Linear Temporal Logic Motion Planning for Teams of Underactuated Robots Using Satisfiability Modulo Convex Programming

Yasser Shoukry[†§] Pierluigi Nuzzo[†] Ayca Balkan[§] Indranil Saha[**]
Alberto L. Sangiovanni-Vincentelli[†] Sanjit A. Seshia[†] George J. Pappas[*] Paulo Tabuada[§]

***Abstract*—We present an efficient algorithm for multi-robot motion planning from Linear Temporal Logic (LTL) specifications. We assume that the dynamics of each robot can be described by a discrete-time, linear system together with constraints on the control inputs and state variables. The workspace is characterized by a set of obstacles and a set of regions of interest, where both the obstacles and the regions are polyhedra. Given an LTL formula ψ, specifying the multi-robot mission, our goal is to construct a set of collision-free trajectories for all robots, and the associated control strategies, to satisfy ψ. We show that the motion planning problem can be formulated as the feasibility problem for a formula φ over Boolean and convex constraints, respectively capturing the LTL specification and the robot dynamics. We then adopt a Satisfiability Modulo Convex (SMC) programming approach that exploits a monotonicity property of φ to decompose the problem into smaller subproblems. At each iteration, a Boolean satisfiability solver searches for candidate high-level discrete plans while completely abstracting the low-level continuous dynamics. Convex programming is then used to check the feasibility of the proposed plans against the dynamic constraints, or generate succinct explanations for their infeasibility to reduce the search space. New candidate plans can then be efficiently generated until a feasible one is found. Simulation results show that our algorithm is more than one order of magnitude faster than state-of-the-art sampling-based techniques for high-dimensional state spaces while supporting complex missions.**

I. Introduction

An increasing number of safety-critical robotics applications (e.g., in rescue missions) as well as autonomous systems (e.g., unmanned aircraft and self-driving cars) require efficient techniques that can rigorously reason about hybrid system behaviors and guarantee the correctness of a controller implementation. Algorithmic control synthesis from formal specifications captured by a logic formalism, such as Linear Temporal Logic (LTL) [1], holds considerable promise for encompassing a rich set of task specifications, while providing implementations that are correct by construction [2–7]. However, the complexity of today's robotics and autonomous systems poses several challenges to synthesis techniques.

A major difficulty stems from the growing need to reason about the tight integration of discrete abstractions (*task planning*) with continuous trajectories (*motion planning*) [8]. This integration can soon become daunting for complex, high-dimensional systems, since a vast hybrid, discrete/continuous space must be searched while accounting for complex geometries, motion dynamics, collision avoidance, and temporal goals. In this paper, we address this challenge by focusing on the problem of *multi-robot motion planning* from LTL specifications. Given the robot model and initial state, a description of the workspace, and a task specification as an LTL formula ψ, we aim at planning collision-free and dynamically-feasible motion trajectories for all robots that satisfy ψ. While a growing body of work has focused, over the years, on the synthesis of reactive controllers that satisfy LTL specifications, a set of computational difficulties in this context still comes from the interplay between motion trajectories and task constraints. The task planner needs to generate trajectories that the robot can execute in the physical world, i.e., which satisfy the robot dynamics constraints (e.g., bounds on the velocity, directions of motions, and control inputs) and are collision-free. However, in complex systems, effective discrete planning techniques may fail to accurately capture the constraints imposed by the dynamics; on the other hand, effective methods for generating collision-free and dynamically-feasible trajectories may turn out to be inefficient in the presence of complex mission requirements imposed by ψ.

A first category of techniques for LTL motion planning utilizes a discrete abstraction of the system, often obtained by partitioning the continuous state space into polytopes, and an automata-theoretic approach to synthesize the controller [2–5]. However, these approaches are subject to the *curse of dimensionality* and become usually impractical for systems with more than five continuous states [9]. Moreover, some of these approaches assume the availability of low-level feedback controllers that are capable of generating collision-free and dynamically-feasible trajectories that are compatible with each automaton action, which may not always be the case for complex robotic systems. A second category of approaches attempts at synthesizing the high-level planner together with the associated low-level controller, by either leveraging mixed integer linear programming (MILP) encodings of LTL specifications [10, 11] or sampling-based methods [12, 13]. MILP-based planners can leverage the empirical performance of state-of-the-art solvers to solve for both the discrete and continuous constraints at the same time; however, they still tend to be impractical when the problem size grows. On the other hand, sampling-based techniques tend to require large computation time for obstacle avoidance problems in the presence of narrow passages [14] and for underactuated systems, e.g., systems with a lower number of actuators than degrees of freedom, under dynamic, in addition to kinematic, constraints. Moreover, sampling-based techniques do not have, in general, control over the length of the generated trajectory.

In this paper, we propose an efficient method for the integration of task planning and robot motion planning from generic LTL specifications based on the coordination of Boolean Satisfiability (SAT) solving and convex programming. We consider the case of robots with dynamics that can be modeled as a discrete-time linear system. We then build on

Affiliations: [†]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, [§]Department of Electrical Engineering, University of California, Los Angeles, CA, [*]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, [**]Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India

This work was partially sponsored by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, by the NSF project ExCAPE: Expeditions in Computer Augmented Program Engineering, and by NSF award 1239085.

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 20, 2017.

our previous results [15, 16] on motion planning for a single robot under reach-avoid specifications to also address multi-robot scenarios, including collision avoidance constraints, and arbitrary LTL specifications.

We show that multi-robot motion planning can be formulated as the feasibility problem for a type of formula φ, called monotone Satisfiability Modulo Convex formula, over a combination of Boolean and convex constraints, respectively capturing the LTL specification and the robot dynamics. We then adopt a *Satisfiability Modulo Convex programming (SMC)* approach [16, 17] that exploits the monotonicity property of φ to decompose the problem into smaller subproblems that can be efficiently solved. In SMC, a SAT solver searches for candidate high-level discrete plans while completely abstracting the low-level continuous dynamics. Convex programming is then used to check the feasibility of the proposed plans against the dynamic constraints and provide the reason for the conflict, an infeasibility certificate, whenever inconsistencies are found. SMC can efficiently generate succinct infeasibility explanations, which is key to reduce the search space and achieve scalability. New candidate plans can then be rapidly generated until a feasible one is found. In this way, we are able to leverage the efficiency and the formal guarantees of state-of-the-art algorithms in both the Boolean and convex analysis domains.

While being inspired by the lazy Satisfiability Modulo Theory (SMT) paradigm [18] and the counterexample-based control approach [19], our framework differs from the compositional motion planning approach by Saha et al. [7], which focuses on generating trajectories out of a library of motion primitives for a set of safe, finite-horizon LTL properties, by leveraging an off-the-shelf SMT solver. When compared with MILP-based techniques, SMC approaches have shown to scale better as the number of Boolean variables and constraints in φ increases [16], which is typically the case for complex multi-robot task planning. In fact, SMC exploits abstraction and conflict-driven learning, together with the structure of φ, to directly reduce the search space and decompose the original problem into a sequence of convex programs with usually a much smaller number of variables and constraints. Conversely, MILP-based approaches leverage branching and cutting planes to generate a sequence of continuous or Lagrangean relaxations of the original problem, whose overall number of variables and constraints may become prohibitively high, sometimes on the order of the number of all the Boolean and real variables and constraints of the original problem. Finally, the simulation results in this paper show that our algorithm can run faster than state-of-the-art sampling-based techniques for high-dimensional state spaces, while supporting complex missions.

