A Sampling-Based ~~Method~~ for Multi-Robot ~~Optimal~~ Path Planning ~~under Temporal Tasks~~

Yiannis Kantaros, *Student Member, IEEE,* and Michael M. Zavlanos, *Member, IEEE*

***Abstract*—This paper proposes a sampling-based algorithm for multi-robot control synthesis under global Linear Temporal Logic (LTL) specifications. The majority of existing LTL planning approaches rely on ~~constructing~~ a discrete abstraction of robot mobility in the workspace, which is then combined with the Büchi automaton that captures the LTL specification to obtain a product automaton among the robots. ~~Using this product automaton,~~ graph search methods can be applied to synthesize correct-by construction motion plans for the robots. ~~Nevertheless, the high computational cost of the discrete abstraction process along with the exponential growth of the size of the product automaton with the number of robots render the control synthesis intractable.~~ In this paper, we propose a new sampling-based LTL planning algorithm that does not employ any discrete abstraction of robot mobility and avoids the construction of a product automaton. Instead, it builds incrementally trees that take into account the robot dynamics and constraints and explores the product state-space without explicitly constructing it, until optimal motion plans are found. By exploring the product state-space using a tree rather than a graph of arbitrary structure, we require much fewer resources to store it, increasing scalability of our algorithm significantly compared to similar temporal planning methods. ~~Moreover by controlling the distance between any two nodes in the tree, we can obtain samples of the workspace that allow for fast exploration of large workspaces. Allowing this distance to decrease as the tree grows, we show~~ that our algorithm is probabilistically complete and asymptotically optimal. ~~Finally,~~ we present numerical experiments ~~that show efficiency of our approach compared to~~ relevant temporal planning methods. ~~Our algorithm can be viewed as an extension of the RRT* algorithm for temporal planning problems.~~**

I. Introduction

MOTION planning is a fundamental problem ~~in the~~ robotics ~~literature~~ that has received considerable ~~research attraction~~ [1]. The basic motion planning problem consists of generating trajectories that reach a given goal region from an initial configuration while avoiding obstacles. Methods for point-to-point navigation range from using potential fields and navigation functions [2], [3] to sampling-based algorithms [4]–[6]. More recently, a new class of planning approaches have been developed that can handle a richer class of tasks, than the classical point-to-point navigation, and can capture temporal and boolean requirements. Such tasks can be, e.g., sequencing or coverage [7], data gathering [8], intermittent communication [9], or persistent surveillance [10], and can be captured using formal languages, such as Linear Temporal Logic (LTL) [11], that are developed in concurrency theory. ~~Given a task described by a formal language, model checking algorithms can be employed to synthesize correct-by construction controllers that satisfy the assigned tasks.~~

~~Control synthesis for mobile robots under complex tasks, captured by Linear Temporal Logic (LTL) formulas, builds upon either bottom-up approaches where independent LTL expressions are assigned to robots [12]–[15] or top-down approaches when a global LTL formula describing a collaborative task is assigned to a team of robots [16], [17], as in our work. Top-down approaches generate discrete high-level motion plans for all robots using the product automaton that combines the individual transition systems, that have to satisfy a bisimulation property [7], [18] and are obtained through an abstraction process [19]–[23] with a Non-deterministic Büchi Automaton (NBA) that represents the global LTL specification. The main limitations of these approaches are the high computational cost of constructing a discrete abstraction and the poor scalability of control synthesis algorithms for large product automata.~~

Motivated by existing sampling-based algorithms for point-to-point navigation [6], ~~we propose~~ a temporal ~~planning~~ method that does not require any discrete abstraction of robot mobility and avoids the construction of a product automaton. In particular, we build incrementally through a Büchi-guided sampling-based algorithm directed trees that explore both the continuous state-space of robot positions and the discrete state-space of the NBA, simultaneously. Specifically, ~~to construct a discrete motion plan that satisfies a LTL specification,~~ we first build a tree ~~incrementally until a path from~~ an initial to an *accepting* state ~~is constructed~~. ~~This path~~ corresponds to the *prefix* part of the motion plan and is executed once. Then, ~~a new~~ tree rooted at an accepting state ~~is constructed~~ in a similar way ~~until~~ a cycle-detection method ~~discovers~~ a loop around the root. This cyclic path corresponds to the *suffix* part of the motion plan and is executed indefinitely. ~~Moreover,~~ by construction of the trees, the continuous execution of the generated ~~synthesized~~ discrete plans satisfies the assigned LTL-based task, ~~as well~~. ~~Finally, our~~ algorithm can be viewed as an extension of the RRT* algorithm [6] for ~~temporal~~ planning problems.

The main contributions of the proposed method are three-fold. First, it does not require any discrete abstraction of the workspace, as opposed to the majority of existing literature on robot planning under ~~temporal~~ tasks. Second, using a tree rather than an arbitrary graph to explore the continuous product state-space, comprised of the continuous state-space of robot positions and the discrete state-space of a NBA, results in significant savings in resources both in terms of memory to ~~save~~ the associated data structures and in terms of computational cost in applying graph search techniques

Yiannis Kantaros and Michael M. Zavlanos are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA. {yiannis.kantaros,michael.zavlanos}@duke.edu. This work is supported in part by NSF under grant IIS #1302283.

to find the desired motion plans. In this way, scalability of the proposed LTL-based planning method is significantly increased compared to existing approaches. Third, we show that the proposed planning algorithm is probabilistically complete and asymptotically optimal.

To the best of our knowledge, the most relevant works are presented in [24]–[28]. In our previous work [25], [26], we have proposed sampling-based planning algorithms for multi-agent systems under global temporal goals. Specifically, [25] transforms given transition systems that abstract robot mobility into trace-included transition systems with smaller state-spaces that are still rich enough to construct motion plans that satisfy the global LTL specification. However, this algorithm does not scale well with the number of robots, since it relies on the construction of a product automaton among all agents. A more tractable approach is proposed in [26] that builds trees incrementally, similar to the approach proposed here, that approximate the product automaton until a motion plan is constructed. Common in [25], [26] is that a discrete abstraction of the environment is required as an input to the planning algorithms, which is not the case in the work considered here. Other temporal planning methods that avoid the construction of a discrete abstraction of the environment are presented in [24], [27], [28]. In [27], [28], sampling-based algorithms are proposed which build incrementally a Kripke structure until it is expressive enough to generate a motion plan that satisfies a task specification expressed in deterministic μ-calculus. Specifically, in [27], [28], since control synthesis from ω-regular languages requires cyclic patterns, an RRG-like algorithm is employed to construct a Kripke structure. However, building arbitrary structures to represent transition systems, compromises scalability of temporal planning methods since, as the number of samples increases, so does the density of the constructed graph increasing in this way the required resources to save the associated structure. Therefore, the algorithms proposed in [27], [28] can be typically used only for single-agent motion planning problems, unlike our approach that can handle more complex planning problems that involve multi-robot networks, complex environments and LTL tasks that correspond to large NBA. Motivated by this limitation, in [24], a sampling-based temporal logic path planning algorithm is proposed, that also builds upon the RRG algorithm, but constructs incrementally sparse graphs representing transition systems that are then used to construct a product automaton. Then correct-by-construction discrete plans are synthesized applying graph search methods on the product automaton. However, similar to the works in [27], [28], as the number of samples increases, the sparsity of the constructed graph is lost and the computational cost of graph search methods increases, compromising scalability of planning algorithms. To the contrary, in our work, we propose a sampling-based approach that builds trees, instead of graphs of arbitrary structure, to explore the continuous product state-space. Due to this tree structure, our proposed algorithm is more economical in terms of memory requirements and does not require the application of expensive graph search techniques to find the optimal motion plan, but instead it tracks the sequence of parent nodes starting from desired accepting states. This allows our method to handle larger problems compared to the ones that can be solved using the approach in [24]. Moreover, we show that our proposed planning algorithm is asymptotically optimal which is not the case in [24].

The rest of the paper is organized as follows. In Section II, we provide a brief overview of LTL and then, in Section III we present the problem formulation. In Section IV we describe our proposed sampling-based planning algorithm and we examine its correctness and optimality in Section V. Simulation results and conclusive remarks are presented in Sections VI and VII, respectively.

II. Preliminaries

In this section we formally describe Linear Temporal Logic (LTL) by presenting its syntax and semantics. Also, we briefly review preliminaries of automata-based LTL model checking. A detailed overview of this theory can be found in [11].

Linear temporal logic is a type of formal logic whose basic ingredients are a set of atomic propositions $\mathcal{AP}$, the boolean operators, i.e., conjunction $\wedge$, and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \ \mathcal{U} \ \phi_2$, where $\pi \in \mathcal{AP}$. For the sake of brevity we abstain from presenting the derivations of other Boolean and temporal operators, e.g., *always* $\Box$, *eventually* $\Diamond$, *implication* $\Rightarrow$, which can be found in [11]. In this paper, we restrict our attention to LTL that exclude the "next" temporal operator, since it violates the continuity of underlying path.

An infinite *word* σ over the alphabet $2^{\mathcal{AP}}$ is defined as an infinite sequence $\sigma = \pi_0\pi_1\pi_2\cdots \in (2^{\mathcal{AP}})^\omega$, where ω denotes an infinite repetition and $\pi_k \in 2^{\mathcal{AP}}$, $\forall k \in \mathbb{N}$. The language $\texttt{Words}(\phi) = \{\sigma \in (2^{\mathcal{AP}})^\omega | \sigma \models \phi\}$ is defined as the set of words that satisfy the LTL formula ϕ, where $\models \subseteq (2^{\mathcal{AP}}) \times \phi$ is the satisfaction relation.

Any LTL formula ϕ can be translated into a Nondeterministic Büchi Automaton (NBA) over $2^{\mathcal{AP}}$ denoted by B [29], which is defined as follows:

Definition 2.1: A *Nondeterministic Büchi Automaton* (NBA) B over $2^{\mathcal{AP}}$ is defined as a tuple $B = \left(\mathcal{Q}_B, \mathcal{Q}_B^0, \Sigma, \rightarrow_B, \mathcal{Q}_B^F\right)$, where $\mathcal{Q}_B$ is the set of states, $\mathcal{Q}_B^0 \subseteq \mathcal{Q}_B$ is a set of initial states, $\Sigma = 2^{\mathcal{AP}}$ is an alphabet, $\rightarrow_B \subseteq \mathcal{Q}_B \times \Sigma \times \mathcal{Q}_B$ is the transition relation, and $\mathcal{Q}_B^F \subseteq \mathcal{Q}_B$ is a set of accepting/final states.

An *infinite run* ρ_B of B over an infinite word $\sigma = \pi_0\pi_1\pi_2\ldots$, $\pi_k \in \Sigma = 2^{\mathcal{AP}}$ $\forall k \in \mathbb{N}$ is a sequence $\rho_B = q_B^0 q_B^1 q_B^2 \ldots$ such that $q_B^0 \in \mathcal{Q}_B^0$ and $(q_B^k, \pi_k, q_B^{k+1}) \in \rightarrow_B$, $\forall k \in \mathbb{N}$. An infinite run ρ_B is called *accepting* if $\texttt{Inf}(\rho_B) \cap \mathcal{Q}_B^F \neq \varnothing$, where $\texttt{Inf}(\rho_B)$ represents the set of states that appear in ρ_B infinitely often. The words σ that result in an accepting run of B constitute the accepted language of B, denoted by $\mathcal{L}_B$. Then it is proven [11] that the accepted language of B is equivalent to the words of ϕ, i.e., $\mathcal{L}_B = \texttt{Words}(\phi)$.

III. Problem Formulation

Consider N mobile robots that reside in a complex workspace $\mathcal{W} \subset \mathbb{R}^d$, $d = 2, 3$, and let $\mathcal{O}$ denote the set of

obstacles in $\mathcal{W}$ and $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{O}$ denote the free workspace. We define $\mathbf{x}_i \in \mathcal{W}$ the position of robot i, for $i \in \{1, \dots, N\}$.

Let $\mathcal{R} = \{\ell_j\}_{j=1}^{W}$ be a set of $W > 0$ disjoint regions of interest in $\mathcal{W}$ that can have any arbitrary shape. Also, let $\mathcal{AP} = \{\{\pi_i^{\ell_j}\}_{j=1}^{W}\}_{i=1}^{N}$ be a set of atomic propositions $\pi_i^{\ell_j}$, so that $\pi_i^{\ell_j}$ is true if robot i is inside region ℓ_j. Moreover, we define the labeling function $L : \mathcal{W}^N \to 2^{\mathcal{AP}}$ that determines the atomic propositions that are satisfied at an arbitrary point $\mathbf{x} = [\mathbf{x}_1^T, \dots, \mathbf{x}_i^T, \dots, \mathbf{x}_N^T]^T \in \mathcal{W}^N$, where $\mathbf{x}_i \in \mathcal{W}$ and $i \in \{1, \dots, N\}$.

We assume that the robots have to accomplish a collaborative high-level task described by a global LTL formula ϕ that is defined over the set of atomic propositions $\mathcal{AP}$. For example, ~~a LTL specified~~ task for a team of 2 robots ~~can be~~: $\phi = (\square\lozenge\pi_1^{\ell_1}) \wedge (\square\lozenge\pi_2^{\ell_6}) \wedge (\neg\pi_1^{\ell_1} \mathcal{U} \pi_2^{\ell_3}) \wedge (\lozenge\pi_1^{\ell_2}) \wedge (\square\neg\pi_2^{\ell_4}) \wedge (\square\lozenge(\pi_1^{\ell_5} \wedge \pi_2^{\ell_5}))$, which requires (i) robot 1 to visit region ℓ_1 infinitely often, (ii) robot 2 to visit region ℓ_6 infinitely often, (iii) robot 1 to avoid region ℓ_1 until robot 2 visits region ℓ_3, (iv) robot 1 to eventually visit location ℓ_2, (v) robot 2 to always avoid region ℓ_4, and (vi) robots 1 and 2 to visit simultaneously region ℓ_5 infinitely often.

~~Such~~ LTL formulas ~~are~~ satisfied by discrete plans τ that are infinite sequences of locations in $\mathcal{W}_{\text{free}}^N$, i.e., $\tau = \tau(1)\tau(2)\dots\tau(k)\dots$, where $\tau(k) \in \mathcal{W}_{\text{free}}^N$. A discrete plan τ satisfies ϕ if the trace generated by τ, defined as $\texttt{trace}(\tau) := L(\tau(1))L(\tau(2))\dots L(\tau(k))$, belongs to $\texttt{Words}(\phi)$. We equivalently denote that $\texttt{trace}(\tau) \in \texttt{Words}(\phi)$ by $\tau \models \phi$. Such discrete plans can be written in a finite representation called prefix-suffix structure, i.e., $\tau = \tau^{\text{pre}} [\tau^{\text{suf}}]^{\omega}$, where the prefix part $\tau^{\text{pre}} = \tau(1)\tau(2)\dots\tau(K)$ is executed once and the suffix part $\tau^{\text{suf}} = \tau(K)\tau(K+1)\dots\tau(K+S)\tau(K+S+1)$, where $\tau(K+S+1) = \tau(K)$, is executed indefinitely [11].[1] The cost of such a plan can be defined as:

$$J(\tau) = \underbrace{\sum_{k=1}^{K-1} C(\tau(k), \tau(k+1))}_{\text{Cost of } \tau^{\text{pre}}} + \underbrace{\sum_{k=K}^{K+S} C(\tau(k), \tau(k+1))}_{\text{Cost of } \tau^{\text{suf}}}. \tag{1}$$

In (1), $C : \mathcal{W}_{\text{free}}^N \times \mathcal{W}_{\text{free}}^N \to \mathbb{R}_+$ is a cost function, which captures the incurred cost due to the execution of the prefix and suffix part of τ, for all robots. For ~~the~~ holonomic planning, the traveled distance can be ~~set~~ as the cost function, ~~i.e.,~~ $C(\tau(k), \tau(k+1)) = \|\tau(k) - \tau(k+1)\|$. ~~Given the above definitions we can~~ define the joint space where the ~~robot network resides~~ by the tuple $E := (\mathcal{W}^N, \mathcal{AP}, L, C)$. Then, the problem we address in this paper can be defined as follows.

