

2016-01-01

Robotic swarm control from spatio-temporal specifications

Iman Haghighi, Sadra Sadraddini, Calin Belta. 2016. "Robotic Swarm Control from Spatio-Temporal Specifications." 2016 IEEE 55TH Conference On Decision And Control (CDC), pp. 5708 - 5713 (6).

<https://hdl.handle.net/2144/29822>

"Downloaded from OpenBU. Boston University's institutional repository."

Robotic Swarm Control from Spatio-Temporal Specifications

Iman Haghighi, Sadra Sadraddini, and Calin Belta

Abstract—In this paper, we study the problem of controlling a two-dimensional robotic swarm with the purpose of achieving high level and complex spatio-temporal patterns. We use a rich spatio-temporal logic that is capable of describing a wide range of time varying and complex spatial configurations, and develop a method to encode such formal specifications as a set of mixed integer linear constraints, which are incorporated into a mixed integer linear programming problem. We plan trajectories for each individual robot such that the whole swarm satisfies the spatio-temporal requirements, while optimizing total robot movement and/or a metric that shows how strongly the swarm trajectory resembles given spatio-temporal behaviors. An illustrative case study is included.

I. INTRODUCTION

Robotic swarms have received a lot of attention from the robotics research community in recent years [1]. Large teams of robots are suitable for a broad range of applications such as distributed task allocation [2], area patrolling and coverage [3], [4], search and rescue missions [5], and simultaneous localization and mapping (SLAM) [6]. With the recent technological developments, producing a large number of inexpensive robots that are equipped with sophisticated sensing, computation and communication tools has become a reality.

Describing complex spatial specifications for swarms is a non-trivial task. The existing methods rely on spatial configurations generated from simple geometrical shapes, potential fields or sets of target points [7], [8], [9], [10]. However, it is practically easier to specify collective spatial behaviors of a swarm as opposed to specifying trajectories for each individual robot. The authors in [11], [12] introduced a method for controlling the abstract behavior of swarms based on the first and second moments of their spatial distribution. This is a useful approach to specify some simple patterns such as ellipsoids and boxes. However, there is a necessity for a more powerful framework of pattern specification that is not only easily definable and interpretable by the user, but is also rich enough to capture a wide range of complex spatial patterns that are not expressible by merely statistical moments. For this reason, we propose to use a formal spatial logic [13] that is capable of describing high level global behaviors in multi agent systems.

This work was partially supported by the National Science Foundation under grants NRI-1426907 and CMMI-1400167.

I. Haghighi is with the Division of Systems Engineering, Boston University, Boston MA, USA haghighi@bu.edu

S. Sadraddini is with the Department of Mechanical Engineering, Boston University, Boston MA, USA sadra@bu.edu

C. Belta is with the Department of Mechanical Engineering, Boston University, Boston MA, USA cbelta@bu.edu

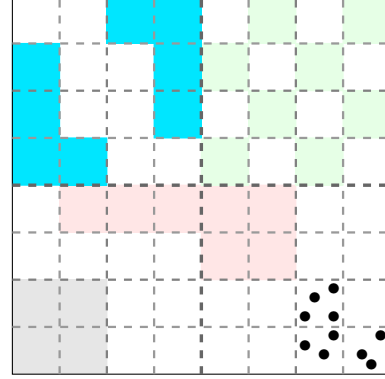


Fig. 1. An example of spatio-temporal patterning requirements for a swarm: While avoiding the unsafe zone (red) at all times, attain the following formations in any order within 30 seconds: 1) form a checkerboard pattern (green) by populating every other cell on the north east quadrant of the workspace. 2) Populate one of the grey squares in the south west quadrant. After completing both tasks, gather in one of the L shaped upload regions (cyan). All the tasks must be completed within 40 seconds and each formation must be maintained for at least 3 seconds.

In this paper, we consider square-shaped two dimensional workspaces that are gridded to equal-sized cells as illustrated in Fig. 1. A user can express spatial requirements for the swarm by defining shapes that can be formed by unions of cells in the grid. The user can give the swarm choices between distinct patterns. Furthermore, they can also specify certain requirements for how these patterns should evolve over time. An example of such a spatio-temporal specification is given in Fig. 1. Such specifications involve logical reasoning and provide different choices for the swarm movement. A wide variety of complex patterns can be defined in this framework that are not easily expressible by earlier work in the literature. However, specifications of this type can be naturally expressed as spatial temporal logic (SpaTeL) [14] formulas.

We formulate the swarm motion planning problem corresponding to a SpaTeL specification as a mixed integer linear programming (MILP) problem. A feasible solution to this problem provides a high-level plan for the movements of the swarm. We are also able to optimize a cost function. For instance, we are able to minimize the total swarm movement (energy), or find a plan that maximizes the satisfaction of a specification (robustness), or a combination of both. Finally, we develop a low-level strategy to move each individual robot according to the high level plan. Two different solutions for the specification given in Fig. 1 corresponding to two different initial conditions are given in Fig. 5 and 6. It can be seen that the optimal solution in Fig. 5 involves populating

the grey region before forming the checkerboard pattern and populating the right cyan region at the end, while the optimal solution in Fig. 6 forms the checkerboard pattern first and populates the left cyan region.

Although temporal logic specifications in the context of mobile robot control and motion planning have been recently explored in the literature, there is very limited prior work in which complex requirements are expressed in *both* space and time. The authors in [12] attempt to solve a similar problem, but spatial specifications are limited to statistical moments of the swarm in their work and thus complex spatial patterns are not easily expressible. The authors in [15] introduced a procedure to specify emergent spatial behaviors in swarms by linear temporal logic and used model checking techniques to verify such behaviors in swarms, but the control problem is not discussed in that work. As opposed to linear temporal logic multi-robot motion planning [16], [17], [18], [19], our solution is optimization-based which is advantageous in the following ways. First, we are able to optimize a general cost function, which is difficult to formalize in automata-based approaches. We are also able to deal with infeasibility by minimizing the *distance* of the swarm trajectory from satisfaction. Furthermore, under some relaxations, the complexity of our approach is independent of the size of the swarm. Therefore, our approach is easily applicable to large swarms.

This paper is organized as follows. First, we provide the necessary background on SpaTeL specifications in Sec. II. Next, the problem is formulated in Sec. III. The solution and the technical details are explained in Sec. IV. Finally, an illustrative case study is presented in Sec. V.