II. Problem Formulation

We consider a set R of N robots that move in a workspace $\mathcal{W} \subset \mathbb{R}^w$ where w can be 2 or 3, corresponding, respectively, to a 2-dimensional or 3-dimensional workspace. We use $\|a\|$ to denote the infinity norm of a and formulate the motion planning problem as follows.

A. Robot Model

We assume that the dynamics of robot R_i, $i \in \{1, \dots, N\}$, is described by a discrete-time linear system of the form:

$$x_{t+1}^i = A_i x_t^i + B_i u_t^i, \qquad \text{(II.1)}$$

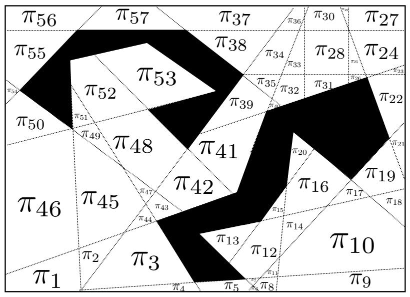

Fig. 1. Pictorial representation of the workspace regions and obstacles for motion planning problem in this paper.

$$x_0^i = \overline{x}_0^i, \qquad \|x_t^i\| \leq \overline{x}_i, \qquad \|u_t^i\| \leq \overline{u}_i, \quad \forall t \in \mathbb{N} \quad \text{(II.2)}$$

where $x_t^i \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state of robot R_i at time $t \in \mathbb{N}$, $u_t^i \in \mathcal{U} \subseteq \mathbb{R}^m$ is the robot input, $\overline{x}_0^i$ is the robot initial state, and $\overline{u}_i$ and $\overline{x}_i$ are bounds on the input and state variables. The matrices A_i and B_i represent the robot dynamics and have appropriate dimensions. For a robot with nonlinear dynamics that is either differentially flat or feedback linearizable, the state space model (II.1) corresponds to its feedback linearized dynamics.

B. Workspace

We assume that the robots must avoid a set of *obstacles* $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_o\}$, with $\mathcal{O}_i \subset \mathbb{R}^w$, and represent the obstacle-free *workspace* as $\mathcal{W} = \bigcup_1^r \mathcal{W}_i$, where $\overline{\mathcal{W}} = \{\mathcal{W}_1, \dots, \mathcal{W}_r\}$ is a set of non-overlapping *regions*, with $\mathcal{W}_i \subset \mathbb{R}^w$. As pictorially represented in Fig. 1, both the regions and the obstacles are assumed to be polygons.

For robot R_i, we can uniquely associate to each of the above regions a proposition in the set $\Pi^i = \{\pi_1^i, \dots, \pi_r^i\}$, where π_j^i evaluates to one (true) if robot R_i is in region $\mathcal{W}_j$ and zero (false) otherwise. We then denote by $h_{\mathcal{W}\to\Pi^i} : \mathcal{W} \to \Pi^i$ the map from each point $w \in \mathcal{W}$ to the proposition $\pi_j^i \in \Pi^i$ that evaluates to one at w for robot R_i. Moreover, a subset of each robot state variables, describing its position (coordinates), is also used to describe $\mathcal{W}$. Therefore, we denote as $h_{\mathcal{X}\to\mathcal{W}} : \mathcal{X} \to \mathcal{W}$ the natural projection of the state x^i onto the workspace $\mathcal{W}$, and by $h_{\mathcal{X}\to\Pi^i}$ the map from the state space of robot R_i to the set of propositions Π^i, obtained after projecting the state onto the workspace, i.e., $h_{\mathcal{X}\to\Pi^i}(x^i) = h_{\mathcal{W}\to\Pi^i}(h_{\mathcal{X}\to\mathcal{W}}(x^i))$.

Finally, we introduce an *adjacency function* $Adj : \overline{\mathcal{W}} \times \overline{\mathcal{W}} \to \mathbb{B}$ over the pairs of elements in $\overline{\mathcal{W}}$ such that $Adj(\mathcal{W}_i, \mathcal{W}_j) = 1$ if $\mathcal{W}_i$ and $\mathcal{W}_j$ are adjacent and 0 otherwise[1]. Because of the one-to-one correspondence between elements in $\overline{\mathcal{W}}$ and propositions in Π^i, we also write $Adj(\pi_j^i, \pi_k^i) = 1$ if π_j^i and π_k^i are associated with adjacent regions in $\overline{\mathcal{W}}$ and 0 otherwise. Moreover, for all i and j, $Adj(\pi_j^i, \pi_j^i) = 1$ holds.

C. Collision Avoidance

We require the distance (with respect to the infinity norm) between any two robots in the workspace at each time to be larger than an arbitrarily small positive number $\epsilon \in \mathbb{R}^+$. Formally,

$$\|h_{\mathcal{X}\to\mathcal{W}}(x_t^i) - h_{\mathcal{X}\to\mathcal{W}}(x_t^j)\| \geq \epsilon, \quad \forall\, t \geq 0, \qquad \text{(II.3)}$$
$$\forall\, i, j \in \{1, \dots, N\}, i \neq j.$$

[1]Two polyhedra in $\mathbb{R}^w$ are adjacent if they share a face of dimension $w-1$.

D. Linear Temporal Logic

We express the specification for a multi-robot mission using Linear Temporal Logic (LTL) [1]. Let $\Pi = \bigcup_{i=1}^{R} \Pi^i$ be the set of propositions associated with the workspace regions for all robots, as defined above. We consider formulas over a set of atomic propositions Σ, where $\sigma(\pi) \in \Sigma$ is a Boolean or pseudo-Boolean predicate on Π. For example, we can express that "either robot R_1 or R_2 must be in $\mathcal{W}_1$" via the proposition $\sigma_1 := \pi_1^1 \vee \pi_1^2$ or that "at least one robot must be in $\mathcal{W}_2$" using the proposition $\sigma_2 := \sum_{i=1}^{N} \pi_2^i \geq 1$.

Syntax. From atomic propositions $\sigma \in \Sigma$, any LTL formula can be generated according to the following grammar:

$$\psi := \sigma \mid \neg\psi_0 \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \bigcirc\psi_0 \mid \psi_1 \,\mathcal{U}\, \psi_2 \mid \psi_1 \,\mathcal{R}\, \psi_2,$$

where ψ_0, ψ_1, ψ_2 are LTL formulas. Given the above grammar, we can define $false$ and $true$ in the following way: $false = \psi \wedge \neg\psi$ and $true = \neg false$. Given the temporal operators *next* ($\bigcirc$), *until* ($\mathcal{U}$), and *release* ($\mathcal{R}$), we can derive additional temporal operators, for example, *eventually* ($\Diamond$) and *always* ($\Box$). These operators are derived as:

$$\Diamond\psi = true\ \mathcal{U}\ \psi \qquad \text{and} \qquad \Box\psi = false\ \mathcal{R}\ \psi.$$

Semantics. The semantics of an LTL formula is defined over an infinite sequence ρ of valuations, i.e., truth assignments to the propositions used in the formula. Let ρ_t denote the set of atomic propositions that are true at the t-th position of ρ. For an LTL formula ψ, we denote by $\rho, t \models \psi$ the fact that the sequence ρ starting with the t-th element satisfies the LTL formula. Satisfaction is recursively defined as follows:

$$\begin{array}{ll}
\rho, t \models \sigma & \text{iff } \sigma \in \rho_t \\
\rho, t \models \neg\psi & \text{iff } \rho, t \not\models \psi \\
\rho, t \models \psi_1 \wedge \psi_2 & \text{iff } \rho, t \models \psi_1 \text{ and } \rho, t \models \psi_2 \\
\rho, t \models \psi_1 \vee \psi_2 & \text{iff } \rho, t \models \psi_1 \text{ or } \rho, t \models \psi_2 \\
\rho, t \models \bigcirc\psi & \text{iff } \rho, t+1 \models \psi \\
\rho, t \models \psi_1 \mathcal{U} \psi_2 & \text{iff there exists } k \geq t \text{ such that } \rho, k \models \psi_2 \\
& \quad \text{and for all } t \leq j < k, \quad \rho, j \models \psi_1 \\
\rho, t \models \psi_1 \mathcal{R} \psi_2 & \text{iff for all } k \geq t\text{: } \rho, k \models \psi_2 \\
& \quad \text{or for some } t \leq j < k, \quad \rho, j \models \psi_1
\end{array}$$

The sequence ρ satisfies a formula ψ, if $\rho, 0 \models \psi$.