Problem 1: Given the tuple $E = (\mathcal{W}^N, \mathcal{AP}, L, C)$, ~~a~~ global LTL specification ϕ defined over the set $\mathcal{AP}$, determine an optimal discrete team plan τ ~~that~~ satisfies ϕ ~~and so does its continuous execution~~, and minimizes the cost function $J(\tau)$ given in (1).

IV. Sampling-Based Optimal Control Synthesis

Our proposed algorithm, denoted as pRRT*, ~~p for product, relies on the incremental constructing trees~~ that explore the product state-space $P := \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$, where $\mathcal{Q}_B$ ~~stands for~~ the discrete state-space of the NBA ~~that corresponds to~~ ϕ. These trees are used to design optimal motion plans ~~τ~~ in a prefix-suffix structure, ~~i.e.,~~ $\tau = \tau^{\text{pre}}[\tau^{\text{suf}}]^{\omega}$, that satisfy ~~a given~~ LTL formula ϕ and minimize the cost function (1). The construction of the optimal plan $\tau = \tau^{\text{pre}}[\tau^{\text{suf}}]^{\omega} \models \phi$ is described in Algorithm 1.

Algorithm 1: Construction of Optimal plans $\tau \models \phi$

Input: Logic formula ϕ, tuple $E = (\mathcal{W}^N, \mathcal{AP}, L, C)$, Initial location $\mathbf{x}^0 \in \mathcal{W}_{\text{free}}^N$, maximum numbers of iterations $n_{\max}^{\text{pre}}$, $n_{\max}^{\text{suf}}$

Output: Optimal plans $\tau \models \phi$

1 Convert ϕ to a NBA $B = (\mathcal{Q}_B, \mathcal{Q}_B^0, \to_B, \mathcal{Q}_B^F)$;
; ▷ Construction of Prefix Plans $\tau^{\text{pre},\alpha}$
2 Define goal set: $\mathcal{X}_{\text{goal}}^{\text{pre}}$;
3 Initial state: $q_P^0 = (\mathbf{x}^0, q_B^0)$;
4 $[\mathcal{G}_T, \mathcal{P}] = \texttt{ConstructTree}(\mathcal{X}_{\text{goal}}^{\text{pre}}, E, B, q_P^0, n_{\max}^{\text{pre}})$;
5 **for** $\alpha = 1 : |\mathcal{P}|$ **do**
6 $\quad \tau^{\text{pre},\alpha} = \texttt{FindPath}(\mathcal{G}_T, q_P^0, \mathcal{P}(\alpha))$;
; ▷ Construction of Suffix Plans $\tau^{\text{suf},\alpha}$
7 **for** $\alpha = 1 : |\mathcal{P}|$ **do**
8 $\quad$ Initial state: $q_P^0 = \mathcal{P}(\alpha)$;
9 $\quad$ Define goal set: $\mathcal{X}_{\text{goal}}^{\text{suf}}(q_P^0)$;
10 $\quad$ **if** $q_P^0 \in \mathcal{X}_{goal}^{suf}$ **then**
11 $\quad\quad \mathcal{G}_T = (\{q_P^0\}, \{q_P^0, q_P^0\}, 0)$;
12 $\quad\quad \mathcal{S}_\alpha = \{q_P^0\}$;
13 $\quad$ **else**
14 $\quad\quad [\mathcal{G}_T, \mathcal{S}_\alpha] = \texttt{ConstructTree}(\mathcal{X}_{\text{goal}}^{\text{suf}}, E, B, q_P^0, n_{\max}^{\text{suf}})$;
15 $\quad$ **for** $e = 1 : |\mathcal{S}_\alpha|$ **do**
16 $\quad\quad \tilde{\tau}^{\text{suf},e} = \texttt{FindPath}(\mathcal{G}_T, q_P^0, \mathcal{S}_\alpha(e))$;
17 $\quad e^* = \text{argmin}_e(J(\tilde{\tau}^{\text{suf},e}))$;
18 $\quad \tau^{\text{suf},\alpha} = \tilde{\tau}^{\text{suf},e^*}$
; ▷ Construction of optimal plans
19 $\alpha^* = \text{argmin}_\alpha(J(\tau^{\text{pre},\alpha}) + J(\tau^{\text{suf},\alpha}))$;
20 Optimal Plan: $\tau = \tau^{\text{pre},\alpha^*}[\tau^{\text{suf},\alpha^*}]^{\omega}$;

In Algorithm 1, first the LTL formula is translated to a NBA $B = \{\mathcal{Q}_B, \mathcal{Q}_B^0, \to_B, \mathcal{Q}_B^F\}$ [line 1, Alg. 1]. Then, in lines 2-6, the prefix plans $\tau^{\text{pre},\alpha}$ are constructed, followed by the construction of their respective suffix plans $\tau^{\text{suf},\alpha}$ in lines 7-18. Finally, using the constructed prefix and suffix plans, the optimal plan $\tau = \tau^{\text{pre},\alpha^*}[\tau^{\text{suf},\alpha^*}]^{\omega} \models \phi$ is synthesized in lines 19-20.

A. Construction of Prefix Parts

In this Section, we describe how the prefix plan is constructed [lines 2-6, Alg. 1]. Motivated by the RRT* algorithm [6], to construct the prefix plan we build incrementally a tree, denoted by $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$, that explores the state-space $P = \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$. The set $\mathcal{V}_T$ consists of nodes/states $q_P = (\mathbf{x}, q_B) \in \mathcal{V}_T \subseteq \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$, ~~where~~ ~~$\mathbf{x} \in \mathcal{W}_{\text{free}}^N$~~ ~~and~~ ~~$q_B \in \mathcal{Q}_B$~~. The set of edges $\mathcal{E}_T$ captures transitions among the nodes in $\mathcal{V}_T$, i.e., $(q_P, q_P') \in \mathcal{E}_T$, if there is a transition from state $q_P \in \mathcal{V}_T$ to state $q_P' \in \mathcal{V}_T$. The

[1] The prefix-suffix structure holds for any LTL formula.

Algorithm 2: Function $[\mathcal{G}_T, \mathcal{Z}] = \texttt{ConstructTree}(\mathcal{X}_{\text{goal}}, E, B, q_P^0, n_{\max})$

Input: Initial state q_P^0, Goal region $\mathcal{X}_{\text{goal}}$, tuple $E = (\mathcal{W}^N, \mathcal{AP}, L, C)$, NBA B, maximum number of iterations $n_{\max}$

Output: Tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$, set $\mathcal{Z} = \{q_P \in \mathcal{V}_T, \mid q_P \in \mathcal{X}_{\text{goal}}\}$

1 $\mathcal{V}_T = \{q_P^0\}$;
2 $\mathcal{E}_T = \emptyset$;
3 $\texttt{Cost}(q_P^0) = 0$;
4 **for** $n = 1 : 1 : n_{max}$ **do**
5 $\quad \mathbf{x}^{\text{rand}} = \texttt{Sample}(\mathcal{W}, \mathcal{V}_T)$;
6 $\quad Q_P^{\text{nearest}} = \texttt{Nearest}(\mathcal{G}_T, \mathbf{x}^{\text{rand}})$;
7 $\quad \mathbf{x}^{\text{new}} = \texttt{Steer}(\mathbf{x}^{\text{nearest}}, \mathbf{x}^{\text{rand}})$;
8 $\quad$ **for** $b = 1 : |\mathcal{Q}_B|$ **do**
9 $\quad\quad q_B^{\text{new}} = \mathcal{Q}_B(b)$;
10 $\quad\quad q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$;
11 $\quad\quad [\mathcal{G}_T, \texttt{added}] = \texttt{Extend}(q_P^{\text{new}}, Q_P^{\text{nearest}}, \mathcal{G}_T)$;
12 $\quad\quad$ **if** $added = 1$ **then**
13 $\quad\quad\quad [\mathcal{E}_T, \texttt{Cost}] = \texttt{Rewire}(q_P^{\text{new}}, Q_P^{\text{nearest}}, \mathcal{G}_T)$;
14 $\mathcal{Z} = \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}$;

function $\texttt{Cost} : \mathcal{V}_T :\to \mathbb{R}_+$ assigns the cost of reaching node $q_P \in \mathcal{V}_T$ from the root of the tree. ~~Since we are interested in constructing~~ a prefix plan, i.e., a path in $\mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$ that connects an initial state q_P^0 to an *accepting* state $q_P = (\mathbf{x},\ q_B)$ with $q_B \in \mathcal{Q}_B^F$ we define the goal region, ~~$\mathcal{X}_{\text{goal}}^{\text{pre}}$, during~~ the construction of $\mathcal{G}_T$, as

$$\mathcal{X}_{\text{goal}}^{\text{pre}} = \{q_P = (\mathbf{x},\ q_B) \in \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B \mid q_B \in \mathcal{Q}_B^F\}. \quad (2)$$

The tree $\mathcal{G}_T$ is rooted at the initial state $q_P^0 = (\mathbf{x}^0, q_B^0)$ where $\mathbf{x}^0 = (\mathbf{x}_1^{0,T}, \ldots, \mathbf{x}_N^{0,T})^T \in \mathcal{W}_{\text{free}}^N$ is a vector that stacks the initial positions $\mathbf{x}_i^0 \in \mathcal{W}_{\text{free}}$ of all robots $i \in \{1, \ldots, N\}$ and $q_B^0 \in \mathcal{Q}_B^0$.[2] In Algorithm 2, the set $\mathcal{V}_T$ initially contains only the initial state/root $q_P^0 = (\mathbf{x}^0, q_B^0)$ [line 1, Alg. 2]. Also, the set of edges is initialized as $\mathcal{E}_T = \emptyset$ [line 2, Alg. 2]. By convention, we assume that the cost of q_P^0 is zero [line 3, Alg. 2]. In what follows, we describe how the tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$ is constructed incrementally [lines 4-13, Alg. 2].

1) Sampling a state $\mathbf{x}^{rand}$: The first step ~~for~~ the construction of the graph $\mathcal{G}_T$ is to sample a state $\mathbf{x}^{\text{rand}}$ that belongs to $\mathcal{W}_{\text{free}}^N$. ~~Sampling such a state $\mathbf{x}^{\text{rand}}$ is accomplished by a~~ sampling function $\texttt{Sample}$ that generates independent samples from a given distribution [line 5, Alg. 2]; see Algorithm 3.

2) Constructing a state $q_P^{new} = (\mathbf{x}^{new}, q_B^{new})$: Next, given the state $\mathbf{x}^{\text{rand}} \in \mathcal{W}_{\text{free}}^N$, we construct a state $q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$ that belongs to $\mathcal{W}^N \times \mathcal{Q}_B$ which will ~~then~~ be examined if it can be added to the tree $\mathcal{G}_T$.

First, we construct the state $\mathbf{x}^{\text{new}} \in \mathcal{W}^N$. To do so, we first find the ~~vertices~~ $Q_P^{\text{nearest}} \subseteq \mathcal{V}_T$, which consists ~~of~~ $q_P^{\text{nearest}} = (\mathbf{x}^{\text{nearest}}, q_B^{\text{nearest}}) \in \mathcal{V}_T$, that ~~is the~~ closest to $\mathbf{x}^{\text{rand}}$ in terms of a given distance function [line 6, Alg. 2]. ~~Note~~ that Q_P^{nearest} may include multiple vertices due to the same position $\mathbf{x}^{\text{nearest}}$. ~~In this paper, Euclidean distance is used. This is accomplished by~~ the function $\texttt{Nearest} : \mathcal{W}^N \to \mathcal{V}_T$ defined as

$$\texttt{Nearest}(\mathbf{x}^{\text{rand}}) = \underset{q_P \in \mathcal{V}_T}{\operatorname{argmin}} \left\| \mathbf{x}^{\text{rand}} - \Pi|_{\mathcal{W}^N} q_P \right\|,$$

where, $\Pi|_{(\cdot)}(\cdot)$ stands for the projection operator, i.e., $\Pi|_{\mathcal{W}^N} q_P \in \mathcal{W}^N$ and $\Pi|_{\mathcal{Q}_B} q_P \in \mathcal{Q}_B$.

Next, given the state $\mathbf{x}^{\text{nearest}} \in \mathcal{W}_{\text{free}}^N$, we construct the state $\mathbf{x}^{\text{new}} \in \mathcal{W}^N$ using the function $\texttt{Steer} : \mathcal{W}_{\text{free}}^N \times \mathcal{W}^N \to \mathcal{W}^N$ [line 7, Alg. 2]. ~~Given the configuration $\mathbf{x}^{\text{nearest}}$~~ the function $\texttt{Steer}$ returns a configuration $\mathbf{x}^{\text{new}}$ that is "closer" to $\mathbf{x}^{\text{rand}}$ than $\mathbf{x}^{\text{nearest}}$ is. The configuration $\mathbf{x}^{\text{new}}$ has to be within distance at most equal to $\epsilon > 0$ from $\mathbf{x}^{\text{nearest}}$, i.e.,

$$\left\| \mathbf{x}^{\text{nearest}} - \mathbf{x}^{\text{new}} \right\| \leq \epsilon,$$

where ϵ is user-specified, and should also satisfy

$$\left\| \mathbf{x}^{\text{new}} - \mathbf{x}^{\text{rand}} \right\| \leq \left\| \mathbf{x}^{\text{nearest}} - \mathbf{x}^{\text{rand}} \right\|.$$

Second, we construct the state q_P^{new}. First, ~~check~~ whether $\mathbf{x}^{\text{new}} \in \mathcal{W}_{\text{free}}^N$. If not, ~~go~~ back to the $\texttt{Sample}$ step. Let $\mathcal{Q}_B(b)$ be a candidate Büchi state to be attached to $\mathbf{x}^{\text{new}}$, where $\mathcal{Q}_B(b)$ denotes the b-th state ~~of in~~ the set $\mathcal{Q}_B$ for $b \in \{1, \ldots, |\mathcal{Q}_B|\}$ [line 9, Alg. 2]. Appending the state $q_B^{\text{new}} = \mathcal{Q}_B(b) \in \mathcal{Q}_B$ to $\mathbf{x}^{\text{new}} \in \mathcal{W}_{\text{free}}^N$, we construct the state $q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$. Next, we examine if we can extend the tree $\mathcal{G}_T$ towards the constructed state q_P^{new} [line 11, Algorithm 2]. This is accomplished by the function $\texttt{Extend}$ described in Algorithm 4. This procedure is repeated for all states $(\mathbf{x}^{\text{new}}, \mathcal{Q}_B(b))$, with $b \in \{1, \ldots, |\mathcal{Q}_B|\}$ [line 8, Alg. 2]. In what follows, we describe the function $\texttt{Extend}$ for a given state $q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$.

