Provably-Correct Coordination of Large Collections of Agents with Counting Temporal Logic Constraints[*]

Yunus Emre Sahin
University of Michigan
Department of Electrical Engineering and Computer Science
Ann Arbor, MI 48109
ysahin@umich.edu

Petter Nilsson
University of Michigan
Department of Electrical Engineering and Computer Science
Ann Arbor, MI 48109
pettni@umich.edu

Necmiye Ozay
University of Michigan
Department of Electrical Engineering and Computer Science
Ann Arbor, MI 48109
necmiye@umich.edu

ABSTRACT

In this paper, we consider the problem of coordinating a large collection of homogeneous agents subject to a novel class of constraints: counting temporal logic constraints. Counting constraints arise naturally in many multi-agent planning problems where the identity of the agents is not important for the task to be completed. We introduce a formal language to capture such tasks and present an optimization-based technique to synthesize plans for large collections of agents in a way to guarantee the satisfaction of tasks specified in this formalism.

CCS CONCEPTS

•Theory of computation →Modal and temporal logics; *Abstraction;* **•Computing methodologies →Planning for deterministic actions; Multi-agent planning; Motion path planning; •Computer systems organization →Robotic autonomy;**

KEYWORDS

Correct-by-construction synthesis

ACM Reference format:
Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. 2017. Provably-Correct Coordination of Large Collections of Agents with Counting Temporal Logic Constraints. In *Proceedings of The 8th ACM/IEEE International Conference on Cyber-Physical Systems, Pittsburgh, PA USA, April 2017 (ICCPS),* 10 pages.
DOI: http://dx.doi.org/10.1145/3055004.3055021

1 INTRODUCTION

Many cyber-physical system technologies that will contribute to the vision of smart and connected communities require coordination of large collections of systems. Scalable design methodologies to control such collections in a way to ensure safety and efficiency are therefore of great importance. This paper makes a step towards this goal by introducing a new logic, namely counting linear temporal logic, and by providing algorithms to synthesize plans for a large number of agents in a way to ensure satisfaction of specifications given in this logic.

Coordination of large numbers of agents is relevant in many applications including evacuation planning [14], formation control [25], coverage [15], and emergency response [9, 18]. Classical techniques for coordination deal with relatively simple objectives, such as reaching a goal state while avoiding collisions or converging to a consensus formation, with various degrees of complexity in the dynamics [17, 24, 29]. More recently, there has been an interest in using temporal logics to describe complex high-level specifications for robots and automatically synthesizing plans or controllers that are guaranteed to satisfy the given specifications by construction [2, 13]. Although there are attempts to apply similar ideas in multi-robot settings [6, 12, 22], the number of robots that can be coordinated is limited due to the high computational complexity of these approaches. Related work also exists in coordinating “swarms” of robots, in which the number of robots can be arbitrary. In this case, however, the swarm is roughly treated as a single agent with deformable shape [3, 11], which renders individual task assignments impossible.

We are motivated by multi-agent planning tasks that are fairly complex but that have a certain structure that allows scalability. Consider an emergency response scenario like an earthquake that requires deployment of hundreds of autonomous (ground and air) vehicles to provide supplies to victims. In such a scenario, the robotic team needs to provide supplies to certain areas, surveil different areas for survivors, and avoid certain regions of danger. The tasks may require *sufficiently many* robots to be in a given region simultaneously to provide the required support. Similarly, narrow passage ways, or potential to trigger further destruction due to overloading damaged structures may require *not too many* robots to be in certain regions at the same time. Therefore, there are temporally varying constraints on the number of robots that must be in different regions over time. We call such constraints *counting constraints* [19] and capture their temporal variation with a novel logic called *counting linear temporal logic.* The role or identity of an individual robot is not important for satisfaction of these constraints; that is, as long as the supplies reach their target, it does not matter which subset of the robots that provide them. Moreover, usually, there are not too many different types of robots; for instance, one class of UAVs and two classes of ground vehicles. That is, there is a small number of classes of robots but robots within each class have the same dynamics and motion capabilities.

[*]This work is supported in part by NSF grants CNS-1239037, CNS-1446298 and ECCS-1553873, and DARPA grant N66001-14-1-4045.

Counting constraints together with robots within a class having the same motion capabilities render the overall coordination problem permutation invariant. This structural property was first exploited in [19] for coordination of large collections of systems in the context of scheduling thermostatically controlled loads with time-invariant counting constraints on system modes. Possible extensions to multi-agent systems and to complex linear temporal logic specifications on counting constraints are also outlined in [19]. We now instantiate these extensions by introducing counting LTL and we propose an integer linear programming (ILP)-based synthesis algorithm to solve the resulting coordination problem. One desirable property of multi-agent plans is the ability to accept new agents during the execution of a plan, or tolerate that agents may become unavailable. We show that such robustness properties of the plans are easy to enforce within the proposed framework.

The rest of the paper is structured as follows. Some preliminaries are given and the problem is formally stated in Section 2. In Section 3, we show how dynamics of multiple robots can be aggregated into a constrained linear system. We present our solution approach in Section 4 and give multiple examples demonstrating the scalability of the approach are in Section 6. Finally, Section 7 concludes the paper with some remarks and directions for future work.

2 PRELIMINARIES AND PROBLEM STATEMENT

2.1 Notation and definitions

This section introduces notation and necessary background information. We follow the exposition in [1] for the basic notions related to transition systems and extend them to collections.

In the rest of the paper, $\mathbb{N}$ denotes the set of non-negative integers and $[N] = \{1, \dots, N\}$ denotes the set of positive integers up to N. The indicator function of a set A is denoted $\mathbb{1}_A(x)$ and is equal to 1 if $x \in A$ and to 0 otherwise. The vector $\mathbf{1}_n$ represents the n-dimensional vector consisting of all 1's.

We use transition systems to model the dynamics of a single agent.

Definition 2.1. A *transition system* is a tuple $T = (S, \rightarrow, AP, L)$ where S is a finite set of states, $\rightarrow \subseteq S \times S$ is a transition relation, AP is a finite set of atomic propositions and $L : S \rightarrow 2^{AP}$ is a labeling function.

We say that a state $s \in S$ satisfies an atomic proposition $a \in AP$ if $a \in L(s)$. We assume that the transition systems considered do not have any blocking states. That is, for all $s \in S$, there exists a state s' such that $(s, s') \in \rightarrow$. This is without loss of generality since the transition system can always be amended with a dummy sink state with a self-transition to obtain a non-blocking transition system that is equivalent for synthesis purposes.

Definition 2.2. A *path* of a transition system $T = (S, \rightarrow, AP, L)$ is an infinite sequence $\pi : s_0 s_1 s_2 \dots$ of states such that $(s_k, s_{k+1}) \in \rightarrow$. A *trace* of T corresponding to a path π is defined as $trace(\pi) = L(s_0)L(s_1)L(s_2) \dots \in (2^{AP})^\omega$.

We are interested in controlling the collective behavior of N homogeneous agents, dynamics of which are represented by identical transition systems T. We assume that all the transitions in a transition system are *controllable* (or, action deterministic), that is, if an agent is in state s and a transition exists from state s to s', then the controller can choose to take this transition and enforce the agent to transition to state s'. In the context of robotics it is common to directly model the motion of a robot using such an action deterministic transition system on some grid world; this approach is based on the assumption that the underlying dynamics allow steering the robot arbitrarily. Another alternative is to design low-level motion primitives (see, for instance, [16]) to enable deterministic transitions among different regions of the workspace. For more complex and general dynamics, abstraction methods proposed in [20, 21, 27] can be used, where each node in the transition system corresponds to a subset of the original state-space.

