Temporal Logic Optimal Control for Large-Scale Multi-Robot Systems: 10^{400} States and Beyond

Yiannis Kantaros and Michael M. Zavlanos

***Abstract*— This paper proposes a new optimal control synthesis algorithm for multi-robot systems with Linear Temporal Logic (LTL) specifications. Existing planning approaches with LTL specifications rely on graph search techniques applied to a product automaton constructed among the robots. In our previous work, we have proposed a more tractable sampling-based algorithm that builds incrementally trees that approximate the state-space and transitions of the synchronous product automaton and does not require sophisticated graph search techniques. Here, we extend our previous work by introducing bias in the sampling process which is guided by transitions in the Büchi automaton that belong to the shortest path to the accepting states. This allows us to synthesize optimal motion plans from product automata with hundreds of orders more states than those that state-of-the-art methods can manipulate. We also show that the proposed algorithm is probabilistically complete and asymptotically optimal.**

I. Introduction

Optimal control synthesis algorithms under complex collaborative tasks captured by global Linear Temporal Logic (LTL) formulas are proposed in [1], [2]. In these works, optimal discrete plans are derived for every robot by constructing a product automaton among all robots using the individual transition systems that capture robot mobility and a Nondeterministic Büchi Automaton (NBA) that represents the global LTL specification. As the number of robots or the size of the NBA increases, the state-space of the product automaton grows exponentially. Consequently, these motion planning algorithms scale poorly with the number of robots and the complexity of the assigned task.

To mitigate these issues, in our previous work we proposed a sampling-based optimal control synthesis algorithm that avoids the explicit construction of the product among the transition systems and the NBA [3]. Specifically, this algorithm builds incrementally directed trees that approximately represent the state-space and transitions among states of the product automaton. The advantage is that approximating the product automaton by a tree rather than representing it explicitly by an arbitrary graph, as existing works do, 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.

In this work, we extend [3] by introducing bias in the sampling process of the algorithm. For this, we first exploit the structure of the atomic propositions to prune the NBA by removing transitions that can never be enabled. Then, we define a metric over the state-space of the NBA that captures the shortest path, i.e., the minimum number of feasible transitions that connect any two NBA states. Given this metric, we define a probability distribution over the nodes that are reachable from the current tree so that nodes that are closer to the final/accepting states of the NBA are sampled with higher probability; no particular sampling probability is proposed in [3]. We show that introducing bias in the sampling process does not violate the sampling assumptions in [3] so that our algorithm inherits the same probabilistic completeness and asymptotic optimality guarantees. Moreover, we show that by biasing the sampling process we can synthesize optimal motion plans from product automata with order 10^{400} states and beyond, which is hundreds of orders more states than those that any existing optimal control synthesis algorithms [1]–[4] can handle. For example, our algorithm in [3], when implemented with uniform sampling, can optimally solve problems with order 10^{10} states, which are still more than existing methods [1], [2], [4]. Compared to off-the-shelf model-checkers, such as NuSMV [5] and nuXmv [6], that can design feasible but not optimal motion plans, our proposed biased sampling-based algorithm can find feasible plans much faster. NuSMV can solve problems with order 10^{30} states, while nuXmv can handle infinite-state synchronous transition systems but it is slower than our algorithm. Note that our algorithm can be implemented in a distributed way, as in our recent work [7], which can further decrease the computational time.

II. Problem Formulation

Consider N mobile robots that evolve in a complex workspace $\mathcal{W} \subset \mathbb{R}^d$ with W disjoint regions of interest in $\mathcal{W}$. The j-th region is denoted by r_j and it can be of any arbitrary shape.[1] Mobility of robot i in $\mathcal{W}$ isz represented by a weighted Transition System (wTS) defined as follows:

Definition 2.1 (wTS): A *weighted Transition System* (wTS) for robot i, denoted by wTS_i is a tuple $\text{wTS}_i = \big(\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}_i, L_i\big)$ where: (a) $\mathcal{Q}_i = \bigcup_{j=1}^{W}\{q_i^{r_j}\}$ is the set of states, where a state $q_i^{r_j}$ indicates that robot i is at location r_j; (b) $q_i^0 \in \mathcal{Q}_i$ is the initial state of robot i; (c) $\rightarrow_i \subseteq \mathcal{Q}_i \times \mathcal{Q}_i$ is the transition relation for robot i. Given the robot dynamics, if there is a control input $\mathbf{u}_i$ that can drive robot i from location r_j to r_e, then there is a transition from state $q_i^{r_j}$ to $q_i^{r_e}$ denoted by

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 and by ONR under grant #N000141812374.

[1] Overlapping regions can also be considered by introducing additional states to wTS_i defined in Definition 2.1 that capture the presence of robot i in more than one region.

$(q_i^{r_j}, q_i^{r_e}) \in \rightarrow_i$; (d) $w_i : \mathcal{Q}_i \times \mathcal{Q}_i \rightarrow \mathbb{R}_+$ is a cost function that assigns weights/costs to each possible transition in wTS. For example, such costs can be associated with the distance that needs to be traveled by robot i in order to move from state $q_i^{r_j}$ to state $q_i^{r_k}$; (e) $\mathcal{AP}_i = \bigcup_{j=1}^{W} \{\pi_i^{r_j}\}$ is the set of atomic propositions, where $\pi_i^{r_j}$ is true if robot i is inside region r_j and false otherwise; and (f) $L_i : \mathcal{Q}_i \rightarrow \mathcal{AP}_i$ is an observation/output function defined as $L_i(q_i^{r_j}) = \pi_i^{r_j}$, for all $q_i^{r_j} \in \mathcal{Q}_i$.

Given the definition of the wTS, we can define the synchronous *Product Transition System* (PTS) as follows [1]:

Definition 2.2 (PTS): Given N transition systems $\text{wTS}_i = (\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}, L_i)$, the *product transition system* $\text{PTS} = \text{wTS}_1 \otimes \text{wTS}_2 \otimes \cdots \otimes \text{wTS}_N$ is a tuple $\text{PTS} = (\mathcal{Q}_{\text{PTS}}, q_{\text{PTS}}^0, \longrightarrow_{\text{PTS}}, w_{\text{PTS}}, \mathcal{AP}, L_{\text{PTS}})$ where (a) $\mathcal{Q}_{\text{PTS}} = \mathcal{Q}_1 \times \mathcal{Q}_2 \times \cdots \times \mathcal{Q}_N$ is the set of states; (b) $q_{\text{PTS}}^0 = (q_1^0, q_2^0, \ldots, q_N^0) \in \mathcal{Q}_{\text{PTS}}$ is the initial state, (c) $\longrightarrow_{\text{PTS}} \subseteq \mathcal{Q}_{\text{PTS}} \times \mathcal{Q}_{\text{PTS}}$ is the transition relation defined by the rule $\frac{\bigwedge_{\forall i}(q_i \rightarrow_i q_i')}{q_{\text{PTS}} \rightarrow_{\text{PTS}} q_{\text{PTS}}'}$, where with slight abuse of notation $q_{\text{PTS}} = (q_1, \ldots, q_N) \in \mathcal{Q}_{\text{PTS}}$, $q_i \in \mathcal{Q}_i$. The state q_{PTS}' is defined accordingly; (d) $w_{\text{PTS}} : \mathcal{Q}_{\text{PTS}} \times \mathcal{Q}_{\text{PTS}} \rightarrow \mathbb{R}_+$ is a cost function that assigns weights/cost to each possible transition in PTS, defined as $w_{\text{PTS}}(q_{\text{PTS}}, q_{\text{PTS}}') = \sum_{i=1}^{N} w_i(\Pi|_{\text{wTS}_i} q_{\text{PTS}}, \Pi|_{\text{wTS}_i} q_{\text{PTS}}')$, where $q_{\text{PTS}}', q_{\text{PTS}} \in \mathcal{Q}_{\text{PTS}}$, and $\Pi_{\text{wTS}_i} q_{\text{PTS}}$ stands for the projection of state q_{PTS} onto the state space of wTS_i. The state $\Pi_{\text{wTS}_i} q_{\text{PTS}} \in \mathcal{Q}_i$ is obtained by removing all states in q_{PTS} that do not belong to $\mathcal{Q}_i$; (e) $\mathcal{AP} = \bigcup_{i=1}^{N} \mathcal{AP}_i$ is the set of atomic propositions; and, (f) $L_{\text{PTS}} = \bigcup_{\forall i} L_i : \mathcal{Q}_{\text{PTS}} \rightarrow \mathcal{AP}$ is an observation/output function giving the set of atomic propositions that are satisfied at a state $q_{\text{PTS}} \in \mathcal{Q}_{\text{PTS}}$.

