holds for $t \geq 1$. Thus, φ_1 is satisfied at node v_2.

IV. MINLP Formulation

In this section, we formulate the Problem 1 as a mixed-integer nonlinear programming (MINLP) problem containing $M(t)$ and $x(t+1)$ as variables, where $t \in \{0, \ldots, T_\varphi - 1\}$ and T_φ is the time horizon of the graph temporal logic (GTL) formula φ that we seek to satisfy.

A. Stochasticity and Motion Constraints

The desired time-varying Markov matrix $M(t)$ at time index t is a column-stochastic matrix, i.e.

$$\mathbf{1}^{\mathrm{T}} M(t) = \mathbf{1}^{\mathrm{T}}. \tag{2}$$

From Remark 1, we have that $M_{i,j}(t) = 0$ if $A_{\text{adj}}[j, i] = 0$, and $M^m_{i,j}(t) \geq 0$ otherwise. Thus,

$$(\mathbf{1}\mathbf{1}^{\mathrm{T}} - A_{\text{adj}}^{\mathrm{T}}) \odot M(t) = \mathbf{0}, \tag{3}$$

$$M(t) \geq \mathbf{0}. \tag{4}$$

B. MILP Encoding of GTL Formulas

In this section, we ignore the constraints implied by the dynamics (1), and seek only for graph trajectories that satisfy a given GTL formula. Given a labeled graph $G = (V, E)$ with $V = \{v_1, \ldots, v_N\}$, a GTL formula φ and its time horizon T_φ, a node $v \in V$, we want to compute a graph trajectory g such that $(g, v) \models \varphi$ under Assumption 1. With a slight abuse of notation, we write $g(t) := [g(v_1, t), \ldots, g(v_N, t)]^{\mathrm{T}}$, where $t \in \{0, \ldots, T_\varphi\}$. Consider $\mathcal{X} \subseteq \mathbb{R}^d$ with $d \geq 1$.

Theorem 1: The existence of a graph trajectory g such that $(g, v) \models \varphi$ can be equivalently formulated as a mixed-integer linear programming (MILP) feasibility problem with constraints given by $A[\boldsymbol{x}^{\mathrm{T}}, w^{\mathrm{T}}]^{\mathrm{T}} \leq b$, where the variables are $\boldsymbol{x} = [g(0)^{\mathrm{T}}, \ldots, g(T_\varphi)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^p$ and $w \in \{0, 1\}^k$, and the parameters $A \in \mathbb{R}^{q \times (p+k)}$, $b \in \mathbb{R}^q$, $p = (T_\varphi + 1)Nd \in \mathbb{N}$, $k \in \mathbb{N}$, and $q \in \mathbb{N}$ depend only on φ and v.

Proof: See the supplementary material on the github[1]. ■

C. Synthesis of a Time-Varying Markov Matrix via MINLP

Theorem 1 shows that the synthesis of a graph trajectory satisfying a GTL formula at a given node can be equivalently formulated as an MILP feasibility problem. However, in the swarm setting (see Definition 8), we are interested in finding graph trajectories that follow the dynamics (1) and satisfy motion constraints.

Corollary 1: Let $G = (V, E)$ be the labeled graph induced by the topology of the bins as in Definition 8, φ be a GTL formula with time horizon T_φ, V' be a subset of V, and $x(0)$ be the node labels at time index 0. Then, the following statements are equivalent:

1) There exists x such that $(x, v) \models \varphi$ for all $v \in V'$ while the motion constraints are satisfied.
2) There exists a feasible solution of the MINLP feasibility problem with constraints given by

$$\begin{aligned}
&A_i \begin{bmatrix} \boldsymbol{x} \\ w_i \end{bmatrix} \leq b_i, && \forall v_i \in V', && (5)\\
&\mathbf{1}^{\mathrm{T}}\, x(t) = 1, && t = 1, \dots, T_\varphi, && (6)\\
&x(t) \geq \mathbf{0}, && t = 1, \dots, T_\varphi, && (7)\\
&\mathbf{1}^{\mathrm{T}}\, M(t) = \mathbf{1}^{\mathrm{T}}, && t = 0, \dots, T_\varphi - 1, && (8)\\
&M(t) \geq \mathbf{0}, && t = 0, \dots, T_\varphi - 1, && (9)\\
&(\mathbf{1}\mathbf{1}^{\mathrm{T}} - A_{\text{adj}}^{\mathrm{T}}) \odot M(t) = \mathbf{0}, && t = 0, \dots, T_\varphi - 1, && (10)\\
&x(t+1) = M(t)\, x(t), && t = 0, \dots, T_\varphi - 1, && (11)
\end{aligned}$$

where the variables are $\boldsymbol{x} = [x(1)^{\mathrm{T}}, \dots, x(T_\varphi)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{n_{\mathrm{r}} T_\varphi}$, $w_i \in \{0,1\}^{p_i}$ for all $v_i \in V'$ and $M(t) \in \mathbb{R}^{n_{\mathrm{r}} \times n_{\mathrm{r}}}$ for $t \in \{0, \dots, T_\varphi - 1\}$, i denotes a bin index, and the parameters A_i, b_i and p_i depend only on v_i and φ for all $v_i \in V'$.

