2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Optimal Temporal Logic Planning for Multi-Robot Systems in Uncertain Semantic Maps

Yiannis Kantaros, and George J. Pappas

Abstract— **This paper addresses a multi-robot motion planning problem in probabilistic maps obtained by semantic simultaneous localization and mapping (SLAM). The goal of the robots is to accomplish complex collaborative high level tasks captured by global temporal logic specifications in the presence of uncertainty in the workspace. Specifically, the robots operate in an unknown environment modeled as a semantic map determined by Gaussian distributions over landmark positions and arbitrary discrete distributions over landmark classes. We extend Linear Temporal Logic by including information-based predicates allowing us to incorporate uncertainty and probabilistic satisfaction requirements directly into the task specification. We propose a new highly scalable sampling-based approach that synthesizes paths that satisfy the assigned task specification while minimizing a user-specified motion cost function. Finally, we show that the proposed algorithm is probabilistically complete, asymptotically optimal and supported by convergence rate bounds. We provide extensive simulation results that corroborate the theoretical analysis and show that the proposed algorithm can address large-scale planning tasks.**

I. Introduction

This paper addresses a multi-robot motion planning problem in uncertain environments with collaborative tasks specified by global temporal logic formulas. The uncertain environment is modeled as a map distribution, obtained from a semantic simultaneous localization and mapping (SLAM) algorithm [1], while the task is specified over the uncertain map in terms of landmark positions and labels. Given a semantic map distribution, the goal of this paper is to design multi-robot control policies that accomplish collaborative tasks with user-specified probabilistic satisfaction requirements in the presence of uncertainty in the environment.

Control in the presence of mapping or localization uncertainty typically gives rise to stochastic optimal control problems with partial observability. To avoid the need of computationally expensive approaches that allow for control in belief space [2]–[5], we extend Linear Temporal Logic (LTL) by including information-based predicates which allows us to incorporate uncertainty and probabilistic satisfaction requirements directly into the task specification. This gives rise to a deterministic optimal control problem with temporal logic constraints. To solve this problem, we propose a sampling-based approach that explores both the robot motion space and the state space of an automaton that corresponds to the temporal logic (TL) specification. Building upon our previous work [6], we design biased sampling functions that allow us to address large-scale planning tasks, as shown by extensive numerical experiments. Finally, we show that the proposed algorithm is probabilistically complete, asymptotically optimal, and supported by convergence rate bounds. Sampling-based approaches for temporal logic planning are presented in [6]–[9], as well, but they consider *known* environments.

Temporal logic control synthesis in the presence of uncertainty has recently received increasing research attention. Uncertainty in sensing and actuation has been studied, see e.g., [10]–[13] while the proposed approaches typically rely on probabilistic model checking methods to synthesize controllers that maximize the satisfaction probability. Uncertainty in the workspace in terms of incomplete (or dynamic) environment models is also considered in [14]–[16]. In these works, the environment and robot mobility are captured by transition systems and nominal controllers are revised as the environments, i.e., the transition systems, are updated. To the contrary in this work, we propose an abstraction-free approach for temporal logic planning in uncertain environments that are modeled using probabilistic semantic maps obtained by available SLAM algorithms. Semantic maps consist of Gaussian distributions over the landmark positions and discrete distributions over the landmark labels. Motion planning in probabilistic semantic maps is also considered in [17]. In particular, [17] considers sequencing tasks captured by co-safe LTL formulas under the assumption that the labels of the landmarks are known. In contrast, this work considers task specifications captured by arbitrary co-safe temporal logic formulas with uncertain landmark labels. Moreover, we show that the proposed sampling-based algorithm can be applied to large-scale planning tasks that involve large workspaces and number of robots. Finally, we provide formal optimality and convergence rate guarantees that do not exist in [17].

Contribution: The contribution of this paper can be summarized as follows. *First*, we propose an abstraction-free sampling-based approach for complex tasks specified by co-safe TL formulas for multi-robot systems that reside in unknown environments modeled as probabilistic semantic maps. *Second*, the proposed algorithm is highly scalable, i.e., it can quickly design control policies which satisfy desired task specifications for large-scale planning tasks that involve large robot teams and large workspaces. *Third*, we show that the proposed algorithm is probabilistically complete, asymptotically optimal, and converges exponentially fast to

The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, 19104, USA. {kantaros,pappasg}@seas.upenn.edu. This material is based upon work supported by the AFRL and DARPA under Contract No. FA8750-18-C-0090. This work was supported in part by the ARL RCTA under Contract No. W911NF-10-2-0016.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.

the optimal solution.

II. Problem Definition

Consider N mobile robots governed by the following dynamics: $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t))$, for all $j \in \mathcal{N} := \{1, \dots, N\}$, where $\mathbf{p}_j(t) \in \Omega \subseteq \mathbb{R}^n$ stands for the state (e.g., position and orientation) of robot j in an environment Ω at discrete time t, $\mathbf{u}_j(t) \in \mathcal{U}_j$ stands for a control input in a *finite* space of admissible controls $\mathcal{U}_j$. Hereafter, we compactly denote the dynamics of all robots as $\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t))$, where $\mathbf{p}(t) \in \Omega^N$, $\forall t \geq 0$, and $\mathbf{u}(t) \in \mathcal{U} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$.

The robots operate in a environment modeled by a semantic map $\mathcal{M} = \{\ell_1, \ell_2, \dots, \ell_M\}$ consisting of M landmarks. Each landmark $\ell_i = \{\mathbf{x}_i, c_i\} \in \mathcal{M}$ is defined by its position $\mathbf{x}_i \in \Omega$ and its class $c_i \in \mathcal{C}$, where $\mathcal{C}$ is a finite set of classes. The robots do not know the true landmark positions but instead they have access to a probability distribution $\mathcal{P}$ over the space of all possible maps. Such a distribution can be produced by a semantic SLAM algorithm and typically consists of a Gaussian distribution over the landmark positions and a discrete distribution over the landmark classes. Specifically, we assume that $\mathcal{P}$ is determined by parameters $(\hat{\mathbf{x}}_i, \Sigma_i, d_{c_i})$, for all landmarks ℓ_i, such that $\mathbf{x}_i \sim \mathcal{N}(\hat{\mathbf{x}}_i, \Sigma_i)$ and the class c_i is determined by a discrete distribution d_{c_i}. Hereafter, we compactly denote by $(\hat{\mathbf{x}}, \Sigma, d_c)$ the parameters that determine $\mathcal{P}$.