In what follows, we give definitions related to the PTS, that we will use throughout the rest of the paper. An *infinite path* τ of a PTS is an infinite sequence of states, $\tau = \tau(1)\tau(2)\tau(3)\ldots$ such that $\tau(1) = q_{\text{PTS}}^0$, $\tau(k) \in \mathcal{Q}_{\text{PTS}}$, and $(\tau(k), \tau_i(k+1)) \in \rightarrow_{\text{PTS}}$, $\forall k \in \mathbb{N}_+$, where k is an index that points to the k-th entry of τ denoted by $\tau(k)$. The *trace* of an infinite path $\tau = \tau(1)\tau(2)\tau(3)\ldots$ of a PTS, denoted by $\texttt{trace}(\tau) \in \left(2^{\mathcal{AP}}\right)^{\omega}$, where ω denotes infinite repetition, is an infinite word that is determined by the sequence of atomic propositions that are true in the states along τ, i.e., $\texttt{trace}(\tau) = L(\tau(1))L(\tau(2))\ldots$. A *finite path* of a PTS can be defined accordingly. Given the definition of the weights w_{PTS} in Definition 2.2, the *cost* of a finite path τ is defined as $\hat{J}(\tau) = \sum_{k=1}^{|\tau|-1} w_{\text{PTS}}(\tau(k), \tau(k+1))$, where, $|\tau|$ stands for the number of states in τ.

We assume that the robots have to accomplish a complex collaborative task captured by a global LTL statement ϕ defined over the set of atomic propositions $\mathcal{AP} = \bigcup_{i=1}^{N} \mathcal{AP}_i$. Due to space limitations, we abstain from formally defining the semantics and syntax of LTL. A detailed overview can be found in [8]. Given an LTL formula ϕ, we define the *language* $\texttt{Words}(\phi) = \left\{\sigma \in (2^{\mathcal{AP}})^{\omega} | \sigma \models \phi\right\}$, where $\models \subseteq (2^{\mathcal{AP}})^{\omega} \times \phi$ is the satisfaction relation, as the set of infinite words $\sigma \in (2^{\mathcal{AP}})^{\omega}$ that satisfy the LTL formula ϕ. Any LTL formula ϕ can be translated into a Nondeterministic Büchi Automaton (NBA) B over $(2^{\mathcal{AP}})^{\omega}$ defined as follows [9].

Definition 2.3 (NBA): 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.

Given the PTS and the NBA B that corresponds to the LTL ϕ, we can now define the *Product Büchi Automaton* (PBA) $P = \text{PTS} \otimes B$, as follows [8]:

Definition 2.4 (PBA): Given the product transition system $\text{PTS} = (\mathcal{Q}_{\text{PTS}}, q_{\text{PTS}}^0, \longrightarrow_{\text{PTS}}, w_{\text{PTS}}, \mathcal{AP}, L_{\text{PTS}})$ and the NBA $B = (\mathcal{Q}_B, \mathcal{Q}_B^0, \Sigma, \rightarrow_B, \mathcal{Q}_B^F)$, we can define the *Product Büchi Automaton* $P = \text{PTS} \otimes B$ as a tuple $P = (\mathcal{Q}_P, \mathcal{Q}_P^0, \longrightarrow_P, w_P, \mathcal{Q}_P^F)$ where (a) $\mathcal{Q}_P = \mathcal{Q}_{\text{PTS}} \times \mathcal{Q}_B$ is the set of states; (b) $\mathcal{Q}_P^0 = q_{\text{PTS}}^0 \times \mathcal{Q}_B^0$ is a set of initial states; (c) $\longrightarrow_P \subseteq \mathcal{Q}_P \times 2^{\mathcal{AP}} \times \mathcal{Q}_P$ is the transition relation defined by the rule: $\frac{(q_{\text{PTS}} \rightarrow_{\text{PTS}} q_{\text{PTS}}') \wedge (q_B \xrightarrow{L_{\text{PTS}}(q_{\text{PTS}})} q_B')}{q_P = (q_{\text{PTS}}, q_B) \longrightarrow_P q_P' = (q_{\text{PTS}}', q_B')}$. Transition from state $q_P \in \mathcal{Q}_P$ to $q_P' \in \mathcal{Q}_P$, is denoted by $(q_P, q_P') \in \longrightarrow_P$, or $q_P \longrightarrow_P q_P'$; (d) $w_P(q_{\text{P}}, q_{\text{P}}') = w_{\text{PTS}}(q_{\text{PTS}}, q_{\text{PTS}}')$, where $q_{\text{P}} = (q_{\text{PTS}}, q_B)$ and $q_{\text{P}}' = (q_{\text{PTS}}', q_B')$; and (e) $\mathcal{Q}_P^F = \mathcal{Q}_{\text{PTS}} \times \mathcal{Q}_B^F$ is a set of accepting/final states.

Given ϕ and the PBA an infinite path τ of a PTS satisfies ϕ if and only if $\texttt{trace}(\tau) \in \texttt{Words}(\phi)$, which is equivalently denoted by $\tau \models \phi$. Specifically, if there is a path satisfying ϕ, then there exists a path $\tau \models \phi$ that 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 τ^{pre} is executed only once followed by the indefinite execution of the suffix part τ^{suf}. The prefix part τ^{pre} is the projection of a finite path p^{pre} that lives in $\mathcal{Q}_P$ onto $\mathcal{Q}_{\text{PTS}}$. The path p^{pre} starts from an initial state $q_P^0 \in \mathcal{Q}_P^0$ and ends at a final state $q_P^F \in \mathcal{Q}_P^F$, i.e., it has the following structure $p^{\text{pre}} = (q_{\text{PTS}}^0, q_B^0)(q_{\text{PTS}}^1, q_B^1)\ldots(q_{\text{PTS}}^K, q_B^K)$ with $(q_{\text{PTS}}^K, q_B^K) \in \mathcal{Q}_P^F$. The suffix part τ^{suf} is the projection of a finite path p^{suf} that lives in $\mathcal{Q}_P$ onto $\mathcal{Q}_{\text{PTS}}$. The path p^{suf} is a cycle around the final state $(q_{\text{PTS}}^K, q_B^K)$, i.e., it has the following structure $p^{\text{suf}} = (q_{\text{PTS}}^K, q_B^K)(q_{\text{PTS}}^{K+1}, q_B^{K+1})\ldots(q_{\text{PTS}}^{K+S}, q_B^{K+S})(q_{\text{PTS}}^{K+S+1}, q_B^{K+S+1})$, where $(q_{\text{PTS}}^{K+S+1}, q_B^{K+S+1}) = (q_{\text{PTS}}^K, q_B^K)$. Then our goal is to compute a plan $\tau = \tau^{\text{pre}}[\tau^{\text{suf}}]^{\omega} = \Pi|_{\text{PTS}} p^{\text{pre}}[\Pi|_{\text{PTS}} p^{\text{pre}}]^{\omega}$, where $\Pi|_{\text{PTS}}$ stands for the projection on the state-space $\mathcal{Q}_{\text{PTS}}$, so that the following objective function is minimized