II. PRELIMINARIES

A. Quad Transition Systems

A *quad transition system* (QTS) [13], [14] is a tree data structure defined as the tuple $Q(t) = (\mathcal{V}, \mathcal{E}, v_l, V_f, \mu, \mathcal{L}, l)$, where: \mathcal{V} is the set of nodes (vertices). $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of directed edges (transitions). We say that v_2 is a *child* of v_1 if and only if $(v_1, v_2) \in \mathcal{E}$; v_l is the root (the only node which is not a child of another node); V_f is the set of leaves (nodes without children); $\mu : \mathcal{V} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^+$ is the valuation function, designating each node a real value at any given time $t \geq 0$; \mathcal{L} is a finite set of labels; $l : \mathcal{E} \rightarrow \mathcal{L}$ is the labeling function that maps each edge to a label.

Let $A(t) \in \mathbb{R}^{2^D \times 2^D}$ represent a time-varying $2^D \times 2^D$ matrix, where $D \in \mathbb{N}$ is the *depth* of the matrix and $t \in \mathbb{R}_{\geq 0}$ is time. We construct a QTS from $A(t)$ as follows. We let the *root* node v_l represent $A(t)$. Next, we partition the matrix into four $2^{D-1} \times 2^{D-1}$ sub-matrices, where each sub-matrix is represented by a child of v_l . We label each edge with a *directional* label from the set $\mathcal{L} = \{NW, NE, SE, SW\}$, where *NW* represents north west, *SE* represents south east, etc (see Fig. 2). Next, we execute the same procedure for each child until the leaf nodes are obtained, i.e. each leaf is a single element matrix. Note that $|V_f| = 2^{2D}$ and $|\mathcal{V}| = \sum_{i=0}^D 2^{2^i}$. For each leaf node $v_f \in V_f$, we let $\mu(v_f, t)$ to be the

value of the corresponding element in $A(t)$. The valuation function for other nodes is recursively defined as the sum of the valuations of its children, i.e.

$$\mu(v, t) = \sum_{(v, v_c) \in \mathcal{E}} \mu(v_c, t) \quad \forall v \in \mathcal{V} \setminus V_f. \quad (1)$$

An example of a QTS construction is given in Fig. 2.

Given a subset of labels $\mathcal{B} \subseteq \mathcal{L}$, a *labeled path* of a QTS is defined as a function that maps a vertex to a set of infinite sequences of nodes:

$$\lambda^{\mathcal{B}}(v_0) := \{(v_0, v_1, v_2 \dots, \overline{v_f}) \mid (v_i, v_{i+1}) \in \mathcal{E}, v_f \in V_f, l(v_i, v_{i+1}) \in \mathcal{B}, i \in \mathbb{N}_{\geq 0}\}, \quad (2)$$

where $\overline{v_f}$ denotes infinite repetitions of leaf node v_f . The i -th element of a labeled path $\pi \in \lambda^{\mathcal{B}}(v)$ is denoted by π_i . For example, in Fig. 2, $(v_0, v_1, \overline{v_5})$ and $(v_0, v_1, \overline{v_6})$ are both members of $\lambda^{\{NW, NE\}}(v_0)$.

A *QTS signal* starting at time t is defined as $Q_t = \{Q(\tau) \mid \tau \geq t\}$.

B. Spatial Temporal Logic (SpaTeL)

SpaTeL formulas are defined by nesting tree spatial superposition logic (TSSL) specifications [13] inside temporal operators of signal temporal logic (STL) [20]. Formal definitions of SpaTeL syntax and semantics can be found in [14]. Informally, SpaTeL formulas are STL formulas in which linear predicates over signals are replaced with spatial formulas over quad transition systems and allow for describing how spatial patterns change over time.

A TSSL formula is recursively formed by linear predicates over the valuation function (1), spatial operators, and boolean operators (\wedge, \vee, \neg). For example, $\varphi := \mu \sim c$, where $\sim \in \{\geq, <\}$, is a very simple TSSL formula consisting of a single predicate that indicates that the function μ of (1) at the initial node v_l must have a value of larger (smaller) than threshold c . If the predicate is true, we write $Q^{v_l} \models \varphi$ (read as QTS Q satisfies formula φ at node v_l). Spatial operators are used to define specifications at nodes located in lower tree levels. For instance, $\exists_{\mathcal{B}} \bigcirc \varphi$ ($\mathcal{B} \subseteq \{NW, NE, SW, SE\}$) is the spatial “there exists next” operator which means that the spatial formula φ has to be satisfied for at least one of the children of the initial node with directional label $l(v_l, v') \in \mathcal{B}$. Furthermore, $\forall_{\mathcal{B}} \bigcirc \varphi$, read as “for all next”, indicates that φ must be satisfied by all such children. Specifications at deeper tree levels can be expressed similarly by nesting several spatial next operators. In addition to spatial next, TSSL is equipped with spatial until operators ($\exists_{\mathcal{B}} \varphi_1 \mathcal{U}_{\kappa} \varphi_2, \forall_{\mathcal{B}} \varphi_1 \mathcal{U}_{\kappa} \varphi_2$). Formal definitions for all these operators are presented in [13].

A SpaTeL formula is recursively formed by TSSL formulae, temporal operators, and boolean operators. Three common temporal operators are eventually (F_I), always (G_I), and until (U_I), where $I = [t_1, t_2]$ is a time interval. For instance, A QTS signal $Q_t \models F_I \varphi$ (read as Q_t satisfies $F_I \varphi$) if there $\exists \tau \in [t + t_1, t + t_2]$ such that $Q_{\tau} \models \varphi$ and $Q_t \models G_I \varphi$ if $\forall \tau \in [t + t_1, t + t_2]$ $Q_{\tau} \models \varphi$. A SpaTeL formula is satisfied by a QTS signal Q_t if and only if it is satisfied by the QTS at time t ($Q(t)$).