The goal of the robots is to accomplish a complex collaborative task captured by a global temporal logic specification ϕ. To define tasks in probabilistic semantic maps we extend Linear Temporal Logic (LTL) by including information/uncertainty-based predicates. The syntax of this specification language can be defined follows.

$$\phi ::= \text{true} \mid \pi \mid g \geq 0 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \; \mathcal{U} \; \phi_2 \quad (1)$$

where π is a predicate over the robot state $\mathbf{x}(t)$ and/or the information space of the landmarks, and $g \geq 0$ is a predicate over the robot state $\mathbf{x}(t)$ and the map distribution $\mathcal{P}$, where $g : \Omega^N \times \mathcal{P} \to \mathbb{R}$.

The operators disjunction $\vee$, eventually $\Diamond$, and always $\Box$, are defined as in [18]. Examples of atomic predicates π defined over the robot states $\mathbf{p}(t)$ and the information space include

$$\pi_j(\mathcal{R}) : \mathbf{p}_j \in \mathcal{R} \subseteq \Omega, \quad (2a)$$
$$\pi_{\ell_i}(\Sigma_i, \zeta) : \det \Sigma_i \leq \zeta. \quad (2b)$$

In (2a), $\pi_j(\mathcal{R})$ is true only if robot j is within a region $\mathcal{R}$ of the workspace Ω. Similarly, in (2b), $\pi_{\ell_i}(\Sigma_i, \zeta)$ is true if the uncertainty of the position of landmark ℓ_i, captured by the determinant of the covariance matrix Σ_i, denoted by $\det \Sigma_i$, is less than a user-specified parameter $\zeta \geq 0$.

Moreover, given a map distribution $\mathcal{P}$, examples of atomic predicates $g \geq 0$ include

$$g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r, \delta) \geq 0 : \mathbb{P}(||\mathbf{p}_j - \mathbf{x}_i|| \leq r) \geq 1 - \delta, \quad (3a)$$
$$g_c(\mathbf{p}_j, r, \delta, \bar{\mathcal{C}}) \geq 0 : \mathbb{P}(||\mathbf{p}_j - \mathbf{x}_i|| \leq r) \geq 1 - \delta, c_i \in \bar{\mathcal{C}} \subseteq \mathcal{C} \quad (3b)$$

The atomic proposition in (3a) is true if the probability of robot j being within distance less than r from landmark ℓ_i is greater than $1 - \delta$, for some user-specified parameters $r, \delta > 0$. Similarly, the atomic predicate in (3b) is true if the probability of robot j being within distance less than r from at least one landmark ℓ_i with class $c_i \in \bar{\mathcal{C}}$ is greater than $1 - \delta$, for some user-specified parameters $r, \delta > 0$ and subset of classes $\bar{\mathcal{C}} \subseteq \mathcal{C}$. Given an infinite sequence of robot states $p = [\mathbf{p}(0), \mathcal{P}][\mathbf{p}(1), \mathcal{P}] \dots$ and a semantic map $\mathcal{P}$, the semantics is

$$\begin{aligned}
&p \models \text{true}, \; p \models \pi \Leftrightarrow \mathbf{p}(0) \models \pi, \\
&p \models g \geq 0 \Leftrightarrow g(\mathbf{p}(0)) \geq 0, \; p \models \neg\phi \Leftrightarrow \neg(p \models \phi), \\
&p \models \phi_1 \wedge \phi_2 \Leftrightarrow (p \models \phi_1) \wedge (p \models \phi_2), \\
&p \models \phi_1 \mathcal{U} \phi_2 \Leftrightarrow \exists \; t \geq 0 \text{ s.t. } (\mathbf{p}(t)\mathbf{p}(t+1) \dots \models \phi_2), \\
&\qquad\qquad \wedge (\mathbf{p}(t')\mathbf{p}(t'+1) \dots \models \phi_1), \; \forall 0 \leq t' < t.
\end{aligned}$$

Hereafter we assume that the assigned task ϕ is expressed as a *co-safe* Temporal Logic (TL) formula defined over a set of atomic propositions $\mathcal{AP}$, which is satisfied by a finite-length robot trajectory. In order to interpret a temporal logic formula over the trajectories of the robot system, we use a labeling function $L : \Omega^N \times \mathcal{P} \to 2^{\mathcal{AP}}$ that determines which atomic propositions are true given the current robot system state $\mathbf{p}(t) \in \Omega^N$ and the distribution of maps $\mathcal{P}$.

Given a task specification ϕ, our goal is to select a stopping time H and a sequence $\mathbf{u}_{0:H}$ of control inputs $\mathbf{u}(t)$, for all $t \in \{0, \dots, H\}$, that maximizes the probability of satisfying ϕ while minimizing a motion cost. This gives rise to the following optimal control problem:

$$\min_{H, \mathbf{u}_{0:H}} \left[J(H, \mathbf{u}_{0:H}) = \sum_{t=0}^{H} w(\mathbf{p}(t), \mathbf{p}(t+1)) \right] \quad (4a)$$
$$\mathbf{p}_{0:H} \models \phi \quad (4b)$$
$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)). \quad (4c)$$

where $w(\mathbf{p}(t), \mathbf{p}(t+1)) \geq 0$ is a motion cost function that captures e.g., the energy consumption or the distance required to travel from $\mathbf{p}(t)$ to $\mathbf{p}(t+1)$. Then, the problem addressed in this paper can be summarized as follows:

Problem 1: Given an initial robot configuration $\mathbf{p}(0)$, a distribution of maps $\mathcal{P}$, and a co-safe TL task ϕ, select a horizon H and compute control inputs $\mathbf{u}(t)$ for all time instants $t \in \{0, \dots, H\}$ as per (4).

III. Planning In Probabilistic Semantic Maps

We propose a sampling-based algorithm to solve Problem 1, which is summarized in Algorithm 1. First, we translate the specification ϕ constructed using a set of atomic predicates $\mathcal{AP}$ into Deterministic Finite state Automaton (DFA), defined as follows [18] [line 1, Alg. 1].