$$J(\tau) = \beta \hat{J}(\tau^{\text{pre}}) + (1-\beta)\hat{J}(\tau^{\text{suf}}), \tag{1}$$

where $\hat{J}(\tau^{\text{pre}})$ and $\hat{J}(\tau^{\text{suf}})$ stand for the cost of the prefix and suffix part, respectively and $\beta \in [0,1]$ is a user-specified parameter. Specifically, in this paper we address the following problem.

Problem 1: Given a global LTL specification ϕ, and transition systems wTS_i, for all robots i, determine a discrete team plan τ that satisfies ϕ, i.e., $\tau \models \phi$, and minimizes (1).

III. Sampling-based Optimal Control Synthesis

In this section, we build upon our previous work [3] to propose a sampling-based optimal control synthesis algorithm

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

Input: LTL formula ϕ, $\{\text{wTS}_i\}_{i=1}^N$, $q^0_{\text{PTS}} \in \mathcal{Q}_{\text{PTS}}$, maximum numbers of iterations $n^{\text{pre}}_{\max}$, $n^{\text{suf}}_{\max}$
Output: Optimal plans $\tau \models \phi$

1 Convert ϕ to a NBA $B = (\mathcal{Q}_B, \mathcal{Q}^0_B, \rightarrow_B, \mathcal{Q}^F_B)$;
2 $[\{\Sigma^{\text{feas}}_{q_B,q'_B}\}_{\forall q_B,q'_B \in \mathcal{Q}_B}] =$ `FeasibleWords`$(\{\Sigma_{q_B,q'_B}\}_{\forall q_B,q'_B \in \mathcal{Q}_B})$;
3 Construct graph $\mathcal{G}_B$ and $d(q_B, q'_B)$;
4 Define goal set: $\mathcal{X}^{\text{pre}}_{\text{goal}}$;
5 **for** $b_0 = 1 : |\mathcal{Q}^0_B|$ **do**
6 Root of the tree: $q^r_P = (q^0_{\text{PTS}}, q^0_B)$, $q^0_B = \mathcal{Q}^0_B(b_0)$;
7 $[\mathcal{G}_T, \mathcal{P}] =$ `ConstructTree`$(\mathcal{X}^{\text{pre}}_{\text{goal}}, \text{wTS}_1, \ldots, \text{wTS}_N, B, q^r_P, n^{\text{pre}}_{\max})$;
8 **for** $a = 1 : |\mathcal{P}|$ **do**
9 $\tau^{\text{pre},a} =$ `FindPath`$(\mathcal{G}_T, q^r_P, \mathcal{P}(a))$;
10 **for** $a = 1 : |\mathcal{P}|$ **do**
11 Root of the tree: $q^r_P = \mathcal{P}(a)$;
12 Define goal set: $\mathcal{X}^{\text{suf}}_{\text{goal}}(q^r_P)$;
13 **if** $(q^r_P \in \mathcal{X}^{suf}_{goal}) \ \wedge \ (w_P(q^r_P, q^r_P) = 0)$ **then**
14 $\mathcal{G}_T = (\{q^r_P\}, \{q^r_P, q^r_P\}, 0)$;
15 $\mathcal{S}_a = \{q^r_P\}$;
16 **else**
17 $[\mathcal{G}_T, \mathcal{S}_a] =$ `ConstructTree`$(\mathcal{X}^{\text{suf}}_{\text{goal}}, \{\text{wTS}_i\}_{i=1}^N, B, q^r_P, n^{\text{suf}}_{\max})$;
18 Compute $\tau^{\text{suf},a}$ (see [3]);
19 $a_{q^0_B} = \text{argmin}_a(\hat{J}(\tau^{\text{pre},a}) + \hat{J}(\tau^{\text{suf},a}))$;
20 $a^* = \text{argmin}_{a_{q^0_B}}(\hat{J}(\tau^{\text{pre}}_{q^0_B}) + \hat{J}(\tau^{\text{suf}}_{q^0_B}))$;
21 Optimal Plan: $\tau = \tau^{\text{pre},a^*}[\tau^{\text{suf},a^*}]^\omega$;

that can synthesize optimal motion plans τ in prefix-suffix structure, i.e., $\tau = \tau^{\text{pre}}[\tau^{\text{suf}}]^\omega$, that satisfy a given global LTL specification ϕ from PBA with arbitrarily large state-space. The main difference with [3] lies in the sampling process. The procedure is based on the incremental construction of a directed tree that approximately represents the state-space $\mathcal{Q}_P$ and the transition relation $\rightarrow_P$ of the PBA defined in Definition 2.4. In what follows, we denote by $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$ the tree that approximately represents the PBA P. Also, we denote by q^r_P the root of $\mathcal{G}_T$. The set of nodes $\mathcal{V}_T$ contains the states of $\mathcal{Q}_P$ that have already been sampled and added to the tree structure. The set of edges $\mathcal{E}_T$ captures transitions between 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 function $\texttt{Cost} : \mathcal{V}_T :\rightarrow \mathbb{R}_+$ assigns the cost of reaching node $q_P \in \mathcal{V}_T$ from the root q^r_P of the tree. In other words, $\texttt{Cost}(q_P) = \hat{J}(\tau_T)$, where $q_P \in \mathcal{V}_T$ and τ_T is the path in the tree $\mathcal{G}_T$ that connects the root to q_P.

The construction of the prefix and the suffix part is described in Algorithm 1. In lines 1-3, first the LTL formula is translated to a NBA B and then B is pruned by removing transitions that can never happen. Then, in lines 4-9, the prefix parts $\tau^{\text{pre},a}$ are constructed, followed by the construction of their respective suffix parts $\tau^{\text{suf},a}$ in lines 10-19. Then, the optimal plan $\tau = \tau^{\text{pre},a^*}[\tau^{\text{suf},a^*}]^\omega \models \phi$ is synthesized in lines 20-21.

A. Feasible Words

In this section, given the NBA B that corresponds to the assigned LTL formula ϕ, we define a function $d : \mathcal{Q}_B \times \mathcal{Q}_B \rightarrow \mathbb{N}$ that returns the minimum number of *feasible* NBA transitions that are required to reach a state $q'_B \in \mathcal{Q}_B$ starting from a state $q_B \in \mathcal{Q}_B$ [lines 1- 3, Alg. 1]. This function will be used in the construction of the prefix and suffix parts to bias the sampling process. A *feasible* NBA transition is defined as follows.