Proof: This is a direct application of Theorem 1. The constraint (5) is obtained by the equivalence shown in Theorem 1. The constraint (11) is resulting from the dynamics (1). The definition of the density distribution, the stochasticity, and the motion constraints are given by (6)–(10). ■

V. Solution

Corollary 1 formulates the problem of synthesizing $M(t)$ as a mixed-integer nonlinear programming (MINLP) feasibility problem. However, MINLPs are NP-hard problems, and they are inefficiently solved by general purpose off-the-shelf MINLP solvers. We use the specific structure of the problem to propose an efficient method to compute a feasible solution of the MINLP.

A. Special Case: MILP Formulation for Complete Graphs

When the graph G is complete, we can reduce the MINLP feasibility problem given by constraints (5)–(11) to a mixed-integer linear programming (MILP) feasibility problem.

Corollary 2: With the notation of Corollary 1, assume that G is a complete graph. Then, the latter statements are equivalent:

1) There exists x such that $(x, v) \models \varphi$ for all $v \in V'$ while the motion constraints are satisfied.
2) There exists a feasible solution of the MILP feasibility problem with constraints given by (5)–(7).

Furthermore, if there exists $\hat{\boldsymbol{x}} = [\hat{x}(1)^{\mathrm{T}}, \dots, \hat{x}(T_\varphi)^{\mathrm{T}}]^{\mathrm{T}}$ satisfying constraints (5)–(7), then $\hat{M}(t)$ given by

$$\hat{M}_{ij}(t) = \hat{x}_i(t+1),\ \forall i, j \in \{1, \dots, n_r\}, \qquad (12)$$

for $t \in \{0, \dots, T_\varphi - 1\}$, satisfies constraints (8)–(11).

Proof: We know that $A_{\text{adj}} = \mathbf{1}\mathbf{1}^T$ when G is a complete graph. Thus, the constraint (10) is automatically satisfied.

(1) implies (2) is trivial by the equivalence of Corollary 1.

Suppose there exist $\hat{\boldsymbol{x}}$ and $\hat{w}_k$ for all $v_k \in V'$ satisfying constraints (5)–(7). With $\hat{M}(t)$ given by (12) and a fixed j,

$$\textstyle\sum_i \hat{M}_{ij}(t) = \sum_i \hat{x}_i(t+1) = \mathbf{1}^{\mathrm{T}} \hat{x}(t+1) = 1.$$

This yields the satisfiability of the constraint (8) by $\hat{M}(t)$. The constraint (9) is satisfied by $\hat{M}(t)$ due to the constraint (7). Finally, for $t \in \{0, \dots, T_\varphi - 1\}$ and a fixed i, we have

$$\textstyle\sum_j M_{ij}(t) \hat{x}_j(t) = \hat{x}_i(t+1) \sum_j \hat{x}_j(t) = \hat{x}_i(t+1).$$

This gives the satisfiability of the constraint (11) by $\hat{M}(t)$. Therefore, there exists $\hat{\boldsymbol{x}}$, $\hat{M}(t)$ and $\hat{w}_k$ for all $v_k \in V'$ satisfying (5)–(10). Hence, Corollary 1 provides that statement (1) also holds. ■

Complexity and correctness analysis: With the notation of Corollary 1, the number of non-binary variables N_{c} and the number of binary variables N_{b} of the equivalent MILP in Corollary 2 are given by $N_{\mathrm{c}} = n_{\mathrm{r}} T_\varphi$ and $N_{\mathrm{b}} = \sum_{v_i \in V'} p_i$. The number of constraints C of the MILP is $C = T_\varphi + N_{\mathrm{c}} + \sum_{v_i \in V'} q_i$, where q_i is the dimension of b_i. Since a linear program (LP) can be solved in polynomial time in the number of variables and constraints via interior-point methods [24], the worst-case time complexity to solve the MILP is $O(2^{N_{\mathrm{b}}} R(N_{\mathrm{c}}, C))$, R is a polynomial. By Corollary 2, a solution to the MILP ensures the satisfaction of the constraints. Therefore, the algorithm for the special case is correct.