Definition 3.1 (DFA): A Deterministic Finite state Automaton (DFA) D over $\Sigma = 2^{\mathcal{AP}}$ is defined as a tuple $D = \left(\mathcal{Q}_D, q_D^0, \delta_D, q_F\right)$, where $\mathcal{Q}_D$ is the set of states, $q_D^0 \in \mathcal{Q}_D$ is the initial state, $\delta_D : \mathcal{Q}_D \times \Sigma \to \mathcal{Q}_D$ is

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Mission Planning in Probabilistic Maps

Input: (i) maximum number of iterations $n_{\max}$, (ii) robot dynamics, (iii) map distribution $\mathcal{P}$, (iv) initial robot configuration $\mathbf{p}(0)$, (v) TL formula ϕ

Output: Terminal horizon H, and control inputs $\mathbf{u}_{0:H}$

1 Convert ϕ into a DFA;
2 Initialize $\mathcal{V} = \{\mathbf{q}(0)\}$, $\mathcal{E} = \emptyset$, $\mathcal{V}_1 = \{\mathbf{q}(0)\}$, $K_1 = 1$, and $\mathcal{X}_g = \emptyset$;
for $n = 1, \ldots, n_{max}$ **do**
3 Sample a subset $\mathcal{V}_{k_{\text{rand}}}$ from $f_{\mathcal{V}}$;
4 **for** $\mathbf{q}_{rand}(t) = [\mathbf{p}_{rand}(t), q_D^{rand}] \in \mathcal{V}_{k_{rand}}$ **do**
5 Sample a control input $\mathbf{u}_{\text{new}} \in \mathcal{U}$ from $f_{\mathcal{U}}$;
6 Compute $\mathbf{p}_{\text{new}}(t+1) = \mathbf{f}(\mathbf{p}_{\text{rand}}(t), \mathbf{u}_{\text{new}})$;
7 Compute $q_D^{\text{new}} = \delta_D(q_D^{\text{rand}}, L([\mathbf{p}_{\text{rand}}(t), \mathcal{P}]))$;
8 **if** $\exists q_D^{new}$ **then**
9 Construct $\mathbf{q}_{\text{new}}(t+1) = [\mathbf{p}_{\text{new}}(t+1), q_D^{\text{new}}]$;
10 Update set of nodes: $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{\text{new}}\}$;
11 Update set of edges: $\mathcal{E} = \mathcal{E} \cup \{(\mathbf{q}_{\text{rand}}, \mathbf{q}_{\text{new}})\}$;
12 Compute cost of new state: $J_{\mathcal{G}}(\mathbf{q}_{\text{new}}) = J_{\mathcal{G}}(\mathbf{q}_{\text{rand}}) + w(\mathbf{p}_{\text{rand}}, \mathbf{p}_{\text{new}})$;
13 **if** $q_D^{new} = q_F$ **then**
14 $\mathcal{X}_g = \mathcal{X}_g \cup \{\mathbf{q}_{\text{new}}\}$;
15 Update the sets $\mathcal{V}_k$;
16 Among all nodes in $\mathcal{X}_g$, find $\mathbf{q}_{\text{end}}(t_{\text{end}})$;
17 $H = t_{\text{end}}$ and recover $\mathbf{u}_{0:H}$ by computing the path $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \ldots, \mathbf{q}(t_{\text{end}})$;

a deterministic transition relation, and $q_F \in \mathcal{Q}_D$ is the accepting/final state.

Next, we define the accepting run of a DFA that will be used in Algorithm 1. A run of ρ_D of D over a finite word $\sigma = \sigma(1)\sigma(2)\ldots\sigma(K) \in (2^{\mathcal{AP}})^*$, is a sequence $\rho_D = q_D^0 q_D^1 q_D^2 \ldots, q_D^K$, where $\delta_D(q_D^k, \sigma(k)) = q_D^{k+1}$, $\forall k \in \mathbb{N}$. A run ρ_D is called *accepting* if $q_K = q_F$. A sequence of robot states $\mathbf{p}_{0:H}$ satisfies ϕ, i.e., $\mathbf{p}_{0:H} \models \phi$, if the word $\sigma = L([\mathbf{p}(0), \mathcal{P}])L([\mathbf{p}(1), \mathcal{P}])\ldots L([\mathbf{p}(H), \mathcal{P}])$ yields an accepting DFA run.

The proposed algorithm relies on incrementally constructing a directed tree that explores both the robot motion space and the state-space of the DFA that corresponds to ϕ. In what follows, we denote the constructed tree by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, J_{\mathcal{G}}\}$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The set of nodes $\mathcal{V}$ contains states of the form $\mathbf{q}(t) = [\mathbf{p}(t), q_D]$, where $\mathbf{p}(t) \in \Omega$ and $q_D \in \mathcal{Q}_D$.[1] The function $J_{\mathcal{G}} : \mathcal{V} \to \mathbb{R}_+$ assigns the cost of reaching node $\mathbf{q} \in \mathcal{V}$ from the root of the tree. The root of the tree, denoted by $\mathbf{q}(0)$, is constructed so that it matches the initial state of the robots $\mathbf{p}(0)$ and the initial state of the DFA D, i.e., $\mathbf{q}(0) = [\mathbf{p}(0), q_D^0]$. By convention the cost of the root $\mathbf{q}(0)$ is $J_{\mathcal{G}}(\mathbf{q}(0)) = 0$, while the cost of a node $\mathbf{q}(t+1) = [\mathbf{p}(t+1), q_D] \in \mathcal{V}$, given its parent node $\mathbf{q}(t) = [\mathbf{p}(t), q_D'] \in \mathcal{V}$, is computed as $J_{\mathcal{G}}(\mathbf{q}(t+1)) = J_{\mathcal{G}}(\mathbf{q}(t)) + w(\mathbf{p}(t), \mathbf{p}(t+1))$. Observe that by applying this equation recursively, we get that $J_{\mathcal{G}}(\mathbf{q}(t+1)) = J(t, \mathbf{u}_{0:t+1})$ which is the objective function in (4).