Definition 3.1 (Feasible NBA transitions): A transition $(q_B, \sigma^*, q'_B) \in \rightarrow_B$ is feasible if the finite word $\sigma^* \in \Sigma = 2^{\mathcal{AP}}$ is a *feasible* word, i.e., if σ^* can be generated by the PTS defined in Definition 2.2.

To characterize the words $\sigma^* \in \Sigma$ that are feasible, we need first to define the words $\sigma^*_i \in \Sigma_i = 2^{\mathcal{AP}_i}$ that are feasible, i.e, the words that can be generated by wTS_i.

Definition 3.2 (Feasible words $\sigma^*_i \in \Sigma_i$): A word $\sigma^*_i \in \Sigma_i$ is *feasible* if and only if $\sigma^*_i \not\models b^{\text{inf}}_i$, where b^{inf}_i is a Boolean formula defined as $b^{\text{inf}}_i = \vee_{\forall r_j}(\vee_{\forall r_e}(\pi^{r_j}_i \wedge \pi^{r_e}_i))$.
Note that the Boolean formula b^{inf}_i is satisfied by any finite word $\sigma^*_i \in \Sigma_i$ that requires robot i to be present in two or more disjoint regions, simultaneously.[2]

Definition 3.3 (Feasible words $\sigma^* \in \Sigma$): A word $\sigma^* \in \Sigma$ is *feasible* if and only if $\sigma^*_i \not\models b^{\text{inf}}_i$, for all robots i, where $\sigma^*_i = \Pi|_{\Sigma_i}\sigma^*$ and $\Pi|_{\Sigma_i}\sigma^*$ stands for the projection of the word σ^* onto $\Sigma_i = 2^{\mathcal{AP}_i}$.

In what follows, we define the function $d : \mathcal{Q}_B \times \mathcal{Q}_B \rightarrow N$ that returns the minimum number of feasible NBA transitions that are required to reach a state $q'_B \in \mathcal{Q}_B$ starting from a state $q_B \in \mathcal{Q}_B$ based on the transition relation $\rightarrow_B$. To define this function, we first construct sets $\Sigma^{\text{feas}}_{q_B,q'_B} \subseteq \Sigma$ that collect feasible finite words σ^* that enable transition from a state $q_B \in \mathcal{Q}_B$ to $q'_B \in \mathcal{Q}_B$ according to $\rightarrow_B$ [line 2, Alg. 1]. To construct these sets, sets $\Sigma_{q_B,q'_B} \subseteq \Sigma$ that collect all finite (feasible or infeasible) words $\sigma^* \in \Sigma$ that enable a transition from $q_B \in \mathcal{Q}_B$ to $q'_B \in \mathcal{Q}_B$, for all $q_B, q'_B \in \mathcal{Q}_B$, are required.

Next, we view the NBA as a directed graph $\mathcal{G}_B = \{\mathcal{V}_B, \mathcal{E}_B\}$, where the set of nodes $\mathcal{V}_B$ is indexed by the states $q_B \in \mathcal{Q}_B$ and the set of edges $\mathcal{E}_B \subseteq \mathcal{V}_B \times \mathcal{V}_B$ collects the edges from nodes/states q_B to q'_B denoted by (q_B, q'_B), where (q_B, q'_B) exists if $\Sigma^{\text{feas}}_{q_B,q'_B} \neq \emptyset$ [line 3, Alg. 1]. Assigning weights equal to one to all edges in the set $\mathcal{E}_B$, we define the function $d : \mathcal{Q}_B \times \mathcal{Q}_B \rightarrow \mathbb{N}$ as

$$d(q_B, q'_B) = \begin{cases} |SP_{q_B,q'_B}|, \text{if } SP_{q_B,q'_B} \text{ exists,} \\ \infty, \qquad \text{otherwise,} \end{cases} \tag{2}$$

where SP_{q_B,q'_B} denotes the shortest path in $\mathcal{G}_B$ from $q_B \in \mathcal{V}_B$ to $q'_B \in \mathcal{V}_B$ and $|SP_{q_B,q'_B}|$ stands for its cost.

B. Construction of Optimal Prefix Parts

In this section we describe the construction of the tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$ that will be used for the synthesis of the prefix part [lines 4-9, Alg. 1]. Since the prefix part connects an initial state $q^0_P = (q^0_{\text{PTS}}, q^0_B) \in \mathcal{Q}^0_P$ to an *accepting* state $q_P = (q_{\text{PTS}}, q_B) \in \mathcal{Q}^F_P$, with $q_B \in \mathcal{Q}^F_B$,

[2]Note that if we consider regions r_j that are not necessarily disjoint, then the Boolean formula b^{inf}_i is defined as $b^{\text{inf}}_i = \vee_{\forall r_j}(\vee_{\forall r_e \text{s.t.} r_j \cap r_e = \emptyset}(\pi^{r_j}_i \wedge \pi^{r_e}_i))$. Note also that definition of infeasible words depends on the problem at hand, i.e, the defition of atomic propositions included in the sets $\mathcal{AP}_i$.

Algorithm 2: Function $[\mathcal{G}_T, \mathcal{Z}] = \texttt{ConstructTree}(\mathcal{X}_{\text{goal}}, \text{wTS}_1, \ldots, \text{wTS}_N, B, q_P^r, n_{\max})$

1 $\mathcal{V}_T = \{q_P^r\}$, $\mathcal{E}_T = \emptyset$, $\texttt{Cost}(q_P^r) = 0$;
2 **if** *prefix* **then**
3 Select a feasible NBA final state $q_B^{F,\text{feas}} \in \mathcal{Q}_B^F$.;
4 **if** $q_B^{F,feas}$ *does not exist* **then**
5 Exit the function and set $\mathcal{Z} = \emptyset$;
6 $\mathcal{D}_{\min} = \{q_P^r\}$;
7 **for** $n = 1 : n_{max}$ **do**
8 **if** *prefix* **then**
9 $q_{\text{PTS}}^{\text{new}} = \texttt{Sample}(\mathcal{V}_T, \text{wTS}_1, \ldots, \text{wTS}_N, q_B^{F,\text{feas}})$;
10 **if** *suffix* **then**
11 $q_{\text{PTS}}^{\text{new}} = \texttt{Sample}(\mathcal{V}_T, \text{wTS}_1, \ldots, \text{wTS}_N, q_B^r)$;
12 **for** $b = 1 : |\mathcal{Q}_B|$ **do**
13 $q_B^{\text{new}} = \mathcal{Q}_B(b)$;
14 $q_P^{\text{new}} = (q_{\text{PTS}}^{\text{new}}, q_B^{\text{new}})$;
15 **if** $q_P^{new} \notin \mathcal{V}_T$ **then**
16 $[\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}] = \texttt{Extend}(q_P^{\text{new}}, \rightarrow_P)$;
17 Update $\mathcal{D}_{\min}$;
18 **if** $q_P^{new} \in \mathcal{V}_T$ **then**
19 $[\mathcal{E}_T, \texttt{Cost}] = \texttt{Rewire}(q_P^{\text{new}}, \mathcal{V}_T, \mathcal{E}_T, \texttt{Cost})$;
20 $\mathcal{Z} = \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}$;