3) Extend $\mathcal{G}_T$ towards q_P^{new}: The first step in Algorithm 4 is to construct the set $\mathcal{P}^{\text{near}} \subseteq \mathcal{V}_T$ [line 2, Alg. 4] that collects all states $q_P^{\text{nearest}} = (\mathbf{x}^{\text{nearest}}, q_B^{\text{nearest}}) \in Q_P^{\text{nearest}}$, ~~and~~ $\tilde{q}_P^{\text{near}} = (\tilde{\mathbf{x}}^{\text{near}}, \tilde{q}_B^{\text{near}}) \in \mathcal{V}_T$ that ~~satisfy $\|\tilde{\mathbf{x}}^{\text{near}} - \mathbf{x}^{\text{new}}\| \leq r(\mathcal{V}_T)$, which is complished by function $\texttt{Near}$,~~

$$\texttt{Near}(q_P^{\text{new}}, \mathcal{G}_T) = \{\tilde{q}_P^{\text{near}} = (\tilde{\mathbf{x}}^{\text{near}}, \tilde{q}_B^{\text{near}}) \in \mathcal{V}_T \mid \|\tilde{\mathbf{x}}^{\text{near}} - \mathbf{x}^{\text{new}}\| \leq r(\mathcal{V}_T)\}, \quad (3)$$

where the radius $r(\mathcal{V}_T)$ is selected as

$$r(\mathcal{V}_T) = \min \left\{ \gamma \left(\frac{\log |\mathcal{V}_T|}{|\mathcal{V}_T|} \right)^{1/\dim}, \epsilon \right\}. \quad (4)$$

In (4), dim is the dimension of the space that states $\mathbf{x} \in \mathcal{W}_{\text{free}}^N \subseteq \mathbb{R}^{\dim}$ reside in, i.e., $\dim := dN$, and γ is a constant

Algorithm 3: Function $\texttt{Sample}(\mathcal{W}, \mathcal{V}_T)$

1 $\mathbf{x}^{\text{rand}} = [\ ]$;
2 **for** $i = 1 : 1 : N$ **do**
3 $\quad$ Draw a sample $\mathbf{x}_i$ from a continuous distribution on $\mathcal{W}$;
4 $\quad \mathbf{x}^{\text{rand}} = [\mathbf{x}^{\text{rand}}, \mathbf{x}_i]$;
5 **return** $\mathbf{x}^{rand}$;

[2] In what follows, for the sake of simplicity we assume that the set of initial states $\mathcal{Q}_B^0$ consists of only one state, i.e., $\mathcal{Q}_B^0 = \{q_B^0\}$. In case there are more than one initial states in B, then the lines 2-6 in Alg. 1 should be executed for each possible initial state q_B^0.

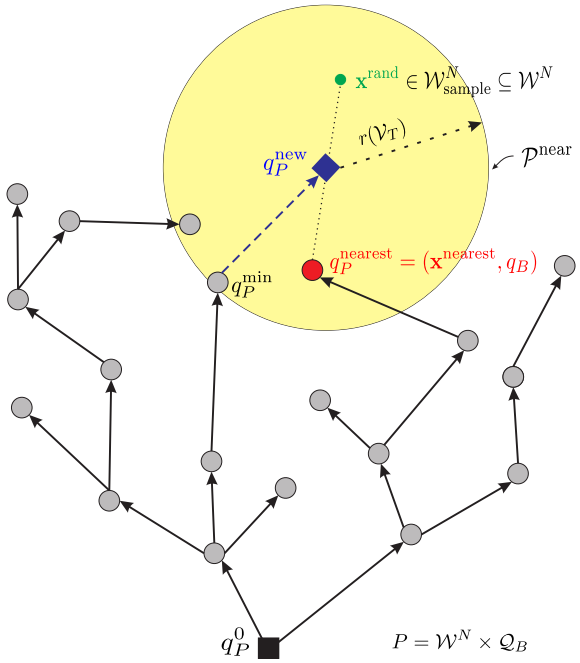


Fig. 1. Graphical depiction of Algorithm 4. The black square stands for the root of the tree and the gray and red disks represent nodes in the set $\mathcal{V}_T$. Black arrows represent transitions captured by $\mathcal{E}_T$. The small green circle stands for the state $\mathbf{x}^{\text{rand}}$ generated by Algorithm 3. The blue diamond and the red circle stand for the states q_P^{new} and q_P^{nearest}, respectively. The dashed blue arrow represents the new edge that will be added to the set $\mathcal{E}_T$ after the execution of Algorithm 4.

Algorithm 4: Function $\texttt{Extend}(q_P^{\text{new}}, Q_P^{\text{nearest}}, \mathcal{G}_T)$

1 $\texttt{added} = 0$;
2 $\mathcal{P}^{\text{near}} = \texttt{Near}(q_P^{\text{new}}, \mathcal{G}_T) \cup Q_P^{\text{nearest}}$;
3 $c = \infty$;
4 **for** *all* $q_P^{near} = (\mathbf{x}^{near}, q_B^{near}) \in \mathcal{P}^{near}$ **do**
5 **if** $\texttt{ObstacleFree}(\mathbf{x}^{new}, \mathbf{x}^{near}) \wedge (q_B^{near}, L(\mathbf{x}^{new}), q_B^{new}) \in \rightarrow_B$ **then**
6 $c' = \texttt{Cost}(q_P^{\text{near}}) + C(\mathbf{x}^{\text{near}}, \mathbf{x}^{\text{new}}, \mathbf{u})$;
7 **if** $c' \leq c$ **then**
8 $\texttt{added} = 1$;
9 $q_P^{\text{min}} = q_P^{\text{near}}$;
10 $c = c'$;
11 $\texttt{Cost}(q_P^{\text{new}}) = c'$;
12 **if** $\texttt{added} = 1$ **then**
13 $\mathcal{V}_T = \mathcal{V}_T \cup \{q_P^{\text{new}}\}$;
14 $\mathcal{E}_T = \mathcal{E}_T \cup \{q_P^{\text{min}}, q_P^{\text{new}}\}$
15 **return** $\mathcal{V}_T$, $\mathcal{E}_T$, $\mathcal{C}_T$, $\texttt{added}$;

selected as

$$\gamma > 4 \left(\frac{\mu(\mathcal{W}^N_{\text{free}})}{\zeta_{\text{dim}}} \right)^{1/\text{dim}}, \tag{5}$$

where $\mu(\mathcal{W}^N_{\text{free}})$ stands for the measure of the set $\mathcal{W}^N_{\text{free}}$, and ζ_{dim} is the volume of the unit sphere in $\mathbb{R}^{\text{dim}}$. This specific selection of the radius $r(\mathcal{V}_T)$ will be discussed in Theorem 5.7.

Next, in lines 3-11 of Algorithm 4, we select the parent of the state q_P^{new} among the states $q_P^{\text{near}} \in \mathcal{P}^{\text{near}}$. Specifically, for each state $q_P^{\text{near}} = (\mathbf{x}^{\text{near}}, q_B^{\text{near}}) \in \mathcal{P}^{\text{near}}$, we

Algorithm 5: Function $\texttt{Rewire}(q_P^{\text{new}}, Q_P^{\text{nearest}}, \mathcal{G}_T)$

1 **for** *all* $q_P^{near} = (\mathbf{x}_P^{near}, q_B^{near}) \in \mathcal{P}^{near} = \texttt{Near}(q_P^{new}, \mathcal{G}_T) \cup Q_P^{nearest}$ **do**
2 **if** $\texttt{ObstacleFree}(\mathbf{x}^{new}, \mathbf{x}^{near}) \wedge (q_B^{new}, L(\mathbf{x}^{new}), q_B^{near}) \in \rightarrow_B$ **then**
3 **if** $\texttt{Cost}(q_P^{near}) > \texttt{Cost}(q_P^{new}) + C(\mathbf{x}_P^{near}, \mathbf{x}_P^{new}, \mathbf{u})$ **then**
4 $\texttt{Cost}(q_P^{\text{near}}) = \texttt{Cost}(q_P^{\text{new}}) + C(\mathbf{x}_P^{\text{near}}, \mathbf{x}_P^{\text{new}}, \mathbf{u})$;
5 $\mathcal{E}_T = \mathcal{E}_T \setminus \{(\texttt{Parent}(q_P^{\text{near}}), q_P^{\text{near}})\}$;
6 $\mathcal{E}_T = \mathcal{E}_T \cup \{(q_P^{\text{new}}, q_P^{\text{near}})\}$;
7 **return** $\mathcal{E}_T$, $\texttt{Cost}$;

check ~~if~~ (i) ~~the~~ straight line starting from position $\mathbf{x}^{\text{near}}$ and ending at position $\mathbf{x}^{\text{new}}$ ~~are~~ inside $\mathcal{W}^N_{\text{free}}$, and the number of times each robot trajectory crosses the boundary of any region $\ell_j \in \mathcal{R}$ is at most one; this is accomplished by the function $\texttt{ObstacleFree}(\mathbf{x}^{\text{near}}, \mathbf{x}^{\text{new}})$, ~~and (ii)~~ $(q_B^{\text{near}}, L(\mathbf{x}^{\text{near}}), q_B^{\text{new}}) \in \rightarrow_B$ [line 5, Algorithm 4]. Notice that ~~the~~ requirement (i) guarantees that if the ~~resulting~~ discrete plan satisfies a LTL formula ϕ, then so does its continuous ~~path,~~ provided all robots move synchronously towards their next positions, as per the generated discrete plan. If (i) and (ii) hold, then, we call such transitions *feasible* and q_P^{near} is a *candidate parent* of q_P^{new}. The cost of the state q_P^{new} is [line 11, Algorithm 4]:[3]

$$\texttt{Cost}(q_P^{\text{new}}) = \underbrace{\texttt{Cost}(q_P^{\text{near}})}_{\text{Cost of reaching } q_P^{\text{near}} \text{ from the root } q_P^0 \text{ of the tree } \mathcal{G}_T} + \underbrace{C(\mathbf{x}^{\text{near}}, \mathbf{x}^{\text{new}})}_{\text{cost of reaching } q_P^{\text{new}} \text{ from } q_P^{\text{near}}}. \tag{6}$$

Among all candidate parents q_P^{near} of q_P^{new}, we pick the one that incurs the minimum cost $\texttt{Cost}(q_P^{\text{new}})$, denoted by q_P^{min} [lines 7-11, Algorithm 4]. If the set of candidate parents is not empty [line 12, Algorithm 4], then the sets $\mathcal{V}_T$, $\mathcal{E}_T$ are updated as: $\mathcal{V}_T = \mathcal{V}_T \cup \{q_P^{\text{new}}\}$ and $\mathcal{E}_T = \mathcal{E}_T \cup \{q_P^{\text{min}}, q_P^{\text{new}}\}$ [lines 13-14, Algorithm 4]. Algorithm 4 is illustrated in Figure 1, as well. If the state q_P^{new} is added to the set of nodes $\mathcal{V}_T$, the *rewiring* process follows [line 13, Algorithm 2].

4) Rewiring through q_P^{new}: Once a new state $q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$ has been added to the tree, we *rewire* the nodes in $q_P^{\text{near}} \in \mathcal{P}^{\text{near}}$ through the node q_P^{new} [line 13, Alg. 2] if this will decrease the cost $\texttt{Cost}(q_P^{\text{near}})$, i.e., if this will decrease the cost of reaching the node q_P^{near} starting from the root q_P^0 of the tree. The rewiring process is described in Algorithm 5 and is illustrated in Figure 2.

Specifically, for each state $q_P^{\text{near}} \in \mathcal{P}^{\text{near}}$, we first check if a transition from q_P^{near} to q_P^{new} is *feasible*, i.e., if $\texttt{ObstacleFree}(\mathbf{x}^{\text{new}}, \mathbf{x}^{\text{near}})$ is true and if $(q_B^{\text{new}}, L(\mathbf{x}^{\text{new}}), q_B^{\text{near}}) \in \rightarrow_B$. If this is the case, then we check if the cost of q_P^{near} will decrease due to rewiring through q_P^{new}, i.e., we check if $\texttt{Cost}(q_P^{\text{near}}) > \texttt{Cost}(q_P^{\text{new}}) + C(\mathbf{x}_P^{\text{new}}, \mathbf{x}_P^{\text{near}})$

[3]Notice in (6) that $\texttt{Cost}(q_P^{\text{new}})$ captures the cost of the path in $\mathcal{G}_T$ that connects the state q_P^{new} to the root q_P^0, i.e., if $q_P^{\text{new}} \in \mathcal{X}_{\text{goal}}^{\text{pre}}$ then $\texttt{Cost}(q_P^{\text{new}})$ captures the cost of a prefix part.

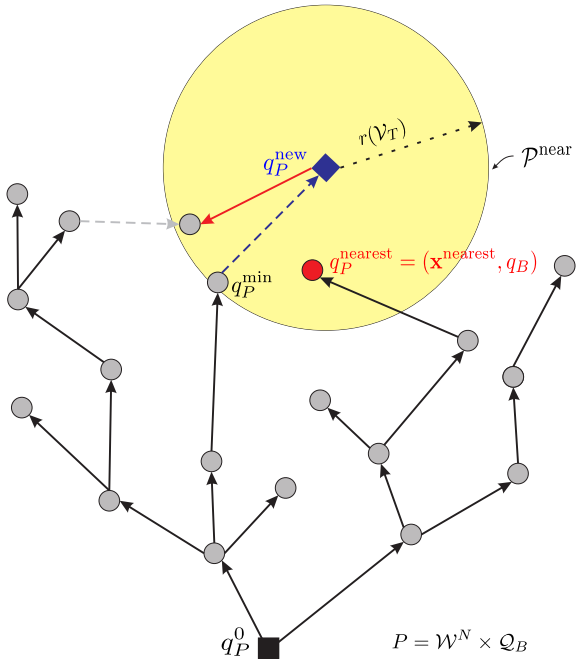


Fig. 2. Graphical depiction of Algorithm 5. The black square stands for the root of the tree and the gray and red disks and the blue diamond represent nodes in the set $\mathcal{V}_T$. Black arrows represent transitions captured by $\mathcal{E}_T$. The blue diamond stands for the state q_P^{new} that was added to $\mathcal{V}_T$. The dashed gray arrow stand for the edge that will be deleted from the set $\mathcal{E}_T$ during the execution of Algorithm 5. The red arrow stand for the new edge that will be added to $\mathcal{E}_T$ during the execution of Algorithm 5.

[line 2, Alg. 5]. If so, then the cost of the node q_P^{near} is updated as [line 4, Alg. 5]

$$\texttt{Cost}(q_P^{\text{near}}) = \texttt{Cost}(q_P^{\text{new}}) + C(\mathbf{x}_P^{\text{near}}, \mathbf{x}_P^{\text{new}}),$$

and the parent of q_P^{near} becomes the node q_P^{new} [line 5-6, Alg. 5].