Closure. For an LTL formula ψ, the closure $cl(\psi)$ is the smallest set of subformulas that satisfy the following conditions:

$$\begin{array}{lll}
\psi \in cl(\psi) & & \\
\psi_1 \in cl(\psi) & \text{if } \circ\,\psi_1 \in cl(\psi) & \text{for } \circ \in \{\neg, \bigcirc\} \\
\psi_1,\ \psi_2 \in cl(\psi) & \text{if } \psi_1 \circ \psi_2 \in cl(\psi) & \text{for } \circ \in \{\wedge, \vee, \mathcal{U}, \mathcal{R}\}
\end{array}$$

E. Problem Definition

Definition 2.1 (Problem Instance): A problem instance is a tuple $\mathcal{P} = (R, \mathcal{W}, \Pi, Adj, \Sigma, D, \overline{x}_0, \overline{x}, \overline{u}, \epsilon, \psi)$, where:

- R is the set of robots,
- $\mathcal{W}$ is the workspace,
- Π is the set of propositions corresponding to the workspace regions and robots,
- Adj is the adjacency function defining the connectivity of the different regions in the workspace,
- Σ is the set of atomic propositions for the robot mission,
- $D = \{(A_1, B_1), \ldots, (A_N, B_N)\}$ is the set of dynamics for the group of robots,
- $\overline{x}_0 = (\overline{x}_0^1, \ldots, \overline{x}_0^N)$ is the set of initial states for the group of robots,
- $\overline{x} = (\overline{x}^1, \ldots, \overline{x}^N)$ is the set of bounds on the states for the group of robot,
- $\overline{u} = (\overline{u}^1, \ldots, \overline{u}^N)$ is the set of bounds on the control inputs for the group of robots,
- $\epsilon \in \mathbb{R}^+$ is the positive margin for collision avoidance,
- ψ denotes the LTL specification defined over the atomic propositions Σ that the robots have to satisfy.

Definition 2.2 (Trajectory): A system *trajectory* for a problem instance $\mathcal{P} = (R, \mathcal{W}, \Pi, Adj, \Sigma, D, \overline{x}_0, \overline{x}, \overline{u}, \epsilon, \psi)$ is a triple $(x, \lambda, \rho(\lambda))$ including the following infinite sequences:

- $x = x_0 x_1 x_2 \ldots$ is a sequence of system states, where the system state $x_t = (x_t^1, \ldots, x_t^N) \in \mathcal{X}^N$ includes the states of all the robots at time t;
- $\lambda = \lambda_0 \lambda_1 \lambda_2 \ldots$ is a sequence of valuations over Π, where $\lambda_t = (\lambda_t^1, \ldots, \lambda_t^N) \in \Pi^1 \times \ldots \times \Pi^N$ is the set of workspace propositions that are true at x_t, i.e., $\lambda_t^i = h_{\mathcal{X} \to \Pi^i}(x_t^i)$ for all $t \geq 0$ and $i \in \{1, \ldots, N\}$;
- $\rho(\lambda) = \rho_0(\lambda_0)\rho_1(\lambda_1)\rho_2(\lambda_2)\ldots$ is a sequence of valuations over Σ, where $\rho_t(\lambda_t)$ is the truth assignment associated with state x_t and propositions λ_t.

We call x and λ, respectively, the *state trajectory* and the *region trajectory* of the multi-robot system. Similarly, we call $x^i = x_0^i x_1^i x_2^i \ldots$ and $\lambda^i = \lambda_0^i \lambda_1^i \lambda_2^i \ldots$, respectively, the state and region trajectory for robot R_i.

Definition 2.3 (Valid Trajectory): A trajectory (x, λ, ρ) for a problem instance $\mathcal{P} = \langle R, \mathcal{W}, \Pi, Adj, \Sigma, D, \overline{x}_0, \overline{x}, \overline{u}, \epsilon, \psi \rangle$ is a *valid trajectory* if the following holds:

- **Initial state constraint**: $x_0 = \overline{x}_0$,
- **Dynamics, input, and state constraints**: for all $i \in \{1, \ldots, N\}$ and $k \geq 0$ there exists u_k^i such that $x_{k+1}^i = A_i x_k^i + B_i u_k^i, \|x_k^i\| \leq \overline{x}^i$, and $\|u_k^i\| \leq \overline{u}^i$,
- **Workspace and obstacle avoidance constraints**: $Adj(\lambda_k^i, \lambda_{k+1}^i) = 1, \forall\ k \geq 0, \forall\ i \in \{1, \ldots, N\}$,
- **Collision avoidance constraints**: $\forall\ k \geq 0, \forall\ i, j \in \{1, \ldots, N\}, i \neq j, \|h_{\mathcal{X} \to \mathcal{W}}(x_k^i) - h_{\mathcal{X} \to \mathcal{W}}(x_k^j)\| \geq \epsilon$,
- **LTL constraints**: ρ satisfies the formula ψ, i.e., $\rho, 0 \models \psi$.

We now formally define the motion planning problem that we solve in this paper.

Problem 2.4 (Motion Planning Problem): Given a problem instance $\mathcal{P} = \langle R, \mathcal{W}, \Pi, Adj, \Sigma, D, \overline{x}_0, \overline{x}, \overline{u}, \epsilon, \psi \rangle$, synthesize a valid trajectory for the multi-robot system.

III. SMC-Based Solution Strategy

Using state space discretizations to account for constraints on the continuous dynamics may lead to state explosion as the number of continuous states and the number of obstacles increase. Our strategy aims, instead, at exploiting coarser abstractions of both the state space and the workspace, thus effectively decoupling the problem of *generating an obstacle-free path* from the ones of *checking physical realizability and collision avoidance*. By leveraging a Satisfiability Modulo Convex (SMC) programming approach, we then partition the planning problem into two smaller subproblems involving reasoning, respectively, on sets of discrete and continuous variables from the original problem. These subproblems can be efficiently solved using specialized techniques.