we can define the goal region for the tree $\mathcal{G}_T$, as $\mathcal{X}_{\text{goal}}^{\text{pre}} = \{q_P = (q_{\text{PTS}}, q_B) \in \mathcal{Q}_P \mid q_B \in \mathcal{Q}_B^F\}$ [line 4, Alg. 1]. The root q_P^r of the tree is an initial state $q_P^0 = (q_{\text{PTS}}^0, q_B^0)$ of the PBA and the following process is repeated for each initial state $q_B^0 \in \mathcal{Q}_B^0$, in parallel [line 5-6, Alg. 1]. The construction of the tree is described in Algorithm 2 [line 7, Alg. 1]. In line 6 of Algorithm 1, $\mathcal{Q}_B^0(b_0)$ stands for the b_0-th initial state assuming an arbitrary enumeration of the elements of the set $\mathcal{Q}_B^0$. The set $\mathcal{V}_T$ initially contains only the root q_P^r, i.e., an initial state of the PBA and, therefore, the set of edges is initialized as $\mathcal{E}_T = \emptyset$ [line 1 , Alg. 2]. By convention, we assume that the cost of q_P^r is zero. Given the root q_P^r we select a *feasible* final state $q_B^F \in \mathcal{Q}_B^F$, such that (i) $d(q_B^0, q_B^F) \neq \infty$ and (ii) $d(q_B^F, q_B^F) \neq \infty$. Among all final states that satisfy both (i) and (ii), we select one randomly denoted by $q_B^{F,\text{feas}}$ [line 3, Alg. 2]. If there does not exist such a state $q_B^{F,\text{feas}}$, then this means that there is no prefix-suffix plan associated with the initial state q_B^0. In this case, the construction of the tree stops without having detected any final states around which a loop exists [lines 4-5, Alg. 2]. The final state $q_B^{F,\text{feas}}$ will be used in the following subsection in order to bias the exploration of the PBA towards this state. We also define the set $\mathcal{D}_{\min}$ that collects the nodes $q_P = (q_{\text{PTS}}, q_B) \in \mathcal{V}_T$ that have the minimum distance $d(q_B, q_B^{F,\text{feas}})$ among all nodes in $\mathcal{V}_T$, i.e., $\mathcal{D}_{\min} = \{q_P = (q_{\text{PTS}}, q_B) \in \mathcal{V}_T \mid d(q_B, q_B^{F,\text{feas}}) = d_{\min}\}$, where $d_{\min} = \min \cup_{\forall q_B \in \Pi|_B \mathcal{V}_T} \{d(q_B, q_B^{F,\text{feas}})\}$ and $\Pi|_B \mathcal{V}_T \subseteq \mathcal{Q}_B$ stands for the projection of all states $q_P \in \mathcal{V}_T \subseteq \mathcal{Q}_P$ onto $\mathcal{Q}_B$. The set $\mathcal{D}_{\min}$ initially collects only the root [line 6, Alg2].

1) Sampling a state $q_P^{new} \in \mathcal{Q}_P$: The first step in the construction of the graph $\mathcal{G}_T$ is to sample a state q_P^{new} from the state-space of the PBA. This is achieved by a sampling function `Sample`; see Algorithm 3. Specifically, we first create a state $q_{\text{PTS}}^{\text{rand}} = \Pi|_{\text{PTS}} q_P^{\text{rand}}$, where q_P^{rand} is sampled from a given discrete distribution $f_{\text{rand}}(q_P|\mathcal{V}_T) : \mathcal{V}_T \rightarrow [0, 1]$ [line 1, Alg. 3]. The probability density function $f_{\text{rand}}(q_P|\mathcal{V}_T)$ defines the probability of selecting the state $q_P \in \mathcal{V}_T$ as the state q_P^{rand} at iteration n of Algorithm 2 given the set $\mathcal{V}_T$. The distribution f_{rand} is defined as follows:

$$f_{\text{rand}}(q_P|\mathcal{V}_T, \mathcal{D}_{\min}) = \begin{cases} p_{\text{rand}} \frac{1}{|\mathcal{D}_{\min}|}, & \text{if } q_P \in \mathcal{D}_{\min} \\ (1 - p_{\text{rand}}) \frac{1}{|\mathcal{V}_T \setminus \mathcal{D}_{\min}|}, & \text{otherwise,} \end{cases} \quad (3)$$

where $p_{\text{rand}} \in (0.5, 1)$ stands for the probability of selecting *any* node $q_P \in \mathcal{D}_{\min}$ to be q_P^{rand}. Note that p_{rand} can change with iterations n but it should always satisfy $p_{\text{rand}} \in (0.5, 1)$ so that states $q_P = (q_{\text{PTS}}, q_B) \in \mathcal{D}_{\min} \subseteq \mathcal{V}_T$ are selected more often to be q_P^{rand} [line 1, Alg. 3]. Given $q_{\text{PTS}}^{\text{rand}}$, in our previous work [3], we sample a state $q_{\text{PTS}}^{\text{new}}$ from a discrete distribution $f_{\text{new}}(q_{\text{PTS}}|q_{\text{PTS}}^{\text{rand}})$ so that $q_{\text{PTS}}^{\text{new}}$ is reachable from $q_{\text{PTS}}^{\text{rand}}$. Here, we sample a state $q_{\text{PTS}}^{\text{new}}$ so that it is both reachable from $q_{\text{PTS}}^{\text{rand}}$ and also it can lead to a final state $q_P^F = (q_{\text{PTS}}, q_B^{F,\text{feas}}) \in \mathcal{Q}_P^F$ by following the shortest path in $\mathcal{G}_B$ that connects q_B^0 to $q_B^{F,\text{feas}}$.

a) Selection of $q_B^{min} \in \mathcal{Q}_B$: First, given $q_P^{\text{rand}} = (q_{\text{PTS}}^{\text{rand}}, q_B^{\text{rand}})$, we first construct the reachable set $\mathcal{R}_B(q_B^{\text{rand}})$ that collects all states $q_B \in \mathcal{Q}_B$ that can be reached in one hop in B from q_B^{rand} given the observation $L_{\text{PTS}}(q_{\text{PTS}}^{\text{rand}})$ [line 2, Alg. 3], i.e., $\mathcal{R}_B(q_B^{\text{rand}}) = \{q_B \in \mathcal{Q}_B \mid (q_B^{\text{rand}}, L_{\text{PTS}}(q_{\text{PTS}}^{\text{rand}}), q_B) \in \rightarrow_B\}$. Given $\mathcal{R}_B(q_B^{\text{rand}})$ we construct the set $\mathcal{R}_B^{\min}(q_B^{\text{rand}})$ that collects the states $q_B \in \mathcal{R}_B(q_B^{\text{rand}})$ that have the minimum distance from $q_B^{F,\text{feas}}$ among all other nodes $q_B \in \mathcal{R}_B(q_B^{\text{rand}})$, i.e., $\mathcal{R}_B^{\min}(q_B^{\text{rand}}) = \{q_B \in \mathcal{R}_B(q_B^{\text{rand}}) \mid d(q_B, q_B^{F,\text{feas}}) = \min_{q_B' \in \mathcal{R}_B(q_B^{\text{rand}})} d(q_B', q_B^{F,\text{feas}})\} \subseteq \mathcal{R}_B(q_B^{\text{rand}})$.