5) Construction of Paths: The construction of the tree $\mathcal{G}_T$ ends after $n_{\max}^{\text{pre}}$ iterations, where $n_{\max}^{\text{pre}}$ is user specified [line 4, Alg. 2]. Then we construct the set $\mathcal{P} = \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}^{\text{pre}}$ [line 14, Algorithm 2] that collects all the states $q_P \in \mathcal{V}_T$ that belong to the goal region $\mathcal{X}_{\text{goal}}^{\text{pre}}$. Given the tree $\mathcal{G}_T$ and the set $\mathcal{P}$ [line 4, Alg. 1] that collects all states $q_P \in \mathcal{X}_{\text{goal}}^{\text{pre}} \cap \mathcal{V}_T$, we can compute the prefix plans [lines 5-6, Alg. 1]. In particular, the path that connects the α-th state of the set $\mathcal{P}$, denoted by $\mathcal{P}(\alpha)$, to the root q_P^0 constitutes the α-th prefix plan and is denoted by $\tau^{\text{pre},\alpha}$ [line 6, Algorithm 1]. Its computation is described in Algorithm 6. Note that for the computation of $\tau^{\text{pre},\alpha}$ in Algorithm 6 only the parent of each node in the graph $\mathcal{G}_T$ is required, due to the tree structure of $\mathcal{G}_T$. Specifically, the prefix plan $\tau^{\text{pre},\alpha}$ is constructed by finding the parent of a node $q_P \in \mathcal{V}_T$ starting from the node that represents the state $\mathcal{P}(\alpha)$, until the root of the tree is reached [lines 1-6, Alg. 6]. The parent of each node is computed by the function $\texttt{parent} : \mathcal{V}_T \to \mathcal{V}_T$ that maps a node $q_P \in \mathcal{V}_T$ to a unique vertex $q_P' \in \mathcal{V}_T$ if $(q_P', q_P) \in \mathcal{E}_T$, i.e., $\texttt{parent}(q_P) = q_P'$ if $(q_P', q_P) \in \mathcal{E}_T$. By convention, we assume that $\texttt{parent}(q_P^0) = q_P^0$, where q_P^0 is the root of the tree $\mathcal{G}_T$. In line 4 of Algorithm 6, $|$ stands for the concatenation of paths in $\mathcal{W}^N$, and $\mathbf{x}^{\text{prev}}$ is defined as $\mathbf{x}^{\text{prev}} = \Pi|_{\mathcal{W}^N} q_P^{\text{prev}}$. Thus, for the resulting prefix plan $\tau^{\text{pre},\alpha}$, it holds

Algorithm 6: Function $\texttt{FindPath}(\mathcal{G}_T, q_P^{\text{initial}}, q_P^{\text{goal}})$

```
1 path = x^goal;
2 q_P^prev = Parent(q_P^goal);
3 while q_P^prev ≠ q_P^initial do
4 |   path = x^prev | path;
5 |   q_P^prev = Parent(q_P^prev);
6 path = x^initial | path;
7 return path;
```

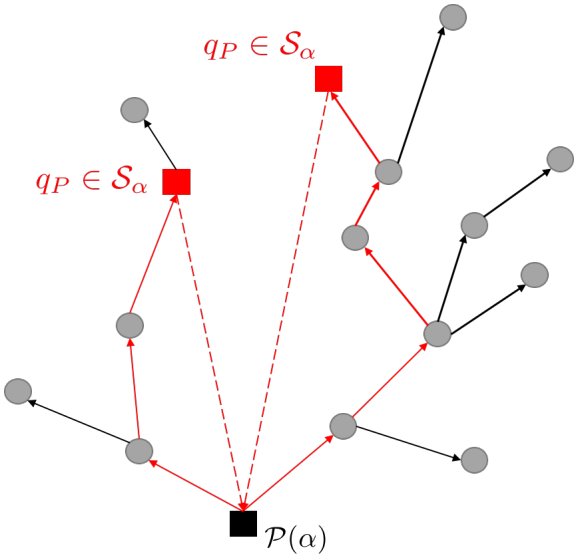


Fig. 3. Graphical depiction of detecting cycles around an accepting state $\mathcal{P}(\alpha)$ (black square) which acts as the root of the tree. The red diamonds stand for states $q_P \in \mathcal{X}_{\text{goal}}^{\text{suf}} \cap \mathcal{V}_T = \mathcal{S}_\alpha$. Solid red arrows stand for the paths that connect the states $q_P \in \mathcal{S}_\alpha$ to the root $\mathcal{P}(\alpha)$. The dashed red arrows imply that a transition from q_P to $\mathcal{P}(\alpha)$ is feasible; however, such transitions are not included in the set $\mathcal{E}_T$. The detected cycles around the final state $\mathcal{P}(\alpha)$ is illustrated by solid and dashed red arrows.

that $\tau^{\text{pre},\alpha}(1) = \Pi|_{\mathcal{W}^N} q_P^0$ and $\tau^{\text{pre},\alpha}(|\tau^{\text{pre},\alpha}|) = \Pi|_{\mathcal{W}^N} \mathcal{S}_\alpha$.

Notice that the computational complexity of constructing the prefix plan $\tau^{\text{pre},\alpha}$ is $O(|\mathcal{V}_T|)$. On the other hand, if the product state-space $\mathcal{W}^N \times \mathcal{Q}_B$ was represented by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ of arbitrary structure, then the computational complexity of the Dijkstra algorithm to find the optimal path $\tau^{\text{pre},\alpha}$ that connects the state $\mathcal{S}_\alpha$ to the root with the minimum cost is $O(|\mathcal{E}| + |\mathcal{V}| \log(|\mathcal{V}|))$, where $|\mathcal{E}| + |\mathcal{V}| \log(|\mathcal{V}|) > |\mathcal{V}|$.

B. Construction of Suffix Plans

Once the prefix plans $\tau^{\text{pre},\alpha}$ are constructed, the construction of the respective suffix plans $\tau^{\text{pre},\alpha}$ follows [lines 7-18, Alg. 1]. The suffix plan $\tau^{\text{pre},\alpha}$ is a sequence of states in $\mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$ that starts from the state $\mathcal{P}(\alpha)$, $\alpha \in \{1, \ldots, |\mathcal{P}|\}$, and ends at the same state $\mathcal{P}(\alpha)$, i.e., a cycle around state $\mathcal{P}(\alpha)$, where any two consecutive states in this sequence should be feasible. Projection of this sequence on $\mathcal{W}_{\text{free}}^N$ gives the suffix plan $\tau^{\text{suf},\alpha}$.

For this purpose, we build a tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$ that explores the product state-space $P = \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$, similarly as in Section IV-A. The only differences are that: (i) the root of the tree is now $q_P^0 = \mathcal{P}(\alpha)$, i.e., it is an *accepting* state [line

8, Alg. 1] detected during the construction of the prefix plans, (ii) the goal region, given the root q_P^0, is defined as

$$\begin{aligned}\mathcal{X}_{\text{goal}}^{\text{suf}}(q_P^0) =&\{q_P = (\mathbf{x},\ q_B) \in \mathcal{W}^N \times \mathcal{Q}_B \mid \\ &(q_B, L(\mathbf{x}), \Pi|_{\mathcal{Q}_B} q_P^0) \in \rightarrow_B \\ &\wedge\ \texttt{ObstacleFree}(\mathbf{x}, \Pi|_{\mathcal{W}^N} q_P^0)\},\end{aligned} \tag{7}$$

i.e., it collects all states q_P for which a transition from q_P to the root q_P^0 is feasible, but is not included in $\mathcal{E}_T$, and (iii) we first check if $q_P^0 \in \mathcal{X}_{\text{goal}}^{\text{suf}}$, i.e., if $(\Pi|_{\mathcal{Q}_B} q_P^0, L(\Pi|_{\mathcal{W}_{\text{free}}^N} q_P^0), \Pi|_{\mathcal{Q}_B} q_P^0)$ [line 10, Alg. 1]. If so, the construction of the tree is trivial, as it consists of only the root, and a loop around it with zero cost [line 11, Alg. 1].[4] If $q_P^0 \notin \mathcal{X}_{\text{goal}}^{\text{suf}}$, then the tree $\mathcal{G}_T$ is constructed by Algorithm 2 [line 14, Alg. 1].

Once a tree rooted at $q_P^0 = \mathcal{P}(\alpha)$ is constructed, a set $\mathcal{S}_\alpha \subseteq \mathcal{V}_T$ is constructed, as well, that collects all states $q_P \in \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}^{\text{suf}}(q_P^0)$ [lines 12, 14, Alg. 1]. Then for each state $q_P \in \mathcal{S}_\alpha$, we compute the cost $J(\tilde{\tau}^{\text{suf,e}})$ of each possible suffix plan $\tilde{\tau}^{\text{suf,e}}$, for all $e \in \{1, \ldots, |\mathcal{S}_\alpha|\}$, associated with the root q_P^0. By construction of the cost functions $\texttt{Cost}$ and $J(\cdot)$, it holds that $J(\tilde{\tau}^{\text{suf,e}}) = \texttt{Cost}(\mathcal{S}_\alpha(e)) + C(\Pi|_{\mathcal{W}^N}\mathcal{S}_\alpha(e), \Pi|_{\mathcal{W}^N} q_P^0)$, where $\mathcal{S}_\alpha(e)$ stands for the e-th state in the set $\mathcal{S}_\alpha$. Among all detected suffix plans $\tilde{\tau}^{\text{suf,e}}$ associated with the *accepting* state $\mathcal{P}(\alpha)$, we pick the suffix plan with the minimum cost, which constitutes the suffix plan $\tau^{\text{suf},\alpha}$ [lines 17-18, Alg. 1]. This process is repeated for all $\alpha \in \{1, \ldots, |\mathcal{P}|\}$ [line 7, Alg. 1]. In this way, for each prefix plan $\tau^{\text{pre},\alpha}$ we construct its respective suffix plan $\tau^{\text{suf},\alpha}$, if it exists.

C. Construction of Optimal Discrete Plan

By construction, any motion plan $\tau^\alpha = \tau^{\text{pre},\alpha}[\tau^{\text{suf},\alpha}]^\omega$, with $\mathcal{S}_\alpha \neq \emptyset$, $\alpha \in \{1, \ldots, |\mathcal{P}|\}$ satisfies the global LTL specification ϕ. The cost $J(\tau^\alpha)$ of each plan τ^α is defined in (1). Among all the motion plans $\tau^\alpha \models \phi$, we pick the one with the smallest cost $J(\tau^\alpha)$ denoted by τ, i.e., $\tau = \tau^{\alpha_*}$, where $\alpha_* = \text{argmin}_\alpha J(\tau^\alpha)$ [lines 19-20, Alg. 1].

V. Correctness and Optimality

In this section ~~the probabilistic completeness and asymptotic optimality of the proposed algorithm pRRT* is analysed. First, some reasonable assumptions are made, which is required to establish the desired properties. Second, the proof of probabilistic completeness of pRRT*is provided. We put forward a RRT based motion planning method, denoted as pRRT, which is very similar to the proposed algorithm, except the missing step of extending and rewiring when incrementally constructing the tree. The probabilistic completeness of pRRT* can directly follow from that of pRRT. Third, the proof of asymptotic optimality based on that of RRT* is provided. Since pRRT*is much like RRT*, execpt constructing the tree in the product state space, it turns out that the proof of asymptotic optimality of RRT* can be employed in our case.~~

[4]The reason that we terminate the construction of the tree if a suffix plan, associated with the accepting state $\mathcal{P}(\alpha)$, with zero cost is found is because any other suffix plan with non-zero cost will be discarded in the construction of the optimal suffix plan associated with $\mathcal{P}(\alpha)$ in lines 17-18 of Algorithm 1.

~~Two reasonable assumptions are stated as follows. When people assign robots a certain task to visit a destination, which, in most cases, can't be a single point or uncountably many points but with measure zero. Physically, we couldn't manipulate anything in a single point or zero measure space, since robots or tools take up certain non-zero measure physical space. Furthermore, a homotopic class of paths exsits, meaning any path in th class can continuously deform into the other. This is because a feasible path resides in the physical space taking up nonzero volume. For instance, a robot can pass through a corridor in infinite paths. Before stating the assumptions formally, recall that an LTL formula capturing a temporal task is defined over a set of atomic propositions $\mathcal{AP} = \{\{\pi_i^j\}_{j=1}^W\}_{i=1}^N$, where π_i^j is true if robot i is inside region l_j and l_j represents certain physical region in the problem we aim to solve.~~

Assumption 5.1 (Nonpoint region): Every atomic proposition in the LTL formula ~~stands for~~ a nonpoint region, ~~more~~ precisely, $\mu(l_j) > 0$ where μ ~~is~~ Lebesgue measure.

Assumption 5.2 (Convergence space): For any product state $(\mathbf{x}, q_B) \subseteq \mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$ which can be reached from the root, there exists a constant $\delta_{\mathbf{x}} > 0$, ~~where $\delta_{\mathbf{x}}$ means~~ δ depends on $\mathbf{x}$, such that any point in the interior of the ball, ~~radius~~ $\delta_{\mathbf{x}}$ centered at $\mathbf{x}$, ~~lying within~~ $\mathcal{W}_{\text{free}}^N$, ~~i.e.,~~ $\texttt{int}(\mathcal{B}_{\delta_{\mathbf{x}}}(\mathbf{x})) \subset \mathcal{W}_{\text{free}}^N$, can ~~pair with~~ same Büchi state q_B, ~~constituting a product state that can be reached from the root, as well.~~

Assumption 5.1 ensures that the algorithm can sample a point within a labeled region with probability larger than 0, ~~otherwise,~~ it's impossible to generate any feasible ~~plan.~~ Assumption 5.2 ensures that there exists ~~certain space~~ around ~~the~~ feasible plan ~~making the~~ homotopy class exist, ~~and also certain space around the optimal plan making convergence feasible.~~

~~Before we give theoretical proof of probabilistic completeness and asymptotic optimality of our algorithm, note that when we say a product state in a ball or some physical space, it means the position part lies in a ball or some physical space. And the distance between two product states means the Euclidean distance between the position part of two product states.~~

A. Probabilistic Completeness

~~The notation of probabilistic completeness is states as follows. Its proof relies on RRT based motion planning algorithm.~~

Theorem 5.3 (Probabilistic Completeness): Algorithm 1 is probabilistically complete, i.e., if there exists a motion plan τ that satisfies a given LTL formula ϕ, ~~as required to solve Problem 1,~~ then Algorithm 1 will find ~~a feasible plan~~ with probability 1.