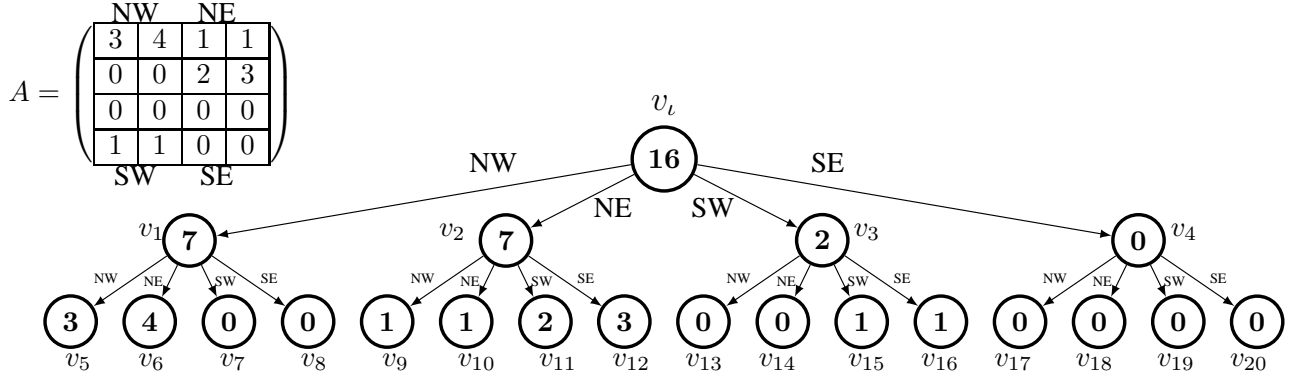


Fig. 2. The QTS corresponding to the matrix A .

Example 1: Consider a 4×4 matrix with the requirement that every other entry is zero (thus forming a checkerboard pattern). There are two different realizations for this pattern, that can be specified by the following TSSL formulas:

$$\begin{aligned}\varphi_{c1} &= \forall_{\mathcal{L}} \circ (\forall_{\{NW, SE\}} \circ (\mu = 0)) \\ \varphi_{c2} &= \forall_{\mathcal{L}} \circ (\forall_{\{NE, SW\}} \circ (\mu = 0)),\end{aligned}$$

where μ is the valuation function in the definition of QTS. Now consider a spatio-temporal requirement that the checkerboard pattern periodically switches tiles. The following SpaTeL formula specifies this requirement:

$$\Phi_c = G_{[0, t_1]}(F_{[0, t_2]}\varphi_{c1} \wedge F_{[0, t_2]}\varphi_{c2}). \quad (3)$$

SpaTeL is equipped with quantitative semantics. Quantitative valuation (robustness) $\rho(\Phi, Q_t)$ of a SpaTeL formula Φ with respect to QTS signal Q_t is calculated recursively:

$$\begin{aligned}\rho(\varphi, Q_t) &= \rho_s(\varphi, v_l), \\ \rho(\neg\Phi, Q_t) &= -\rho(\Phi, Q_t), \\ \rho(\Phi_1 \wedge \Phi_2, Q_t) &= \min(\rho(\Phi_1, Q_t), \rho(\Phi_2, Q_t)), \\ \rho(\Phi_1 \vee \Phi_2, Q_t) &= \max(\rho(\Phi_1, Q_t), \rho(\Phi_2, Q_t)), \\ \rho(F_{I_1, I_2}\Phi, Q_t) &= \sup_{t' \in [t+I_1, t+I_2]} \rho(\Phi, Q_{t'}), \\ \rho(G_{I_1, I_2}\Phi, Q_t) &= \inf_{t' \in [t+I_1, t+I_2]} \rho(\Phi, Q_{t'}), \\ \rho(\Phi_1 U_{[I_1, I_2]} \Phi_2, Q_t) &= \sup_{t' \in [t+I_1, t+I_2]} (\min(\rho(\Phi_2, Q_{t'}), \inf_{t'' \in [t+I_1, t']} \rho(\Phi_1, Q_{t''}))),\end{aligned} \quad (4)$$

where $\rho_s(\varphi, v)$ is the robustness of TSSL formula φ with respect to node $v \in \mathcal{V}$:

$$\begin{aligned}\rho_s(\top, v) &= 1, \\ \rho_s(\mu \sim c, v) &= (\mu - c) \text{ if } (\sim \text{ is } \geq), (c - \mu) \text{ if } (\sim \text{ is } \leq), \\ \rho_s(\neg\varphi, v) &= -\rho_s(\varphi, v), \\ \rho_s(\varphi_1 \wedge \varphi_2, v) &= \min(\rho_s(\varphi_1, v), \rho_s(\varphi_2, v)), \\ \rho_s(\varphi_1 \vee \varphi_2, v) &= \max(\rho_s(\varphi_1, v), \rho_s(\varphi_2, v)), \\ \rho_s(\exists_{\mathcal{B}} \circ \varphi, v) &= \max_{\pi \in \lambda^{\mathcal{B}}(v)} \rho_s(\varphi, \pi_1), \\ \rho_s(\forall_{\mathcal{B}} \circ \varphi, v) &= \min_{\pi \in \lambda^{\mathcal{B}}(v)} \rho_s(\varphi, \pi_1), \\ \rho_s(\exists_{\mathcal{B}} \varphi_1 U_k \varphi_2, v) &= \sup_{\pi \in \lambda^{\mathcal{B}}(v), i \in (0, k]} (\min(\rho_s(\varphi_2, \pi_i), \inf_{j \in [0, i]} \rho_s(\varphi_1, \pi_j))), \\ \rho_s(\forall_{\mathcal{B}} \varphi_1 U_k \varphi_2, v) &= \inf_{\pi \in \lambda^{\mathcal{B}}(v), i \in (0, k]} (\min(\rho_s(\varphi_2, \pi_i), \inf_{j \in [0, i]} \rho_s(\varphi_1, \pi_j))).\end{aligned} \quad (5)$$

Positive and negative robustness indicate satisfaction and violation, respectively.

$$\begin{aligned}\rho(\Phi, Q_t) &> 0 \Rightarrow Q_t \models \Phi, \\ \rho(\Phi, Q_t) &< 0 \Rightarrow Q_t \not\models \Phi.\end{aligned} \quad (6)$$

The absolute robustness value can be viewed as a measure of "distance to satisfaction". In other words, a higher absolute value for robustness indicates stronger satisfaction (violation) of a specification. We use the definition of robustness ((4), (5)) in subsequent sections to translate SpaTeL specifications into mixed integer constraints.

Example 2 (Example 1 continued): Consider a stationary QTS $Q(t) = \mathcal{Q}$, where \mathcal{Q} is the QTS depicted in Fig. 2. By computing quantitative semantics from (5), it is straightforward to verify that specification Φ_c in (3) is violated by Q_0 with a robustness of -4 .

The horizon of a SpaTeL formula is defined similar to STL [21]. Intuitively, the horizon T of a SpaTeL formula Φ is the maximum time for which some specification in Φ must be checked against Q_t . For instance, the time horizon of $G_{[0, 20]} F_{[0, 5]} \forall_{\mathcal{L}} \circ (\mu \geq 1)$ is $T = 25$.