Algorithm 1 SMC-BASED MOTION PLANNER

```
1: Initialize horizon:    $L := 1$;
2: while Trajectory is not found do
3:     $|[\mathcal{P}, L]|_D :=$ ENCODE-DIS-PLAN($\mathcal{P}, L$)
4:     $|[\mathcal{P}, L]|_C :=$ ENCODE-CON-PLAN($\mathcal{P}, L$)
5:     (STATUS, $\lambda, x, u$) := SMC.SOLVE($|[\mathcal{P}, L]|_D, |[\mathcal{P}, L]|_C$);
6:     if STATUS == UNSAT then
7:         Increase horizon:    $L := L + 1$;
8: return $(\lambda, x, u)$;
```

As summarized in Algorithm 1, we first observe that the multi-robot motion planning problem for a fixed horizon L can be formulated as the feasibility problem for a special type of formula φ over Boolean and convex constraints, respectively capturing the constraints in the LTL specification ψ (subformula $|[\mathcal{P}, L]|_D$) and the dynamics (subformula $|[\mathcal{P}, L]|_C$). The formula φ is a monotone SMC formula that can be solved via a finite number of convex programs [16]. Specifically, SMC.SOLVE follows an iterative approach combining efficient SAT solving with a convex programming engine. At each iteration, the SAT solver generates candidate high-level paths λ that satisfy the set of constraints expressed by ψ. These paths are only defined over the set of Boolean propositions Π and ignore the robots' dynamics, input and state constraints, as well as the collision avoidance constraints.

The feasibility of the generated paths λ is then checked with respect to the system dynamics D, the control input bounds $\overline{u}$, the state bounds $\overline{x}$, the robots' initial states $\overline{x}_0$, and the collision avoidance constraints, by casting a convex optimization problem. If both the Boolean and the real-valued constraints are satisfied, a valid trajectory is returned, consisting of the proposed plan and the corresponding state and control input trajectories for the group of robots. Otherwise, the proposed sequence λ is marked as infeasible and new candidate plans are generated until either a feasible one is found, or SMC.SOLVE returns UNSAT, meaning that no trajectory is feasible for the current horizon length. A prominent feature of SMC is the generation of compact infeasibility certificates, i.e., "succinct explanations" that can capture the root causes for the infeasibility of a plan and rule out the largest possible number of invalid plans for the SAT solver to accelerate the search. In what follows, we provide details on the encodings of both the discrete and continuous planning problems and discuss the formal guarantees of Algorithm 1.

IV. SMC ENCODING OF THE PROBLEM

A. Encoding the High-Level Discrete Planning Problem

We translate the high-level, discrete planning problem into a conjunction of Boolean constraints using the Bounded Model Checking (BMC) encoding technique for LTL model checking by Biere et al. [20]. Though a trace that satisfies an LTL formula is given as an infinite execution path of the system, such trace can be represented by a finite path under the following conditions: (i) the finite path is a valid prefix of all its infinite extensions, (ii) a portion of the finite path can loop to generate a valid infinite path. Let $(x, \lambda, \rho(\lambda))$ be a valid trajectory of the system under generic LTL constraints. The system trajectory $\rho = \rho_0\rho_1\rho_2\ldots$ can then be represented as

$$\rho = (\rho_0\rho_1\ldots\rho_{k-1})(\rho_k\ldots\rho_L)^\omega,$$

where $0 < k \leq L$ and $\rho_L = \rho_{k-1}$. Such a representation of a trajectory is called an (L, k)-loop. The trajectory $(\rho_k\ldots\rho_L)^\omega$ denotes an infinite trajectory that can be obtained by repeating the sequence $(\rho_k\ldots\rho_L)$.

Given a problem instance $\mathcal{P} = (R, \mathcal{W}, \Pi, Adj, \Sigma, D, \overline{x}_0, \overline{x}, \overline{u}, \epsilon, \psi)$, and a positive constant L, let $(\overline{x}_0, \overline{\lambda}_0)$ represent the initial state of the system, where $\overline{x}_0 = (\overline{x}_0^1, \ldots, \overline{x}_0^N)$, $\overline{\lambda}_0 = (\overline{\lambda}_0^1, \ldots, \overline{\lambda}_0^N)$ and, for all $i \in \{1, \ldots, N\}$, $\overline{\lambda}_0^i = h_{\mathcal{X}\to\Pi^i}(\overline{x}_0^i)$. Our objective is to generate a formula that represents any valid trajectory of the multi-robot system in the form of an (L, k)-loop. The decision variables for the formula are ultimately given by the propositions associated with the workspace regions to be occupied by each robot and the variable k represents the location at which the loop starts. Specifically, for all $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, r\}$, $t \in \{0, \ldots, L\}$, we introduce a Boolean variable π_{jt}^i which evaluates to one if and only if robot R_i is in region $\mathcal{W}_j$ at time t. Let $\tilde{\Pi}$ be the set of all these decision variables. Based on these variables, the encoding of the discrete trajectory synthesis problem is linear in L and captures three kinds of constraints:

- Workspace and obstacle avoidance constraints, denoted by $|[\mathcal{W}]|$,
- LTL formula constraints, denoted by $|[LTL]|$,
- Loop constraints, denoted by $|[LOOP]|$.

1) Workspace Constraints: A set of workspace constraints can be captured by the following formula:

$$|[\mathcal{W}]| := (\lambda_0 = \overline{\lambda}_0) \wedge \bigwedge_{i=1}^{N} \bigwedge_{t=1}^{L} \lambda_t^i \in \mathcal{N}(\lambda_{t-1}^i),$$

where $\mathcal{N}(\lambda_t^i) = \{\pi_j^i \in \Pi^i \,|Adj(\lambda_t^i, \pi_j^i) = 1\}$ denotes the set of regions that are adjacent (neighbors) to λ_t^i. The above formula enforces that the trajectory starts with the initial regions in $\overline{\lambda}_0$ and proceeds by only visiting regions that are adjacent. For instance, adjacency constraints can be encoded using the variables in $\tilde{\Pi}$ as follows: $\forall i \in \{1, \ldots, N\}, t \in \{1, \ldots, L-1\}$, $j \in \{1, \ldots, r\}$,

$$\pi_{j(t-1)}^i \Rightarrow \bigvee_{j' \in \mathcal{I}(j)} \pi_{j't}^i,$$

where $\mathcal{I}(j) = \{j' | Adj(\pi_j^i, \pi_{j'}^i) = 1, \pi_{j'}^i \in \Pi^i\}$. At each time t and for each robot R_i only one of the π_{jt}^i can be one, which can be captured by the following pseudo-Boolean constraints:

$$\sum_{j=1}^{r} \pi_{jt}^i = 1, \quad \forall i \in \{1, \ldots, N\}, \quad \forall t \in \{0, \ldots, L\},$$

which are also part of the formula $|[\mathcal{W}]|$. Obstacle avoidance is implicitly encoded by the fact that Π and Adj are defined only over the regions in the free space.

2) LTL Constraints: We generate Boolean constraints capturing the LTL formula specification using the *eventuality encoding* [20]. Specifically, we leverage this encoding to generate constraints that only hold for the discrete plan, as in expressions (IV.1)-(IV.7) below. The full SMC formula encoding, including constraints on both Boolean and real variables, will follow in Section IV-B.

A fresh unconstrained propositional variable b_t^φ, called *formula variable*, is introduced for every subformula $\varphi \in cl(\psi)$

and for all $0 \leq t \leq L$. The variable b_t^{φ} is true iff φ holds at time point t on the trajectory. We also introduce the Boolean variable $loopexists$, which is true if the bounded trajectory contains a loop. Moreover, the Boolean variable $inloop_t$ is true if the t-th index of the trajectory is in the loop part of the trajectory. The encoding is then defined in terms of Boolean constraints over the formula variables as described below.