The tree $\mathcal{G}$ is initialized so that $\mathcal{V} = \{\mathbf{q}(0)\}$, $\mathcal{E} = \emptyset$, and $J_{\mathcal{G}}(\mathbf{q}(0)) = 0$ [line 2, Alg. 1]. Also, the tree is built incrementally by adding new states $\mathbf{q}_{\text{new}}$ to $\mathcal{V}$ and corresponding edges to $\mathcal{E}$, at every iteration n of Algorithm 1, based on a *sampling* [lines 3-5, Alg. 1] and *extending-the-tree* operation [lines 6-15, Alg. 1]. After taking $n_{\max} \geq 0$ samples, where $n_{\max}$ is user-specified, Algorithm 1 terminates and returns a solution to Problem 1, i.e., a terminal horizon H and a sequence of control inputs $\mathbf{u}_{0:H}$.

To extract such a solution, we need first to define the set $\mathcal{X}_g \subseteq \mathcal{V}$ that collects all states $\mathbf{q}(t) = [\mathbf{p}(t), q_D] \in \mathcal{V}$ of the tree that satisfy $q_D = q_F$, which captures the constraint (4b) [lines 13-14, Alg. 1]. Then, among all nodes $\mathcal{X}_g$, we select the node $\mathbf{q}(t) \in \mathcal{X}_g$, with the smallest cost $J_{\mathcal{G}}(\mathbf{q}(t))$, denoted by $\mathbf{q}(t_{\text{end}})$ [line 16, Alg. 1]. Then, the terminal horizon is $H = t_{\text{end}}$, and the control inputs $\mathbf{u}_{0:H}$ are recovered by computing the path $\mathbf{q}_{0:t_{\text{end}}}$ in $\mathcal{G}$ that connects $\mathbf{q}(t_{\text{end}})$ to the root $\mathbf{q}(0)$, i.e., $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \ldots, \mathbf{q}(t_{\text{end}})$ [line 17, Alg. 1]. Note that satisfaction of the constraint (4c) is guaranteed by construction of $\mathcal{G}$; see Section III-A. In what follows, we describe the core operations of Algorithm 1, '*sample*' and '*extend*' that are used to construct the tree $\mathcal{G}$.

A. Incremental Construction of Trees

At every iteration n of Algorithm 1, a new state $\mathbf{q}_{\text{new}}(t+1) = [\mathbf{p}_{\text{new}}(t+1), q_D^{\text{new}}]$ is sampled. The construction of the state $\mathbf{q}_{\text{new}}(t+1)$ relies on two steps. Specifically, first we sample a state $\mathbf{p}_{\text{new}}(t+1)$; see Section III-A.1. Second, we append to $\mathbf{p}_{\text{new}}(t+1)$, a DFA state q_D^{new} giving rise to $\mathbf{q}_{\text{new}}(t+1)$ which is then added to the tree structure, if possible; see Section III-A.2.

1) Sampling Strategy: To construct the state $\mathbf{p}_{\text{new}}$, we first divide the set of nodes $\mathcal{V}$ into a *finite* number of sets, denoted by $\mathcal{V}_k \subseteq \mathcal{V}$, based on the DFA component of the states $\mathbf{q} \in \mathcal{V}$. Specifically, $\mathcal{V}_k$ collects all states $\mathbf{q} \in \mathcal{V}$ that share the same DFA state $\mathbf{q}_D$, for some given $\mathbf{q}_D \in \mathcal{Q}_D$. By construction of $\mathcal{V}_k$, we get that $\mathcal{V} = \cup_{k=1}^{K_n} \mathcal{V}_k$, where K_n is the number of subsets $\mathcal{V}_k$ at iteration n. Also, notice that K_n is finite for all iterations n by construction of the DFA. At iteration $n = 1$ of Algorithm 1, it holds that $K_1 = 1$, $\mathcal{V}_1 = \mathcal{V}$ [line 2, Alg. 1].

Second, given the sets $\mathcal{V}_k$, we first sample from a given discrete distribution $f_{\mathcal{V}}(k|\mathcal{V}) : \{1, \ldots, K_n\} \to [0, 1]$ an index $k \in \{1, \ldots, K_n\}$ that points to the set $\mathcal{V}_k$ [line 3, Alg. 1]. The density function $f_{\mathcal{V}}(k|\mathcal{V})$ defines the probability of selecting the set $\mathcal{V}_k$ at iteration n of Algorithm 1 given the set $\mathcal{V}$. Any density function $f_{\mathcal{V}}$ can be used to draw samples k_{rand} as long as it satisfies the following assumption.

Assumption 3.2 (Density function $f_{\mathcal{V}}$): (i) The probability density function $f_{\mathcal{V}}(k|\mathcal{V})$ satisfies $f_{\mathcal{V}}(k|\mathcal{V}) \geq \epsilon$, $\forall\, k \in \{1, \ldots, K_n\}$ and for all $n \geq 0$, for some $\epsilon > 0$ that remains constant across all iterations n. (ii) Independent samples k_{rand} can be drawn from $f_{\mathcal{V}}$.

Next, given the set $\mathcal{V}_{k_{\text{rand}}}$ sampled from $f_{\mathcal{V}}$ and for each state $\mathbf{q}_{\text{rand}} \in \mathcal{V}_{k_{\text{rand}}}$, we sample a control input $\mathbf{u}_{\text{new}} \in \mathcal{U}$ from a discrete distribution $f_{\mathcal{U}}(\mathbf{u}) : \mathcal{U} \to [0, 1]$ [line 5, Alg. 1]. Given a control input $\mathbf{u}_{\text{new}}$ sampled from $f_{\mathcal{U}}$, we construct the state $\mathbf{p}_{\text{new}}$ as $\mathbf{p}_{\text{new}} = \mathbf{f}(\mathbf{p}_{\text{rand}}, \mathbf{u}_{\text{new}})$ [line 6, Alg. 1]. Any density function $f_{\mathcal{U}}$ can be used to draw samples $\mathbf{u}_{\text{new}}$ as long as it satisfies the following assumption.

[1] Throughout the paper, when it is clear from the context, we drop the dependence of $\mathbf{q}(t)$ on t.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.