III. PROBLEM FORMULATION AND APPROACH

Consider N homogenous planar robots with negligible sizes in a two-dimensional space. The position of robot r at time t is denoted by $x_r(t) \in \mathcal{X}$, $r = 1, \dots, N$, where $\mathcal{X} \subset \mathbb{R}^2$ is the workspace of the robots, which is assumed to be the following square: $\mathcal{X} := [-\frac{a}{2}, \frac{a}{2}] \times [-\frac{a}{2}, \frac{a}{2}]$, where a is the length of the square. Note that any rectangular workspace can be normalized to meet this assumption. We denote the state of the swarm by $x(t) = (x_1(t)^T, x_2(t)^T, \dots, x_N(t)^T)^T$. The kinematics of each individual robot is assumed as follows:

$$\dot{x}_r(t) = u_r(t), \quad r = 1, \dots, N, \quad (7)$$

where $u_r(t) \in \mathcal{U}$ is the control applied to robot r at time t and $\mathcal{U} = \{u_r \mid \|u_r\|_2 \leq u_m\}$, where u_m is the maximum speed that a robot can attain.

\mathcal{X} is partitioned into $2^D \times 2^D$ number of equal-sized cells, where D is the depth of the grid. A user expresses desirable patterns by defining shapes that are formed by unions of cells in the workspace, defining thresholds for the number

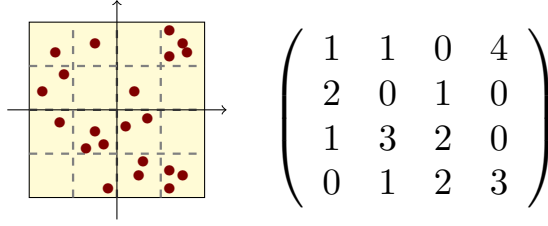


Fig. 3. (Left) A swarm of 21 robots in a square region. The square is gridded into 16 cells. (Right) The matrix representing the number of the robots in each cell.

of agents populating each shape, and expressing temporal requirements for those pre-identified patterns. The objective in this paper is to synthesize a control policy for (7) such that the spatio-temporal requirements expressed by the user are met. We will provide a formal formulation for this problem later in this section.

We construct the matrix $\mathcal{N}(t) \in \mathbb{N}^{2^D \times 2^D}$, where the value of each element is the number of robots in the corresponding cell, as illustrated in an example in Fig. 3. We construct the time varying QTS $Q(t)$ from $\mathcal{N}(t)$ using the procedure outlined in Sec. II-A. Note that the shapes defined by unions of cells can be easily expressed using the spatial next operator in tree spatial superposition logic (see Example 3). Consequently, a SpaTeL specification Φ can be automatically generated from the input specification.

Remark 1: A supervised learning algorithm is proposed in [13] for automatically learning TSSL formulae that are satisfied by a set of positive spatial configurations (images) and violated by a set of negative images. This method can be used to learn TSSL (and SpaTeL) formulas describing more complex high level patterns (circular clusters, ellipsoids, etc). Two sets of images, one illustrating the desirable pattern and the other lacking the pattern, should be artificially created. The learning algorithm generates a TSSL formula by using the generated images as a training set. Although this is a very effective method to find SpaTeL descriptors for arbitrary patterns, the resulting formulas are often too long and complex. Therefore, The mixed integer linear programming problems that result from the framework presented in subsequent sections become unsolvable by existing solvers. As a result, the input spatio-temporal requirements in this paper are limited to unions of squares. As explained in Sec. II-B, such requirements can be intuitively translated into TSSL/SpaTeL specifications and the resulting formulae are small and manageable.

Example 3: The specification that was introduced in Sec. I (Fig. 1) is formalized by the following SpaTeL formula:

$$\Phi_d = G_{[0,40)}(\neg\varphi_1) \wedge (F_{[0,30)}G_{[0,3)}\varphi_2) \wedge (F_{[0,30)}G_{[0,3)}\varphi_3) \wedge (F_{[30,40)}\varphi_4), \quad (8)$$

where φ_i are TSSL formulas describing different patterns illustrated in Fig. 1: φ_1 represents the red danger zone in Fig. 1, φ_2 specifies formation of a checkerboard pattern in the north west quadrant, φ_3 specifies gathering inside one of the grey cells in the south west quadrant, and φ_4 represents populating one of the L-shaped cyan regions. These formulas

are automatically generated by representing each cell in the gridded workspace using appropriate spatial next operators of TSSL.

$$\begin{aligned} \varphi_1 &= \forall_{SE} \circ \forall_{NW} \circ (\mu \leq 0) \wedge \\ &\quad \forall_{SW} \circ \forall_{NE} \circ \forall_{\{NW, NE\}} \circ (\mu \leq 0) \wedge \\ &\quad \forall_{SW} \circ \forall_{NW} \circ \forall_{NE} \circ (\mu \leq 0), \\ \varphi_2 &= \forall_{NE} \circ (\forall_{\mathcal{L}} \circ \forall_{\{NW, SE\}} \circ (\mu \geq \gamma_1)), \\ \varphi_3 &= \forall_{SW} \circ (\forall_{SW} \circ \exists_{\mathcal{L}} \circ (\mu \geq \gamma_2)), \\ \varphi_4 &= \forall_{NW} \circ (\varphi_5 \vee \varphi_6), \\ \varphi_5 &= (\forall_{NE} \circ \forall_{\{NW, NE, SE\}} \circ (\mu \geq \gamma_3)) \wedge \\ &\quad (\forall_{SE} \circ \forall_{NE} \circ (\mu \geq \gamma_4)), \\ \varphi_6 &= (\forall_{SW} \circ \forall_{\{NW, SW, SE\}} \circ (\mu \geq \gamma_5)) \wedge \\ &\quad (\forall_{NW} \circ \forall_{SW} \circ (\mu \geq \gamma_6)), \end{aligned} \quad (9)$$

where $\mathcal{L} = \{NW, NE, SW, SE\}$, and μ is the the number of robots residing in a subregion of the workspace identified by spatial operators and γ_{1-6} are thresholds for the minimum number of robots that are required to populate each pattern.