B. General Case: Modified Coordinate Descent (MCD)

When the graph G is not complete, we develop a coordinate descent type of technique to find a solution that satisfies constraints (5)–(11) of Corollary 1.

When either $M(t)$ or $\boldsymbol{x}$ in Corollary 1 are known, the MINLP is either an MILP or an LP, which can be solved efficiently by current MILP/LP solvers. We develop a method that generates iterates $\boldsymbol{x}^k = [x^k(1)^{\mathrm{T}}, \dots, x^k(T_\varphi)^{\mathrm{T}}]^{\mathrm{T}}, M^k(t)$ satisfying constraints (5)–(10), and such that the sequence $(\sum_{t=0}^{T_\varphi - 1} ||x^k(t+1) - M^k(t)\, x^k(t)||_p)_{k \in \mathbb{N}}$ is non-increasing. Note that $p \in \{1, 2, \infty\}$ specifies the norm that is being used. The iterates $\boldsymbol{x}^k, M^k(t)$ are generated by alternating between solving two relaxations of the MINLP. If $p \in \{1, \infty\}$, the relaxations are an LP and an MILP. For $p = 2$, we obtain a quadratic program and a mixed-integer quadratic program.

We start with a feasible solution $\boldsymbol{x}^0, w_k^0$ for all $v_k \in V'$ of the relaxation of the original MINLP with the constraints

$$\begin{aligned}
&A_i \begin{bmatrix} \boldsymbol{x} \\ w_i \end{bmatrix} \leq b_i, && \forall v_i \in V',\\
&\mathbf{1}^{\mathrm{T}}\, x(t) = 1, && t = \mathbf{1}, \dots, T_\varphi,\\
&x(t+1) \leq A_{\text{adj}}^{\mathrm{T}} x(t), && t = 0, \dots, T_\varphi - 1,\\
&x(t) \leq A_{\text{adj}} x(t+1), && t = 0, \dots, T_\varphi - 1,\\
&x(t) \geq \mathbf{0}, && t = 1, \dots, T_\varphi,
\end{aligned} \qquad (13)$$

where the variables are $\boldsymbol{x} = [x(1)^{\mathrm{T}}, \dots, x(T_\varphi)^{\mathrm{T}}]^{\mathrm{T}}$ and w_i for all $v_i \in V'$. If the MILP with constraints (13) is infeasible, then the initial problem is infeasible and we stop at this point. Note that the constraint $x(t+1) \leq A_{\text{adj}}^{\mathrm{T}} x(t)$ comes directly from (11) with $A_{\text{adj}}^{\mathrm{T}}$ as an upper bound for $M(t)$. On the other side, from the law of total probability, we have that

$$\begin{aligned}
x_i(t) &= \sum_{j=1}^{n_{\mathrm{r}}} Pr(r_i^m(t) = 1 | r_j^m(t+1) = 1) x_j(t+1)\\
&\leq \sum_{j=1}^{n_{\mathrm{r}}} A_{\text{adj}}[i, j] x_j(t+1),
\end{aligned}$$

for any agent m and $i \in \{1, \dots, n_{\mathrm{r}}\}$. Hence the inequality $x(t) \leq A_{\text{adj}} x(t+1)$ also holds. Observe that the constraints

$x(t) \leq A_{\text{adj}} x(t+1)$ and $\boldsymbol{x}(t+1) \leq A_{\text{adj}}^{\mathrm{T}} x(t)$ are relaxations of the constraint $x(t+1) = M(t)\, x(t)$.

In case of feasibility of the problem (13), in each iteration $k \geq 1$, we solve the following problem for fixed $\boldsymbol{x}^{k-1}$

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{t=0}^{T_\varphi - 1} \|x^{k-1}(t+1) - M(t)\, x^{k-1}(t)\|_p & \\
\text{subject to} & \mathbf{1}^{\mathrm{T}} M(t) = \mathbf{1}^{\mathrm{T}}, & t = 0, \ldots, T_\varphi - 1, \\
& (\mathbf{1}\mathbf{1}^T - A_{\text{adj}}^{\mathrm{T}}) \odot M(t) = \mathbf{0}, & t = 0, \ldots, T_\varphi - 1, \\
& M(t) \geq \mathbf{0}, & t = 0, \ldots, T_\varphi - 1,
\end{array} \tag{14}$$

where the variable is $M(t) \in \mathbb{R}^{n_{\text{r}} \times n_{\text{r}}}$. Let $M^{k-1}(t)$ be the optimal solution of problem (14), we additionally solve the following problem at the k-th iteration