Assumption 3.3 (Density function $f_{\mathcal{U}}$): (i) The distribution $f_{\mathcal{U}}(\mathbf{u})$ satisfies $f_{\mathcal{U}}(\mathbf{u}) \geq \zeta$, for all $\mathbf{u} \in \mathcal{U}$, for some $\zeta > 0$ that remains constant across all iterations n. (ii) Independent samples $\mathbf{u}_{\text{new}}$ can be drawn from the probability density function $f_{\mathcal{U}}$.

Remark 3.4 (Density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$): An example of a distribution $f_{\mathcal{V}}$ that satisfies Assumption 3.2 is the discrete uniform distribution $\frac{1}{k}$, for all $k \in \{1, \ldots, K_n\}$. Observe that the uniform function trivially satisfies Assumption 3.2(ii). Also, observe that Assumption 3.2(i) is also satisfied, since there exists an $\epsilon > 0$ that satisfies Assumption 3.2(i), which is $\epsilon = \frac{1}{|\mathcal{Q}_D|}$. Similarly, uniform density functions $f_{\mathcal{U}}$ satisfy Assumption 3.3. Note that any functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ can be employed as long as they satisfy Assumptions 3.2 and 3.3. Nevertheless, the selection of $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ affects the performance of Algorithm 1; see Theorem 4.3.

2) Extending the tree: To build incrementally a tree that explores both the robot motion space and the DFA state space, we append to $\mathbf{p}_{\text{new}}(t+1)$ the DFA state $q_D^{\text{new}} = \delta_D(q_D^{\text{rand}}, L([\mathbf{p}_{\text{rand}}(t), \mathcal{P}]))$. If such a state does not exist, then this means the path connecting the root to $(q_D^{\text{rand}}, L([\mathbf{p}_{\text{rand}}(t), \mathcal{P}]))$ violates ϕ. Otherwise, the state $\mathbf{q}_{\text{new}} = (\mathbf{p}_{\text{new}}, q_D^{\text{new}})$ is constructed [line 9, Alg. 1] which is then added to the tree. Particularly, we update the set of nodes and edges of the tree as $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{\text{new}}(t+1)\}$ and $\mathcal{E} = \mathcal{E} \cup \{(\mathbf{q}_{\text{rand}}(t), \mathbf{q}_{\text{new}}(t+1))\}$, respectively [lines 10-11, Alg. 1]. The cost of the new node $\mathbf{q}_{\text{new}}(t+1)$ is computed as discussed before, i.e., $J_{\mathcal{G}}(\mathbf{q}_{\text{new}}(t+1)) = J_{\mathcal{G}}(\mathbf{q}_{\text{rand}}(t)) + w(\mathbf{p}_{\text{rand}}(t), \mathbf{p}_{\text{new}}(t+1))$ [line 12, Alg. 1]. Finally, the sets $\mathcal{V}_k$ are updated, so that if there already exists a subset $\mathcal{V}_k$ associated with the DFA state $\mathbf{q}_D^{\text{new}}$, then $\mathcal{V}_k = \mathcal{V}_k \cup \{\mathbf{q}_{\text{new}}(t+1)\}$. Otherwise, a new set $\mathcal{V}_k$ is created, i.e., $K_n = K_n + 1$ and $\mathcal{V}_{K_n} = \{\mathbf{q}_{\text{new}}\}$ [line 15, Alg. 1]. Recall that this process is repeated for all states $\mathbf{q}_{\text{rand}}(t) \in \mathcal{V}_{k_{\text{rand}}}$ [line 4, Alg. 1].

IV. Completeness, Optimality & Convergence

In this section, we examine the correctness, optimality, and convergence rate of Algorithm 1. The proofs are omitted due to space limitations.

Theorem 4.1 (Probabilistic Completeness): If there exists a feasible solution to (4), then Algorithm 1 is probabilistically complete, i.e., it will find with probability 1 a path $\mathbf{q}_{0:H}$, defined as a sequence of states in $\mathcal{V}$, i.e., $\mathbf{q}_{0:H} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \ldots, \mathbf{q}(H)$, that satisfies the constraints of (4), where $\mathbf{q}(h) \in \mathcal{V}$, for all $h \in \{0, \ldots, H\}$.

Theorem 4.2 (Asymptotic Optimality): Assume that there exists an optimal solution to (4). Then, Algorithm 1 is asymptotically optimal, i.e., the optimal path $\mathbf{q}^*_{0:H} = \mathbf{q}(0), \mathbf{q}(1), \mathbf{q}(2), \ldots, \mathbf{q}(H)$, will be found with probability 1, as $n \to \infty$. In other words, the path generated by Algorithm 1 satisfies $\mathbb{P}\left(\{\lim_{n\to\infty} J(H, \mathbf{u}_{0:H}) = J^*\}\right) = 1$, where J is the objective function of (4) and J^* is the optimal cost.[2]

Theorem 4.3 (Convergence rate bounds): Let $\mathbf{q}^*_{0:H}$ denote the optimal solution to (4). Then, there exist parameters $\alpha_n(\mathbf{q}^*_{0:H}) \in (0, 1]$, which depend on the selected density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$, for every iteration n of Algorithm 1, such that $1 \geq \mathbb{P}(A^n(\mathbf{q}^*_{0:H})) \geq 1 - e^{-\frac{\sum_{k=1}^{n} \alpha_n(\mathbf{q}^*_{0:H})}{2} k + H}$, if $n > H$, where $A^n(\mathbf{q}^*_{0:H})$ denotes the event that Algorithm 1 constructs the path $\mathbf{q}^*_{0:H}$ within n iterations.

[2]Note that the horizon H and $\mathbf{u}_{0:H}$ returned by Algorithm 1 depend on n. For simplicity of notation, we drop this dependence.

V. Designing Biased Sampling Functions

As discussed in Section III, any density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ that satisfy Assumptions 3.2 and 3.3, respectively, can be employed. In what follows, we design density functions that allow us to address large-scale planning tasks that involve large teams of robots and workspaces.