~~An RRT based motion planning is put forward here. The~~ function $\texttt{ConstructTree}$ ~~of pRRT algorithm is stated in Algorithm 7, which is the main difference from the pRRT*. Function~~ $\texttt{Nearest}$ [line 6, Alg. 7] returns a set of vertices ~~that~~ are closest to $\mathbf{x}^{\text{rand}}$ in the Euclidean distance, ~~meaning they all have the same configuration state, denoted as~~ $\mathbf{x}^{\text{nearest}}$. In lines 11-15 of Algorithm 7, we select the parent of the state

Algorithm 7: $\texttt{Function}[\mathcal{G}_T, \mathcal{Z}] = \texttt{ConstructTree}(\mathcal{X}_{\text{goal}}, E, B, q_P^0, n_{\max})$

Input : Initial state q_P^0, Goal region $\mathcal{X}_{\text{goal}}$, tuple $E = (\mathcal{W}^N, \text{ap}, L, C)$, NBA B, maximum number of iterations $n_{\max}$

Output: Tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$, set $\mathcal{Z} = \{q_P \in \mathcal{V}_T \,|\, q_P \in \mathcal{X}_{\text{goal}}\}$

$\mathcal{V}_T = \{q_P^0\}$;
$\mathcal{E}_T = \emptyset$;
$\texttt{Cost}(q_P^0) = 0$;
for $n = 1 : 1 : n_{max}$ **do**
  $\mathbf{x}^{\text{rand}} = \texttt{Sample}(E, \mathcal{V}_T)$;
  $Q_P^{\text{nearest}} = \texttt{Nearest}(\mathcal{G}_T, \mathbf{x}^{\text{rand}})$;
  $\mathbf{x}^{\text{new}} = \texttt{Steer}(\mathbf{x}^{\text{nearest}}, \mathbf{x}^{\text{rand}})$;
  for $b = 1 : |\mathcal{Q}_B|$ **do**
    $q_B^{\text{new}} = \mathcal{Q}_B(b)$;
    $q_P^{\text{new}} = (\mathbf{x}^{\text{new}}, q_B^{\text{new}})$;
    for $q_P^{nearest} = (\mathbf{x}^{nearest}, q_B^{nearest}) \in Q_P^{nearest}$ **do**
      if $\texttt{ObstacleFree}(\mathbf{x}^{nearest}, \mathbf{x}^{new})$ $\wedge (q_B^{nearest}, L(\mathbf{x}^{nearest}), q_B^{new}) \in \rightarrow_B$ **then**
        $\mathcal{V}_T = \mathcal{V}_T \cup \{q_P^{\text{new}}\}$;
        $\mathcal{E}_T = \mathcal{E}_T \cup \{(q_P^{\text{nearest}}, q_P^{\text{new}})\}$;
        break;
$\mathcal{Z} = \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}$;

q_P^{new} among the ~~states~~ Q_P^{nearest}. Once a parent is found, the for loop breaks. If q_P^{new} can be connected to the tree through more than one ~~state~~ in Q_P^{nearest}, ~~it's fine to select the first candidate parent,~~ because they all respect the specification and incur the same ~~distance cost of~~ q_P^{new}. Let $\mathcal{V}_n^{\text{pRRT}^*}$ and $\mathcal{V}_n^{\text{pRRT}}$ denote the set of nodes at the n-th iteration for the pRRT* and pRRTalgorithm, respectively. By construction, $\mathcal{V}_n^{\text{pRRT}^*} \subseteq \mathcal{V}_n^{\text{pRRT}}, \forall n \in \mathbb{N}$. ~~Hence, the probabilistic completeness of pRRT* directly from that of pRRT.~~

Before ~~presenting the proof of probabilistic completeness in detail, some notation are provided as follows.~~

Let P denote the subspace of $\mathcal{W}_{\text{free}}^N \times \mathcal{Q}_B$, consisting of states that can be reached from the root. The states in P can be divided into two categories, P_r and P_v. P_r contains the states that ~~respect the~~ specifcation, whereas any state in P_v violates the specification. More precisely, suppose a state $(\mathbf{x}, q_B)$ ~~is on~~ the tree and another state $(\mathbf{x}', q_B')$ can be connected to the tree ~~through it. However,~~ the label generated by $\mathbf{x}'$ ~~couldn't make any~~ Büchi state transition starting from q_B' ~~enabled,~~ then node $(\mathbf{x}', q_B')$ ~~will be the dead-end forever.~~ ~~Here,~~ $(\mathbf{x}, q_B) \in P_r$ and $(\mathbf{x}', q_B') \in P_v$. In the rest of the paper, ~~the term "leaf" only refers to the dead-end, not the normal sense that the node with no children.~~

~~For a state~~ $q_P = (\mathbf{x}, q_B) \in P_r$, $D_k(q_P)$, ~~associated with a tree of k nodes,~~ ~~denotes~~ a random variable whose value is the Euclidean distance ~~of~~ q_P ~~to~~ the nearest candidate parent node $q_P^{\text{ncp}} = (\mathbf{x}^{\text{ncp}}, q_B^{\text{ncp}})$, ~~meaning the state q_P can be connected to the tree through q_P^{nep}.~~ One specific ~~observation~~ of D_k is denoted as d_k.

Lemma 5.4 states ~~that for any~~ product state $q_P = (\mathbf{x}, q_B) \in P_r$ that can be reached in one step, ~~the tree rooted at $q_P^0 = (\mathbf{x}^0, q_B^0)$ will grow arbitrarily close toward it.~~

Lemma 5.4: ~~For any state~~ $q_P = (\mathbf{x}, q_B) \in P_r$ ~~that can be established a one-step connection with~~ the root $(\mathbf{x}_0, q_B^0)$, ~~i.e. (i) $\texttt{ObstacleFree}(\mathbf{x}_0, \mathbf{x})$ is true, (ii) $q_B^0 \xrightarrow{L(\mathbf{x}_0)} q_B$, which means the root is the candidate parent of q_P, and~~ for any positive constant $\delta > 0$,

$$\lim_{k\to\infty} \mathbb{P}(D_k(q_P) < \delta) = 1. \tag{8}$$

Meanwhile, the probability that there exists a node in the tree, which shares the same Büchi state as q_P and lies within ~~the ball of radius δ centered at~~ q_P, ~~goes to 1 as iteration increases to infinity, i.e.,~~

$$\lim_{k\to\infty} \mathbb{P}\left(\left\{\exists\, q_P' = (\mathbf{x}', q_B) \in \mathcal{V}_k^{\text{pRRT}} : \|\mathbf{x} - \mathbf{x}'\| < \delta\right\}\right) = 1. \tag{9}$$

~~In other words, for any one-step reachable state, the tree has a node which is almost identical to it in the sense that they share the same Büchi state and are arbitrarily close to each other in terms of Eulidean distance.~~

Proof: Recall that $D_k(q_P)$ denotes the Euclidean distance ~~of~~ q_P ~~to~~ the nearest candidate parent node in the tree with k nodes. For simplicity, ~~let~~ D_k ~~denote~~ $D_k(q_P)$. The proof ~~process is divided into~~ four steps. ~~The~~ first three steps aim ~~to derive~~ (8), and the last step ~~for~~ (9). (i) ~~Prove~~ that the expectation of D_{k+1} conditioned on the previous observation d_k is ~~samller~~ than that observation, namely, $\mathbb{E}(D_{k+1}|D_k = d_k) < d_k$. (ii) ~~Derive~~ that the sequence $\{\mathbb{E}(D_k)\}$ ~~strictly decreases,~~ i.e. $\mathbb{E}(D_{k+1}) < \mathbb{E}(D_k)$. (iii) ~~Prove~~ that the limit ~~of~~ sequence $\{\mathbb{E}(D_k)\}$ goes to 0, i.e. $\mathbb{E}(D_k) \to 0$. (iv) ~~Derive~~ the probability in (9).

First, note that d_1 is the distance between the root and state q_P. Given the observation d_k, if the newly-added node q_P^{new} is not the candidate parent of ~~q_P~~, we have $D_{k+1} = d_k$. Otherwise, there are two cases. ~~One is that~~ the distance between the ~~newly added~~ node q_P^{new} and q_P may be larger than or ~~euqal~~ to d_k, hence, the minimum distance remains the same, i.e. $D_{k+1} = d_k$. ~~Futhermore, it's also possible that it~~ is smaller than d_k, meaning $D_{k+1} < d_k$. Let p_k denote the probability of ~~event~~ $\{D_{k+1} < d_k\}$, so ~~with probability $1 - p_k$,~~ $\{D_k = d_k\}$ ~~occurs.~~ $\mathbb{E}(D_{k+1})$ can be written as

$$\mathbb{E}(D_{k+1}|D_k = d_k) = (1 - p_k)\, d_k + p_k\, d \tag{10}$$

where $d < d_k$ is a positive number.

Next, we ~~aim to prove~~ that p_k is strictly positive. ~~Now consider one possible scenario where the minimum distance drops, as shown in Figure 4. A~~ ball $\mathcal{B}_{d_k}(q_P)$ ~~of radius d_k is~~ centered at q_P. ~~q_P^{nep} is~~ the nearest candidtate parent node of q_P, ~~located at the boundary of the ball $\mathcal{B}_{d_k}(q_P)$. When $k = 1$,~~ $q_P^{\text{ncp}} = q_P^0$. ~~As tree grows, it's possible that another node becomes new q_P^{nep}.~~ Since ~~q_P^{nep} can transit to q_P,~~ there exists a straight feasible path between ~~them,~~ which is also an edge in the tree. By assumption 5.2, ~~there exists a ball, centered at each state on the path, within which~~ all states have the same Büchi state ~~as the center, then all superposed balls constitute a neighborhood region around the path, referring to the shaded area, denoted as~~ $\mathcal{R}_{d_k}(q_P)$, ~~in the~~ Figure 4.

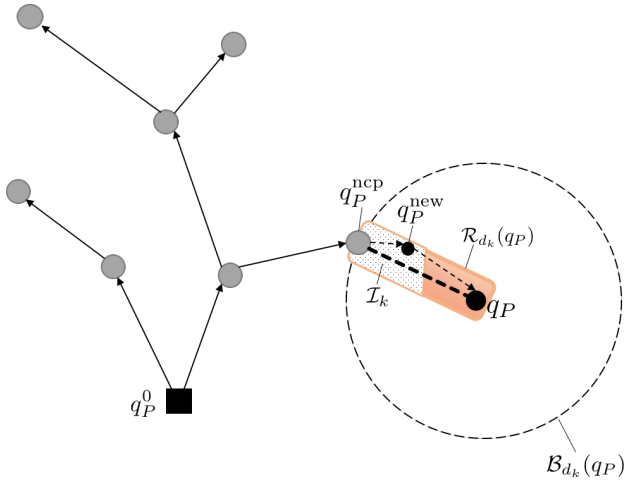


Fig. 4. An illustration of the scenario where the distance between q_P and its nearest candidate parentnode drops. The shaded region denotes the neighborhood around the feasible path depicted as the thick dashed line, from q_P^{ncp} to q_P, and the thin dashed line illustrated alternative of the path on the tree. The dotted area denotes the intersection $\mathcal{I}_k$ of cell $\mathcal{C}(q_P^{\text{ncp}})$ around q_P^{ncp} and neighborhood around the path. q_P^{new} is inside $\mathcal{I}_k$.

Recall that, in the Algorithm 2, to get a new state, the configuration state $\mathbf{x}^{\text{new}}$ is obstained first. It's possbile that $\mathbf{x}^{\text{new}}$ lies within $\mathcal{R}_{d_k}(q_P)$ due to $\mu(\mathcal{R}_{d_k}(q_P)) > 0$. If it occurs, it's highly probable that q_P^{ncp} is the nearest node in the tree. The reason is that, if the whole workspace is partitioned into Voronoi diagram, the cell $\mathcal{C}(q^{\text{ncp}})$ around q_P^{ncp} will intersect $\mathcal{R}_{d_k}(q_P)$ with strictly positive measure, denoted as $\mathcal{I}_k \subseteq \mathcal{R}_{d_k}(q_P)$. In this case, the first condition, $\texttt{ObstacleFree}(\mathbf{x}^{\text{ncp}}, \mathbf{x}^{\text{new}})$, is true. Considering the existence of feasible path $(\mathbf{x}^{\text{ncp}}, q_B^{\text{ncp}}) \to (\mathbf{x}', q_B') \to (\mathbf{x}, q_B)$ where $\mathbf{x}' = \mathbf{x}^{\text{new}}$, q_B' is either q_B^{ncp} or q_B, and traversal of Büchi state space, q_B' will be selected as q_B^{new} and paired with $\mathbf{x}^{\text{new}}$ to constitute q_P^{new}, which can be added to the tree and becomes the new nearest candidate parent of q_P. Hence, $p_k > 0$ in (10) and it becomes;

$$\begin{aligned}\mathbb{E}(D_{k+1}|D_k = d_k) &= (1-p_k)\, d_k + p_k\, d \\ &< (1-p_k)\, d_k + p_k\, d_k \\ &= d_k\end{aligned} \tag{11}$$

Next, prove the inequality $\mathbb{E}(D_{k+1}) < \mathbb{E}(D_k)$. Let $f(d_k)$ denote the probability density function of random variable D_k, and $f(d_{k+1}|d_k)$ denote the conditional probability density function of D_{k+1} conditioned on D_k. Multiplying both sides of (11) by $f(d_k)$ and integrate over the support of D_k, we have

$$\int_{D_k} \mathbb{E}(D_{k+1}|D_k = d_k) f(d_k)\, \mathrm{d}d_k < \int_{D_k} d_k f(d_k)\, \mathrm{d}d_k \tag{12}$$

The left-hand side can be written as

$$\begin{aligned}&\int_{D_k} \left(\int_{D_{k+1}} d_{k+1} f(d_{k+1}|d_k)\, \mathrm{d}d_{k+1} \right) f(d_k)\, \mathrm{d}d_k \\ &= \int_{D_k} \int_{D_{k+1}} d_{k+1} f(d_{k+1}|d_k) f(d_k)\, \mathrm{d}d_{k+1}\, \mathrm{d}d_k \\ &= \int_{D_k} \int_{D_{k+1}} d_{k+1} f(d_{k+1}, d_k)\, \mathrm{d}d_{k+1}\, \mathrm{d}d_k \\ &= \int_{D_{k+1}} \int_{D_k} d_{k+1} f(d_{k+1}, d_k)\, \mathrm{d}d_k\, \mathrm{d}d_{k+1} \\ &= \int_{D_{k+1}} d_{k+1} \left(\int_{D_k} f(d_{k+1}, d_k)\, \mathrm{d}d_k \right) \mathrm{d}d_{k+1} \\ &= \int_{D_{k+1}} d_{k+1} f(d_{k+1})\, \mathrm{d}d_{k+1} \\ &= \mathbb{E}(D_{k+1})\end{aligned} \tag{13}$$

where the first equality is obtained by the definition of conditional expectation, the fourth equliaty obtained by switching the order of double integraion.

The right-hand side of (12) is

$$\int_{D_k} d_k f(d_k) = \mathbb{E}(D_k) \tag{14}$$

Hence,

$$\mathbb{E}(D_{k+1}) < \mathbb{E}(D_k) \tag{15}$$

Next, let $\{\mathbb{E}(D_k)\}$ denote the sequence of $\mathbb{E}(D_1), \mathbb{E}(D_2), \ldots$, which has lower bound 0. We aim to prove that the limit of sequence $\{\mathbb{E}(D_k)\}$ goes to 0, i.e. $\mathbb{E}(D_k) \to 0$. This is done by proving, through contradiction, that 0 is the infimum of $\{\mathbb{E}(D_k)\}$. Assume that $\inf\{\mathbb{E}(D_k)\} = b > 0$, which means for any $\epsilon > 0$, there exists $K \in \mathbb{N}^+$ and for all $k > K$, $b < \mathbb{E}(D_k) < b + \epsilon$. Note that $\mathbb{E}(D_k) \neq b$, otherwise $\mathbb{E}(D_{k+1}) < b$ according to strict decrease of $\{\mathbb{E}(D_k)\}$. For a fixed ϵ, denote $k_\epsilon = \inf\{k : \mathbb{E}(D_k) < b + \epsilon\}$, which means the number of nodes when it's the first time that $\mathbb{E}(D_k) < b + \epsilon$.

As shown in Figure 5, $\mathcal{B}_{k_\epsilon/2}(q_P)$ represents the ball with radius $d_{k_\epsilon}/2$, where d_{k_ϵ} is the observation of D_{k_ϵ}. Remeber $\mathcal{I}_{k_\epsilon}$ denotes the intersection of Voronoi cell around q_P^{ncp} and neighborhood of the feasible path from $\mathbf{x}^{\text{ncp}}$ to $\mathbf{x}$. If $\mathbf{x}^{\text{new}} \in \mathcal{I}_{k_\epsilon}$, $D_{k_\epsilon+1} < d_{k_\epsilon}$. Let $\gamma_{k_\epsilon} = \text{suf}\{\|\mathbf{x} - \mathbf{x}^{\text{ncp}}\| : \mathbf{x} \in \mathcal{I}_{k_\epsilon}\}$, which means possibly the maximum value can be dropped from d_{k_ϵ}, $\gamma_{k_\epsilon} > 0$ since $\mu(\mathcal{I}_{k_\epsilon}) > 0$. Then a ball of radius $d_{k_\epsilon} - \gamma_{k_\epsilon}/2$ is drawn at center q_P. If $\mathbf{x}^{\text{new}}$ is in the intersection of $\mathcal{I}_{k_\epsilon}$ and $\mathcal{B}_{(d_{k_\epsilon} - \gamma_{k_\epsilon}/2)}(q_P)$, we have $D_{k_\epsilon+1} < d_{k_\epsilon} - \gamma_{k_\epsilon}/2$, of which the probability is denoted as $p_{k_\epsilon+1} > 0$.