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{t=0}^{T_\varphi - 1} \|x(t+1) - M^{k-1}(t)\, x(t)\|_p & \\
\text{subject to} & A_i \begin{bmatrix} \boldsymbol{x} \\ w_i \end{bmatrix} \leq b_i, & \forall v_i \in V', \\
& \mathbf{1}^{\mathrm{T}} x(t) = 1, & t = 1, \ldots, T_\varphi, \\
& x(t+1) \leq A_{\text{adj}}^{\mathrm{T}} x(t), & t = 0, \ldots, T_\varphi - 1, \\
& x(t) \leq A_{\text{adj}} x(t+1), & t = 0, \ldots, T_\varphi - 1, \\
& x(t) \geq \mathbf{0}, & t = 1, \ldots, T_\varphi,
\end{array} \tag{15}$$

where the variables are $\boldsymbol{x}$ and w_i for all $v_i \in V'$.

We solve the problem (14) and the problem (15) at each iteration k until we find a pair $M^k(t), \boldsymbol{x}^k$ such that

$$\sum_{t=0}^{T_\varphi - 1} \|x^k(t+1) - M^k(t)\, x^k(t)\|_p \leq \varepsilon_{\text{tol}},$$

for a small fixed $\varepsilon_{\text{tol}} > 0$. In this case, we have obtained a solution of the original MINLP with an accuracy of the bilinear constraints less than ε_{tol}. However, when the sum converges toward a value greater than ε_{tol}, we cannot certify that a solution of the original MINLP does not exist.

Complexity and correctness analysis: We consider in this analysis the notation of Corollary 1. Consider $N_{\text{c}}^{\text{a}} = n_{\text{r}} T_\varphi$, $N_{\text{c}}^{\text{b}} = n_{\text{r}}^2 T_\varphi$, $N_{\text{b}} = \sum_{v_i \in V'} p_i$, $C_1 = T_\varphi + 3N_{\text{c}} + \sum_{v_i \in V'} q_i$, where q_i is the dimension of b_i, $C_2 = n_{\text{r}} T_\varphi + n_{\text{r}}^2 T_\varphi$. Using the arguments in the complexity analysis of Section V-A, the worst-case time complexity to solve the problems (13), (14), (15) are respectively $O(2^{N_b} R_1(N_c^a, C_1))$, $O(R_2(N_c^b, C_2))$ and $O(2^{N_b} R_3(N_c^a, C_1))$, where R_1, R_2, R_3 are polynomials. Finally, the MCD has a worst-case time complexity that is linear in the number of iterations for achieving the ε_{tol}-accuracy, exponential in N_{b} and polynomial in $N_{\text{c}}^{\text{a}}, N_{\text{c}}^{\text{b}}, C_1, C_2$. Furthermore, since a solution returned by MCD is proven to ensure the satisfaction of the GTL specifications, the MCD is correct-by-construction.

C. Agent-Level Probabilistic Swarm Guidance

Assuming the MCD algorithm outputs a solution $M(t)$, each agent uses $M(t)$ to choose its bin-to-bin transition. The decentralized Algorithm 1 specifies how each agent probabilistically computes its target bin at each time index, in order for the high-level task specifications to be satisfied.

Algorithm 1: Probabilistic Swarm Guidance

```
  // For each agent at time index $t$
1 Identify the current bin $\mathcal{R}_i$;
2 Query $M_{ji}(t)$, for all $j \in \{1, \ldots, n_r\}$;
3 Generate $z$ from the uniform distribution on $[0, 1]$;
4 Select bin $\mathcal{R}_j$ such that
    $\sum_{l=1}^{j-1} M_{li}(t) \leq z \leq \sum_{l=1}^{j} M_{li}(t)$;
5 Transit to bin $\mathcal{R}_j$ while achieving collision avoidance.
```

VI. Numerical experiments

In this section, we first demonstrate how the MCD outperforms current off-the-shelf mixed-integer nonlinear programming (MINLP) solvers. Then, we show that MCD combined with Algorithm 1 can be used for real-world applications. The simulations were performed on a computer with an Intel Core i5-7300HQ 2.5GHz $\times 4$ processors and 8Gb of RAM. We use gurobi [13] to solve the optimization problems (13), (14), (15).

A. Optimization Comparisons

We empirically show that the MCD algorithm solves the MINLP feasibility problem of Corollary 1 much more efficiently than general purpose MINLP solvers. For this purpose, we compare the MCD algorithm with two open-source and efficient [20] MINLP solvers: Couenne [4] and SCIP [12].