In what follows, we denote by $q_B^{\text{cand,min}}$ the states that belong to $\mathcal{R}_B^{\min}(q_B^{\text{rand}})$. For every state $q_B^{\text{cand,min}} \in \mathcal{R}_B^{\min}(q_B^{\text{rand}})$, we construct the set $\mathcal{R}_B^{\text{decr}}(q_B^{\text{cand,min}})$ that collects all states $q_B \in \mathcal{Q}_B$, for which (i) there exists a feasible word $\sigma^* \in \Sigma^{\text{feas}}_{q_B^{\text{cand,min}}, q_B}$ such that $q_B^{\text{cand,min}} \xrightarrow{\sigma^*} q_B$, and (ii) q_B is closer to $q_B^{F,\text{feas}}$ than $q_B^{\text{cand,min}}$ is, i.e., $\mathcal{R}_B^{\text{decr}}(q_B^{\text{cand,min}}) = \{q_B \in \mathcal{Q}_B \mid (\Sigma^{\text{feas}}_{q_B^{\text{cand,min}}, q_B} \neq \emptyset) \wedge d(q_B, q_B^{F,\text{feas}}) = d(q_B^{\text{cand,min}}, q_B^{F,\text{feas}}) - 1\} \subseteq \mathcal{Q}_B$. We collect all states $q_B^{\text{cand,min}} \in \mathcal{R}_B^{\min}(q_B^{\text{rand}})$ that satisfy $\mathcal{R}_B^{\text{decr}}(q_B^{\text{cand,min}}) \neq \emptyset$ in the set $\mathcal{M}(q_B^{\text{rand}})$, i.e., $\mathcal{M}(q_B^{\text{rand}}) = \left\{q_B^{\text{cand,min}} \mid \mathcal{R}_B^{\text{decr}}(q_B^{\text{cand,min}}) \neq \emptyset\right\} \subseteq \mathcal{R}_B^{\min}(q_B^{\text{rand}})$ [line 4, Alg.3]. Next, given the set $\mathcal{M}(q_B^{\text{rand}})$, we sample a state $q_B^{\min} \in \mathcal{M}(q_B^{\text{rand}})$ from a discrete distribution $f_B^{\min}(q_B|\mathcal{M}(q_B^{\text{rand}})) : \mathcal{M}(q_B^{\text{rand}}) \rightarrow [0, 1]$ defined as $f_B^{\min}(q_B|\mathcal{M}(q_B^{\text{rand}})) = \frac{1}{|\mathcal{M}(q_B^{\text{rand}})|}$ [line 6, Alg.3] . Note that if $\mathcal{M}(q_B^{\text{rand}}) = \emptyset$, then no sample will be taken at this iteration [line 12, Alg. 3].

b) Selection of $q_B^{decr} \in \mathcal{R}_B^{decr}(q_B^{min})$: Given the state $q_B^{\min}$ and its respective set $\mathcal{R}_B^{\text{decr}}(q_B^{\min})$, we sample a state $q_B^{\text{decr}} \in \mathcal{R}_B^{\text{decr}}(q_B^{\min})$ from a given discrete distribution $f_B^{\text{decr}}(q_B|\mathcal{R}_B^{\text{decr}}(q_B^{\min})) : \mathcal{R}_B^{\text{decr}}(q_B^{\min}) \rightarrow [0, 1]$ [line 7, Alg. 3] defined as $f_B^{\text{decr}}(q_B|\mathcal{R}_B^{\text{decr}}(q_B^{\min})) = \frac{1}{|\mathcal{R}_B^{\text{decr}}(q_B^{\min})|}, \forall q_B \in \mathcal{R}_B^{\text{decr}}(q_B^{\min})$. Note that if $\mathcal{R}_B^{\text{decr}}(q_B^{\min}) = \emptyset$, then no sample $q_{\text{PTS}}^{\text{new}}$ is generated [line 10, alg. 3].

Algorithm 3: Function $q_{\text{PTS}}^{\text{new}} = \texttt{Sample}(\mathcal{V}_T, \mathcal{D}_{\min}, \{\text{wTS}_i\}_{i=1}^N, q_B^{\text{goal}})$

Pick a state $q_P^{\text{rand}} = (q_{\text{PTS}}^{\text{rand}}, q_B^{\text{rand}}) \in \mathcal{V}_T$ from f_{rand};
Compute $\mathcal{R}_B(q_B^{\text{rand}})$;
if $\mathcal{R}_B(q_B^{rand}) \neq \emptyset$ **then**
  Compute $\mathcal{M}(q_B^{\text{rand}}) \subseteq \mathcal{R}_B(q_B^{\text{rand}})$;
  if $\mathcal{M}(q_B^{rand}) \neq \emptyset$ **then**
    Sample $q_B^{\min} \in \mathcal{M}(q_B^{\text{rand}})$ from $f_B^{\min}$;
    Select $q_B^{\text{decr}} \in \mathcal{R}_B^{\text{decr}}(q_B^{\min})$;
    Pick $\sigma^* \in \Sigma_{q_B^{\min}, q_B^{\text{decr}}}$;
  else
    $q_{\text{PTS}}^{\text{new}} = \varnothing$;
else
  $q_{\text{PTS}}^{\text{new}} = \varnothing$;
if $(\mathcal{R}_B(q_B^{rand}) \neq \emptyset) \wedge (\mathcal{M}(q_B^{rand}) \neq \emptyset)$ **then**
  Pick a state q_i^{new} from a given probability distribution $f_{\text{new},i}$, for all robots i;
  Construct $q_{\text{PTS}}^{\text{new}} = (q_1^{\text{new}}, \ldots, q_N^{\text{new}})$;

Given $q_B^{\min}$ and q_B^{decr}, we select a word σ^* from $\Sigma_{q_B^{\min}, q_B^{\text{decr}}} \neq \emptyset$ [line 8, Alg. 3].[3] Given the word σ^*, we construct the set $\mathcal{L}$ so that the i-th element of $\mathcal{L}$ captures the region where robot i has to be so that the word σ^* can be generated.For instance, the set $\mathcal{L}$ corresponding to the word $\sigma^* = \pi_i^{r_j} \pi_z^{r_e}$ is constructed so that $\mathcal{L}(i) = r_j$, $\mathcal{L}(z) = r_e$ and $\mathcal{L}(h) = \varnothing$, for all robots $h \neq i, z$. Then, we sample a state q_i^{new}, for all robots i, from a discrete distribution $f_{\text{new},i}^{\text{pre}}(q_i | q_i^{\text{rand}}) : \mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}}) \to [0,1]$ [line 14, Alg. 3] defined as

$$f_{\text{new},i}^{\text{pre}}(q_i|q_i^{\text{rand}}) = \begin{cases} \frac{1}{|\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})|}, \text{ if } \mathcal{L}(i) = \varnothing, \\ p_{\text{new}}, \text{if } (L(i) \neq \varnothing) \wedge (q_i = SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)) \\ (1 - p_{\text{new}}) \frac{1}{|\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})| - 1}, \text{if } (\mathcal{L}(i) \neq \varnothing) \wedge \\ \quad (q_i \in \mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}}) \setminus SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)), \end{cases} \tag{4}$$

where $\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})$ collects all states in $\mathcal{Q}_i$ that can be reached in one hop from $q_i^{\text{rand}} = \Pi|_{\text{wTS}_i} q_{\text{PTS}}^{\text{rand}}$, i.e., $\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}}) = \{q_i \in \mathcal{Q}_i \mid q_i^{\text{rand}} \to_i q_i\}$. Also, in (4), viewing wTS_i as a graph, $SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}$ stands for the shortest path from the node/state q_i^{rand} to the state $q_i^{\mathcal{L}(i)}$, i.e., $SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}$ is a finite sequence of states in wTS_i that start from q_i^{rand} and end at the state $q_i^{\mathcal{L}(i)}$. Also, $SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)$ stands for the second state in this sequence. Moreover, in (4), p_{new} stands for the probability of selecting the state $SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)$ to be q_i^{new} if $\mathcal{L}(i) \neq \varnothing$. Note that p_{new} can change with iterations n but it should always satisfy $p_{\text{new}} \in (0.5, 1)$ so that the state $SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)$ is selected more often to be q_i^{new}, as it is closer to the state $q_i^{\mathcal{L}(i)}$. Finally, given the states q_i^{new}, we construct $q_{\text{PTS}}^{\text{new}} = (q_1^{\text{new}}, \ldots, q_N^{\text{new}}) \in \mathcal{Q}_{\text{PTS}}$. Observe that by construction of $f_{\text{new},i}^{\text{pre}}$ the state q_i^{new} always lies in $\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})$. As a result, $q_{\text{PTS}}^{\text{new}}$ is reachable from $q_{\text{PTS}}^{\text{rand}}$.