Definition 2.3. Let $\Pi = \{\pi^n\}_{n \in [N]}$ be the collection of paths followed simultaneously by N agents, where each agent n has the dynamics $T^n = (S, \rightarrow, AP, L)$ and path $\pi^n : s_0^n s_1^n s_2^n \dots$. The *collective trace* corresponding to Π is a sequence $ctrace(\Pi) = \sigma = \sigma_0 \sigma_1 \sigma_2 \dots$ of functions $\sigma_k : AP \rightarrow [N]$ such that for all $a \in AP$, $\sigma_k(a) = \sum_{n \in [N]} \mathbb{1}_{L(s_k^n)}(a)$.

In words, a collective trace is a sequence of functions σ_k that, at step k, maps each atomic proposition to the number of agents that are in a state satisfying that atomic proposition.

2.2 Counting LTL

We introduce a temporal logic that is useful to specify and reason about the collective behavior of multiple agents. We call this logic *counting linear temporal logic*, or, for short, counting LTL.

The syntax of a counting LTL formula over a set of atomic propositions AP is given by the following grammar:

$$\varphi ::= True \mid cp \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \; \mathcal{U} \; \varphi_2, \tag{1}$$

where $cp \in AP \times \mathbb{N}$ is a *counting proposition* and φ, φ_1 and φ_2 are counting LTL formulas. The symbols $\neg$, $\wedge$, $\bigcirc$ $\mathcal{U}$ are, respectively, the logical operators negation and conjunction, and the temporal operators next and until. These operators can be used to define additional operators such as disjunction ($\varphi_1 \vee \varphi_2 \doteq \neg(\neg\varphi_1 \wedge \neg\varphi_2)$), false ($False \doteq \neg True$), eventually ($\Diamond\varphi \doteq True \; \mathcal{U} \; \varphi$), and always ($\Box\varphi \doteq \neg\Diamond\neg\varphi$).

Next, we present the semantics of counting LTL. Given a collective trace σ, satisfaction of a counting LTL formula φ by σ at step k, denoted as $\sigma, k \models \varphi$, is inductively defined as follows:

- $\sigma, k \models True$,
- for any counting proposition $cp = [a, m] \in AP \times \mathbb{N}$, $\sigma, k \models [a, m]$ if and only if $\sigma_k(a) \geq m$,
- $\sigma, k \models \varphi_1 \wedge \varphi_2$ if and only if $\sigma, k \models \varphi_1$ and $\sigma, k \models \varphi_2$,
- $\sigma, k \models \neg\varphi$ if and only if $\sigma, k \not\models \varphi$,
- $\sigma, k \models \bigcirc\varphi$ if and only if $\sigma, k+1 \models \varphi$, and
- $\sigma, k \models \varphi_1 \; \mathcal{U} \; \varphi_2$ if and only if there exists $l \geq 0$ such that $\sigma, k+l \models \varphi_2$ and $\sigma, k+l' \models \varphi_1$ for all $0 \leq l' < l$.

We say that a collective trace σ satisfies a counting LTL formula φ, and write $\sigma \models \varphi$, if $\sigma, 0 \models \varphi$.

To exemplify, assume that a collective trace σ is generated by N agents. Then σ satisfies the property $\Diamond[a, 3]$ if there exists a time $k \geq 0$ when the states of *at least* three agents satisfy a. Similarly,

σ satisfies $\Box\neg[b,4]$ if at all times $k \geq 0$, the state of *at most* three agents satisfy b. We also point out the tautologies $[c,0] = \mathit{True}$, $[c,m] = \mathit{False}$ for $m > N$, and that the negation of a counting proposition is $\neg[c,k] = [\neg c, N-k+1]$.

Some remarks on the relation of counting LTL to other logics are in order. It is possible to consider a metric version of counting LTL, i.e. similar to metric temporal logic (MTL). Since we only deal with systems that evolve in discrete-time steps, considering MTL would not add to the expressivity of the logic. If we consider counting propositions as the set of new atomic propositions, counting LTL is essentially equivalent to standard LTL, and can be interpreted over the product transition system of all agents. However, such a product construction leads to an explosion both in the number of states and in the number of atomic propositions, and is therefore not scalable. The key to achieve efficiency in our algorithms is to avoid this construction by working with collective traces and counting propositions. Two logics related to counting LTL, a spatiotemporal logic for swarms [8] and censusSTL [28], have been proposed concurrently at the time this paper was being written. The former [8] specializes to planar robotic swarms and regular grid worlds, and is less expressive than counting LTL. Meanwhile, the latter [28] generalizes counting LTL by allowing "temporal counting propositions" but the paper only focuses on inference of a logic formula from data and does not consider control synthesis for censusSTL.

2.3 Problem statement

Now, we are ready to formally state the multi-agent coordination problem with counting LTL constraints.

Problem 1. *Given a counting LTL formula φ over a set of atomic propositions AP, and a collection of N agents with identical dynamics $T^n = T \doteq (S, \rightarrow, AP, L)$ and (non-identical) initial conditions s_0^n for $n \in [N]$, synthesize, for each agent n, a path π^n starting at s_0^n such that the collective trace corresponding to $\Pi = \{\pi^n\}_{n\in[N]}$ satisfies φ. That is, $ctrace(\Pi) \models \varphi$. We will refer to an instance of this problem as $\left(\varphi, T, \{s_0^n\}_{n\in[N]}\right)$.*

3 REFORMULATION

In this section we present a problem equivalent to Problem 1 that is—as we show in Section 4—more amenable to optimization-based solutions. The alternative formulation is obtained by exploiting the homogeneity of the transition system graphs.

A transition system $T = (S, \rightarrow, AP, L)$ can be seen as a directed graph with nodes S and edges $\rightarrow$. Since all the agents we consider in this work are identical, we aim to represent the paths of all agents simultaneously. To this end, we assign weights to each node and each edge. Let the joint state space of the agents be $S = \{v_1, v_2, \ldots, v_I\}$. Then the weight assigned to $v_i \in S$, denoted by $w_i \in \mathbb{N}$, represents the number of agents that are in state v_i. The weight assigned to an edge $(v_i, v_j) \in\rightarrow$, denoted by $w_{i,j} \in \mathbb{N}$, represents the number of agents that will move from v_i to v_j in the next time step. The dynamics governing the node weights then take the following form:

$$(w_i)^+ = \sum_{j\in[I]} w_{j,i}, \quad \forall i \in [I], \tag{2}$$

subject to the input being in the (state-dependent) constraint set

$$\Upsilon = \left\{ \{w_{i,j}\}_{i,j\in[N]} : \begin{array}{c} w_{i,j} = 0 \text{ if } (v_i, v_j) \notin\rightarrow \\ w_{i,j} \in \mathbb{N} \\ \sum_{j\in[I]} w_{i,j} = w_i \end{array} \right\}. \tag{3}$$

The first constraint assures that no transitions take place between nodes that are not connected by $\rightarrow$, while the second constraint guarantees that $(w_i)^+$ is a non-negative integer for all $i \in [I]$. The third constraint preserves the total number of agents: $\sum_{i\in I}(w_i)^+ = \sum_{i\in I} w_i$. Eq. (2) is a linear system with state-dependent input constraints. By introducing states $\mathbf{w} \doteq [w_1, \ldots, w_I]^\intercal$ and inputs $\mathbf{u} \doteq [w_{1,1}, \ldots, w_{I,1}, w_{1,2}, \ldots, w_{I,2}, \ldots w_{1,I}, \ldots, w_{I,I}]^\intercal$ it can be written as

$$\Sigma : \begin{cases} \mathbf{w}^+ = B\mathbf{u}, \\ \mathbf{u} \in \Upsilon(\mathbf{w}), \end{cases} \tag{4}$$

where the state dependence in the input set has been made clear. The system matrix B is equal to $B = \mathbf{I}_I \otimes \mathbf{1}_I^\intercal$ where $\mathbf{I}_I$ is the $I \times I$ identity matrix and $\otimes$ is the Kronecker product. In the following we will refer to the system Σ defined in (4) as the *aggregate dynamics*.