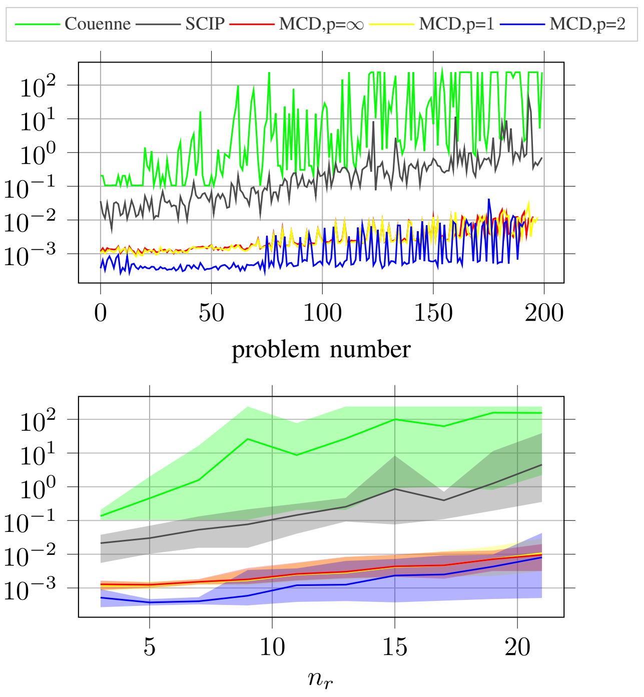

Fig. 1. Computational time, on a logarithm scale, of each solver on 200 non-trivial and randomly generated problems. The figure on the top shows the solving time for each problem. The figure on the bottom shows the average solving time as a function of the number of bins n_r.

Figure 1 shows that no matter what norm is used for the MCD, it always performs better than the MINLP solvers. The MCD with $p = 2$ has the drawback of having an increasing computational time with n_{r} compared to the MCD with $p = 1$ and $p = \infty$. Additionally, the figure shows that the MCD sometimes performs 1000 times better than SCIP while Couenne always has the worst solving time amongst all the solvers. As n_{r} increases, we expect an even bigger gap.

Furthermore, we want to compare $\sum_{t=0}^{T_\varphi - 1} ||x(t+1) - M(t)\,x(t)||_2$, where $\boldsymbol{x}$ and $M(t)$ are the solutions given by each solver for each generated problem. This value specifies the accuracy of the bilinear constraint in the MINLP problem.

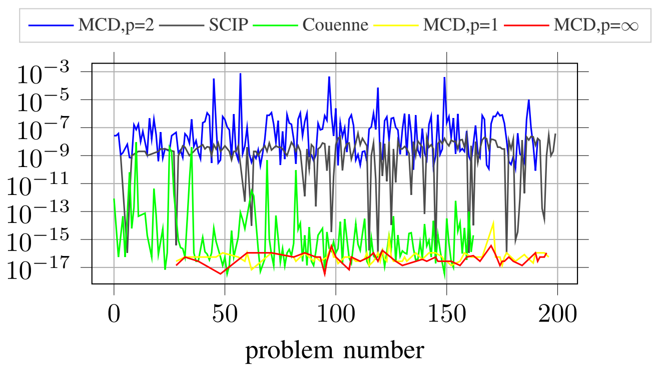

Fig. 2. Variation of $\sum_{t=0}^{T_\varphi - 1} ||x(t+1) - M(t)\,x(t)||_2$, on a logarithm scale, for the solutions $\boldsymbol{x}$, $M(t)$ obtained by the different solvers.

Figure 2 shows that, with $p = 1$ and $p = \infty$, the MCD algorithm achieves better accuracy than the MINLP solvers.

B. A Gridworld Example

We consider in this section a simulation example with a swarm of autonomous agents navigating in a gridworld environment as shown in Figure 3. The task of the swarm is, from a given initial distribution, to converge towards the desired distribution in a fixed amount of time. Moreover, the swarm must satisfy some capacity constraints in the bins, i.e. each bin can contain only a fixed maximum number of agents at a time index. Specifically, we use GTL to express the task specifications of the swarm as following

- $\Diamond_{[11,12]}(x \geq 0.2)$ is specified for bins $1, 5$, and $\Diamond_{[11,12]}(x \geq 0.3)$ is specified for bins $0, 6$. Intuitively, we want the swarm to eventually have between time indexes 11 and 12 at least 20% of agents in bins $1, 5$ and at least 30% of agents in bins $0, 6$.
- $\Box_{[0,12]}(x \leq 0.15)$ is specified for bins $2, 3, 10$, $\Box_{[0,12]}(x \leq 0.5)$ is specified for bins $19, 23, 24$, and $\Box_{[0,12]}(x \leq 0.3)$ is specified for the remaining bins. These constraints are safety constraints limiting the maximum allowable agents per bins at each time index between 0 and 12.
- $\Box_{[0,12]}(x = 0)$ is specified for bins $7, 8, 11, 16$. Intuitively, these bins represent obstacles and no agent should be in these bins.