Now denote $\{D_{k_\epsilon+n}\}$ as a sequence starting from D_{k_ϵ} for $n \in \mathbb{N}$. Note that as tree grows, within the ball $\mathcal{B}_{(d_{k_\epsilon} - \gamma_{k_\epsilon}/2)}(q_P)$, the number of candidate parent of q_P will increase as well. Considering one case, we know q_P^{ncp} is the nearest candidate parent when the number of node is k_ϵ, but there may exist other candidate parents which are further from q_P than q_P^{ncp}. Take a specific candidate parent as an example, as long as a new point $\mathbf{x}^{\text{new}}$ lies within the intersection of Voronoi cell around it, the neighborhood of the feasible path and the ball $\mathcal{B}_{(d_{k_\epsilon} - \gamma_{k_\epsilon}/2)}(q_P)$, a new candidate parent of q_P

will appear within $\mathcal{B}_{(d_{k_\epsilon}-\gamma_{k_\epsilon}/2)}(q_P)$. Let $Q^{\text{cp}}_{k_\epsilon+n}$ denote the set of all candidate parents within $\mathcal{B}_{(d_{k_\epsilon}-\gamma_{k_\epsilon}/2)}(q_P)$, and, by an abuse of notation, $\mathcal{I}_{k_\epsilon+n}$ denote the union of aforementioned intersections incurred by each node in $Q^{\text{cp}}_{k_\epsilon+n}$.

Follow the same logic about $p_{k_\epsilon+1}$ above, we can state that, when the number of node is $k_\epsilon + n + 1$, with probability $p_{k_\epsilon+n+1}$, $D_{k_\epsilon+n+1} < d_{k_\epsilon} - \gamma_{k_\epsilon}/2$, which occurs when $\mathbf{x}^{\text{new}}$ lies within $\mathcal{I}_{k_\epsilon+n}$. Denote $\Delta = d_{k_\epsilon} - \gamma_{k_\epsilon}/2$, then given $D_{k_\epsilon} = d_{k_\epsilon}$, with probability $\prod_{i=0}^{n}(1-p_{k_\epsilon+i})$, event $\{d_{k_\epsilon} - \Delta < D_{k_\epsilon+n} \le d_{k_\epsilon}\}$ occurs. Then with probability $1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})$, $D_{k_\epsilon+n} \le d_{k_\epsilon} - \Delta$.

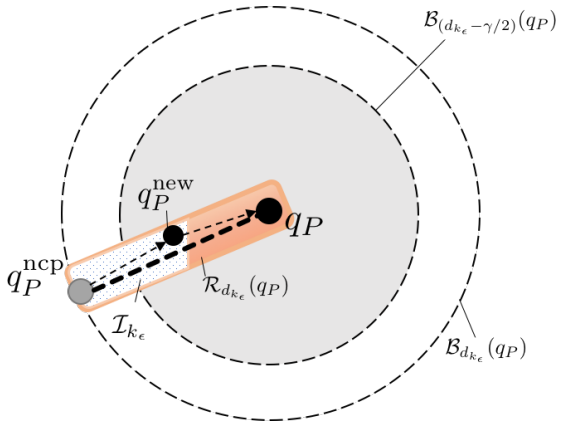


Fig. 5. An illustration of the scenario where $D_{k_\epsilon+1}$ drops below $d_{k_\epsilon} - \gamma_{k_\epsilon}$. The shaded region, inside the ball $\mathcal{B}_{(d_{k_\epsilon}-\gamma_{k_\epsilon}/2)}(q_P)$, denotes neighborhood around the feasible path depicted as the thick dashed line, from q_P^{ncp} to q_P, and the thin dashed line illustrated alternative of the path on the tree. The dotted area denotes the intersection $\mathcal{I}_{k_\epsilon}$ of cell around q_P^{ncp} and neighborhood around the path. q_P^{new} is inside $\mathcal{I}_{k_\epsilon}$.

Then, $\mathbb{E}(D_{k_\epsilon+n} | D_{k_\epsilon} = d_{k_\epsilon})$ can be written as:

$$\mathbb{E}(D_{k_\epsilon+n} | D_{k_\epsilon} = d_{k_\epsilon}) \le \prod_{i=1}^{n}(1-p_{k_\epsilon+i}) d_{k_\epsilon} + \left(1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})\right)(d_{k_\epsilon} - \Delta) \quad (16)$$

Multiplying both sides of (16) by p.d.f $f(d_{k_\epsilon})$ and integrating over the support of D_{k_ϵ}, (16) turns into:

$$\begin{aligned}\mathbb{E}(D_{k_\epsilon+n}) &\le \prod_{i=1}^{n}(1-p_{k_\epsilon+i})\mathbb{E}(D_{k_\epsilon}) + \left(1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})\right)(\mathbb{E}(D_{k_\epsilon}) - \Delta) \\ &< \left(\prod_{i=1}^{n}(1-p_{k_\epsilon+i})\right)(b+\epsilon) + \left(1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})\right)(b+\epsilon-\Delta) \quad (17) \\ &= (b+\epsilon) - \left(1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})\right)\Delta\end{aligned}$$

$\lim_{n\to\infty}\prod_{i=1}^{n}(1-p_{k_\epsilon+i}) < 1$ due to $p_{k_\epsilon+i}$ doesn't convert to 0. Hence, $\lim_{n\to\infty} 1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i}) > 0$. We assume $\lim_{n\to\infty} 1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i}) \ge c$, where $c > 0$ is a constant. (17) becomes

$$\mathbb{E}(D_{k_\epsilon+n}) \le (b+\epsilon) - c\,\Delta \quad (18)$$

For ϵ is an arbitrarily smaller non-negative number, let $\epsilon = c\Delta/2$, then $\mathbb{E}(D_{k_\epsilon+n}) \to b - c\Delta/2 < b$ as $n \to \infty$, which contradicts against the assumption that b is the infimum. Hence, 0 is the infimum of the sequence $\{\mathbb{E}(D_k)\}$. According to monotone convergence theorem, $\mathbb{E}(D_k) \to 0$.

Following the same logic, $\mathbb{E}(D_k^2) \to 0$. Hence, The variance $\text{Var}(D_k) = E(D_k^2) - [E(D_k)]^2 \to 0$, which is used to finish the proof with the help of a theorem introduced below.

Theorem 5.5: Assume that Y_n are random variables and C is a constant such that $\mathbb{E}Y_n \to C$, $\text{Var}(Y_n) \to 0$ as $n \to \infty$. Then $Y_n \to C$ as $n \to \infty$ in probability.

Since $\mathbb{E}(D_k) \to 0$ and $\text{Var}(D_k) \to 0$, $D_k \to 0$ in probabilty, namely, for any $\delta > 0$,

$$\lim_{k\to\infty} \mathbb{P}(D_k \le \delta) = 1 \quad (19)$$

Finally, we prove that those nodes arbitrarily close to q_P share the same Büchi state as q_P. Recall above that with probability $1 - \prod_{i=1}^{n}(1-p_{k_\epsilon+i})$, $D_{k_\epsilon+n} \le d_{k_\epsilon} - \Delta$, which is also the probability that there is a node which has the same Büchi state as q_P. The reason is that when the new point lies within any intersection $\mathcal{I}_{k_\epsilon+i}$, q_B can be selected to be paired with $\mathbf{x}^{\text{new}}$ to constitute a new node, because (i) $\texttt{ObstacleFree}(\mathbf{x}^{\text{ncp}}, \mathbf{x})$ is true, (ii) $q_B^{\text{ncp}} \xrightarrow{L(\mathbf{x}^{\text{ncp}})} q_B$. When $n \to \infty$, the probability goes to 1 due to $p_{k_\epsilon+i} > 0$. ■

Below we extend Lemma 5.4 to any reachable state in P_r.

Theorem 5.6: For any state $q_P = (\mathbf{x}, q_B) \in P_r$ and any positive number $\delta > 0$, $\lim_{k\to\infty}\mathbb{P}(D_k(q_P) < \delta) = 1$. Also, the probability that there exists anode in the tree, which shares the same Büchi state as q_P and lies within the ball of radius δ centered at q_P, goes to 1 as iteration increases to infinity.

This can be proved using Lemma 5.4 inductively.

Proof: Let q_P^0 denote the initial node. If q_P can be reached from the root, then a feasible plan must exist. Discretizing the plan, there exists a sequence of product states, $\{q_P^0, q_P^1, \ldots, q_P^k\}$, where $q_P^i = (\mathbf{x}^i, q_B^i)$ and q_P^i can make one-step transition to q_P^{i+1} for $i \in \{0, \ldots, k-1\}$.

Following Theorem 5.4, Since q_P^1 is reachable from q_P^0 through one-step transition, we have $\lim_{k\to\infty}\mathbb{P}\left(\{\exists\,(\mathbf{x}_1', q_B^1) \in \mathcal{V}_k^{\text{pRRT}} : \|\mathbf{x}_1 - \mathbf{x}_1'\| < \delta\}\right) = 1$ for any $\delta > 0$. In other words, the tree has a node with same Büchi state as q_P^1. The dashed line in Figure 6 means the initial state $(\mathbf{x}_0, q_B^0)$ can reach $(\mathbf{x}_1', q_B^1)$ in multiple steps. Then again Theorem 5.4 can be applied inductively to each pair, $(\mathbf{x}_i', q_B^i)$ and $(\mathbf{x}_{i+1}, q_B^{i+1})$, for $i \in \{1, \ldots, k-1\}$. Since $(\mathbf{x}_{i+1}, q_B^{i+1})$ is a candidate parent of $(\mathbf{x}_i, q_B^i)$ and both are arbitrarily close, $(\mathbf{x}_{i+1}, q_B^{i+1})$ can be reachede from $(\mathbf{x}_i', q_B^i)$ through one step transition. Let $(\mathbf{x}_i', q_B^i)$ be the new initial state, then it can reach $(\mathbf{x}_{i+1}', q_B^{i+1})$ through multiple edges. Due to $q_P = q_P^k$, we can conclude that $\lim_{k\to\infty}\mathbb{P}(D_k(q_P) < \delta) = 1$, and

$$\lim_{k\to\infty}\mathbb{P}\left(\left\{\exists\, q_P' = (\mathbf{x}', q_B) \in \mathcal{V}_k^{\text{pRRT}} : \|\mathbf{x} - \mathbf{x}'\| < \delta\right\}\right) = 1.$$

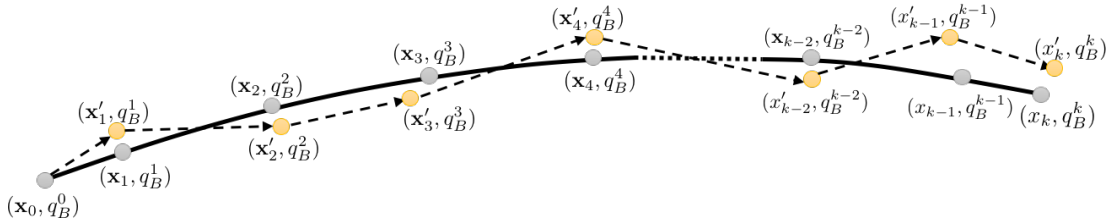


Fig. 6. Depiction proof of Theorem 5.6. The thick curve represents actual pathfrom the root to q_P, and the dashed broken line illustrates the path generated by our algorithm.

~~Finally, for any accepting state,~~ the tree will grow arbitrarily ~~towards~~ it and ~~have~~ a node with the same Büchi state, ~~which, by~~ definition ~~of~~ accepting state, ~~is itself a~~ accepting state. Therefore a feasbile path will be found with probability 1. ■

~~Next, we examine the optimality of the resulting motion plan τ constructed in Section IV-C.~~

B. Asymptotic Optimality

Theorem 5.7 (Asymptotic Optimality): Consider parameters $r(\mathcal{V}_T)$ defined (4), ~~then,~~ Algorithm 1 is asymptotically optimal, i.e., the discrete motion plan $\tau_{n^{\text{pre}}_{\max}}^{n^{\text{suf}}_{\max}}$ that is generated by this algorithm satisfies

$$\mathbb{P}\left(\left\{\lim_{n^{\text{pre}}_{\max}\to\infty, n^{\text{suf}}_{\max}\to\infty} J(\tau_{n^{\text{pre}}_{\max}}^{n^{\text{suf}}_{\max}}) = J(\tau^*)\right\}\right) = 1, \tag{20}$$

where J is the cost function defined in (1), $\tau*$ is the optimal plan, $n^{\text{pre}}_{\max}$ and $n^{\text{suf}}_{\max}$ are the maximum number of iterations used in Algorithm 2, and $\tau_{n^{\text{pre}}_{\max}}^{n^{\text{suf}}_{\max}} = \tau^{\text{pre}, n^{\text{pre}}_{\max}}[\tau^{\text{suf}, n^{\text{suf}}_{\max}}]^\omega$.

~~By the abuse of~~ notation, let $\tau^* = \Pi|_{\mathcal{W}^N} p^* = \Pi|_{\mathcal{W}^N} p^{\text{pre}} [\Pi|_{\mathcal{W}^N} p^{\text{suf}}]^\omega = \tau^{\text{pre}}[\tau^{\text{suf}}]^\omega$ be the discretized optimal plan that satisfies ~~a~~ LTL formula ϕ, where

$$\begin{aligned} p &= (\mathbf{x}_0, q_B^0), (\mathbf{x}_1, q_B^1), \dots (\mathbf{x}_K, q_B^K) \\ &\quad [(\mathbf{x}_K, q_B^K), \dots, (\mathbf{x}_{K+S}, q_B^{K+S}), (\mathbf{x}_{K+S+1}, q_B^{K+S+1})]^\omega \\ &= p^{\text{pre}}[p^{\text{suf}}]^\omega, \end{aligned} \tag{21}$$

where $\mathbf{x}_k \in \mathcal{W}^N_{\text{free}}$, $q_B^k \in \mathcal{Q}_B$, for all $k = \{1, \dots, K+S+1\}$, $q_B^K \in \mathcal{Q}_B^F$, $(\mathbf{x}_{K+S+1}, q_B^K) = (\mathbf{x}_K, q_B^K)$. ~~We~~ show that the optimal prefix part p^{pre} will be found and then following the same logic we show that the respective optimal suffix part p^{suf} can be found. This result is ~~proved using~~ the asymptotic optimality of the RRT* algorithm in [6].