A trajectory of the aggregate dynamics is an infinite sequence $\Omega = \mathbf{w}_0\mathbf{w}_1\mathbf{w}_2\ldots$. We will use the atomic propositions of the transition system T to assign collective traces to the trajectories of the aggregate dynamics. A collective trace corresponding to a trajectory Ω with the invariant $\sum_{i\in[I]}(\mathbf{w}_k)_i = N$ is a sequence $ctrace(\Omega) = \mu = \mu_0\mu_1\mu_2\ldots$ of functions $\mu_k : AP \rightarrow \{0, \ldots, N\}$ such that for all $a \in AP$, $\mu_k(a) = \sum_{i\in[I]}(\mathbf{w}_k)_i \mathbb{1}_{L(v_i)}(a)$.

Problem 2. *Given the aggregate dynamics Σ, an initial condition $\mathbf{w}_0 \in \mathbb{N}^I$, and a counting LTL formula φ over AP, synthesize an infinite sequence of control inputs $\mathbf{u} = \mathbf{u}_0\mathbf{u}_1\mathbf{u}_2\ldots$ within the allowable input set* (3) *such that the collective trace corresponding to the resulting trajectory of Σ satisfies φ. We will refer to an instance of this problem as $(\varphi, \Sigma, \mathbf{w}_0)$.*

We now relate Problem 1 to Problem 2. By construction, the following holds.

Proposition 1. *For any instance $P = \left(\varphi, T, \{s_0^n\}_{n\in[N]}\right)$ of Problem 1, there exists an instance $P' = (\varphi, \Sigma, \mathbf{w}_0)$ of Problem 2 such that P has a solution if and only if P' has a solution.*

Proof. We first construct the instance P' from P. The aggregate dynamics Σ are constructed from the transition system T as presented above and the initial condition $\mathbf{w}_0$ of P' is given in terms of the initial condition $\{s_0^n\}_{n\in[N]}$ as $(\mathbf{w}_0)_i = \sum_{n\in[N]} \mathbb{1}_{\{v_i\}}\left(s_0^n\right)$.

We show that a solution of P can be mapped to a solution of P'. Let the paths $\pi^n = s_0^n s_1^n s_2^n \ldots$ for $n \in [N]$ be such that $\sigma \doteq ctrace\left(\{\pi^n\}_{n\in[N]}\right) \models \varphi$. They can be mapped to aggregate states $\mathbf{w}_k$ and aggregate controls $\mathbf{u}_k$ for $k \geq 0$ by

$$(\mathbf{w}_k)_i \doteq \sum_{n\in[N]} \mathbb{1}_{\{v_i\}}\left(s_k^n\right), \tag{5a}$$

$$(\mathbf{u}_k)_{i,j} \doteq \sum_{n,m\in[N]} \mathbb{1}_{\{v_i\}}\left(s_k^n\right) \mathbb{1}_{\{v_j\}}\left(s_{k+1}^n\right). \tag{5b}$$

It is clear that $\mathbf{u}_k \in \Upsilon(\mathbf{w}_k)$ and that $\mathbf{w}_{k+1} = B\mathbf{u}_k$, so $\Omega = \mathbf{w}_0\mathbf{w}_1\mathbf{w}_2\ldots$ is a valid trajectory of Σ. Furthermore, the corresponding collective trace $\mu \doteq ctrace(\Omega)$ will satisfy φ since the truth evaluation is identical to that of σ for any $a \in AP$:

$$\begin{aligned}\mu_k(a) &= \sum_{i\in[I]} (\mathbf{w}_k)_i \, \mathbb{1}_{L(s_i)}(a) \\ &= \sum_{i\in[I]} \sum_{n\in[N]} \mathbb{1}_{\{v_i\}}\left(s_k^n\right) \mathbb{1}_{L(v_i)}(a) \\ &= \sum_{n\in[N]} \mathbb{1}_{L(s_k^n)}(a) = \sigma_k(a).\end{aligned}$$

Therefore $\sigma \models \varphi$ implies that $\mu \models \varphi$, so P' has a solution if P has a solution.

Next, given a solution $\mathbf{u}_0\mathbf{u}_1\mathbf{u}_2\dots$ of P', we construct traces $\pi^n = s_0^n s_1^n s_2^n \dots$ of P such that the collective trace satisfies φ. We do so iteratively by assuring that (5a) holds. The invariant (5a) evidently holds at $k = 0$ by construction of P'. For induction, assume that it holds at time k. The control $(\mathbf{u}_k)_{i,j}$ dictates for each node pair (v_i, v_j) how many traces π^n for which $s_k^n = v_i$ that should transition such that $s_{k+1}^n = v_j$. Since $\mathbf{u}_k \in \Upsilon(\mathbf{w}_k)$, it follows that it is possible to assign a destination to each agent such that (5a) holds also at time $k+1$. As above, the truth evaluations of the collective traces are identical due to (5a). Thus P has a solution if P' has a solution. □

Remark 1. *The construction in the proof above incrementally yields traces π^n and requires central coordination. From an implementation standpoint it may be desirable to distribute complete traces π^n at deployment to not require central communication during the mission. This can be achieved by executing the system Σ off-line to extract individual traces as described in the proof. As discussed in Section 4; due to finiteness of the state space of $\mathbf{w}$ any solution trajectory $\Omega = \mathbf{w}_0\mathbf{w}_1\mathbf{w}_2\dots$ must be on a finite prefix-suffix form. In particular, our solution approach searches the solution space for such solutions that are also relatively short. When Ω is on a prefix-suffix form also the generated individual traces π^n will necessarily be on a finite prefix-suffix form, although the suffix may be longer than that of Ω. Therefore infinite individual traces π^n can be generated off-line which enables deployment without need of central coordination.*

In Section 4 we propose an optimization-based approach to solve Problem 2, from which a solution to Problem 1 can be extracted through the construction in the proof of Proposition 1. Note that the size of the linear system (4) is independent of the number of agents N, which makes the approach applicable to very large collections of agents.

3.1 Multiple classes of agents

Before moving to the solution of Problem 2, we extend the theory to the situation when there are multiple heterogeneous classes of agents that are homogeneous within each class. Each class of agents is supposed to provide different services, so we want the ability to pose class-specific counting constraints. For instance, we might require that a certain region in a robotic application should have at least one first aid robot, and between 2 and 5 cleaning robots that clear up debris. We first state a class-augmented generalization of Problem 1 and then show that also this problem can be reformulated to an instance of Problem 2.

Consider C different classes of agents, with N_c agents of type c for $c \in [C]$. The dynamics of agents in class c are described by a transition system $T^c = (S, \to^c, AP, L)$, i.e., the difference between classes lies in the transitions $\to^c$.