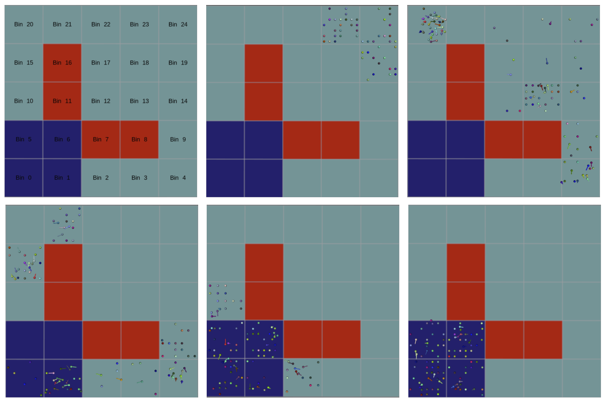

Fig. 3. Evolution of the swarm in a 5 by 5 gridworld. From the second frame to the last frame, the time indexes are respectively $0, 3, 8, 11, 12$.

More specifically, we consider a scenario with a swarm comprised of 100 agents able to perform collision avoidance in a decentralized manner. For this purpose, each agent in the simulation uses optimal reciprocal collision avoidance (ORCA) [16] to dynamically compute safe velocities to reach a given goal region. We first apply the MCD to find a time-varying Markov matrix $M(t)$ such that the specifications above are satisfied. Then, given the current time index $t_{\text{curr}} \leq 12$, each agent independently and probabilistically chooses their target bin based on Algorithm 1 with given $M(t_{\text{curr}})$. When the target bin is obtained, the line 5 of Algorithm 1 consists of using ORCA to generate in real-time, at a fixed frequency, control velocities to reach the target bin while avoiding the fixed obstacles and the other agents in the gridworld.

Figure 3 shows the evolution of the swarm at different time indexes in the gridworld. The video[1] shows no collisions between the agents while they safely avoid the obstacles and satisfy the GTL specifications. Specifically, Figure 5 shows the real-time evolution of the density of the swarm in some specific bins. On one side, we can observe that the density constraints are sometimes violated (bin $2, 3, 10$) between two consecutive time indexes. Intuitively, since the flow of agents passing through the bins between two consecutive time indexes was not constrained in the GTL specifications, these violations are allowed to happen. On the other side, Figure 5 shows that for each bin, the density of the swarm satisfies the prescribed capacity constraints at each new time index. Moreover, for bins $0, 1, 5, 6$, we can observe that the final density distribution is the one imposed by the GTL constraints. Thus, we demonstrate with this example the correctness of the control algorithm.

C. ROS-Gazebo Simulation

In this section, we evaluate the algorithm developed in this paper in a high-fidelity simulation environment. For this purpose, we upgraded the implementation of the gridworld example to interact with Gazebo (a robust physics engine and high-quality graphics simulator). We consider a virtual scenario in which a building of a custom-made world has

[1] https://github.com/u-t-autonomous/RSS2020_SwarmControlGTL.git

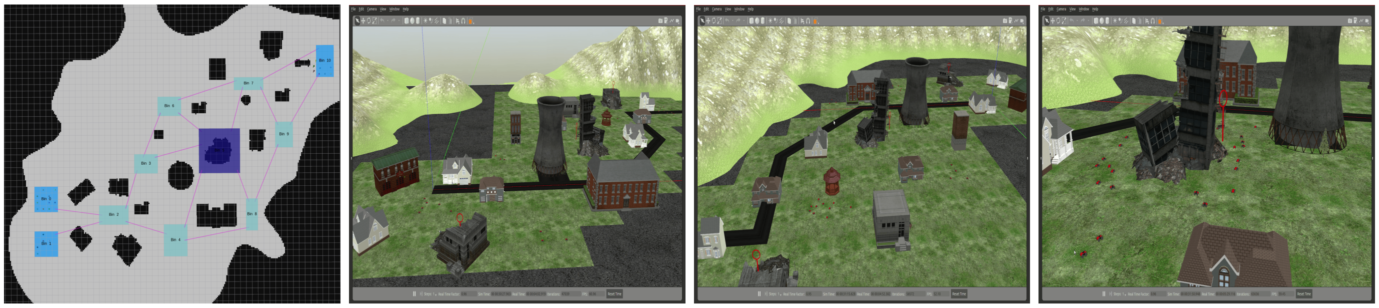

Fig. 4. Gazebo[1] environment for the rescue mission. From the left to the right: the 2D occupancy grid map of the gazebo world with the bins topology and the distribution of the swarm in the gazebo world at time indexes $3, 4$ and 5.