Proof: The proof of optimality is ~~heavily~~ based on ~~the~~ that of RRT*. ~~For better understanding, we first restate the outline of~~ proof in [6] which also ~~constitute~~ part of our proof, ~~then put most efforts to the part incurred by involvement of~~ temporal tasks. ~~Before it, some preliminaries are provided.~~

Define δ-interior of $\mathcal{W}^N_{\text{free}}$, strong and weak δ clearances which are important concepts for the formulation of robustly optimal problems. For any real number $\delta > 0$, the δ-interior of $\mathcal{W}^N_{\text{free}}$ ~~means~~ the subset of $\mathcal{W}^N_{\text{free}}$ ~~within which any state is~~ at least δ distance away from any obstacle, denoted as $\texttt{int}_\delta(\mathcal{W}^N_{\text{free}})$. ~~Then~~ a feasible path $\tau : [0,1] \to \mathcal{W}^N_{\text{free}}$ has strong δ clearance if $\tau \subset \mathcal{W}^N_{\text{free}}$, which means path τ lies entirely within $\mathcal{W}^N_{\text{free}}$. Furthermore, a feasible path $\tau : [0,1] \to \mathcal{W}^N_{\text{free}}$ has weak δ clearance if there exist a strong δ clearance path τ' which is homotopic to it, i.e, τ' can be continuously transformed to τ ~~through~~ $\mathcal{W}^N_{\text{free}}$. Finally, a feasible path $\tau^* \in \mathcal{W}^N_{\text{free}}$ is called ~~the~~ robustly optimal ~~path~~ if for any sequence of paths $\{\tau_n\}$, where $\tau_n \in \mathcal{W}^N_{\text{free}}$, $\forall n \in \mathbb{N}$, ~~such~~ that $\{\tau_n\}$ converges to τ^*, $\lim_{n\to\infty} J(\tau_n) = J(\tau^*)$. ~~See~~ [6] for more technical details.

~~The outline of proof of RRT* is given below.~~

First, ~~define~~ a sequence $\{\delta_n\}$ of positive real numbers ~~which~~ converges to 0 as n goes to infinity, Then a sequence $\{\tau_n\}$ of paths is constructed in a way that τ_n has strong δ_n clearance and the sequence converges to ~~the~~ robustly optimal path τ^* as n goes to infinity.

Second, ~~define~~ a sequence $\{q_n\}$ of postive real ~~numbers, of which each q_n is derermined by δ_n in the form~~ $q_n = \delta_n/(1+\theta_1)$, where θ_1 is a small positive constant, ~~then~~ a sequence $\{\mathcal{B}_{n,M_n}\}$ of M_n balls, each with radius q_n and center on τ_n, is constructed to cover ~~path~~ τ_n ~~from the beginning to the end, and the center of two~~ consecutive balls are $\theta_1 q_n$ apart. ~~[6] proves~~ that for any two ~~point,~~ $\mathbf{x}_m \in \mathcal{B}_{n,m}$ and $\mathbf{x}_{m+1} \in \mathcal{B}_{n,m+1}$, in any two consecutive balls, the following ~~hold.~~ (i) the Euclidean distance between $\mathbf{x}_m$ and $\mathbf{x}_{m+1}$ is not more than the radius $r(\mathcal{V}_T)$ used in Algorithm 4 and 5; (ii) the straight line connecting $\mathbf{x}_m$ and $\mathbf{x}_{m+1}$ lies enritely within the free workspace.

Third, ~~Construct~~ a graph $\mathcal{G}_n$ ~~by~~ connecting an edge from node $\mathbf{x}$ to node $\mathbf{x}'$ when (i) $\|\mathbf{x}-\mathbf{x}'\| \le r(\mathcal{V}_T)$, (ii) $\mathbf{x}$ is sampled before $\mathbf{x}'$. Observe that the cost of the best path in $\mathcal{G}_n$ from the initial state to the goal region is not less than that generated by RRT*. Hence, if the algorithm generating $\mathcal{G}_n$ is optimal, so is RRT*. ~~[6] proves~~ that the probability that, ~~in any two consecutive balls in $\mathcal{B}_n$,~~ there ~~exists~~ two nodes ~~which~~ satisfy (i) and (ii) converges to 1 ~~as~~ iteration goes to infinity.

Finally, ~~show that~~ any sequence of ~~paths~~ starting from the root and reaching the goal region, ~~of which each is the shortest path in n-th iteration,~~ will converge to the optimal path $\tau*$.

The main difference ~~in our proof of pRRT*~~ is ~~the~~ third step, ~~aiming to~~ combine Büchi ~~state~~ with ~~robots' position.~~ ~~Now, each~~ position state on the path τ_n is paired with a Büchi state, denoted as plan p_n, ~~making it satisfy the specified task. By the abuse of notation of $\mathcal{G}_n$,~~ an edge between $(\mathbf{x}, q_B)$ and $(\mathbf{x}', q'_B)$ will be added to the ~~graph if~~ the straight line connecting $\mathbf{x}$ and $\mathbf{x}'$ crosses through any boundary of regions at most once, ~~and~~ $q_B \xrightarrow{L(\mathbf{x})} q'_B$ ~~besides the conditions (i) and (ii) mentioned above~~. Note that the radius ~~of balls in~~ $\mathcal{B}_n$ decreases to 0 as δ_n ~~drops~~ to 0, and the distance

between centers of any two consecutive balls also goes to 0. By assumption, there exists a neighborhood $\mathcal{R}_i$ around the path connecting any two consecutive centers, $o_i = (\mathbf{x}_i, q_B^i)$ and $o_{i+1} = (\mathbf{x}_{i+1}, q_B^{i+1})$, $\forall i \in \{0, 1, \ldots, M_n - 1\}$. Hence, we can conclude that there exists a large number $N \in \mathbb{N}$, for any iteration $n > N$ and any two consective centers, o_i and o_{i+1}, the former can make a one-step transition to the latter, i.e. (i) $\texttt{ObstacleFree}(\mathbf{x}_i, \mathbf{x}_{i+1})$ holds, (ii) $q_B^i \xrightarrow{L(\mathbf{x}_i)} q_B^{i+1}$. Furthermore, $\mathcal{B}_i \subset \mathcal{R}_i$ and $\mathcal{B}_{i+1} \subset \mathcal{R}_i$, because both the radius of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ shrink to 0 but R_i remain the same. Recall that [6] proves that eventually there exist $\mathbf{x}$ and $\mathbf{x}'$ in any consecutive balls such that $\mathbf{x}$ in the former ball is sampled before $\mathbf{x}'$ in the latter ball. We can inductively use this to finish our proof. Note that the rest assumes $n > N$. See Figure 7 for graphical illustration.

Consider the first two balls. State $(\mathbf{x}_0, q_B^0)$ can make one-step transition to $(\mathbf{x}_1, q_B^1)$, therefore, $\texttt{ObstacleFree}(\mathbf{x}_0, \mathbf{x}_1)$ is true. Since $\mathcal{B}_0 \subset \mathcal{R}_0$, $\mathcal{B}_1 \subset \mathcal{R}_0$ and $\mathbf{x}'_1 \in \mathcal{B}_1$, $\texttt{ObstacleFree}(\mathbf{x}_0, \mathbf{x}'_1)$ is true. In our algorithm, the Büchi state space is searched in an exhaustive way, hence, q_B^1 will be selected to be paired with $\mathbf{x}'_1$, adding an edge from $(\mathbf{x}_0, q_B^0)$ to $(\mathbf{x}'_1, q_B^1)$. Inductively, for a state $(\mathbf{x}'_i, q_B^i)$ in $\mathcal{B}_i$, there will be an edge from it to the state $(\mathbf{x}'_{i+1}, q_B^{i+1})$ in $\mathcal{B}_{i+1}$. Eventually, there will be a path from the inital state to the goal region which satisfy the specified temporal task.

Note as n grows to infinity, it gradually becomes impossbile that two different boundary segments, ∂W_1 and ∂W_2, cross through two consecutive balls, since the radius decreases to 0. See Figure 8(a). But it's possible that a boundary segment ∂W of one region crosses through the ball, even through two consecutive balls. See Figure 8(b) for graphical detail. When n is very large, the boundary segment crossing through the ball can be approximated as a line segment. In this case, $\mathbf{x}'_i$ and $\mathbf{x}'_{i+1}$ should lie within the same part where the corresponding center is inside the ball. Since $\texttt{ObstacleFree}(\mathbf{x}_i, \mathbf{x}_{i+1})$ is true, $\mathbf{x}'_i$ and $\mathbf{x}'_{i+1}$ being inside the same part with $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, respectively, guarantee that $\texttt{ObstacleFree}(\mathbf{x}'_i, \mathbf{x}'_{i+1})$ is true. This further guarantees the inductive process proceeds forward. Actually, since the total number of balls increases to infinity while that boundaries cross through balls occurs rarely, the balls in $\mathcal{B}_n$ crossed through by boundaries only make up a small portion, which does not impact the value of γ_{pRRT^*}.

Like the case in RRT*, the cost of the best path in $\mathcal{G}_n$ from the initial state to the goal region is not less than that generated by pRRT*. Follwing the same final step above in the outline of proof of RRT*, pRRT* will converge to the optimal path. ■

VI. Simulation Results

In this section, we present three case studies, implemented using Python 3.6.3, that illustrate the efficiency and scalability of the proposed algorithm. We also compare our sampling-based algorithm to the one proposed in [24], that also scales well with the size of the network. Note that a direct comparison with works that require a discrete abstraction of robot mobility as e.g., in [25], [26] cannot be made, since in this work a discrete abstraction is not needed. In all the following case studies, we consider circular regions of interest ℓ_j with radius $\delta > 0$.

A. Case Study I

In the first simulation study, we consider a motion planning problem for a single robot that lies in a workspace with $W = 5$ regions of interest of radius $\delta = 0.1$, as depicted in Figure 11. The parameter ϵ of the `Steer` function is 0.25 distance units. The robot is initially located at $\mathbf{x}_0 = (0.8, 0.1)$. The assigned task is described by the following LTL formula:

$$\phi = \Diamond(\pi_1^{\ell_4}) \wedge \Box\Diamond(\pi_1^{\ell_3} \wedge (\Diamond\pi_1^{\ell_1})) \wedge (\neg\pi_1^{\ell_1} \mathcal{U} \pi_1^{\ell_2}) \wedge \Box(\neg\pi_1^{\ell_5}). \tag{22}$$

In words, the LTL-based task in (22) requires robot 1: (a) to eventually visit region ℓ_4, (b) to visit region ℓ_1 and ℓ_3 infinitely often, and the scenario where visiting ℓ_1 between two occurance of visiting ℓ_3 happens infinitely often (c) before the first time it visits region ℓ_1, it should finish visiting region ℓ_2 and (d) always avoid region ℓ_5. The considered LTL formula corresponds to a NBA with $|\mathcal{Q}_B| = 11$ states, $|\mathcal{Q}_B^0| = 1$, and $|\mathcal{Q}_B^F| = 2$.[5]

Algorithm 1 was run under five different choices of parameters $n_{\text{max}}^{\text{pre}}$, $n_{\text{max}}^{\text{pre}}$ and, after running 20 times for each choice, the cost of the resulting plan for each case is depicted in Figure 9. Observe in Figure 9 that as we increase the number of iterations that Algorithm 2 runs, the cost of the resulting plans decreases, as expected due to Theorem 5.7. The number of detected final states and runtime for each case are also depicted in the same figure. Figure 10 illustrates the trees that were built for the construction of the prefix and suffix part when $n_{\text{max}}^{\text{pre}} = n_{\text{max}}^{\text{suf}} = 200$. The resulting motion plans for the cases $n_{\text{max}}^{\text{pre}} = n_{\text{max}}^{\text{suf}} = 200$ and $n_{\text{max}}^{\text{pre}} = n_{\text{max}}^{\text{suf}} = 1000$ are depicted in Figure 11.

B. Case Study II

In the second simulation study, we consider a network of $N = 2$ robots residing in the same workspace as in the first case study; see Figure 11. The assigned task is expressed in the following temporal logic formula:

$$\phi = \Box\Diamond(\pi_1^{\ell_1}) \wedge \Box(\pi_1^{\ell_1} \rightarrow \bigcirc(\neg\pi_1^{\ell_1} \mathcal{U} \pi_2^{\ell_2})). \tag{23}$$

In words, the LTL-based task in (23) requires robot 1: (a) to visit region ℓ_1, infinitely often, and (b) once it visits region ℓ_1, its next position should never be inside ℓ_1 until robot 2 visits region ℓ_2. The considered LTL formula corresponds to a NBA with $|\mathcal{Q}_B| = 5$ states, $|\mathcal{Q}_B^0| = 1$, and $|\mathcal{Q}_B^F| = 2$. Recall that, previously, we claim to exlude "next" operator in the formula in order to ensure asymptotic optimality. However, in this simulation study, we aim to compare the performance of the proposed temporal planning algorithm to the one proposed in [24], which terminates when the first feasible path is found, only providing guarantee on probabilistic completeness. Without considering optimality, this case can demonstrate our

[5]The translation of the LTL formula to a NBA was made by the tool developed in [31].

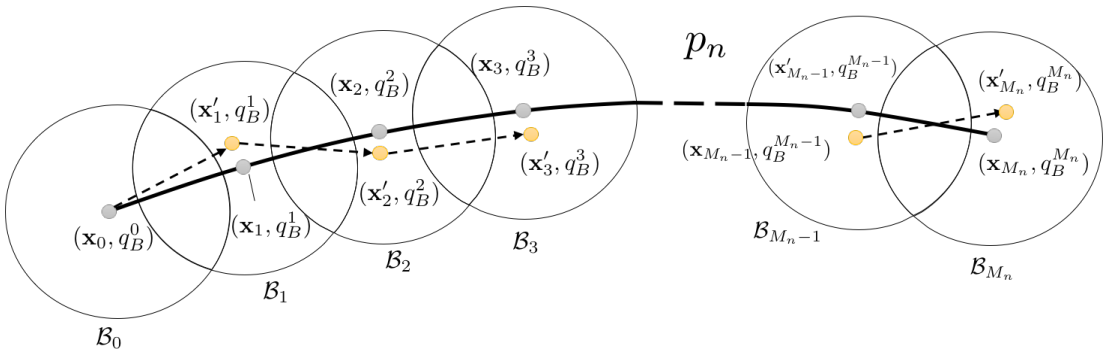
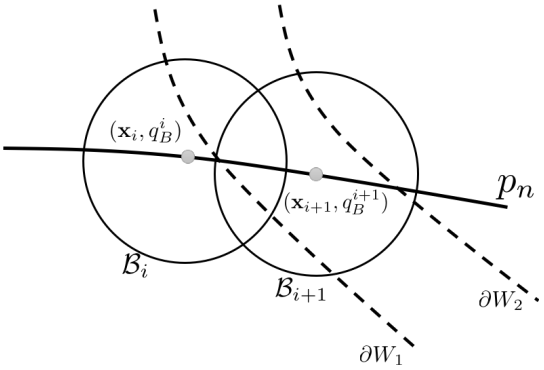
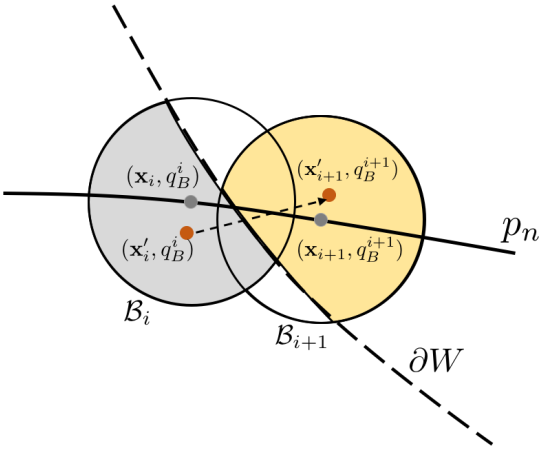


Fig. 7. Graphical depiction of proof of asymptotic optimality. The thick line represents path p_n and the thin dashed arrowline is the path connecting those nodes of which the position state are mentioned in the third step above. Any center can make a one-step transition to its subsequent center.



(a) One boundary segment crossing through two consecutive balls.



(b) Two different boundary segments crossing through two consecutive balls.