We wish to find a control strategy that steers the swarm such that Φ is satisfied. Such a policy is not usually unique. Therefore, we choose a policy that optimizes a cost function. For instance, we can minimize the total number of robot displacements (one displacement is defined as moving one robot from its current location to a neighboring cell). In addition, a natural candidate for optimization is maximizing the SpaTeL robustness. The problem that we consider in this paper is formulated as follows:

Problem 1: Given a swarm of N agents with initial positions at $x(0)$ and a SpaTeL formula Φ that describes time varying spatial requirements of the user, find an *optimal and correct* control strategy such that:

$$\begin{aligned} u_r(t) &= \underset{r=1, \dots, N, t \in [0, T]}{\operatorname{argmin}} \quad -\alpha \rho(\Phi, Q_0) + J_f(x(T)) \\ &\quad + \int_0^T J_r(x(t), u(t)) dt, \\ \text{s.t.} \quad &Q_0 \models \Phi, \end{aligned} \quad (10)$$

where ρ is the SpaTeL robustness, Q_0 is the QTS signal starting at time 0, $J_f : \mathbb{R}^{2N} \rightarrow \mathbb{R}$, $J_r : \mathbb{R}^{2N} \times \mathbb{U}^N \rightarrow \mathbb{R}$, are the endpoint cost and the running cost (Lagrangian), respectively. The end time T is the time horizon of the SpaTeL formula and α is a positive constant designating a weight for SpaTeL robustness.

Our approach to problem 1 can be summarized as follows. First, we find $\mathcal{N}(t), 0 \leq t \leq T$ such that $Q_0 \models \Phi$. It is known that this problem is undecidable in continuous time [22]. Therefore, we (approximately) solve the problem in discrete time assuming that at each time step, each robot can be displaced by one cell to its right, left, up or down. Therefore, we choose a sampling time such that: $\Delta t \geq \frac{a}{2^{D-1}u_m}$. We assume that the time intervals of the temporal operators of Φ are multiples of Δt . This assumption can be matched by increasing D such that the time intervals can be reasonably approximated by multiples of Δt . We also denote the last discrete time by $K := \frac{T}{\Delta t}$. The matrix \mathcal{N} at time $t = k\Delta t$ is denoted by $\mathcal{N}[k]$. We construct a discrete time model for the evolution of $\mathcal{N}[k]$. Next, we find the required values at each time such that the SpaTeL specification is

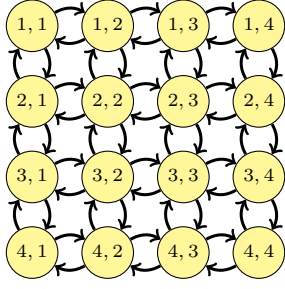


Fig. 4. A flow network with 16 cells. The robots are only able to move from one cell to a neighboring cell in one time step.

satisfied using a MILP-based approach that is explained in Sec. IV. Finally, we find continuous time controls for each individual robot such that the number of each cell at time $t = k\Delta t$ matches its corresponding value in $\mathcal{N}[k]$.

IV. SOLUTION

A. Swarm Flow in Discrete Time

In this section, we develop a discrete time model that characterizes the evolution of $\mathcal{N}[k]$. At each time step, each robot is only able to remain at its current cell or move to an adjacent cell (The cell to its right, left, top, or down). All the robots move synchronously during one time step. The flow of the robots between the cells can be thought as a network as depicted in Fig. 4. The index of each cell is represented by $[i, j]$, where i is the row and j is the column of the element in the matrix $\mathcal{N}[k]$. The set of cells that are adjacent to $[i, j]$ is denoted by $\Omega([i, j])$. We denote the number of robots in the cell $[i, j]$ at time step k by $\mathcal{N}_{[i, j]}[k]$. The number of robots that move from cell $[i, j]$ to an adjacent cell $[i', j'] \in \Omega([i, j])$ during time $[k, k + 1]\Delta t$ is denoted by $f_{[i, j]}^{[i', j']}[k]$, which is a non-negative integer. The total number of robots that move out from cell $[i, j]$ at time step k is:

$$f_{[i, j]}^{out}[k] := \sum_{[i', j'] \in \Omega([i, j])} f_{[i, j]}^{[i', j']}[k]. \quad (11)$$

We add the following constraint:

$$\mathcal{N}_{[i, j]}[k] \geq f_{[i, j]}^{out}[k], \quad (12)$$

which indicates that the number of robots moving out from a cell can not be more than the number of robots in the cell. The number of robots that enter cell $[i, j]$ at k is:

$$f_{[i, j]}^{in}[k] := \sum_{[i', j'] \in \Omega([i, j])} f_{[i', j']}^{[i, j]}[k]. \quad (13)$$

The discrete time evolution of $\mathcal{N}[k]$ is:

$$\mathcal{N}_{[i, j]}[k + 1] = \mathcal{N}_{[i, j]}[k] - f_{[i, j]}^{out}[k] + f_{[i, j]}^{in}[k], \quad (14)$$

which is a function of decisions made on the values of $f_{[i, j]}^{[i', j']}[k]$. In a compact form, we define the decision variable $f[k]$ as the set:

$$f[k] = \left\{ f_{[i, j]}^{[i', j']} \mid \forall [i', j'] \in \Omega([i, j]), \forall [i, j] \right\}, \quad (15)$$

and the discrete time evolution of $\mathcal{N}[k]$ is written as:

$$\mathcal{N}[k + 1] = \mathcal{F}(\mathcal{N}[k], f[k]). \quad (16)$$

B. Mixed-Integer Formulation of SpaTeL Specifications

In this section, we explain how to recursively transform a SpaTeL formula into a set of mixed-integer constraints. Our method is inspired by the binary mixed-integer encoding of STL formulas presented in [22].

For a predicate of a SpaTeL formula $\sigma = (\mu \geq c)$, a set of binary variables $z_\sigma[v, k] \in \{0, 1\}$, $v \in \mathcal{V}$, $0 \leq k \leq K$, is associated such that values 1 and 0 indicate *True* and *False*, respectively. The corresponding mixed integer constraints are:

$$\begin{cases} \mu[v, k] - Mz_\sigma[v, k] & \leq c, \\ \mu[v, k] + M(1 - z_\sigma[v, k]) & \geq c, \end{cases} \quad (17)$$

where M is a sufficiently large positive number. Mixed integer constraints for all predicates in the form of $\sigma' = (\mu \leq c)$ are defined similarly. For encoding a SpaTeL formula, The following rules are used to map boolean, temporal, and spatial operators into mixed integer constraints. These rules are derived from the definition of SpaTeL robustness (4),(5).