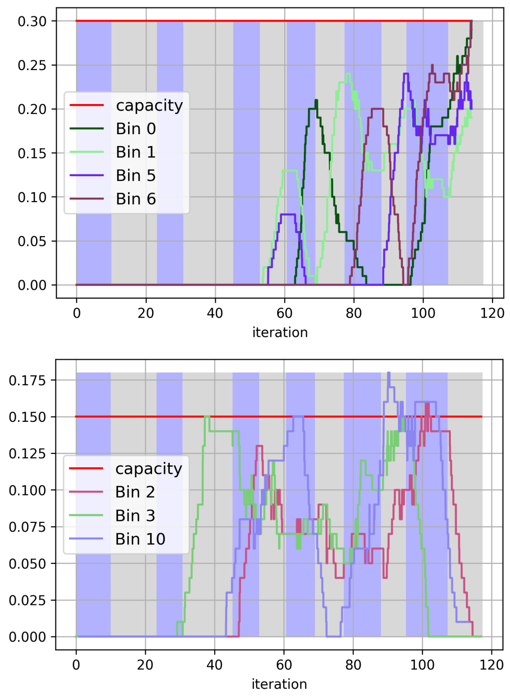

Fig. 5. Evolution of the density distribution of the swarm in specific bins. The switch in the color bands represents the synchronization time (called time index in this paper) at which each agent decides to transit to a neighbor bin.

collapsed, and a swarm of 24 robots are constrained to first secure some surrounding buildings before trying to secure the collapsed building. For that purpose, we compute the 2D occupancy grid map of the environment and build on top of it the graph in which the agents move. Figure 4 and video[1] show the evolution of the swarm in the gazebo world. The specifications of the mission are defined as follows.

- $(x = 0)\mathcal{U}_{\leq 5}(\exists^4 \bigcirc (0.1 \leq x \leq 0.4))$ is specified for bin 5. This formula specifies that the density of the swarm in bin 5 must be 0 until the densities of the swarm in at least 4 of the neighbors of bin 5 become between 0.1 to 0.4. Intuitively, this means that no agents should enter the damaged area (bin 5) until a certain number of agents have surrounded the area.
- $\lozenge_{[0,5]}(x \geq 0.9)$ is specified for bin 5. This formula specifies that at least 90% of the agents should eventually be in the damaged area in less than 5 time units.
- $\square_{[0,5]}(x \leq 0.35)$ is specified for all bins, except for bins $0, 1, 5, 10$. This formula specifies the capacity constraints in bins other than bins $0, 1, 5, 10$.

We first compute $M(t)$ using the MCD, then the robots compute their target node in a decentralized manner according to Algorithm 1. On the low-level control, collision avoidance is achieved using ORCA and the local sensing capabilities of the robots. The video of the gazebo simulation demonstrates the correctness of the MCD algorithm. We observe that to satisfy the GTL specifications, the synthesized controller leads the agents in the swarm towards the neighbor bins of the damaged area, before sending them in the area.

VII. Conclusions

The paper develops a correct-by-construction algorithm to control, in a decentralized and probabilistic manner, the density distribution of a swarm of autonomous agents subject to GTL specifications. The algorithm is independent of the number of agents in the swarm and relies on synthesizing a time-varying Markov matrix by solving an optimization problem. The synthesized Markov matrix is independently used by each agent to determine its next target while the entire swarm satisfies the GTL specifications. The complexity analysis of the developed algorithm shows that it significantly improves scalability over existing approaches. We also demonstrated the correctness of the algorithm in the special case and in the general case using the MCD. We successfully demonstrated the efficiency and the correctness of the algorithm in simulation. Although the paper uses finite-horizon GTL to specify the collective behavior of the swarm, future work will extend the finite-horizon GTL to infinite-horizon GTL, and investigate efficient and more accurate modifications of the algorithm. We will also consider experiments in a real-world setting involving swarms of low-cost autonomous vehicles.

Acknowledgments

This work is supported by the grants AFRL FA9550-19-1-0169, and DARPA D19AP00004.