Fig. 8. Graphical depiction of cases where boundaries of regions cross through balls. In Figure 8(a), the sequence of balls $\mathcal{B}_n$ are constructred along the path p_n. Two different boundary segments cross through two consecutive balls, which is eventually unlikely to occur as n grows to infinity. In Figure 8(b), a boundary segment crosses through two consecutive balls, the shaded part illlustrate the one where the center belongs.

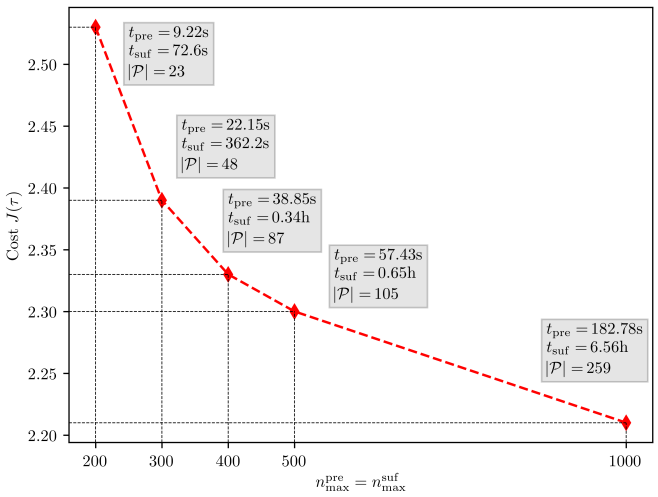
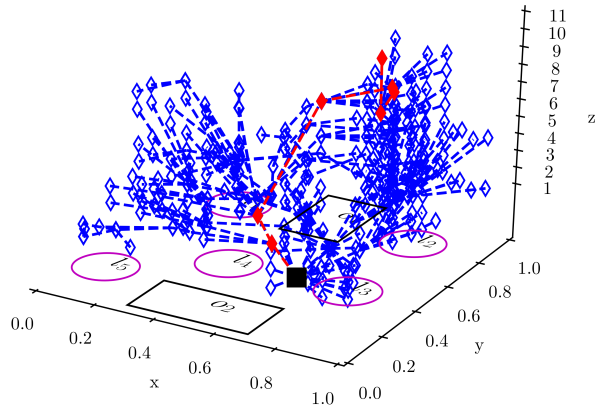


Fig. 9. Simulation Study I: Evolution of the cost $J(\tau)$ of the resulting optimal motion plan τ for various maximum numbers of iterations for Algorithm 2. The time required for the construction of the optimal prefix and suffix part along with the number of detected final states for each case are included in the gray colored box.

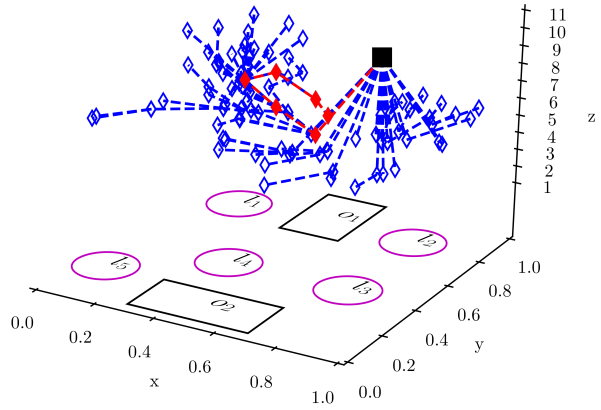
proposed algorithm's capability of satisfying specifiacations involving "next" operator.

The step-size ϵ for both Algorithms is selected to be $0.25N$, where N is the number of robots. To provide a fair comparison between the two approaches, we set the parameter $\eta_1(k)$ in [24] to be $\frac{1}{\sqrt{\pi}} \sqrt[n]{\frac{\mu(\mathcal{W}^N)\Gamma(dN/2+1)}{k}}$ for all $k \geq 1$, where $\mu(\mathcal{W}^N)$ is the total measure (volume) of the configuration space, dN is the dimension of $\mathcal{W}^N$, and Γ is the gamma function and k is the current size of transition system, similar to the number of different position states in the tree constructred by our algorithm, and $\eta_2(k)$ to be equal to $2\eta_1(k)$, which satisfies (i) $\eta_1(k) < \eta_2(k)$ for all $k \geq 1$, (ii) $c\,\eta_1(k) > \eta_2(k)$ for some finite $c > 1$ and all $k \geq 0$.

Figure 12 depicts the average runtime of both algorithms, after running them 20 times, for various choices of the radius δ of the regions of interest ℓ_j. Observe that as the radius δ decreases, [24] consumes more and more time than our algorithm. The reason is that as the graph built by [24] grows, the parameter η_1 goes to zero. Therefore, as $\eta_1 \to 0$, the constructed graph loses its sparsity and is degenerated into a graph of arbitrary structure, where $|\mathcal{E}| >> |\mathcal{V}|$. However, in our case, it always holds that $|\mathcal{E}_T| = |\mathcal{V}_T| - 1$ due to the tree that is used for the exploration of the product state-space $P = \mathcal{W}^N \times \mathcal{Q}_B$, which results in significant savings in resources in terms of memory. Also, recall that the sampling-



(a) Tree for prefix.

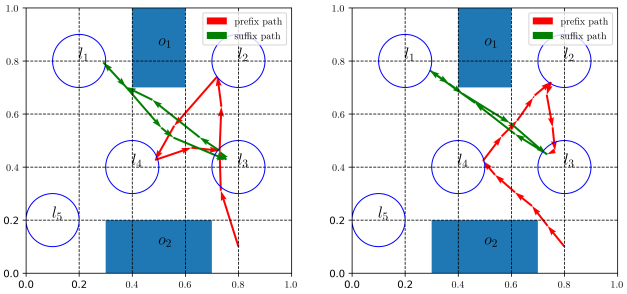


(b) Tree for suffix.

Fig. 10. Simulation Study I: Figures 10(a) and 10(b) depict the trees built by Algorithm 2 to synthesize the prefix and suffix part, respectively, until the first feasible path is found. The black square stands for the root of the tree and the red path represent the prefix and the suffix part. The z axis captures the NBA states. Specifically, $z = 1$ captures the initial state q_B^0, $z = 9$ and $z = 11$ represent the two accepting states of $\mathcal{Q}_B$, and the bottom represent the workspace $\mathcal{W}$. In this simulation, the Büchi state in the final state of the tree corresponds to $z = 11$.

based method in [24] is not asymptotically optimal, which is not the case in our work.

Figure 13 depicts the difference of average cost $J(\tau)$ of both algorithms, after running them 20 times, for various choices of the radius δ of the regions of interest. $\Delta J(\tau)$ equals the average cost incurred by [24] minus that of Algorithm 1. To make the comparison clear, the average cost $J(\tau)$of our algorithm is also drawn in the figure. Observe that the path found by our algorithm is much better than [24] in terms of total cost, which is guaranteed by the fact that our algorithm connects each node to the one, within distance of $r(\mathcal{V}_T)$, that incurs the smallest cost, via extending and rewiring steps.



(a) $n_{\max}^{\text{pre}} = n_{\max}^{\text{suf}} = 200$ (b) $n_{\max}^{\text{pre}} = n_{\max}^{\text{suf}} = 1000$

Fig. 11. Simulation Study I: Figures 11(a) and 11(b) illustrate the optimal motion plans synthesized by Algorithm 1, when $n_{\max}^{\text{pre}} = n_{\max}^{\text{suf}} = 200$ and $n_{\max}^{\text{pre}} = n_{\max}^{\text{suf}} = 1000$, respectively. The red and green arrow lines represent the prefix and the suffix part, respectively.

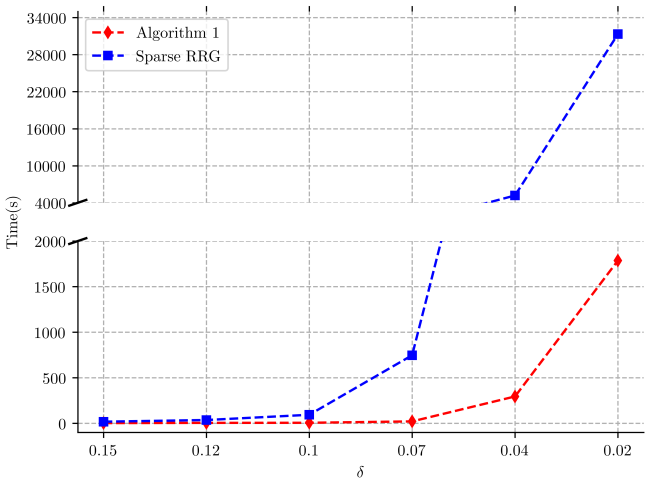


Fig. 12. Simulation Study II: Comparison of runtime between Algorithm 1 and the work presented in [24] with respect to the radius δ of the regions of interest ℓ_j.

C. Case Study III

In the third simulation study, we consider a network of $N = 4$ robots residing in a square workspace with dimensions 1×1 with $W = 6$ disjoint regions of interest of radius $\delta = 0.15$. The assigned task captures an intermittent communication problem that is defined in [25]:

$$\phi = \Box\Diamond(\pi_1^{\ell_1} \wedge \pi_2^{\ell_1}) \wedge \Box\Diamond(\pi_2^{\ell_2} \wedge \pi_3^{\ell_2}) \wedge \Box\Diamond(\pi_3^{\ell_3} \wedge \pi_4^{\ell_3}) \wedge \Box\Diamond(\pi_1^{\ell_4} \wedge \Diamond(\pi_4^{\ell_5} \wedge \Diamond\pi_3^{\ell_6})). \quad (24)$$

In words, the LTL-based task in (24) requires: (a) robots 1 and 2 to meet in region ℓ_1 infinitely often, (b) robots 2 and 3 to meet in region ℓ_2 infinitely often, (c) robots 3 and 4 to meet in region ℓ_3 infinitely often, (d) robot 1 to visit region ℓ_4, (e) once (d) occurs, robot 4 to visit region ℓ_5, and (f) once (e) happens, robot 3 should visit region ℓ_6. The steps (d)-(f) occur in this specific order infinitely often. The requirements (a)-(c) capture the intermittent communication task and (d)-(f) describe a sequencing task [7]. This LTL formula ensures that the information collected by the robots during the sequencing task will eventually be transmitted intermittently through a multi-hop communication path to any other robot in the

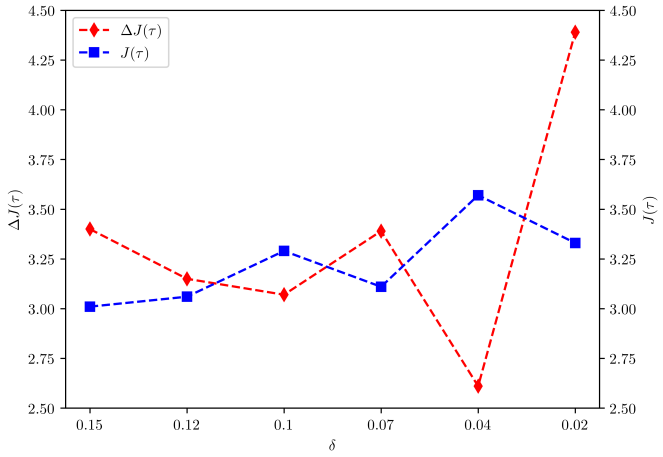


Fig. 13. Simulation Study II: Comparison of cost $J(\tau)$ between Algorithm 1 and the work presented in [24] with respect to the radius δ of the regions of interest ℓ_j.

network. The considered LTL formula corresponds to a NBA with $|\mathcal{Q}_B| = 21$ states, $|\mathcal{Q}_B^0| = 1$, $|\mathcal{Q}_B^F| = 4$ and 223 transitions. The parameter ϵ of the `Steer` function is $0.25N$ distance units. We ran Algorithm 1 until a discrete plan that satisfies the considered LTL task is found. The prefix and suffix parts were found in 13.8 and 35.6 minutes building trees of 5781 and 11863 nodes, respectively. Note that the temporal planning algorithm [24] failed to provide motion plans that satisfy the LTL formula (24) for various selections of the parameters $\eta_1(\mathcal{V})$, $\eta_2(\mathcal{V})$ and the step-size ϵ due to excessive memory requirements of saving the constructed graph.

VII. Conclusion

In this paper we proposed a sampling-based algorithm for multi-robot temporal task planning. The majority of existing LTL planning approaches rely on obtaining a discrete abstraction of robot mobility in the workspace and then constructing a product automaton among the robots which is used to synthesize discrete motion plans. The limitations of these approaches are that both the abstraction process and the control synthesis are computationally expensive. In this paper, we proposed a new sampling-based LTL planning algorithm that does not require any discrete abstraction of robot mobility and avoids the construction of a product automaton. Instead, it builds incrementally a tree that can explore the workspace and the state-space of a NBA that captures a given LTL specification, simultaneously. We showed that our algorithm is probabilistically complete and asymptotically optimal and presented numerical experiments that show that our method scales much better compared to relevant temporal planning methods.

References

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and T. S., "Principles of robot motion: theory, algorithms, and implementations," Boston, MA, 2005.

[3] I. Arvanitakis, K. Giannousakis, and A. Tzes, "Mobile robot navigation in unknown environment based on exploration principles," in *Control Applications (CCA), Conference on*. Buenos Aires, Argentina: IEEE, September 2016, pp. 493–498.

[4] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, San Fransisco, CA, USA, pp. 995–1001.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005, pp. 2020–2025.

[8] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *Robotics and Automation (ICRA), IEEE International Conference on*, Singapore (to appear), 2017.

[9] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, 2016.

[10] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

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

[12] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[13] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 5025–5032.

[14] ——, "Motion and action planning under LTL specifications using navigation functions and action description language," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo Big Sight, Japan, November 2013, pp. 240–245.

[15] M. Guo and D. V. Dimarogonas, "Reconfiguration in motion planning of single-and multi-agent systems under infeasible local LTL specifications," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, Florence, December 2013, pp. 2758–2763.

[16] Y. Chen, X. C. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global LTL specifications," in *50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, USA, December 2011, pp. 2718–2723.

[17] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[18] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *International Conference on Robotics and Automation (ICRA)*, Anchorage, AL, May 2010, pp. 2689–2696.

[19] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Intelligent Robots and Systems, Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4, Las Vegas, NV, USA, October 2003, pp. 3546–3551.

[20] C. Belta and L. Habets, "Constructing decidable hybrid systems with velocity bounds," in *Decision and Control (CDC) 43rd Conference on*, vol. 1. Bahamas: IEEE, December 2004, pp. 467–472.

[21] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[22] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2006, pp. 348–362.

[23] D. Boskos and D. V. Dimarogonas, "Decentralized abstractions for multi-agent systems under coupled constraints," in *Conference on Decision and Control (CDC),*, Osaka, Japan, December 2015, pp. 7104–7109.

[24] 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.

[25] Y. Kantaros and M. M. Zavlanos, "Intermittent connectivity control in mobile robot networks," in *49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, November, 2015, pp. 1125–1129.

[26] ——, "Sampling-based control synthesis for multi-robot systems under global temporal specifications," in *Cyber-Physical Systems (ICCPS), 8th International Conference on*. Pittsburgh, PA, USA: ACM/IEEE, 2017.

[27] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, 2009, pp. 2222–2229.

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

[29] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *1st Symposium in Logic in Computer Science (LICS)*. IEEE Computer Society, 1986.

[30] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[31] P. Gastin and D. Oddoux, "Fast LTL to büchi automata translation," in *International Conference on Computer Aided Verification*. Springer, 2001, pp. 53–65.