Proposition and logical operators: The propositional LTL subformulas are encoded as follows: $\forall t \in \{0, \ldots, L+1\}$:

$$\begin{array}{ll} b_t^{\sigma(\pi)} \Leftrightarrow (\sigma(\pi) \in \rho_t) & b_t^{\neg\psi} \Leftrightarrow \neg b_t^{\psi} \\ b_t^{\psi_1 \wedge \psi_2} \Leftrightarrow b_t^{\psi_1} \wedge b_t^{\psi_2} & b_t^{\psi_1 \vee \psi_2} \Leftrightarrow b_t^{\psi_1} \vee b_t^{\psi_2} \end{array} \tag{IV.1}$$

Specifically, let $f_\sigma(\pi_1^1, \pi_2^1, \ldots, \pi_{r-1}^N, \pi_r^N)$ be the Boolean or pseudo-Boolean predicate providing the truth assignment for a proposition σ as a function of an evaluation over Π. We enforce this predicate to hold at each time point in the trajectory as follows: $\forall t \in \{0, \ldots, L+1\}$, $\forall \sigma \in \Sigma$:

$$b_t^{\sigma} \Leftrightarrow f_\sigma(\pi_{1t}^1, \pi_{2t}^1, \ldots, \pi_{(r-1)t}^N, \pi_{rt}^N). \tag{IV.2}$$

Temporal operators: The temporal components of the closure of an LTL formula can be encoded as follows: $\forall t \in \{0, \ldots, L\}$,

$$\begin{array}{l} b_t^{\bigcirc\ \psi_1} \Leftrightarrow b_{t+1}^{\psi_1} \\ b_t^{\psi_1\ \mathcal{U}\ \psi_2} \Leftrightarrow b_t^{\psi_2} \vee (b_t^{\psi_1} \wedge b_{t+1}^{\psi_1\ \mathcal{U}\ \psi_2}) \\ b_t^{\psi_1\ \mathcal{R}\ \psi_2} \Leftrightarrow b_t^{\psi_2} \wedge (b_t^{\psi_1} \vee b_{t+1}^{\psi_1\ \mathcal{R}\ \psi_2}) \end{array} \tag{IV.3}$$

When the trajectory contains a loop, auxiliary constraints are required to ensure the correct semantics of the $\mathcal{U}$ operator. In the constraints below, let $q_t^{\diamondsuit\psi_2}$ be an auxiliary variable denoting the evaluation of the formula $\diamondsuit\psi_2$ at the t-th location of the trajectory. Then, we enforce:

$$\begin{array}{l} q_0^{\diamondsuit\psi_2} = false \\ loopexists \Rightarrow (b_L^{\psi_1\ \mathcal{U}\ \psi_2} \Rightarrow q_L^{\diamondsuit\psi_2}) \\ \forall t \in \{1, \ldots, L\}: \\ \qquad q_t^{\diamondsuit\psi_2} = q_{t-1}^{\diamondsuit\psi_2} \vee (inloop_t \wedge b_t^{\psi_2}) \end{array} \tag{IV.4}$$

The constraints ensure that if a loop exists, then $q_L^{\diamondsuit\psi_2}$ is the evaluation of the formula $\diamondsuit\psi_2$ at the L-th location of the trajectory. The conjunction of the formula constraints obtained as in (IV.1), (IV.2), (IV.3), and (IV.4) forms the $|[LTL]|$ formula.

3) Loop Constraints: The loop constraints are used to non-deterministically decide the value of k for the (L,k)-loop. For this purpose, $L+1$ fresh variables l_0, ..., l_L are introduced as loop selector variables. At most one of these variables can be true. If l_j is true then $\rho_L = \rho_{j-1}$ and for all $\varphi \in cl(\psi)$, $b_L^{\varphi} = b_{j-1}^{\varphi}$. In this case the bounded trajectory is a (L,j)-loop. If none of the variables is true then the LTL formula can be satisfied using a finite length trajectory. The loop constraints are encoded as the conjunction of the constraints given below:

$$\begin{array}{l} l_0 \Leftrightarrow false,\ inloop_0 \Leftrightarrow false \\ \forall t \in \{1, \ldots, L\}: \\ \qquad inloop_t \Leftrightarrow inloop_{t-1} \vee l_t \\ \qquad inloop_{t-1} \Rightarrow \neg l_t \\ loopexists \Leftrightarrow inloop_L. \end{array} \tag{IV.5}$$

Maintaining region consistency: The region occupied by each robot at the L-th time instance is equal to the region of the robot at the time instance before the loop starts, i.e.,

$$\forall t \in \{1, \ldots, L\}:\ l_t \Rightarrow (\rho_L = \rho_{t-1}). \tag{IV.6}$$

Maintaining formula consistency: For each subformula $\varphi \in cl(\psi)$, we ensure that their evaluation at the $(L+1)$-th location of the trajectory is equivalent to their evaluation at the location where the loop starts. In the no-loop case, the evaluation of the sub formulas at the $(L+1)$-th location is $false$.

$$\begin{array}{l} \forall t \in \{1, \ldots, L\} : l_t \Rightarrow (b_{L+1}^{\varphi} = b_t^{\varphi}) \\ \neg loopexists \Rightarrow (b_{L+1}^{\varphi} = false) \end{array} \tag{IV.7}$$

The conjunction of constraints (IV.5), (IV.6), and (IV.7) provides the formula $|[LOOP]|$.

4) Full Discrete Problem Encoding: The full encoding of the discrete portion of the problem is denoted by $|[\mathcal{P}, L]|_D$, and is given by the conjunction of the above three sets of constraints and the constraint that ensures that the LTL formula ψ holds in the initial state:

$$|[\mathcal{P}, L]|_D \Leftrightarrow |[\mathcal{W}]| \wedge |[LTL]| \wedge |[LOOP]| \wedge b_0^{\psi}$$

B. Encoding the Low-Level Motion Planning Problem

A valid trajectory must satisfy a set of dynamic, input, and state constraints, as well as collision avoidance constraints. We encode them via a conjunction of hybrid constraints including Boolean variables, as well as convex constraints on the reals.

Dynamics, State, and Input Constraints. We enforce that valid trajectories progress according to the robots' dynamics with the conjunction of the following constraints:

$$\begin{array}{ll} x_{t+1}^i = A_i x_t^i + B_i u_t^i, & \forall t \in \{0, \ldots, L-1\}, \forall i \in \{1, \ldots, N\} \\ x_0^i = \overline{x}_0^i, & \forall i \in \{1, \ldots, N\} \\ \|x_t^i\| \leq \overline{x}^i & \forall t \in \{0, \ldots, L-1\}, \forall i \in \{1, \ldots, N\} \\ \|u_t^i\| \leq \overline{u}^i & \forall t \in \{0, \ldots, L-1\}, \forall i \in \{1, \ldots, N\} \end{array} \tag{IV.8}$$

Maintaining Consistency Between Regions and States. The state of each robot must be consistent with the workspace region occupied at each time. Since each workspace region $\mathcal{W}_j$ is a polyhedron, it can be captured by an affine inequality of the form $(P_j w + q_i \leq 0)$. We therefore obtain, $\forall i \in \{1, \ldots, N\}$:

$$\begin{array}{ll} \pi_{jt}^i \Rightarrow \Big(P_j h_{\mathcal{X}\rightarrow\mathcal{W}}(x_t^i) + q_j \leq 0\Big), & \forall t \in \{0, \ldots, L\}, j \in \{1, \ldots, r\} \\ l_t \Rightarrow (x_L^i = x_{t-1}^i), & \forall t \in \{1, \ldots, L\}, \end{array} \tag{IV.9}$$

where $h_{\mathcal{X}\rightarrow\mathcal{W}}(.)$, the natural projection of the state space onto the workspace, is also an affine function. We also require that, if there is a loop, the state of each robot at time L is identical to its state before the loop starts. While this is only a sufficient condition for the existence of a continuous trajectory that is consistent with the discrete plan, it can be shown from reachability analysis that this condition becomes necessary under some technical assumptions on the robot dynamics [21].