In order to build incrementally a graph whose set of nodes approximates the state-space $\mathcal{Q}_P$ we need to append to $q_{\text{PTS}}^{\text{new}}$ a state from the state-space $\mathcal{Q}_B$ of the NBA B. Let $q_B^{\text{new}} = \mathcal{Q}_B(b)$ [line 13, Alg. 2] be the candidate Büchi state that will be attached to $q_{\text{PTS}}^{\text{new}}$, where $\mathcal{Q}_B(b)$ stands for the b-th state in the set $\mathcal{Q}_B$ assuming an arbitrary enumeration of the elements of the set $\mathcal{Q}_B$. The following procedure is repeated for all $q_B^{\text{new}} = \mathcal{Q}_B(b)$ with $b \in \{1, \ldots, |\mathcal{Q}_B|\}$. First, we construct the state $q_P^{\text{new}} = (q_{\text{PTS}}^{\text{new}}, q_B^{\text{new}}) \in \mathcal{Q}_P$ [line 14, Alg. 2] and then we check if this state can be added to the tree $\mathcal{G}_T$ [lines 15-16, Alg. 2]. If the state q_P^{new} does not already belong to the tree from a previous iteration of Algorithm 2, i.e, if $q_P^{\text{new}} \notin \mathcal{V}_T$ [line 12, Alg. 2], we check which node in $\mathcal{V}_T$ (if there is any) can be the parent of q_P^{new} in the tree $\mathcal{G}_T$. If there exist candidate parents for q_P^{new} then the tree is *extended* towards q_P^{new} and the set $\mathcal{D}_{\min}$ is updated [line 17, Alg. 2]. If $q_P^{\text{new}} \in \mathcal{V}_T$, then the *rewiring* step follows [lines 18-19, Alg. 2] that aims to reduce the cost of nodes $q_P \in \mathcal{V}_T$. A detailed description of the 'extend' and 'rewire' steps can be found in [3].

2) 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 7, Alg. 2]. Then, we construct the set $\mathcal{P} = \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}^{\text{pre}}$ [line 20, Alg. 2] that collects all the states $q_P \in \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}^{\text{pre}}$. Given the tree $\mathcal{G}_T$ and the set $\mathcal{P}$ we compute the prefix plans $\tau^{\text{pre},a}$, for all $a \in \{1, \ldots |\mathcal{P}|\}$ [lines 8-9, Alg. 1].

C. Construction of Optimal Suffix Parts

Once a prefix plan $\tau^{\text{pre},a}$ is constructed, the corresponding suffix plan $\tau^{\text{suf},a}$ is constructed [lines 10-18, Alg. 1]. To construct the suffix part $\tau_i^{\text{suf},a}$ we build a tree $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \texttt{Cost}\}$ that approximates the PBA P, in a similar way as in Section III-B, and implement a cycle-detection mechanism to identify cycles around the state $P(a)$. The only differences are that: (i) the root of the tree is now $q_P^r = \mathcal{P}(a)$ [line 11, Alg. 1] detected during the construction of the prefix plans, (ii) the goal region corresponding to the root $q_P^r = \mathcal{P}(a)$, is defined as $\mathcal{X}_{\text{goal}}^{\text{suf}}(q_P^r) = \{q_P = (q_{\text{PTS}}, q_B) \in \mathcal{Q}_P \mid (q_P, L(q_{\text{PTS}}), q_P^r) \in \to_P\}$, (iii) we first check if $q_P^r \in \mathcal{X}_{\text{goal}}^{\text{suf}}$ and $w_P(q_P^r, q_P^r) = 0$ [line 13, Alg. 1], and (iv) a probability density function $f_{\text{new},i}^{\text{suf}}$ that is different from (4) is employed. If (iii) holds, then the construction of the tree is trivial [line 14, Alg. 1]. Otherwise the tree $\mathcal{G}_T$ is constructed by Algorithm 2 [line 17, Alg. 1]. As for (iv), $f_{\text{new},i}^{\text{suf}}(q_i|q_i^{\text{rand}}) : \mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}}) \to [0,1]$ [line 14, Alg. 3] that generates q_i^{new} is defined as

$$f_{\text{new},i}^{\text{suf}}(q_i|q_i^{\text{rand}}) = f_{\text{new},i}^{\text{pre}}(q_i|q_i^{\text{rand}}), \tag{5}$$

if $q_B^r \notin \mathcal{R}_B^{\text{decr}}$, where $q_B^r = \Pi|_B q_P^r \in \mathcal{Q}_B^F$. If $q_B^r \in \mathcal{R}_B^{\text{decr}}$, then this means that the NBA part of the root (final state), i.e., $q_B^r \in \mathcal{Q}_B^F$, can be reached in one hop from $q_B^{\min}$. Then in this case, to steer the tree towards the root $q_P^r = (q_{\text{PTS}}^r, q_B^r)$, we select the following probability density function $f_{\text{new},i}^{\text{suf}}(q_i|q_i^{\text{rand}})$ to generate the samples $q_{\text{PTS}}^{\text{new}}$.

$$f_{\text{new},i}^{\text{suf}}(q_i|q_i^{\text{rand}}) = \begin{cases} \frac{1-p_{\text{new}}}{|\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})|}, \text{if } (\mathcal{L}(i)=\varnothing) \wedge (q_i \neq SP_{q_i^{\text{rand}}, q_i^r}(2)), \\ p_{\text{new}}, \text{if } (\mathcal{L}(i) = \varnothing) \wedge (q_i = SP_{q_i^{\text{rand}}, q_i^r}(2)) \\ p_{\text{new}}, \text{if } (L(i) \neq \varnothing) \wedge (q_i = SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)) \\ \frac{1-p_{\text{new}}}{|\mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}})| - 1}, \text{if } (\mathcal{L}(i) \neq \varnothing) \wedge \\ \quad (q_i \in \mathcal{R}_{\text{wTS}_i}(q_i^{\text{rand}}) \setminus SP_{q_i^{\text{rand}}, q_i^{\mathcal{L}(i)}}(2)) \end{cases} \tag{6}$$

[3] To speed up the detection of final states, we always select the same word $\sigma^* \in \Sigma_{q_B^{\min}, q_B^{\text{decr}}}$ for a given pair of states $q_B^{\min}$ and q_B^{decr}.