- *Negation:*

$$\Psi = \neg\Phi \rightarrow z_\Psi[v, k] = 1 - z_\Phi[v, k];$$

- *Conjunction:*

$$\Psi = \bigwedge_{i=1}^m \Phi_i \rightarrow \begin{cases} z_\Psi[v, k] \leq z_{\Phi_i}[v, k], i = 1, \dots, m, \\ z_\Psi[v, k] \geq 1 - m + \sum_{i=1}^m z_{\Phi_i}[v, k]; \end{cases}$$

- *Disjunction:*

$$\Psi = \bigvee_{i=1}^m \Phi_i \rightarrow \begin{cases} z_\Psi[v, k] \geq z_{\Phi_i}[v, k], \\ z_\Psi[v, k] \leq \sum_{i=1}^m z_{\Phi_i}[v, k]; \end{cases}$$

- *There exists spatial next:*

$$\Psi = \exists_{\mathcal{B}} \bigcirc \Phi \rightarrow z_\Psi[v, k] = \bigvee_{\pi \in \lambda^{\mathcal{B}}(v)} z_\Phi[\pi_1, k];$$

- *For all spatial next:*

$$\Psi = \forall_{\mathcal{B}} \bigcirc \Phi \rightarrow z_\Psi[v, k] = \bigwedge_{\pi \in \lambda^{\mathcal{B}}(v)} z_\Phi[\pi_1, k];$$

- *There exists spatial until:*

$$\begin{aligned} \Psi &= \exists_{\mathcal{B}} \Phi_1 \mathcal{U}_\kappa \Phi_2 \rightarrow \\ z_\Psi[v, k] &= \bigvee_{\pi \in \lambda^{\mathcal{B}}(v), i \in (0, \kappa]} (z_{\Phi_2}[\pi_i, k] \wedge \bigwedge_{j \in [0, i)} z_{\Phi_1}[\pi_j, k]); \end{aligned}$$

- *For all spatial until:*

$$\begin{aligned} \Psi &= \forall_{\mathcal{B}} \Phi_1 \mathcal{U}_\kappa \Phi_2 \rightarrow \\ z_\Psi[v, k] &= \bigwedge_{\pi \in \lambda^{\mathcal{B}}(v), i \in (0, \kappa]} (z_{\Phi_2}[\pi_i, k] \wedge \bigwedge_{j \in [0, i)} z_{\Phi_1}[\pi_j, k]); \end{aligned}$$

- *Temporal eventually:*

$$\Psi = F_{[k_1 \Delta t, k_2 \Delta t]} \Phi \rightarrow z_\Psi[v, k] = \bigvee_{k' = k_1, \dots, k_2} z_\Phi[v, k'];$$

- *Temporal always:*

$$\Psi = G_{[k_1 \Delta t, k_2 \Delta t]} \Phi \rightarrow z_\Psi[v, k] = \bigwedge_{k'=k_1, \dots, k_2} z_\Phi[v, k'];$$

- *Temporal until:*

$$\begin{aligned} \Psi &= \Phi_1 U_{[k_1 \Delta t, k_2 \Delta t]} \Phi_2 \rightarrow \\ z_\Psi[v, k] &= \bigvee_{k'=k_1, \dots, k_2} (z_{\Phi_2}[v, k'] \wedge \bigwedge_{k''=k_1, \dots, k'} z_{\Phi_1}[v, k'']). \end{aligned}$$

Note that $z_\Psi[v, k] \in [0, 1]$ is not required to be declared an integer since it is automatically enforced to take binary values. Finally, the problem of satisfying a general SpaTeL formula, $Q_0 \models \Phi$, reduces to the following constraint:

$$z_\Phi(v_i, 0) = 1, \quad (18)$$

where v_i is the root node of quad transition system.

C. Robustness-Based Encoding

In this section, we briefly explain how to incorporate SpaTeL robustness into the mixed-integer encoding. The method is much in spirit of the method in [23], where the authors characterize the changes in the satisfaction of the specification with respect to the changes in the predicates. For a predicate in the form of $\sigma = (\mu \sim c)$, it is straightforward to see from (4) and (5) that $\frac{\partial \rho(\Phi, Q_0)}{\partial c} \in \{0, 1\}$ (non-decreasing) or $\frac{\partial \rho(\Phi, Q_0)}{\partial c} \in \{0, -1\}$ (non-increasing), depending on the operators preceding the predicate. Therefore, by increasing (decreasing) the value of c for a non-increasing (non-decreasing) predicate, a constraint is *tightened*. Therefore, we alter the values of c in the predicates as follows:

$$\begin{cases} c \leftarrow c + \varrho & \frac{\partial \rho(\Phi, Q_0)}{\partial c} \in \{0, -1\}, \\ c \leftarrow c - \varrho & \frac{\partial \rho(\Phi, Q_0)}{\partial c} \in \{0, 1\}. \end{cases} \quad (19)$$

Next, we add the constraint $\varrho \geq 0$ to ensure satisfaction of Φ . It is easy to show that the maximum ϱ that renders $Q_0 \models \Phi$ is equal to $\rho(\Phi, Q_0)$.

D. High Level Planning

In the previous sections, we formulated the dynamics and SpaTeL objectives as mixed-integer constraints. We formulate the discrete-time version of Problem 1 as the following MILP:

$$\begin{aligned} f[k] &= \underset{k=0,1,\dots,K}{\operatorname{argmin}} && -\alpha \varrho + \mathcal{J}_f(\mathcal{N}[K]) \\ &&& + \sum_0^K \mathcal{J}_r(\mathcal{N}[k], f[k]), \\ \text{s.t.} &&& \mathcal{N}[k+1] = \mathcal{F}(\mathcal{N}[k], f[k]), \\ &&& z_\Phi(v_i, 0) = 1, \\ &&& \varrho \geq 0, \end{aligned} \quad (20)$$

where \mathcal{J}_f and \mathcal{J}_r are the discrete time versions of the end-point and running cost, respectively. Note that we assume the costs are linear functions. In this paper, we are particularly interested in the following cost:

$$J_r(\mathcal{N}[k], f[k]) = \sum f[k], \quad (21)$$

which corresponds to the total number of robot displacements (energy). Note that all the values of f are non-negative.

In case the MILP above is infeasible, no control strategy is able to satisfy the SpaTeL formula. In this case, we relax the last constraint $\varrho \geq 0$, and choose a very large value for α (or remove the other costs). Therefore, the resulting solution solely maximizes the SpaTeL robustness, which is a negative value. In other words, the SpaTeL violation is minimized.