References

[1] Behçet Açikmeşe and David S Bayard. A Markov chain approach to probabilistic swarm guidance. In *2012 American Control Conference (ACC)*, pages 6300–6307. IEEE, 2012.

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49 (5):672–713, September 2002. ISSN 0004-5411. doi: 10.1145/585265.585270.

[3] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, 33(5):1103–1123, 2017.

[4] Pietro Belotti. Couenne: a user's manual. Technical report, Technical report, Lehigh University, 2009.

[5] Spring Berman, Ádám Halász, M Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25 (4):927–937, 2009.

[6] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Souymodip Chakraborty and Joost-Pieter Katoen. Parametric LTL on markov chains. In *IFIP International Conference on Theoretical Computer Science*, pages 207–221. Springer, 2014.

[9] Ishanu Chattopadhyay and Asok Ray. Supervised self-organization of homogeneous swarms using ergodic projections of Markov chains. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6): 1505–1515, 2009.

[10] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[11] Nazlı Demir, Utku Eren, and Behçet Açıkmeşe. Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39 (4):537–554, 2015.

[12] Gerald Gamrath, Tobias Fischer, Tristan Gally, Ambros M. Gleixner, Gregor Hendel, Thorsten Koch, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Stefan Vigerske, Dieter Weninger, Michael Winkler, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 3.2. ZIB-Report 15-60, Zuse Institute Berlin, February 2016. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-57675.

[13] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019. URL http://www.gurobi.com.

[14] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Gros, and Calin Belta. Spatel: a novel spatial-temporal logic and its applications to networked systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 189–198, 2015.

[15] Iman Haghighi, Sadra Sadraddini, and Calin Belta. Robotic swarm control from spatio-temporal specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5708–5713. IEEE, 2016.

[16] Aldo Jaimes, Srinath Kota, and Jose Gomez. An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav (s). In *2008 IEEE International Conference on System of Systems Engineering*, pages 1–6. IEEE, 2008.

[17] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[18] Marius Kloetzer, Xu Chu Ding, and Calin Belta. Multi-robot deployment from LTL specifications with reduced communication. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 4867–4872. IEEE, 2011.

[19] Hadas Kress-Gazit, Tichakorn Wongpiromsarn, and Ufuk Topcu. Correct, reactive, high-level robot control. *IEEE Robotics & Automation Magazine*, 18(3):65–74, 2011.

[20] Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. A review and comparison of solvers for convex minlp. *Optimization and Engineering*, 20(2):397–455, 2019.

[21] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[22] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior*, 15(3):289–305, 2007.

[23] Salar Moarref and Hadas Kress-Gazit. Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. *Autonomous Robots*, pages 1–16, 2019.

[24] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[25] Nikhil Nigam, Stefan Bieniawski, Ilan Kroo, and John Vian. Control of multiple uavs for persistent surveillance: algorithm and flight test results. *IEEE Transactions on Control Systems Technology*, 20(5):1236–1251, 2011.

[26] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977.

[27] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[28] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In

2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS), pages 249–258. IEEE, 2017.

[29] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Synchronous and asynchronous multi-agent coordination with cltl+ constraints. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 335–342. IEEE, 2017.

[30] Martin Saska, Vojtěch Vonásek, Jan Chudoba, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems*, 84(1-4):469–492, 2016.

[31] Mohit Tawarmalani, Nikolaos V Sahinidis, and Nikolaos Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, volume 65. Springer Science & Business Media, 2002.

[32] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[33] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[34] Wayne L Winston and Jeffrey B Goldberg. *Operations research: applications and algorithms*, volume 3. Thomson/Brooks/Cole Belmontˆ eCalif Calif, 2004.

[35] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[36] Z. Xu and A. A. Julius. Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Trans. Autom. Sci. Eng.*, 15(1):264–277, Jan. 2018. ISSN 1545-5955. doi: 10.1109/TASE.2016.2611536.

[37] Z. Xu, F. M. Zegers, B. Wu, W. Dixon, and U. Topcu. Controller synthesis for multi-agent systems with intermittent communication. a metric temporal logic approach. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1015–1022, Sep. 2019. doi: 10.1109/ALLERTON.2019.8919727.

[38] Zhe Xu, Alexander J Nettekoven, A. Agung Julius, and Ufuk Topcu. Graph temporal logic inference for classification and identification, 2019.

[39] R. Yan, Z. Xu, and A. Julius. Swarm signal temporal logic inference for swarm behavior analysis. *IEEE Robotics and Automation Letters*, 4(3):3021–3028, July 2019. ISSN 2377-3774. doi: 10.1109/LRA.2019.2924843.