Problem 3. *Given a counting LTL formula φ over an augmented set of atomic propositions $AP' \doteq AP \times [C]$; and C agent classes, each containing N_c agents with identical dynamics $T^c \doteq (S, \to^c, AP, L)$ and (non-identical) initial conditions $s_0^{n_c}$ for $n_c \in [N_c]$, synthesize, for each agent n_c, a path π^{n_c} starting at $s_0^{n_c}$ such that the collective trace corresponding to $\Pi = \{\pi^{n_c}\}_{n\in[N_c], c\in[C]}$ satisfies φ, i.e., $ctrace(\Pi) \models \varphi$.*

Using the procedure above, we can construct C different aggregate systems Σ^c that describe the aggregate behavior of individual classes. By concatenating the states as $\mathbf{w}^{[C]} = [(\mathbf{w}^1)^T, \dots, (\mathbf{w}^C)^T]^T$ and $\mathbf{u}^{[C]} = [(\mathbf{u}^1)^T, \dots, (\mathbf{u}^C)^T]^T$, we can write

$$\Sigma^{[C]} : \begin{cases} (\mathbf{w}^{[C]})^+ = B^{[C]}\mathbf{u}^{[C]}, \\ \mathbf{u}^{[C]} \in \Upsilon^{[C]}(\mathbf{w}^{[C]}), \end{cases}$$

for $B^{[C]}$ being the block-diagonal matrix formed by the B^c's, and $U^{[C]}(\mathbf{w}^{[C]}) = \prod_{c\in[C]} \Upsilon^c(\mathbf{w}^c)$. We also introduce an augmented collective trace that allows us to assign class-specific tasks. For a trajectory $\mathbf{w}_0^{[C]}\mathbf{w}_1^{[C]}\mathbf{w}_2^{[C]}\dots$ of $\Sigma^{[C]}$ we define the collective trace $ctrace(\Sigma) = \mu = \mu_0\mu_1\mu_2\dots$ where the functions $\mu_k : AP \times [C] \to \{0, \dots, \max_{c\in[C]} N_c\}$ are such that for all $a \in AP$ and $c \in [C]$,

$$\mu_k(a, c) = \sum_{i\in[I]} (\mathbf{w}_k^c)_i \mathbb{1}_{L(v_i)}(a).$$

The result of Proposition 1 now carries over to this augmented construction: we can construct an instance of Problem 2 that has a solution if and only if Problem 3 has a solution. The additional complexity introduced by classes does not affect the conceptual difficulty of solving the instance of Problem 2, and a solution can be transformed into a solution of Problem 3.

4 SOLUTION APPROACH

Given an instance of Problem 2, we encode the aggregate dynamics and the counting LTL specifications as integer-linear-programming (ILP) constraints and solve the corresponding feasibility or optimization problem to obtain a solution. This approach is inspired by bounded model-checking for LTL [4, 5]. Similar ideas have recently been used for control synthesis for continuous-state and hybrid systems, where mixed-integer-linear programming constraints are used [10, 23, 26]. Here, we show that counting propositions also lead to linear constraints. In what follows, M is a sufficiently large positive number, in particular, $M > N + 1$.

4.1 Dynamic constraints

Loop constraints: As is common in LTL model-checking and logic synthesis, we restrict the solutions we seek to those on prefix-suffix form. That is, for a given positive integer h, we search for trajectories of the form $\Omega = \mathbf{w}_0\mathbf{w}_1\dots\mathbf{w}_h\dots$ and an integer $l \in \{0, \dots, h-1\}$ such that for all $k \geq h$, $\mathbf{w}_k = \mathbf{w}_{k+l-h}$. This restriction allows us to formulate optimization problems with finitely many variables. The resulting collective trace has the form $ctrace(\Omega) = \mu_0\mu_1\dots\mu_{l-1}(\mu_l\dots\mu_{h-1})^\omega$.

These "loop constraints" are captured by introducing h integer-valued vector variables $\mathbf{w}_1, \dots, \mathbf{w}_h$ for states and h binary variables

$z_0^{loop}, \dots z_{h-1}^{loop} \in \{0,1\}$ such that

$$\sum_{k=0}^{h-1} z_k^{loop} = 1, \tag{6a}$$

$$\mathbf{w}_h \leq \mathbf{w}_k + M(1 - z_k^{loop}), \qquad k = 0, \dots, h-1, \tag{6b}$$

$$\mathbf{w}_h \geq \mathbf{w}_k - M(1 - z_k^{loop}), \qquad k = 0, \dots, h-1. \tag{6c}$$

Note that (6a) requires exactly one of the loop variables to be 1. Assume $z_l^{loop} = 1$. Then $\mathbf{w}_h = \mathbf{w}_l$ and (6b), (6c) are trivially satisfied for any $k \neq l$.

Aggregate dynamics: The evolution of the states are constrained by the aggregate dynamics (2). That is, we have:

$$\mathbf{w}_{k+1} = B\mathbf{u}_k, \qquad k = 0, \dots, h-1, \tag{7}$$

where integer-valued vector variables $\mathbf{u}_k$ are constrained such that

$$\mathbf{u}_k \in \Upsilon(\mathbf{w}_k), \qquad k = 0, \dots, h-1, \tag{8}$$

which are linear constraints in variables $\mathbf{u}_k$ and $\mathbf{w}_k$.

4.2 Counting LTL constraints

Given a counting LTL formula generated according to the grammar in (1), we recursively parse this formula in order to generate the corresponding ILP constraints. Given any formula φ in the parse tree, we introduce h binary variables $z_k^{\varphi} \in \{0,1\}$ for $k = 0, \dots, h-1$ such that $z_k^{\varphi} = 1$ if and only if $\sigma, k \models \varphi$, where σ is the collective trace of a prefix-suffix trajectory. Using these newly introduced binary variables, a counting LTL formula can be encoded as a set of ILP constraints. We denote the set of all ILP constraints corresponding to a counting LTL formula φ by $ILP_h(\varphi)$; and all the binary variables within $ILP_h(\varphi)$ as $\mathbf{z}^{LTL,h}$.

Since counting propositions are the main difference between counting LTL and standard LTL, we propose an encoding for the counting propositions. The rest of the encodings can be found in bounded model checking literature but we state them here for completeness.

cp (counting proposition): Let $\phi = cp = [a, m] \in AP \times \mathbb{N}$. We define $\mathbf{v}_a = [\mathbb{1}_{L(v_1)}(a), \dots, \mathbb{1}_{L(v_I)}(a)]^T$ and introduce optimization variables $z_k^{cp} \in \{0,1\}$ for $k = 0, \dots, h-1$ such that

$$\begin{aligned} \mathbf{v}_a^T \mathbf{w}_k &\geq \min(m, N+1) - M(1 - z_k^{cp}), \\ \mathbf{v}_a^T \mathbf{w}_k &\leq \min(m, N+1) + M z_k^{cp}. \end{aligned} \tag{9}$$

By construction, *cp* holds (fails to hold) at time k when $z_k^{cp} = 1$ ($z_k^{cp} = 0$). In other words, $\sigma_k(a) = \mathbf{v}_a^T \mathbf{w}_k \geq m$ if and only if $z_k^{cp} = 1$, noting that $\sigma_k(a)$ is at most N.

$\neg$ *(negation):* Let $\phi = \neg\varphi$. Then

$$z_k^{\phi} = 1 - z_k^{\varphi}, \qquad k = 0, \dots, h-1. \tag{10}$$

$\wedge$ *(conjunction):* Let $\phi = \bigwedge_{i=1}^n \varphi_i$. Then for all $k = 0, \dots, h-1$

$$\begin{aligned} z_k^{\phi} &\leq z_k^{\varphi_i}, \qquad i = 1, \dots, n, \\ z_k^{\phi} &\geq 1 - n + \sum_{i=1}^n z_k^{\varphi_i}. \end{aligned} \tag{11}$$

$\vee$ *(disjunction):* Let $\phi = \bigvee_{i=1}^n \varphi_i$. Then for all $k = 0, \dots, h-1$

$$\begin{aligned} z_k^{\phi} &\geq z_k^{\varphi_i}, \qquad i = 1, \dots, n, \\ z_k^{\phi} &\leq \sum_{i=1}^n z_k^{\varphi_i}. \end{aligned} \tag{12}$$

With slight abuse of notation, we also use Boolean operators on these optimization variables. For example for $\phi = \bigvee_{i=1}^n \varphi_i$, we say $z^{\phi} = \bigvee_{i=1}^n z^{\varphi_i}$ instead of stating inequalities as in (12). Finally, we encode the temporal operators as follows:

$\bigcirc$ *(next):* Let $\phi = \bigcirc\varphi$, then

$$\begin{aligned} z_k^{\phi} &= z_{k+1}^{\varphi}, \qquad k = 0, \dots, h-2, \\ z_{h-1}^{\phi} &= \bigvee_{k=0}^{h-1} (z_k^{\varphi} \wedge z_k^{loop}). \end{aligned} \tag{13}$$