E. Low Level Control Policy

The decision variables $f_{[i,j]}^{[i',j']}[k]$ are obtained from the solution to (20). The only remaining problem is to choose the set of individual robots that must be moved from cell $[i, j]$ to adjacent cells. For this purpose, we choose $f_{[i,j]}^{[i',j']}[k]$ number of robots that are closest to the edge between $[i, j]$ and $[i', j']$ and move them with a constant velocity on a straight line. In other words, if $\mathcal{R}_{[i,j]}[k]$ is the set of robot indices that are located inside cell $[i, j]$ at time step k and $\mathcal{R}_{[i,j]}^{[i',j']}[k]$ is the set of robot indices that are supposed to be moved from $[i, j]$ to $[i', j'] \in \Omega([i, j])$ at time step k , the control law would be:

$$u_r(t) = \frac{a}{2^D \Delta t} \cdot \begin{pmatrix} j' - j \\ i - i' \end{pmatrix}, \quad r \in \mathcal{R}_{[i,j]}^{[i',j']}[k], \quad k\Delta t \leq t < (k+1)\Delta t. \quad (22)$$

Algorithm 1 presents the procedure to determine $\mathcal{R}_{[i,j]}^{[i',j']}[k]$.

Data: $[i, j]$ (cell index), $\mathcal{R}_{[i,j]}[k]$ (robots inside that cell), $f[k]$ (flow variables)

Result: $\mathcal{R}_{[i,j]}^{[i',j']}[k]$ (set of robots that are moved from $[i, j]$ to a neighbor)

```

while  $\sum_{[i',j'] \in \Omega([i,j])} f_{[i,j]}^{[i',j']}[k] > 0$  do
    find robot  $r \in \mathcal{R}_{[i,j]}[k]$  that has the minimum
    distance to the edges of  $[i, j]$ ;
    add  $r$  to  $\mathcal{R}_{[i,j]}^{[i',j']}[k]$  where  $[i', j']$  is the neighbor of
     $[i, j]$  that  $r$  was closest to;
    remove  $r$  from  $\mathcal{R}_{[i,j]}[k]$ ;
     $f_{[i,j]}^{[i',j']}[k] \leftarrow f_{[i,j]}^{[i',j']}[k] - 1$ ;
end

```

Algorithm 1: How to assign robots to move from a cell to its neighbors

Remark 2: As mentioned earlier, we do not consider physical sizes for robots in this paper. In practice, careful strategies are required for collision avoidance among the robots. This issue will be further investigated in future work. As a preliminary solution, we propose the following approach. Let n^{cap} be the maximum number of robots that can populate one cell without physically overlapping one another. We basically add the specification that for all cells the number of robots should not exceed n^{cap} , which guarantees there is always enough empty space for swarm movements. Localized policies such as the methods in [24] can be used for guaranteeing collision avoidance.

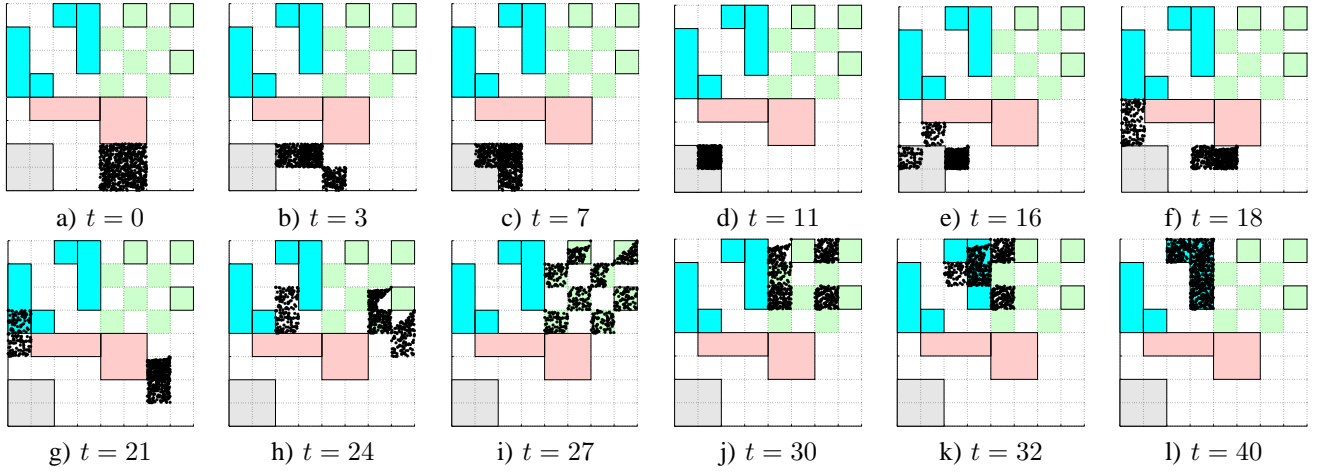


Fig. 5. Case study: Snapshots of the optimal swarm movement satisfying SpaTel formula (8) starting from the initial condition shown in figure a). First the robots are gathered in one cell in the grey region to satisfy φ_3 , then robots move toward forming the checkerboard pattern, satisfying φ_2 . Finally robots move to the populate the upper L-shaped pattern, satisfying φ_5 .

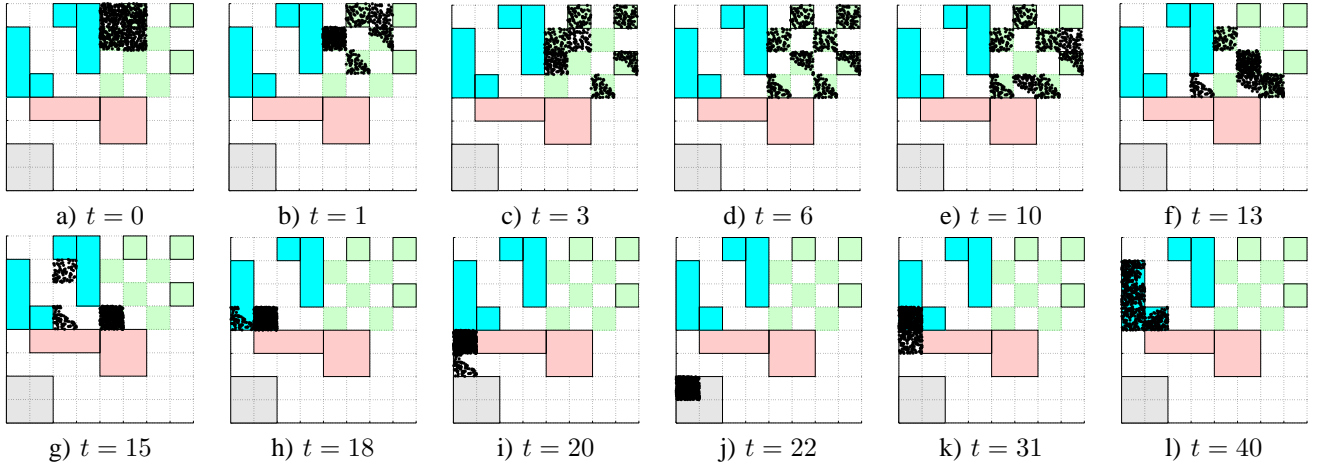


Fig. 6. Case study: Snapshots of the optimal swarm movement satisfying SpaTel formula (8) starting from the initial condition shown in figure a). First the robots are forming the checkerboard pattern φ_2 , then they gather in one grey cell to satisfy φ_3 , and finally robots move to the populate the lower L-shaped pattern, satisfying φ_6 .