Pruning the DFA: We first prune the DFA by removing transitions that are unlikely to be enabled. Particularly, these are DFA transitions that are enabled when a robot j has to satisfy atomic propositions that require it to be present in more than one location *simultaneously*. Hereafter, these DFA transitions are called *infeasible*. For instance, a DFA transition that is enabled if $\pi_j(\mathcal{R}_1) \wedge \pi_j(\mathcal{R}_2)$ is true, is classified as infeasible if the regions $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint, and feasible otherwise. Hereafter, for simplicity, we assume that all regions $\mathcal{R}_e$ are disjoint. Consider also the following example of infeasible transitions. Assume that transition from q_D to q'_D is enabled when the following condition is true: $g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r_1, \delta_1) \wedge g_{\mathbf{x}}(\mathbf{p}_j, \ell_e, r_2, \delta_2), e \neq i$ which requires robot j to be close to landmarks ℓ_i and ℓ_e, simultaneously. Note that existence of a robot position $\mathbf{p}_j$ that satisfies $g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r_1, \delta_1) \wedge g_{\mathbf{x}}(\mathbf{p}_j, \ell_e, r_2, \delta_2)$ depends on the distribution of the position of landmarks ℓ_e and ℓ_j and the parameters $r_1, \delta_1, r_2, \delta_2$. Hereafter, for simplicity, we classify such DFA transitions as *infeasible*.[3] Second, we define a distance function $d : \mathcal{Q}_D \times \mathcal{Q}_D \to \mathbb{N}$ between any two DFA states, which captures the minimum number of transitions in the pruned DFA to reach a state q'_D from a state q_D. This function is defined as follows

$$d(q_D, q'_D) = \begin{cases} |SP_{q_D, q'_D}|, & \text{if } SP_{q_D, q'_D} \text{ exists,} \\ \infty, & \text{otherwise,} \end{cases} \tag{5}$$

where SP_{q_D, q'_D} denotes the shortest path (in terms of hops) in the pruned DFA from q_D to q'_D and $|SP_{q_D, q'_D}|$ stands for its cost (number of hops).

Density function $f_{\mathcal{V}}$: First, we define the set $\mathcal{D}_{\text{min}}$ that collects the nodes $q = (\mathbf{p}, q_D) \in \mathcal{V}$ that have the minimum distance $d(q_D, q_F)$, denoted by d_{min}, among all nodes in $\mathcal{V}$, i.e., $\mathcal{D}_{\text{min}} = \{\mathbf{q} = (\mathbf{p}, q_D) \in \mathcal{V} \mid d(q_D, q_F) = d_{\text{min}}\}$. The set $\mathcal{D}_{\text{min}}$ initially collects only the root and is updated (along with d_{min}) as new states are added to the tree. Given the set $\mathcal{D}_{\text{min}}$, we define the set $\mathcal{K}_{\text{min}}$ that collects the indices k of the subsets $\mathcal{V}_k$ that satisfy $\mathcal{V}_k \cap \mathcal{D}_{\text{min}} \neq \emptyset$. Given the set $\mathcal{K}_{\text{min}}$, the probability density function $f_{\text{rand}}(k|\mathcal{V})$ is defined so that

[3]Note that this is a conservative approach as transitions that are in fact feasible may be classified as infeasible. Nevertheless, this does not affect the correctness of the proposed algorithm, since the constructed biased sampling functions satisfy Assumptions (3.2) and (3.3); see [6]. If the pruned DFA is disconnected then this means that either the specification is infeasible or the task is feasible but the pruning was very conservative. In this case, either uniform distributions or the proposed biased distributions can be employed but using the original (and not the pruned) DFA.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.

it is biased to select more often subsets $\mathcal{V}_k \subseteq \mathcal{V}$ that satisfy $k \in \mathcal{K}_{\min}$. Specifically, $f_{\text{rand}}(k|\mathcal{V})$ is defined as follows

$$f_{\text{rand}}(k|\mathcal{V}) = \begin{cases} p_{\text{rand}} \frac{1}{|\mathcal{K}_{\min}|}, & \text{if } k \in \mathcal{K}_{\min} \\ (1-p_{\text{rand}}) \frac{1}{|\mathcal{V} \setminus \mathcal{K}_{\min}|}, & \text{otherwise,} \end{cases} \quad (6)$$

where $p_{\text{rand}} \in (0.5, 1)$ stands for the probability of selecting *any* subset $\mathcal{V}_k$ that satisfies $k \in \mathcal{K}_{\min}$. Note that p_{rand} can change with iterations n but it should always satisfy $p_{\text{rand}} \in (0.5, 1)$ to ensure that subsets $\mathcal{V}_k$ with $k \in \mathcal{K}_{\min}$ are selected more often.

Density function $f_{\mathcal{U}}$: The sampling function $f_{\mathcal{U}}(\mathbf{u}|k_{\text{rand}})$ is designed so that a state $\mathbf{q} = (\mathbf{p}, q_F)$ is reached by following the shortest path (in terms of hops) in the pruned DFA that connects q_D^0 to q_F. First, given a state $\mathbf{q}_{\text{rand}} = (\mathbf{p}_{\text{rand}}, q_D^{\text{rand}})$, we compute the next DFA state defined as $q_D^{\text{next}} = \delta_D(q_D^{\text{rand}}, L([\mathbf{p}_{\text{rand}}), \mathcal{P}])$. Next, we construct the reachable set $\mathcal{R}_D(q_D^{\text{next}})$ that collects all states $q_D \in \mathcal{Q}_D$ that can be reached in one hop in the *pruned* DFA from q_D^{rand}, defined as $\mathcal{R}_D(q_D^{\text{next}}) = \{q_D \in \mathcal{Q}_D | \exists \sigma \in 2^{\mathcal{AP}}$ s.t. $\delta_D(q_D^{\text{next}}, \sigma) = q_D\}$. Among all states in $\mathcal{R}_D(q_D^{\text{next}})$ we select the state, denoted by $q_D^{\min}$, with the minimum distance from q_F.