Collision Avoidance Constraints. For each pair of robots at each time and a workspace of dimension w, we create pairs of fresh Boolean variables $\{(c_{kt}^{ij}, d_{kt}^{ij}) | t \in \{0, \dots, L\}, k \in \{1, \dots, w\}, i, j \in \{1, \dots, N\}, i \neq j\}$ and encode the collision avoidance conditions via the conjunction of the following constraints:

$$\forall i, j \in \{1, \dots, N\}, i \neq j, \forall t \in \{0, \dots, L\}:$$

$$c_{kt}^{ij} \Rightarrow (h_{\mathcal{X}\to\mathcal{W}}^{k}(x_t^i) - h_{\mathcal{X}\to\mathcal{W}}^{k}(x_t^j)) \geq \epsilon, \forall k \in \{1, \dots, w\}$$
$$d_{kt}^{ij} \Rightarrow (-h_{\mathcal{X}\to\mathcal{W}}^{k}(x_t^i) + h_{\mathcal{X}\to\mathcal{W}}^{k}(x_t^j)) \geq \epsilon, \forall k \in \{1, \dots, w\}$$
$$\sum_{k=1}^{w} (c_{kt}^{ij} + d_{kt}^{ij}) \geq 1, \qquad \text{(IV.10)}$$

where $h_{\mathcal{X}\to\mathcal{W}}^{k}(.)$ is the natural projection of the state space onto the k-th dimension of the workspace.

The conjunction of the sets of constraints (IV.8), (IV.9), and (IV.10), denoted as $|[\mathcal{P}, L]|_C$, is conjoined with the formula $|[\mathcal{P}, L]|_D$ for the discrete plan to provide the overall formula $|[\mathcal{P}, L]|$ encoding the motion planning problem in this paper:

$$|[\mathcal{P}, L]| \Leftrightarrow |[\mathcal{P}, L]|_D \wedge |[\mathcal{P}, L]|_C. \qquad \text{(IV.11)}$$

The following result states that $|[\mathcal{P}, L]|$ can be efficiently solved by combining SAT solving, convex programming, and conflict-driven learning techniques, as shown in Algorithm 1, since it falls into the category of monotone SMC formulas.

Proposition 4.1 (Monotone SMC-Based Encoding): Given a multi-robot motion planning problem instance $\mathcal{P}$ and a finite horizon L, let $|[\mathcal{P}, L]|$ be the formula obtained in (IV.11). $|[\mathcal{P}, L]|$ is a monotone SMC formula, hence its satisfiability problem can be cast as the feasibility problem for a finite disjunction of convex programs, and solved as shown in Algorithm 1.

Proof: The formula $|[\mathcal{P}, L]|$ is expressed in conjunctive normal form, where clauses can be pseudo-Boolean predicates or disjunctions of literals, and literals can be Boolean variables or convex constraints. By inspection of all its clauses, $|[\mathcal{P}, L]|$ complies with the formal syntax of SMC formulas [16, Definition 3.1]. By the properties of SMC formulas [16, Proposition 3.5], the satisfiability problem for $|[\mathcal{P}, L]|$ can be solved as the feasibility problem for a finite disjunction of convex programs. Finally, Algorithm 1, for a fixed horizon L, turns into the SMC decision procedure [16, Algorithm 1] and can be used to solve the satisfiability problem for $|[\mathcal{P}, L]|$ [16, Proposition 4.1 and 5.2]. ■

We can finally state the formal guarantees of Algorithm 1.

Theorem 4.2 (Correctness of Algorithm 1): Given a multi-robot motion planning problem instance $\mathcal{P}$, Algorithm 1, leveraging the SMC-based encoding $|[\mathcal{P}, L]|$ in (IV.11), is sound.

Proof: Termination and soundness are guaranteed by the correctness of the SMC decision procedure [16, Proposition 4.1 and 5.2] and by the fact that the purely Boolean encoding for the system trajectories, based on the eventuality encoding, is sound and complete [20, Theorem 3.2]. However, as discussed above, $|[\mathcal{P}, L]|$ offers only a sufficient encoding for the multi-robot motion planning problem $\mathcal{P}$ and horizon length L. Any satisfying assignment for $|[\mathcal{P}, L]|$ is an L-horizon valid trajectory for $\mathcal{P}$. On the other hand, the proposed encoding cannot guarantee completeness because of the condition on the states in (IV.9) in the presence of a loop. ■

V. RESULTS

We implemented Algorithm 1 in PYTHON on top of the SATEX solver [16], using Z3 [22] as a SAT solver and

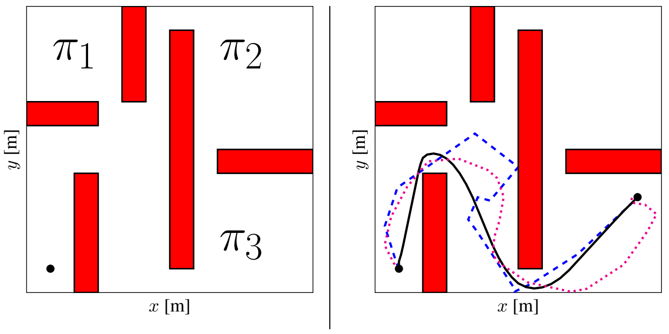

Fig. 2. (Left) Workspace and propositions used in our experiments; (right) trajectories generated by the SMC-based (black), Synergistic RRT (dashed blue), and Synergistic EST (dotted red) motion planners for the double integrator dynamics.

CPLEX [23] as a convex optimization solver. We generate infeasibility certificates that are minimal by providing, at each iteration, an Irreducibly Inconsistent Set (IIS) of linear constraints [24]. Moreover, in each convex program, we instruct SATEX to search for a continuous trajectory that minimizes the ℓ_1-norm of the overall control "effort" $\sum_{0 \leq t \leq L, 1 \leq i \leq N} \|u_t^i\|_1$, over all robots at all times, among the trajectories compatible with the discrete plan from the SAT solver. All the experiments were executed on an Intel Core i7 3.4-GHz processor with 16 GB of memory.

A. Single-Robot Reach-Avoid Specification

As a first scenario to validate our approach, we consider a single robot subject to a reach-avoid specification, an essential motion planning problem, which is embedded in almost all robotics applications. A single-robot scenario also allows comparing against alternative sampling-based algorithms which do not support, as yet, multi-robot formulations. We consider the workspace represented on the left side of Figure 2, where the black dot marks the initial position of the robot. We compare the performance of our algorithm against state-of-the-art sampling-based techniques implemented in Syclop (Synergistic Combination of Layers Of Planning), which have been shown to outperform traditional sampling-based algorithms by orders of magnitude [25]. Syclop, available from the Open Motion Planning Library (OMPL)[2], is a meta-planner that combines a high-level guide computed over a decomposition of the state space with a low-level planning algorithm. The progress that the low-level planner makes is fed back to the high-level planner which uses this information to update the guide. We consider two versions, namely, Syclop RRT (Rapidly-exploring Random Trees) and Syclop EST (Expansive Space Trees) using, respectively, the RRT and EST algorithms as their low-level planners. We also compare with state-of-the-art SMT solvers supporting nonlinear constraints on the reals, namely, Z3 and DREAL [26], when directly applied to the monotone SMC formula encoding the motion planning problem. MILP-based techniques are not considered in this paper, since the SMC-based planner has already been shown to scale better on similar problems [15, 16].