$\mathcal{U}$ *(until):* if $\phi = \varphi_1 \,\mathcal{U}\, \varphi_2$, then

$$z_k^{\phi} = z_k^{\varphi_2} \vee \left(z_k^{\varphi_1} \wedge z_{k+1}^{\phi}\right), \qquad k = 0, \dots, h-2, \tag{14}$$

$$z_{h-1}^{\phi} = z_{h-1}^{\varphi_2} \vee \left(z_{h-1}^{\varphi_1} \wedge \bigvee_{i=0}^{h-1} \left(z_i^{loop} \wedge \tilde{z}_i^{\phi}\right)\right), \tag{15}$$

$$\tilde{z}_k^{\phi} = z_k^{\varphi_2} \vee \left(z_k^{\varphi_1} \wedge \tilde{z}_{k+1}^{\phi}\right), \qquad k = 0, \dots, h-2, \tag{16}$$

$$\tilde{z}_{h-1}^{\phi} = z_{h-1}^{\varphi_2}, \tag{17}$$

where $\tilde{z}_k^{\phi}$ are auxiliary binary variables. As shown in [4], not introducing auxiliary variables results in trivial satisfaction of the until formula ϕ.

While any counting LTL formula can be represented as ILP constraints using the encodings above, more efficient implementations with less overhead are possible for specialized cases. For example, the temporal operators $\Diamond$ (eventually) and $\Box$ (always) are commonly used in specifications. Likewise, the combination of these two operators, $\Box\Diamond$ and $\Diamond\Box$, are frequently used, and they specify *infinitely often* and *reach-stay* types of tasks, respectively. In our implementation, we provide more succinct encodings for these specialized cases and call them *templates*. Template formulas can be combined using Boolean operations, but if they are nested with other temporal operators that may lead to incorrect or incomplete solutions. The template encodings are as follows:

($\Box$) *(always template):* Let $\phi = \Box\varphi$, then

$$z_0^{\phi} = \bigwedge_{k=0}^{h-1} z_k^{\varphi}. \tag{18}$$

($\Diamond$) *(eventually template):* Let $\phi = \Diamond\varphi$, then

$$z_0^{\phi} = \bigvee_{k=0}^{h-1} z_k^{\varphi}. \tag{19}$$

($\Box\Diamond$) *(always-eventually template):* Let $\phi = \Box\Diamond\varphi$, then

$$z_0^{\phi} = \bigvee_{k=0}^{h-1} \left(z_k^{\varphi} \wedge \left(\bigvee_{i=1}^{k} z_i^{loop}\right)\right). \tag{20}$$

($\Diamond\Box$) *(eventually-always template):* Let $\phi = \Diamond\Box\varphi$, then

$$z_0^{\phi} = \bigwedge_{k=0}^{h-1}\left(z_k^{\varphi} \vee \left(\bigwedge_{i=1}^{k} \neg z_i^{loop}\right)\right). \tag{21}$$

In the examples section we compare the computation times for solutions with the template encodings and the generic encodings.

4.3 Overall ILP problem and analysis

Given an instance of Problem 2 and a trajectory length h, the following feasibility problem can be formed:

$$\begin{aligned} &\text{Find } \mathbf{u}_{0:h-1},\ \mathbf{w}_{1:h},\ z_{0:h-1}^{loop},\ \mathbf{z}^{LTL,h} \\ &\text{s.t. (6), (7), (8), } ILP_h(\varphi),\ z_0^{\varphi} = 1. \end{aligned} \tag{OPT}$$

Note that by construction all the optimization variables are integers and all the constraints in (OPT) are linear in the optimization variables, therefore it is an integer linear programming problem. Next we provide some analysis of the solution approach, starting with a soundness result.

Proposition 2. *Given a trajectory length h, if the ILP problem* (OPT) *is feasible, then Problem 2 has a solution.*

Proof. Constraints (6), (7), (8) guarantee that a trajectory $\Omega^* = \mathbf{w}_0^*\mathbf{w}_1^*\ldots\mathbf{w}_{l-1}^*(\mathbf{w}_l^*\ldots\mathbf{w}_{h-1}^*)^{\omega}$ generated from a feasible solution of (OPT) is consistent with the system dynamics and initial condition, while the input sequence $\mathbf{U}^* = \mathbf{u}_0^*\mathbf{u}_1^*\ldots\mathbf{u}_{l-1}^*(\mathbf{u}_l^*\ldots\mathbf{u}_{h-1}^*)^{\omega}$ generated from a feasible solution of (OPT) is in the allowable input set. By the soundness of bounded model checking [4] and the correctness of the encoding for counting constraints, $ILP_h(\varphi)$ together with $z_1^{\varphi} = 1$ guarantees that $ctrace(\Omega^*) \models \varphi$. Therefore, $\mathbf{U}^*$ solves Problem 2. □

The following result shows that there is no loss of generality in using prefix-suffix trajectories, therefore the approach is complete if computational resources allow picking a large enough trajectory length h.

Proposition 3. *If Problem 2 has a solution, then there exists a finite trajectory length $\tilde{h}$ such that* (OPT) *is feasible.*

Proof. (Sketch) The proof proceeds by reducing Problem 2 into a standard LTL control synthesis problem, where the goal is to find a path of a finite transition system that satisfies a given LTL formula or to prove no such path exists. For an action deterministic finite state system T', the control synthesis problem with LTL specification ϕ is equivalent to checking whether T' satisfies $\neg\phi$. If T' satisfies $\neg\varphi$, then no satisfying path exists. If T' violates $\neg\varphi$, then there always exists a counter-example in the form of a path in prefix-suffix form [1]. Since this path satisfies φ, it is a solution to the control synthesis problem.

The reduction proceeds as follows: consider a transition system $T' = (S', \rightarrow', AP', L')$, each state $v' \in S'$ of which is one possible valuation of $\mathbf{w}$. Since $\mathbf{w}$ is an integer-valued vector with $\sum_{i\in[I]}(\mathbf{w})_i = N$, $|S'|$ is finite. The transition relation $\rightarrow'$ of T' is defined based on (4). The set AP' contains an atomic propositions cp' for each counting proposition $cp = [a, m]$ for $0 \le m \le N+1$ (since any $m > N+1$ has the same satisfaction properties with $m = N+1$), and is therefore finite. Similarly, the LTL formula φ' over AP' is defined by replacing each cp that appears in the counting LTL formula φ with the corresponding cp'. Finally the labeling function L' is defined such that $cp' \in L'(v')$ if the valuation of $\mathbf{w}$ corresponding to v' satisfies the counting proposition cp corresponding to cp'. Problem 2 is then equivalent to existence of a path in T' with initial condition $v_0' \in S'$ that corresponds to $\mathbf{w}_0$, that satisfies φ'. This problem admits a prefix-suffix solution. □

Remark 2. *We point out that in the proof, $|S'|$ depends on N, so the $\tilde{h}$ that is required for completeness also depends on N.*

Note that the number of ILP constraints and the number of variables in (OPT) do not depend on the number of agents N. Hence the proposed approach easily scales to a large number of agents as demonstrated in Section 6. Another interesting property of Problem 2 is its invariance to integer scaling as captured in the following proposition.

Proposition 4. *An instance $(\varphi, \Sigma, \mathbf{w}_0)$ of Problem 2 with N agents (i.e., $\sum_{i\in[I]}(\mathbf{w}_0)_i = N$) has a solution if and only if, for any positive integer p, the instance $(\varphi', \Sigma, p\mathbf{w}_0)$ with pN agents and a counting LTL specification φ' that is obtained by multiplying the second terms of all counting propositions in φ by p, has a solution.*

Proof. This amounts to scaling both sides of the linear constraints in (6), (7), (8), (9), which contain state and input variables, by p. The feasibility of (OPT) is not affected by this scaling. Then, by Proposition 3, it does not affect the existence of solutions of Problem 2 either. □

As a result of Proposition 4, the size of a problem instance can be reduced if N, the integer terms in the counting constraints, and the elements of $\mathbf{w}_0$ share a common divisor d (i.e., are not co-prime). In this case it is equivalent to solve a scaled problem with N/d agents and scale the solution by d.