Given q_D^{next} and $q_D^{\min}$, we select a symbol $\sigma \in 2^{\mathcal{AP}}$ that enables a transition from q_D^{next} to $q_D^{\min}$ in the pruned DFA. Given the symbol σ, we select locations that if every robot visits, then σ is generated and transition from q_D^{next} to $q_D^{\min}$ is achieved. To this end, first we select the atomic proposition that appears in σ and is associated with robot j. Note that by definition of the feasible transitions and by construction of the pruned DFA, there is only one (if any) atomic proposition in σ related to robot j. With slight abuse of notation, we denote this atomic proposition by $\pi_j \in \mathcal{AP}$. For instance, the atomic proposition π_j corresponding to the symbol $\sigma = g_c(\mathbf{p}_j, r, \delta, \bar{\mathcal{C}}) g_c(\mathbf{p}_z, r, \delta, \bar{\mathcal{C}})$ is $\pi_j = g_c(\mathbf{p}_j, r, \delta, \bar{\mathcal{C}})$. Also, observe that $\pi_s = \varnothing$ for all robots $s \neq j, z$. Next, given π_j, we compute all possible landmarks (or regions) that if robot j visits, then π_j may be satisfied. These landmarks/regions are collected in a set $\mathcal{L}_j$. For instance, if $\pi_j = g_c(\mathbf{p}_j, r, \delta, \bar{\mathcal{C}})$, then all landmarks that have a non-zero probability of having a class that belongs to $\bar{\mathcal{C}}$ are collected in a set $\mathcal{L}_j$. Among all landmarks (or regions) in the set $\mathcal{L}_j$, we select one landmark/region, denoted by t_j. This landmark/region is used to design $f_{\text{new},i}(\mathbf{u}_i|k)$ so that control inputs that drive robot j toward t_j are selected more often than others. Next, we discuss how we select t_j from $\mathcal{L}_j$ for robot j. If the atomic proposition π_j is in the form of (3a), i.e., $\pi_j = g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r, \delta)$, then it trivially holds that $\mathcal{L}_j = \{\ell_i\}$ and, therefore, $t_j = \ell_i$. If the atomic proposition π_j is in the form of (3b), i.e., $\pi_j = g_c(\mathbf{p}_j, r, \delta, \bar{\mathcal{C}})$, then we select as t_j from $\mathcal{L}_j$ the landmark ℓ_i with maximum value $\frac{d_{c_i}(c_i \in \bar{\mathcal{C}})}{\det \Sigma_i}$, where recall that $\det \Sigma_i$ is the determinant of the covariance matrix Σ_i. In words, the larger the value of $\frac{d_{c_i}(c_i \in \bar{\mathcal{C}})}{\det \Sigma_i}$ is, the smaller the uncertainty about the position and the class of landmark ℓ_i. Similarly, we construct t_j for atomic propositions π_j in the form of (2a). Note that for robots j for which it holds that $\pi_j = \varnothing$, we have that $t_j = \varnothing$, as well. In other words, the location of such robots j does not play any role in generating the symbol σ. Given the assigned landmark/region denoted by t_j, we construct the density function $f_{\text{new},j}(\mathbf{u}_j|k)$ from which we sample a control input $\mathbf{u}_j$ as follows:

$$f_{\text{new},j}(\mathbf{u}_j|k) = \begin{cases} \frac{1}{|\mathcal{U}_j|}, \text{ if } t_j = \varnothing, \\ p_{\text{new}}, \text{if } (t_j \neq \varnothing) \wedge (\mathbf{u}_j = \mathbf{u}_j^*) \\ (1-p_{\text{new}}) \frac{1}{|\mathcal{U}_j \setminus \{\mathbf{u}_j^*\}|}, \text{if } (t_j \neq \varnothing) \\ \qquad \wedge (\mathbf{u}_j \neq \mathbf{u}_j^*), \end{cases} \quad (7)$$

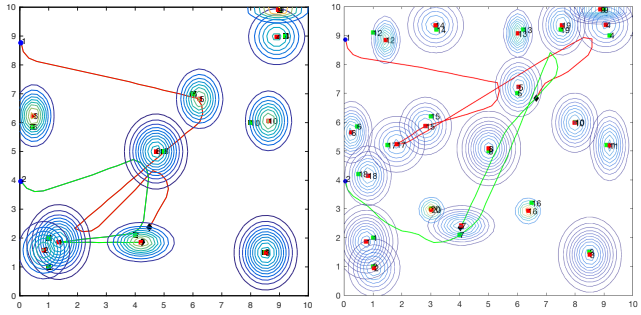
where $\mathbf{u}_j^* \in \mathcal{U}_j$ is the control input that minimizes the *geodesic* distance between $\mathbf{p}_j(t+1)$ and the location of t_j; see [19]. To compute the *geodesic* distance between $\mathbf{p}_j(t+1)$ and the location of t_j, we first treat as 'virtual/temporal' obstacles all landmarks/regions that if visited, the Boolean condition under which transition from $q_D^{\min}$ to q_D^{decr} is enabled, is false. For example, assume that transition from $q_D^{\min}$ to q_D^{decr} is possible if the following Boolean condition is true $\pi_j(\mathcal{R}_1) \wedge \pi_z(\mathcal{R}_2) \wedge (\neg g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r, \delta))$, which is satisfied by the following symbol $\sigma = \pi_j(\mathcal{R}_1)\pi_z(\mathcal{R}_2)$ and is violated if $g_{\mathbf{x}}(\mathbf{p}_j, \ell_i, r, \delta)$ is observed. For such 'virtual' obstacles, we first compute the δ-confidence interval around the mean position of landmark ℓ_i. This δ-confidence interval is then treated as a (known) physical obstacle in the workspace. Then, we compute the *geodesic* distance between $\mathbf{p}_j(t+1)$ and t_j given such virtual obstacles. Also, note that these obstacles may depend on the global state $\mathbf{p}(t)$. For instance, consider the Boolean condition $\pi_j(\mathcal{R}_1) \wedge \pi_z(\mathcal{R}_2) \wedge (\neg \pi_j(\mathcal{R}_1) \wedge \neg \pi_z(\mathcal{R}_1))$. This is violated only if *both* robots j and z are present in region $\mathcal{R}_1$. Such 'virtual' obstacles are ignored, i.e., $\mathcal{R}_1$ is not treated as an obstacle.