We consider robot dynamics captured by chains of integrators, one chain for each coordinate of the workspace, and a sampling time of 0.5 s. The robot starts at the point with coordinates $(0.5, 0.5)$ (in meters) and is required to reach the

[2]https://ompl.kavrakilab.org/planners.html

TABLE I
EXECUTION TIME OF DIFFERENT MOTION PLANNING ALGORITHMS AS A FUNCTION OF THE NUMBER OF CONTINUOUS STATES FOR THE WORKSPACE IN FIG. 2. RESULTS ARE AVERAGED ACROSS 20 TRIALS. TIMEOUT IS SET TO 1 HOUR.

Number of States	**SMC-Based [s]**	**Synergistic RRT [s]**	**Synergistic EST [s]**	**dReal [s]**	**Z3 [s]**
4	3.007	33.166	0.6151	time out	time out
6	4.590	3216.402	791.444	time out	time out
8	7.502	time out	time out	time out	time out
10	10.207	time out	time out	time out	time out
12	34.775	time out	time out	time out	time out
14	60.413	time out	time out	time out	time out
16	39.070	time out	time out	time out	time out
18	70.631	time out	time out	time out	time out
20	75.843	time out	time out	time out	time out

TABLE II
EXECUTION TIME OF THE SMC-BASED MOTION PLANNER VERSUS SYCLOP LTL. TIMEOUT IS SET TO 15 MINUTES.

Number of States	**Number of Propositions**	**SMC-Based [s]**	**Synergistic LTL [s]**
4	1	6.6614	2.4978
	2	16.0456	44.3815
	3	36.651	153.389
6	1	25.2670	13.003
	2	54.517	timeout
	3	73.0913	timeout
8	1	6.280	timeout
	2	26.385	timeout
	3	225.255	timeout

point $(5.5, 2.0)$, while higher order derivatives are set to 0 both at the initial and target points. The corresponding LTL formula is $\psi_1 := (\bigwedge_j \neg\eta_j)\ \mathcal{U}\ \gamma$, where η_j, $j \in \{1, \ldots, o\}$, are propositions associated with the obstacles and γ is the proposition associated with the goal region in the workspace. The upper bound on the control input is $\overline{u} = 0.2$, in appropriate units based on the number of integrators in the chain. Table I reports the execution times of different algorithms as the number of integrators in the chain, hence the number of state variables, increases. Times are averaged over 20 trials. RRT and EST-based planners show much higher variability in execution time than the SMC-based planner, as is expected because of their randomized search schemes. Syclop EST performs better for a small number of continuous states, but its runtime rapidly increases and reaches a 1-hour timeout for a chain of four integrators. Our algorithm scales better over the whole range of continuous states scoring more than one order of magnitude reduction in computation time. Moreover, the generated trajectory is usually smoother because of the ℓ_1-norm minimization. Z3 and DREAL exceed the timeout threshold in all the experiments.

B. Single Robot Under LTL Specifications

While our encoding supports generic LTL specifications, in this scenario, we only focus on co-safe LTL formulas, since this is the only fragment supported by the Syclop LTL motion planner version. We consider the same workspace and initial condition as in Figure 2 under LTL formulas of the form $\psi_2 := \bigwedge_i \Diamond\pi_i \bigwedge_j \Box\neg\eta_j$, where η_j, $j \in \{1, \ldots, o\}$, are propositions associated with the obstacles in the workspace. The robot must visit a set of regions, in arbitrary order, while avoiding obstacles.

Table II reports the execution times of our algorithm and Syclop LTL as both the number of chained integrators capturing the robot dynamics, hence the number of continuous variables, and the number of regions to be visited, hence the number of Boolean variables in the specification, increase.

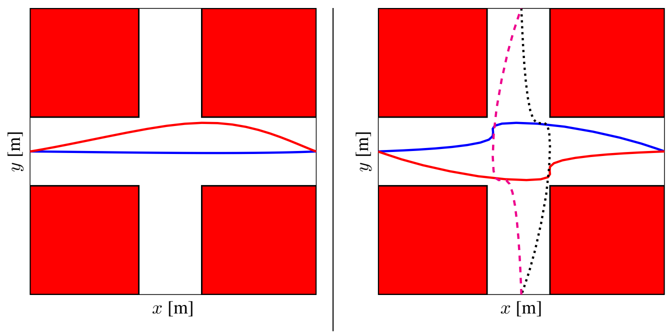

Fig. 3. Workspace and trajectories under reach avoid specifications for a 2-robot scenario (left) and a 4-robot scenario (right).

TABLE III
PERFORMANCE OF THE SMC-BASED MOTION PLANNER AND SIZE OF THE PROBLEM IN MULTI-ROBOT SCENARIOS WITH REACH-AVOID SPECIFICATION AND SPECIFICATION ψ_3. TIMEOUT IS SET TO 30 MINUTES.

Number of Robots	**Number of States (per robot)**	**SMC-Based** Reach-avoid specification			**SMC-Based** $\psi_3 := (\Box\Diamond\sigma_1) \wedge (\Box\Diamond\sigma_2) \wedge (\Box\Diamond\sigma_3)$		
		time [s]	#real vars	#Boolean vars	time [s]	#real vars	#Boolean vars
2	4	0.3346	336	66	19.269	960	2370
	6	0.822	420		44.72625	1200	
	8	1.0625	504		67.6701	1440	
	10	0.915	588		76.3877	1680	
	12	2.444	672		665.4057	1920	
3	4	0.7170	504	117	105.661	1440	3449
	6	2.1074	630		196.425	1800	
	8	3.8263	756		253.077	2160	
	10	15.005	882		1151.087	2520	
	12	8.654	1008		466.6257	2880	
4	4	0.9621	672	180	444.354	1920	4648
	6	5.1138	840		829.665	2400	
	8	6.3493	1008		986.9366	2880	
	10	44.4658	1176		timeout	3360	
	12	80.0632	1344		timeout	3840	
5	4	5.8121	840	255	1334.822	2400	5967
6	4	26.4051	1008	342	timeout	2880	7406
7	4	142.896	1008	441	timeout	3360	8965
8	4	1229.5425	1344	552	timeout	3840	10644

Results are averaged over 10 trials. Syclop exceeds the 15-minute timeout threshold for a system with 3 integrators and and an LTL specification including two regions (i=2). In the case of one region to be visited for a 3-integrator chain, Syclop reaches the timeout value in 4 runs; we then report the average time over the remaining 6 runs.

C. Multi-Robot Scenarios

We first show the effectiveness of the proposed collision avoidance encoding on the workspace in Figure 3, where we force the robots to "cross" each other in the same region as they move from their initial positions to their targets subject to reach-avoid specifications. Table III reports the performance of our motion planner as the number of robots (hence the number of Boolean variables in the problem) and the number of integrators (hence the continuous states in the problem) increase. Trajectories for a 2-robot and a 4-robot scenario are visualized, respectively, on the left and right sides of Figure 3, illustrating the satisfaction of the collision avoidance constraints with a safety margin $\epsilon = 0.2$ m.