5 OTHER CONSIDERATIONS

In this section we discuss some considerations that are relevant in the context of multi-robot systems and show that they can be easily addressed within the proposed framework.

5.1 Static obstacle avoidance

If some part $\mathcal{P}$ of the workspace is blocked by obstacles or considered unsafe, the number of agents in those states should be exactly zero, at all times. Assuming robot states include workspace location information, and introducing $a \in AP$ to be such that $a \in L(s)$ if and only if the state s is in $\mathcal{P}$, static obstacle avoidance specification is captured by the counting LTL formula $\varphi_{obs} = \Box(\neg[a, 1])$.

5.2 Collision avoidance

Assume that the distance between any two agents needs to be at least ε at all times to avoid collisions. Indeed, counting propositions are ideally suited to achieve this task. Let $I(s_i)$ denote the set of states that are ε-close to s_i in the spatial domain. Now introduce a new atomic proposition a_{s_i} for each state such that $a_{s_i} \in L(s)$ if and only if $s \in I(s_i)$. The conjunction $\varphi_{ca} = \Box\left(\bigwedge_{i=1}^{I} \neg[a_{s_i}, 2]\right)$ then enforces the desired minimal distance between any two agents. The counting proposition $cp_i = \neg[a_{s_i}, 2]$ limits the maximum number of agents present in a window $I(s_i)$ around s_i to one, hence enforcing cp_i for each state at all times ensures a minimum separation between any two agents.

5.3 Robustness to entering and exiting agents

Counting propositions constrain the number of agents in a certain state by inequalities as denoted in (9). In practical situations the number of available agents may be uncertain. To account for agents that are momentarily unavailable for task completion, or uncontrolled agents that move across the work space, we want to satisfy those inequalities with an ϵ bound in order to be robust against such deviations. That is, specifications should be satisfied even if ϵ number of agents fail unexpectedly, or if ϵ new agents are for some reason introduced into the workspace.

Definition 5.1. A collective trace σ is called *ϵ-robust* with respect to a specification φ, if $\sigma \models \varphi'$ for all φ' obtained from φ by modifying the counting propositions of φ as follows:

$$[a, m] \mapsto [a, m + \delta], \quad \text{for some } \delta \in [-\epsilon, \epsilon]. \tag{23}$$

This type of robustness can easily be incorporated into the optimization formulation by modifying (9) as follows:

$$\begin{aligned} \mathbf{v}_a^T \mathbf{w}_k &\geq \min(m, N+1) + \epsilon - M(1 - z_k^{cp}) \\ \mathbf{v}_a^T \mathbf{w}_k &\leq \min(m, N+1) - \epsilon + M z_k^{cp}. \end{aligned} \tag{24}$$

In order to find the *most robust* solution possible, the ILP (OPT) with (24) can be posed as an optimization problem with ϵ as a variable that is to be maximized.

6 EXAMPLES

In this section, we demonstrate the proposed approach with some examples. All the examples are executed on a laptop with 2.50GHz Intel Core i-7 processor and 16GB of RAM using GUROBI [7] as the underlying ILP solver. Our prototype implementation is available at https://github.com/sahiny/cLTL-synth.

6.1 Numerical examples

We first demonstrate the scalability of the approach by varying several parameters, such as the number of agents N, the transition system size I, the trajectory length h and the complexity of the specification. In each case, the transitions $\rightarrow$ are generated from an Erdős-Rényi graph with edge probability 0.25.

We start by investigating the effect of transition system size I on runtime. The number of agents N and the trajectory length h are set as $N = 20$ and $h = 20$. Half of the states are selected randomly and are labeled with $S_1 \in AP$ and the rest of the states are labeled with $S_2 \in AP$. Three goal regions G_1, G_2 and G_3 are created such that each of these regions have $\frac{I}{10}$ states that are randomly selected. We randomly initialize all of our agents in S_1. Finally, the specification is set to

$$\varphi \doteq (\Diamond\Box[S_2, N/2]) \wedge \Box\Diamond \bigwedge_{i=1}^{3} ([G_i, N/5]).$$

In words, we require that there should be a point in time such that S_2 (which has no agents initially) is populated by at least half of the agents, and that from that point on, the number of agents present in S_2 should always be greater than $\frac{N}{2}$. In addition, the three goal states should be visited infinitely often, without any specific order. The average computation time and average number of variables and constraints created over ten trials for an increasing transition system size is displayed in Table 1. The same experiments are also run with the template implementation and the corresponding statistics are shown in parentheses.

Table 1: Average computation time and statistics regarding optimization problem size for different number of states.

#State	100	200	500
#variables	51k (50k)	199k (198k)	1250k (1250k)
#constraints	72k (56k)	226k (212k)	1395k (1280k)
Creating cons (sec)	19.2 (2.3)	28.5 (4.8)	116.2 (37.1)
Solving ILP (sec)	16.1 (11.4)	75.9 (85.2)	625.3 (285.1)

Secondly, we show that the proposed approach scales well with respect to the number of agents. We fix the number of states to $I = 100$ and leave the remaining problem parameters intact. The average computation time over ten trials can be seen from Table 2 for different numbers of agents. As before, the numbers in parentheses are for the template implementation. As can be seen, the number of agents has almost no effect on the problem. In fact it is sometimes faster to solve the problems with more agents, since there are more feasible solutions.

Table 2: Average computation time and statistics regarding optimization problem size for different number of agents.

#Agents	20	100	500
#variables	51k (50k)	51k (50k)	51k (50k)
#constraints	72k (56k)	72k (56k)	72k (56k)
Creating cons (sec)	20.6 (11.2)	20.8 (9.4)	20.6 (11.4)
Solving ILP (sec)	16.2 (11.4)	13.6 (10.5)	12.8 (11.2)

Finally, we investigate the effects of the trajectory length h. Only increasing the trajectory length may still return trivial solutions, so we simultaneously increase the complexity of the specification. The number of states is fixed to $I = 100$ and the number of agents to $N = 20$ for this example. The regions S_1 and S_2 are created as before and we create three goal regions using the procedure described above. We consider specifications of the form

$$\begin{aligned} \varphi \doteq &(\Diamond\Box([S_2, N/2])) \wedge (\Diamond\Box([G_1, N/5])) \\ &\wedge (\Diamond\Box([G_2, N/5])) \wedge (\Diamond\Box([G_3, N/5])) \wedge \varphi_* \end{aligned}$$

where φ_* contains different number of consecutive goals in different trials. For instance, for two goals we have

$$\varphi_* \doteq (\Diamond([G_1, 3] \wedge \Diamond([G_2, 3]))).$$

By nesting $\Diamond$ operators as in φ_*, we force the counting propositions $cp_1 = [G_1, 3]$ and $cp_2 = [G_2, 3]$ to be satisfied in given order. As we increase h, we also increase the number of counting propositions to be satisfied in a given order. For example, for three goals, φ_* is given as follows:

$$\varphi_* \doteq (\Diamond([G_1, 3] \wedge \Diamond([G_2, 3] \wedge \Diamond([G_3, 3])))).$$

The average computation time over ten trials for different values of h and for different numbers of goals is presented in Table 3. As expected, time horizon noticeably affects the size of the problem. Again both the generic encodings and template encodings are used and their computational statistics are compared. Note that although the specification φ_* is not among the templates given in Section 4.2 and it has some nesting, using similar ideas, it is possible to construct a simple encoding for it.