VI. Numerical Experiments

In this section, we present numerical experiments that Algorithm 1 can solve large-scale planning tasks in uncertain environments. All case studies have been implemented using MATLAB 2016b on a computer with Intel Core i7 3.1GHz and 16Gb RAM. Hereafter, we employ the density functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ designed in Section V and we select a distance-based motion cost function: $w(\mathbf{p}(t), \mathbf{p}(t+1)) = \sum_{j=1}^{N} ||\mathbf{p}_j(t) - \mathbf{p}_j(t+1)||$. Also, we consider robots with differential drive dynamics where the robot state captures both the position and the orientation of the robots. The available motion primitives are $u \in \{0, 0.15\}$m/s and $\omega \in \{0, \pm\pi/4, \pm\pi/2, \pm\pi/1.33, \pm\pi\}$ rad/s.

We examine the performance of Algorithm 1 with respect to the number N of robots and number M of landmarks. The results are summarized in Table I. The third column in Table I shows the runtime to compute the first feasible path. The robot paths for the cases $N = 2, M = 10$ and $N = 2, M = 20$ are shown in Figure 1. In all case studies, we assume that the set of classes is $\mathcal{C} = \{1, 2, 3, 4\}$ and that the robots have to accomplish a collaborative task captured by the following TL formula. $\phi = \Diamond(\phi_1 \wedge \Diamond(\phi_2 \wedge \Diamond\phi_3)) \wedge (\neg\phi_1 \mathcal{U} \phi_4) \wedge \Diamond(\phi_5) \wedge \Box(\neg\phi_6) \wedge \Box(\neg\phi_7)$. In words, this specification requires that (i) the formulas ϕ_1, ϕ_2, and ϕ_3 should be satisfied sequentially in this order; (ii) ϕ_1

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.



(a) $N = 2, M = 10$ (b) $N = 2, M = 20$

Fig. 1. The contours of the Gaussian distributions for each landmark $\mathcal{N}(\hat{\mathbf{x}}_i, \Sigma_i)$ are shown. The locations $\hat{\mathbf{x}}_i$ are depicted by red squares. The green squares represent the true landmark locations. The initial (final) locations of the robots are depicted with blue circles (black diamonds).

TABLE I

SCALABILITY ANALYSIS

N	M	Time (mins)	H	Cost (m)
2	10	0.06	74	32.02
5	10	1.46	139	79.59
10	10	1.51	127	133.21
2	20	0.63	95	40.10
10	20	3.22	122	134.63
20	20	4.21	137	247.49
30	20	5.15	137	342.41

should not be satisfied until ϕ_4 is satisfied; (iii) ϕ_5 should eventually be satisfied; and (iv) ϕ_6 and ϕ_7 should never be satisfied. The definition of the subformulas ϕ_i depends on the number of robots and landmarks. For example, for the case $N = 2, M = 20$, the subformulas ϕ_1, ϕ_3, ϕ_6 and ϕ_7 are defined as follows.

- $\phi_1 = g_{\mathbf{x}}(\mathbf{p}_1, \ell_{17}, 1, 0.85) \wedge g_{\mathbf{x}}(\mathbf{p}_2, \ell_8, 1, 0.65)$,
- $\phi_3 = [g_c(\mathbf{p}_1, 1, 0.6, \{3\}) \wedge (\vee_{c_i=3} \pi_{\ell_i}(\Sigma_i, 0.1))]$
- $\phi_6 = g_c(\mathbf{p}_1, 1, 0.8, \{4\}) \wedge g_c(\mathbf{p}_2, 1, 0.8, \{4\})$,
- $\phi_7 = g_{\mathbf{x}}(\mathbf{p}_1, \ell_5, 1, 0.8) \vee g_{\mathbf{x}}(\mathbf{p}_1, \ell_{10}, 1, 0.8)$.

For instance, ϕ_1 requires robot 1 to be within distance of at most 1m from landmark ℓ_{17} with probability at least 0.85 and robot 2 to be within distance of at most 1m from ℓ_8 with probability at least 0.65, at the same time. Also, ϕ_3 requires both robots to eventually be within distance less than 1m from a landmark with class $c_i = 3$, simultaneously, with probability greater than 0.6. The selected landmarks should also satisfy $\det \Sigma_i \leq 0.1$. Also, $\square(\neg\phi_6)$ captures an obstacle avoidance constraint. Specifically, the landmarks with class $c_i = 4$ are considered obstacles and all robots should always maintain a distance of at least 1m from them with probability greater than 0.8. The discrete distribution d_c over the labels is designed by randomly selecting a probability within the range $[0.7, 1]$ that a landmark has the correct label, and by randomly selecting probabilities for the remaining classes.

VII. CONCLUSION

In this paper, we proposed a new highly scalable temporal logic planning approach in uncertain environments modeled as probabilistic semantic maps. Finally, we show that the proposed sampling-based algorithm is probabilistically complete and asymptotically optimal.

REFERENCES

[1] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic SLAM," in *IEEE International Conference on Robotics and Automation*, Singapore, May-June 2017, pp. 1722–1729.

[2] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 723–730.

[3] H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Autonomous Robots*, vol. 33, no. 3, pp. 255–272, 2012.

[4] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[5] C.-I. Vasile, K. Leahy, E. Cristofalo, A. Jones, M. Schwager, and C. Belta, "Control in belief space with temporal logic specifications," in *IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, NV, December 2016, pp. 7419–7424.

[6] Y. Kantaros and M. M. Zavlanos, "Temporal logic optimal control for large-scale multi-robot systems: 10^{400} states and beyond," in *IEEE Conference on Decision and Control (CDC)*, December 2018, pp. 2519–2524.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *American Control Conference (ACC)*, Montreal, Canada, June 2012, pp. 735–742.

[8] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, November 2013, pp. 4817–4822.

[9] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multi-robot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2018.

[10] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain markov decision processes with temporal logic specifications," in *IEEE 51st Annual Conference on Decision and Control (CDC)*, Maui, Hawaii, December 2012, pp. 3372–3379.

[11] B. Johnson and H. Kress-Gazit, "Analyzing and revising synthesized controllers for robots with sensing and actuation errors," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 816–832, 2015.

[12] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[13] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4051–4066, 2018.

[14] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[15] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 583–599, 2016.

[16] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[17] J. Fu, N. Atanasov, U. Topcu, and G. J. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," in *IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, 2016, pp. 3690–3697.

[18] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[19] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments." *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:22:35 UTC from IEEE Xplore. Restrictions apply.