We finally demonstrate the capabilities of our algorithm in a multi-robot scenario under generic LTL specifications. We consider the workspace in Figure 2, an even number of robots N, and the LTL formula $\psi_3 := (\Box\Diamond\sigma_1) \wedge (\Box\Diamond\sigma_2) \wedge (\Box\Diamond\sigma_3)$, where $\sigma_1 := \sum_{i=1}^{N} \pi_1^i = N$, $\sigma_2 := \sum_{i=1}^{N/2} \pi_2^i = N/2$, and $\sigma_3 := \sum_{i=N/2+1}^{N} \pi_3^i = N/2$. In words, we require the robots to visit, all together, region 1 infinitely often. Similarly, a first half of robots must also visit region 2, while the second half must visit region 3, all together, infinitely often. Again, we report in Table III the performance of our motion planner as the number of robots and chained integrators increase together

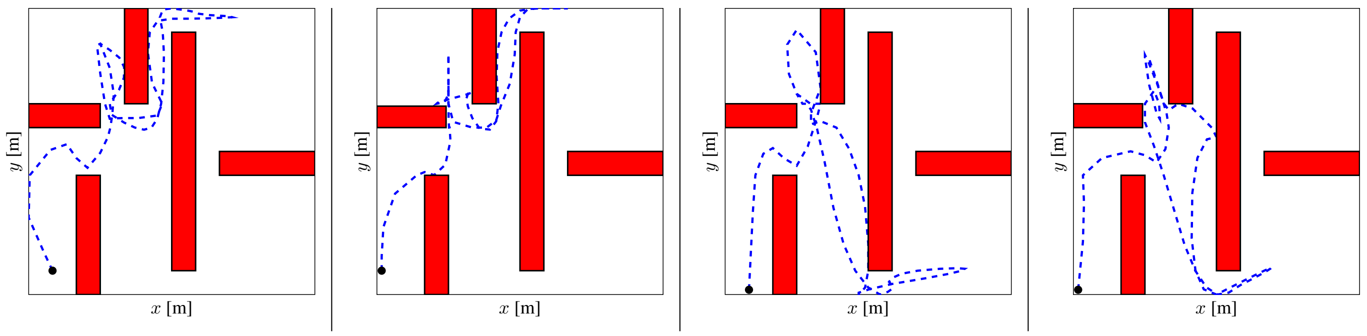

Fig. 4. Trajectories of robots R_1, R_2, R_3, and R_4 (from left to right), subject to $\psi_3 := \left(\Box\Diamond \sum_{i=1}^{4} \pi_1^i = 4\right) \wedge \left(\Box\Diamond \sum_{i=1}^{2} \pi_2^i = 2\right) \wedge \left(\Box\Diamond \sum_{i=3}^{4} \pi_3^i = 2\right)$. The trajectories of R_1 and R_2 visit region 2 while the ones of R_3 and R_4 touch region 3 as specified.

with the problem size in terms of number of Boolean and real variables. The trajectories for the 4-robot scenario are separately shown in Figure 4.

VI. Conclusions

We presented an efficient SMC-based algorithm for multi-robot motion planning from generic Linear Temporal Logic (LTL) specifications, under the assumption of discrete-time, linear dynamics and workspaces described by unions of polyhedra. Our algorithm is more than one order of magnitude faster than state-of-the-art sampling-based techniques for high-dimensional state spaces, while supporting complex missions. Future work includes investigating further strategies for efficient generation of infeasibility certificates, and extending the proposed techniques to planning in the presence of uncertainties in the dynamics and bounded disturbances.

References

[1] A. Pnueli, “The temporal logic of programs,” in *FOCS*, 1977, pp. 46–57.

[2] P. Tabuada and G. J. Pappas, “Linear time logic control of discrete-time linear systems,” *IEEE Trans. Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[3] M. Kloetzer and C. Belta, “A fully automated framework for control of linear systems from temporal logic specifications,” *IEEE Trans. Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[4] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for dynamic robots,” *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, “Receding horizon temporal logic planning,” *IEEE Trans. Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[6] P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia, “A contract-based methodology for aircraft electric power system design,” *IEEE Access*, vol. 2, pp. 1–25, 2014.

[7] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, “Automated composition of motion primitives for multi-robot systems from safe LTL specifications,” in *Int. Conf. Intelligent Robots and Systems*, 2014, pp. 1525–1532.

[8] E. Plaku and S. Karaman, “Motion planning with temporal-logic specifications: Progress and challenges,” *AI Communications*, no. Preprint, pp. 1–12.

[9] M. Rungger, M. Mazo Jr, and P. Tabuada, “Specification-guided controller synthesis for linear systems and safe linear-time temporal logic,” in *Proc. Int. Conf. Hybrid Systems: Computation and Control*, 2013, pp. 333–342.

[10] S. Karaman and E. Frazzoli, “Linear temporal logic vehicle routing with applications to multi-UAV mission planning,” *Int. J. Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

[11] E. M. Wolff, U. Topcu, and R. M. Murray, “Optimization-based trajectory generation with linear temporal logic specifications,” in *IEEE Int. Conf. Robotics and Automation*, 2014, pp. 5319–5325.

[12] S. Karaman and E. Frazzoli, “Sampling-based motion planning with deterministic μ-calculus specifications,” in *Proc. IEEE Conf. Decision and Control*, 2009, pp. 2222–2229.

[13] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, “Sampling-based motion planning with temporal goals,” in *IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2689–2696.

[14] L. Zhang and D. Manocha, “An efficient retraction-based RRT planner,” in *IEEE Int. Conf. Robotics and Automation*, 2008, pp. 3743–3750.

[15] Y. Shoukry, P. Nuzzo, I. Saha, A. Sangiovanni-Vincentelli, S. Seshia, G. Pappas, and P. Tabuada, “Scalable lazy SMT-based motion planning,” in *Proc. IEEE Conf. Decision and Control*, 2016, pp. 6683–6688.

[16] Y. Shoukry, P. Nuzzo, A. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, “SMC: Satisfiability modulo convex optimization,” in *Proc. Int. Conf. Hybrid Systems: Computation and Control*, Apr. 2017.

[17] P. Nuzzo, A. Puggelli, S. A. Seshia, and A. Sangiovanni-Vincentelli, “CalCS: SMT solving for non-linear convex constraints,” in *Int. Conf. Formal Methods in Computer-Aided Design*, Oct. 2010, pp. 71–79.

[18] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, *Satisfiability Modulo Theories, Chapter in Handbook of Satisfiability*. IOS Press, 2009.

[19] T. A. Henzinger, R. Jhala, and R. Majumdar, “Counterexample-guided control,” in *Proc. Int. Colloquium on Automata, Languages and Programming*, 2003, pp. 886–902.

[20] A. Biere, K. Heljanko, T. Junttila, T. latvala, and V. Schuppan, “Linear encoding of bounded LTL model checking,” *Logical Methods in Computer Science*, vol. 2, no. 5:5, pp. 1–64, 2006.

[21] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[22] L. De Moura and N. Björner, “Z3: An efficient SMT solver,” in *Proc. Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[23] (2012, Feb.) IBM ILOG CPLEX Optimizer. [Online]. Available: www.ibm.com/software/integration/optimization/cplex-optimizer/

[24] J. W. Chinneck and E. W. Dravnieks, “Locating minimal infeasible constraint sets in linear programs,” *ORSA Journal on Computing*, vol. 3, no. 2, pp. 157–168, 1991.

[25] E. Plaku, L. E. Kavraki, and M. Y. Vardi, “Motion planning with dynamics by a synergistic combination of layers of planning,” *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, Jun. 2010.

[26] S. Gao, S. Kong, and E. M. Clarke, “dReal: An SMT solver for nonlinear theories over the reals,” in *Int. Conf. Automated Deduction*, 2013, vol. 7898, pp. 208–214.