F. Complexity

The worst case complexity of the framework outlined in previous sections depends on the complexity of the MILP formulation in Sec. IV-B. The complexity of a MILP problem grows exponentially, in the worst case, with respect to the number of integer variables and polynomially with respect to the continuous variables. It is worth to note that for large swarms, the flow variables can be approximated as continuous numbers and be rounded off after obtaining the MILP solution. This approximation significantly reduces the computational complexity without significantly altering the optimal solution and makes the complexity independent from the total number of robots. The number of integer variables in (20) is $KP|\mathcal{V}_f|$, where K is the total number of time steps, P is the number of linear predicates in Φ , and $|\mathcal{V}_f|$ is the number of cells in the gridded workspace. This suggests that the framework might not be scalable for grids with high resolutions or extremely complicated and

long spatio-temporal requirements. However, as illustrated in the examples in Sec. V, quite complicated patterns are achievable in practice with relatively low computation time.

V. CASE STUDY

640 robots in a workspace partitioned into a 8×8 grid. The SpaTel formula of (8) corresponding to the specification of Fig. 1 is the target, where we set $\gamma_1 = 80, \gamma_2 = 640, \gamma_{3-6} = 160$. The cost function that is minimized is the total number of robot displacements given by (21). We demonstrate the results for two different initial conditions. A movie illustrating both cases is available on <https://youtu.be/x-uI8N9iN3I>.

A. Case 1

We set the initial configuration of robots to be in the uniformly distributed in the *SW* quadrant of the *SE* quadrant (see Fig. 5 a). We formulate (20) as a MILP, which we solve

using Gurobi¹. The MILP is solved in 54 seconds on a 3GHz Dual core Macbook Pro. Next, we move the robots according to the plan obtained from the solution of the MILP. Fig. 5 shows snapshots of the swarm movement during its completion of the mission described by (8). It is seen that the swarm first satisfies φ_3 , then φ_2 and then φ_5 .

B. Case 2

Now we set the initial condition to be uniformly distributed in the *NW* quadrant of the *NE* quadrant (see Fig. 6 a). The MILP is solved in 43 seconds. The snapshots of the swarm movement are shown in Fig. 6. This time, the optimal plan is to satisfy φ_2 first, then φ_3 and finally φ_6 .

VI. CONCLUSION AND FUTURE WORK

We presented a fully automated framework to synthesize controls that steer a fully actuated robotic swarm while satisfying high-level spatio-temporal requirements. Such specifications are naturally expressed as spatial temporal logic formulae. A computationally efficient framework was introduced to determine the high level plan from which a low level control strategy executes the swarm movements.

Directions for future research include extending the framework to under actuated swarms and developing distributed control strategies for coordination of movements among robots. We also plan to incorporate machine learning methods from [13] in order to synthesize control policies for more complex spatial patterns which are automatically learned from training data. Furthermore, we plan to create a graphical user interface in which a potential user can define required patterns for execution. The user would draw the patterns that they want to emerge and specify time requirements. The interface will use machine learning techniques to generate SpaTeL formulas for those patterns. These formulas will then be used by algorithms presented in this paper to synthesize control policies for the swarm.

REFERENCES

- [1] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.
- [2] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 128–133.
- [3] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1327–1332.
- [4] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.
- [5] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. Springer, 2006, pp. 529–538.
- [6] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filters," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 254–266.
- [7] Y. Chen and Z. Wang, "Formation control: a review and a new consideration," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3181–3186.
- [8] L. C. Pimenta, N. Michael, R. C. Mesquita, G. A. Pereira, and V. Kumar, "Control of swarms based on hydrodynamic models," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1948–1953.
- [9] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 17, no. 6, p. 947, 2001.
- [10] S. G. Lee and M. Egerstedt, "Multi-robot control using time-varying density functions," *arXiv preprint arXiv:1404.0338*, 2014.
- [11] P. Yang, R. Freeman, K. M. Lynch, and Others, "Multi-agent coordination by decentralized estimation and control," *Automatic Control, IEEE Transactions on*, vol. 53, no. 11, pp. 2480–2496, 2008.
- [12] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 320–330, 2007.
- [13] E. A. Gol, E. Bartocci, and C. Belta, "A formal methods approach to pattern synthesis in reaction diffusion systems," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 108–113.
- [14] I. Haghghi, A. Jones, Z. Kong, E. Bartocci, R. Gros, and C. 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*. ACM, 2015, pp. 189–198.
- [15] A. F. Winfield, J. Sa, M.-C. Fernández-Gago, C. Dixon, and M. Fisher, "On formal specification of emergent behaviours in swarm robotic systems," *International journal of advanced robotic systems*, vol. 2, no. 4, pp. 363–370, 2005.
- [16] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 158–171, 2012.
- [17] J. Tumova and D. V. Dimarogonas, "A receding horizon approach to multi-agent planning from local LTL specifications," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 1775–1780.
- [18] A. Ulusoy, S. L. Smith, X. C. Ding, and C. Belta, "Robust multi-robot optimal path planning with temporal logic constraints," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4693–4698.
- [19] Y. Diaz-Mercado, A. Jones, C. Belta, and M. Egerstedt, "Correct-by-construction control synthesis for multi-robot mixing," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 221–226.
- [20] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.
- [21] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *Runtime Verification*. Springer, 2014, pp. 1–20.
- [22] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 81–87.
- [23] S. Sadraddini and C. Belta, "Robust Temporal Logic Model Predictive Control," *53rd Annual Conference on Communication, Control, and Computing (Allerton)*, 2015.
- [24] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

¹www.gurobi.com