Table 3: Average computation time and statistics regarding optimization problem size for different trajectory lengths and different number of consecutive goals.

h / #goals	20 / 1	50 / 2	100 / 3
#variables	50k (49k)	126k (123k)	252k (246k)
#constraints	72k (55k)	262k (139k)	560k (279k)
Creating cons (sec)	16.9 (1.9)	218.5 (10.5)	1537.2 (44.6)
Solving ILP (sec)	10.0 (13.8)	63.9 (25.7)	522.3 (130.3)

6.2 Abstraction-based synthesis

In this section we highlight the option of solving a counting problem when the transition system is an abstraction of underlying continuous-state dynamics. We solve for a specification similar to those in the previous subsection and generate a transition system for each agent by abstracting their continuous dynamics. To construct the abstraction, we employ the backwards reachability-based approach in [20, 27], with the difference that instead of iteratively partitioning the state-space, we use a fixed size partition. The workspace is partitioned with discretization step 0.25, resulting in a 13×9 grid. We assume each agent has the following dynamics:

$$x(t+1) = \begin{bmatrix} 0.9 & 0.1 \\ 0 & 0.9 \end{bmatrix} x(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} d(t), \qquad (25)$$

where $x \in [-1.125, 2.125] \times [-1.125, 1.125]$ is the state, $u \in [-0.4, 0.4]^2$ is the control input, and $d \in [-0.04, 0.04]^2$ is the disturbance.

The regions of interest are defined as follows:

$$\begin{aligned} G_1 &= [1.375, 2.125] \times [-1.125, -0.375], \\ G_2 &= [1.375, 2.125] \times [0.375, 1.125], \\ G_3 &= [-1.125, -0.375] \times [0.375, 1.125], \\ O &= [-0.125, 0.625] \times [0.375, 1.125]. \end{aligned}$$

The specification considered is the following:

$$\begin{aligned} \varphi =& \square(\neg[O,1]) \wedge \lozenge\left([G_1, 23] \wedge \lozenge([G_2, 23])\right) \wedge \\ & \lozenge\square([G_1, 5]) \wedge \square\lozenge([G_2, 18]) \wedge \square\lozenge([G_3, 18]). \end{aligned}$$

We start with $N = 23$ agents all with initial conditions in the set $[-1.125, -0.375] \times [-1.125, -0.375]$. Trajectory length is set to $h = 50$. The proposed algorithm finds a trajectory for the agents that satisfy φ in approximately 30 seconds. Note that, by construction of the abstraction, this solution trajectory is implementable by the continuous dynamics (25) of each agent. A simulation video implementing the synthesized plan together with concrete low-level continuous controllers for individual agents can be viewed at https://youtu.be/4ttFAB53egE.

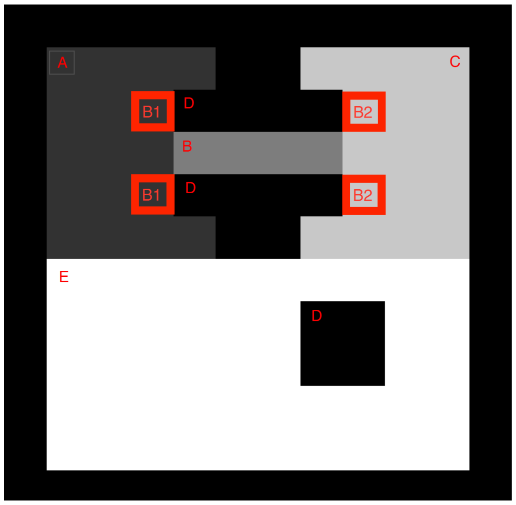

Figure 1: A map of a small area: A, C, **and** E **represent different neighborhoods,** B **represents a fragile bridge and** D **represents inaccessible zones.**

6.3 An emergency response scenario

We finally consider an earthquake emergency response scenario where a collection of ten[1] robots are to be deployed for various tasks. Figure 1 shows the affected area, divided into different regions. Regions A, C, and E represent different neighborhoods, region B represents a possibly damaged bridge, and region D represents inaccessible zones. We model each robot with a transition system with 100 states, where each state corresponds to one cell on a 10×10 grid laid out on the area. The state labels are defined according to the robot's position on this grid. Transitions are defined such that from each cell, the robot can travel to four neighboring cells.

The specifications are as follows. The robots should always avoid the inaccessible zones:

$$\varphi_1 \doteq \square(\neg([D,1])).$$

The bridge should not be used by any robot before there is at least one robot in both ends (to do an inspection to verify that it is safe to use the bridge):

$$\varphi_2 \doteq (\neg([B,1]))\ \mathcal{U}\ ([B_1,1] \wedge [B_2,1]),$$

and it should never be occupied by more than one robot (as it might collapse due to weight):

$$\varphi_3 \doteq \square(\neg([B,2])).$$

After the exploration starts, eventually always at most one robot should be present in the less affected neighborhood E:

$$\varphi_4 \doteq \lozenge\square(\neg([E,2])).$$

Infinitely often (i) there should be more than five robots in neighborhood A, (ii) there should be more than five robots in neighborhood C, (iii) neighborhood A should be left empty, and (iv) neighborhood C should be left empty:

$$\varphi_5 \doteq \square\lozenge(([A,5]), \quad \varphi_6 \doteq \square\lozenge(\neg([A,1]),$$

[1]Our approach can easily handle much larger collections as demonstrated in Section 6.1. We use ten agents in this example to make the visualization easier.

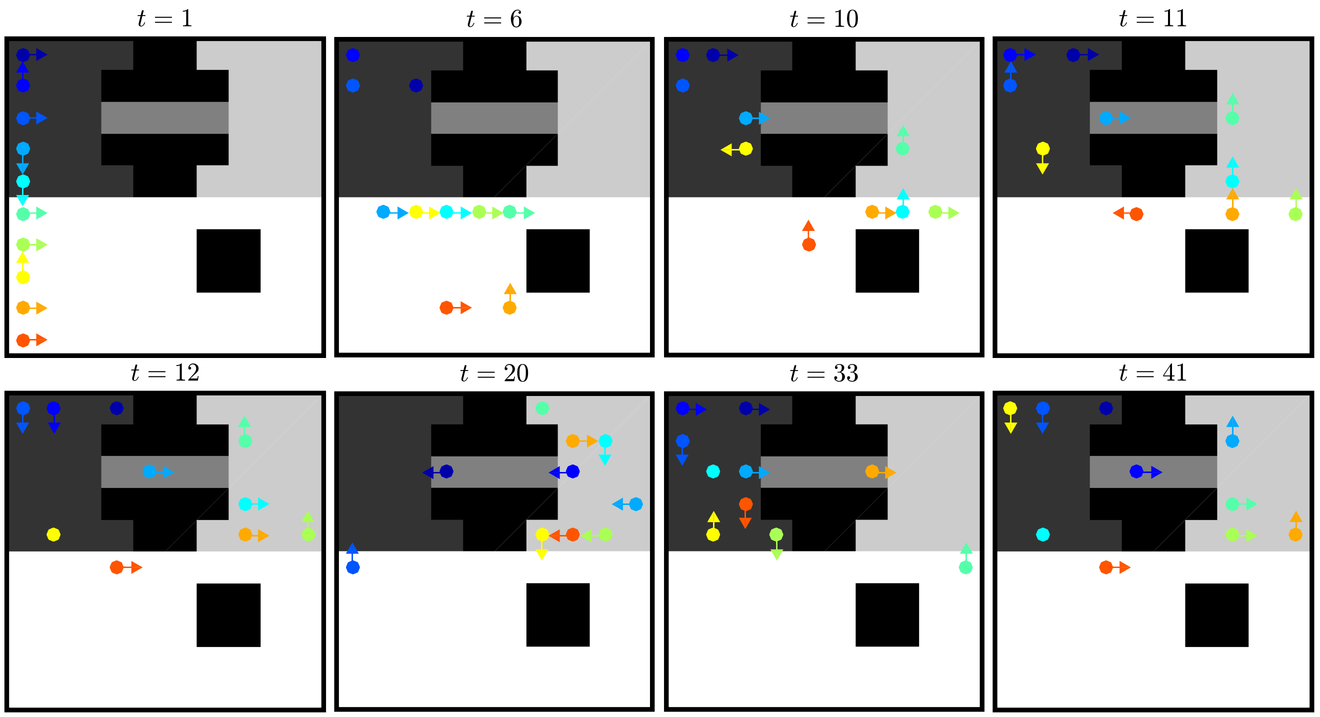

Figure 2: Illustration of individual trajectories in the emergency response scenario at selected time instants. The arrows indicate the direction robots are about to move in; robots without an arrow remain at the same position in the following time step. The prefix-suffix character of the solution is illustrated by the fact that the configuration at $t = 41$ is identical (modulo permutation) to the configuration at $t = 12$. Furthermore, all counting constraints are satisfied: at $t = 20$ region A is left empty and there are eight robots in region C. In addition there are eight robots in region A at $t = 33$ while the region C is empty. There are also no robots on the bridge B until it is examined from both ends at $t = 10$ and never more than one robot afterwards. During the suffix (after $t = 12$) there is at most one robot in region E. Finally, the black inaccessible regions are avoided throughout the execution.

$$\varphi_7 \doteq \Box\Diamond(([C,5]), \ \varphi_8 \doteq \Box\Diamond(\neg([C,1]).$$

Putting everything together, overall specification is

$$\varphi = \bigwedge_{i=1}^{8} \varphi_i.$$

We take the trajectory length h to be 40 and solve the corresponding instance of Problem 1. The resulting individual trajectories are demonstrated in Fig. 2 and together satisfy φ. A video simulating the synthesized plan can be seen at https://youtu.be/EJ-v2yD-6_I.

7 CONCLUSION

In this paper we introduced a logic for specifying the correct behavior of a multi-agent system in which agents are equal and the identity of individual agents is not important for the task to be completed. We proposed an optimization-based approach for synthesizing plans for the agents so that the resulting collective trajectories satisfy the specification. Due to the structure of the problem—permutation invariance in the counting constraints and in agent roles—the proposed approach works well for large numbers of agents. We provided some analysis of the proposed approach and demonstrated it with several examples. An extension of the problem in the presence of different classes of agents, and considerations regarding the robustness of the synthesized plans when agents are added to and/or removed from the collection, were also presented.

As future work, we are looking into extending the logic to allow specifying individual roles to certain agents, with the main difficulty being sustaining the advantages of permutation invariance. One of the assumptions in this work was that the agents move synchronously; however, many large collections of cyber-physical systems may lack this property. As part of the future work, we would like to extend the approach to handle asynchronous agents, or agents that can synchronize only when they are spatially close. Finally, we are interested in implementing the synthesized plans on hardware by, for instance, coordinating a collection of quadrotors.

REFERENCES

[1] Christel Baier and Joost-Pieter Katoen. 1999. *Principles of Model Checking*. MIT Press.
[2] Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George J Pappas. 2007. Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine* 14, 1 (2007), 61–70.

[3] Calin Belta and Vijay Kumar. 2004. Abstraction and control for groups of robots. *IEEE Transactions on Robotics* 20, 5 (2004), 865–875.

[4] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. 1999. Symbolic model checking without BDDs. In *Proc. of TACAS*. Springer, 193–207.

[5] Armin Biere, Keijo Heljanko, Tommi Junttila, Timo Latvala, and Viktor Schuppan. 2006. Linear Encodings of Bounded LTL Model Checking. *Logical Methods in Computer Science* 2 (2006), 1–64.

[6] Yushan Chen, Xu Chu Ding, Alin Stefanescu, and Calin Belta. 2012. Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics* 28, 1 (2012), 158–171.

[7] Gurobi Optimization, Inc. 2016. Gurobi Optimizer Reference Manual. (2016). http://www.gurobi.com

[8] I. Haghighi, S. Sadraddini, and C. Belta. 2016. Robotic swarm control from spatio-temporal specifications. In *Proc. of IEEE CDC*. 5708–5713.

[9] Geoffrey Hollinger, Sanjiv Singh, and Athanasios Kehagias. 2009. Efficient, Guaranteed Search with Multi-Agent Teams. In *Proc. of RSS*. 265–272.

[10] Sertac Karaman, Ricardo G Sanfelice, and Emilio Frazzoli. 2008. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Proc. of IEEE CDC*. 2117–2122.

[11] Marius Kloetzer and Calin Belta. 2007. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics* 23, 2 (2007), 320–330.

[12] Marius Kloetzer and Calin Belta. 2010. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics* 26, 1 (2010), 48–61.

[13] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics* 25, 6 (2009), 1370–1381.

[14] Peter B Luh, Christian T Wilkie, Shi-Chung Chang, Kerry L Marsh, and Neal Olderman. 2012. Modeling and optimization of building emergency evacuation considering blocking effects on crowd movement. *IEEE Transactions on Automation Science and Engineering* 9, 4 (2012), 687–700.

[15] John-Michael McNew, Eric Klavins, and Magnus Egerstedt. 2007. Solving coverage problems with embedded graph grammars. In *Proc. of HSCC*. 413–427.

[16] Daniel Mellinger and Vijay Kumar. 2011. Minimum snap trajectory generation and control for quadrotors. In *Proc. of IEEE ICRA*. 2520–2525.

[17] Mehran Mesbahi and Magnus Egerstedt. 2010. *Graph theoretic methods in multiagent networks*. Princeton University Press.

[18] Keiji Nagatani, Seiga Kiribayashi, Yoshito Okada, Kazuki Otake, Kazuya Yoshida, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, Mineo Fukushima, and Shinji Kawatsuma. 2013. Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics* 30, 1 (2013), 44–63.

[19] Petter Nilsson and Necmiye Ozay. 2016. Control Synthesis for Large Collections of Systems with Mode-Counting Constraints. In *Proc. of HSCC*. 205–214.

[20] Petter Nilsson, Necmiye Ozay, Ufuk Topcu, and Richard M. Murray. 2012. Temporal logic control of switched affine systems with an application in fuel balancing. In *Proc. of ACC*. 5302–5309.

[21] Giordano Pola, Antoine Girard, and Paulo Tabuada. 2008. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica* 44, 10 (2008), 2508–2516.

[22] Vasumathi Raman. 2014. Reactive switching protocols for multi-robot high-level tasks. In *Proc. of IEEE/RSJ IROS*. 336–341.

[23] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. 2014. Model predictive control with signal temporal logic specifications. In *Proc. of IEEE CDC*. 81–87.

[24] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. 2016. Implan: Scalable Incremental Motion Planning for Multi-Robot Systems. In *Proc. of ACM/IEEE ICCPS*. 1–10.

[25] Herbert G Tanner, George J Pappas, and Vijay Kumar. 2004. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation* 20, 3 (2004), 443–455.

[26] Eric M Wolff, Ufuk Topcu, and Richard M Murray. 2014. Optimization-based trajectory generation with linear temporal logic specifications. In *Proc. of IEEE ICRA*. 5319–5325.

[27] Tichakorn Wongpiromsarn. 2010. *Formal methods for design and verification of embedded control systems: application to an autonomous vehicle*. Ph.D. Dissertation. California Institute of Technology.

[28] Z. Xu and A. A. Julius. 2016. Census Signal Temporal Logic Inference for Multiagent Group Behavior Analysis. *IEEE Transactions on Automation Science and Engineering* PP, 99 (2016), 1–14.

[29] Jingjin Yu and Steven M LaValle. 2013. Planning optimal paths for multiple robots on graphs. In *Proc. of IEEE ICRA*. 3612–3617.