TABLE I: Feasibility & Scalability Analysis

N	$\lvert\mathcal{Q}_i\rvert$	$\lvert\mathcal{Q}_P\rvert$	$n_{\text{pre}}/n_{\text{suf}}$	Pre1+Suf1	[5]/ [6]
9	9	10^{10}	24/48	0.4+0.5 (secs)	<1 sec
10	10^2	10^{21}	37/33	0.6+0.2 (secs)	<1 sec
10	10^3	10^{30}	33/26	0.6+0.5 (secs)	50/40 secs
10	10^4	10^{40}	11/12	1.5+2.4 (mins)	M/M
100	10^2	10^{200}	69/36	2.4+0.8 (secs)	F/F
100	10^4	10^{400}	22/51	3.1+5.7 (mins)	M/M
200	10^4	10^{800}	41/50	14.1.9+25.7 (mins)	M/M

Table I: The first, second, and third column show the number N of robots, $|\mathcal{Q}_i|$ which is the same for all wTSs, and the size of the PBA, defined as $|\mathcal{Q}_P| = |\mathcal{Q}_B|\Pi_{i=1}^{N}|\mathcal{Q}_i|$. The fourth and fifth column show the average number of iterations n and the average runtime of Algorithm 2 for 10 experiments required to find the first prefix and suffix part, respectively. The runtime of NuSMV [5] and nuXmv [6] are reported in the last column. The 'M' means that the model of the multi-robot system failed to build. The 'F' means that the model of the multi-robot system was built but the LTL formula is too long to input it to NuSMV and nuXmv in a user-friendly way.

where $q_i^r = \Pi|_{\text{wTS}_i} q_{\text{PTS}}^r$. Note that when (5) is employed, the sampling process is biased towards any state $q_P = (q_{\text{PTS}}, q_B^r)$ (and not $q_P = (q_{\text{PTS}}, q_B^{F,\text{feas}})$ as in the prefix part); see also [line 11, Alg. 2]. On the other hand, when (6) is employed the sampling process is biased towards the root $q_P^r = (q_{\text{PTS}}^r, q_B^r)$. Once a tree rooted at $q_P^r = \mathcal{P}(a)$ is constructed, a set $\mathcal{S}_a \subseteq \mathcal{V}_T$ is formed that collects all states $q_P \in \mathcal{V}_T \cap \mathcal{X}_{\text{goal}}^{\text{suf}}(q_P^r)$ [lines 15, 17, Alg. 1]. Given the set $\mathcal{S}_a$, we compute the best suffix plan $\tau^{\text{suf},a}$ associated with $q_P^r = \mathcal{P}(a) \in \mathcal{Q}_P^F$, as in [3], [line 18, Alg. 1], in parallel for all $a \in \{1, \ldots, |\mathcal{P}|\}$.

D. Construction of Optimal Discrete Plans

Given any motion plan $\tau^a = \tau^{\text{pre},a}[\tau^{\text{suf},a}]^\omega$, with $\mathcal{S}_a \neq \emptyset$, and $a \in \{1, \ldots, |\mathcal{P}|\}$ satisfies the LTL task ϕ. Among all synthesized motion plans, the optimal motion plan is selected as in [3] [line 19-21, Alg. 1]. By showing that the proposed biased sampling functions satisfy Assumptions 4.1 and 4.2 in [3] we get the following result; the proof is omitted due to space limitations.

Theorem 3.4: Let p_{rand}^n and p_{new}^n denote the parameters p_{rand} and p_{new} defined in (3) and (4)-(6), respectively, at iteration n of Algorithm 2. If there exists a solution to Problem 1, $p_{\text{rand}}^n > \epsilon_1$, and $p_{\text{new}}^n > \epsilon_2$, for some constants $\epsilon_1 > 0$ and $\epsilon_2 > 0$, then Algorithm 1 is probabilistically complete and asymptotically optimal.

IV. Numerical Experiments

In this section, we present case studies, implemented using MATLAB R2015b on a computer with Intel Core i7 2.2GHz and 4Gb RAM. In what follows, we examine the performance of Algorithm 1 with respect to the size of the network and the size of the wTSs. The results are reported in Table I. For all case studies shown in Table I, we consider the following LTL task $\phi = G(\xi_1 \rightarrow (\bigcirc\neg\xi_1 \mathcal{U} \xi_2)) \wedge (\square\Diamond\xi_1) \wedge (\square\Diamond\xi_3) \wedge (\square\Diamond\xi_4) \wedge (\neg\xi_1 \mathcal{U} \xi_5) \wedge (\square\Diamond\xi_5) \wedge (\square\neg\xi_6) \wedge (\Diamond(\xi_7 \vee \xi_8))$, where ξ_e is a Boolean formula in conjunctive or disjunctive normal form defined over the atomic propositions $\pi_i^{r_j}$. For instance, for the planning problem with order 10^{21}, ξ_1 is selected as $\xi_1 = (\pi_1^{r_{100}}) \wedge (\pi_8^{r_{20}} \vee \pi_8^{r_{90}} \vee \pi_8^{r_{11}}) \wedge (\pi_9^{r_1} \vee \pi_9^{r_{10}} \vee \pi_9^{r_{25}} \vee \pi_9^{r_{35}})$. All other formulas ξ_e are defined similarly. The considered LTL formula corresponds to a NBA with $|\mathcal{Q}_B| = 21$, $|\mathcal{Q}_B^0| = 1$, $|\mathcal{Q}_B^F| = 2$, and 125 transitions. Given the NBA, we construct the sets $\Sigma_{q_B, q_B'}^{\text{feas}} \subseteq 2^{\mathcal{AP}}$ in 0.1 seconds approximately for all case studies in Table I. Note that to speed up the detection of the first feasible plan, we do not execute the rewiring step until the first final state and a loop around it are detected. The resulting average runtimes for 10 experiments are reported in the fifth column of Table I. Note that without the rewiring step, Algorithm 1 can only generate feasible plans. Observe in Table I that Algorithm 1 outperforms NuSMV in terms of both runtime and the size of problems that it can handle. Also, while nuXmv can handle infinite-state transition systems, it is significantly slower than the algorithm proposed here.

Since optimality depends on the discovery of prefix plans to all feasible final states, bias can be used sequentially for every feasible final state $q_B^{\text{F,feas}}$ to discover such plans. Once all feasible final states $q_B^{\text{F,feas}}$ have been detected, or after a pre-determined number of iterations n, the rewiring step is activated and uniform sampling is used to better explore the PBA towards all directions in $\mathcal{Q}_P$. Instead, biased sampling favors exploration towards the shortest paths that lead to the final states. Numerical experiments confirming that Algorithm 1 is optimal can be found in [3].

V. Conclusion

This paper proposed a new temporal logic optimal control synthesis algorithm for large multi-robot systems that scales significantly better than state-of-the-art methods.

References

[1] 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.

[2] A. Ulusoy, S. L. Smith, and C. Belta, "Optimal multi-robot path planning with ltl constraints: guaranteeing correctness through synchronization," in *Distributed Autonomous Robotic Systems*. Springer, 2014, pp. 337–351.

[3] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multi-robot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, 2018. [Online]. Available: DOI:10.1109/TAC.2018.2853558

[4] ——, "Sampling-based control synthesis for multi-robot systems under global temporal specifications," in *Proc. 8th ACM/IEEE International Conference on Cyber-Physical Systems*, Pittsburgh, PA, USA, April 2017, pp. 3–13.

[5] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in *International Conference on Computer Aided Verification*. Springer, 2002, pp. 359–364.

[6] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuxmv symbolic model checker," in *International Conference on Computer Aided Verification*. Springer, 2014, pp. 334–342.

[7] Y. Kantaros and M. M. Zavlanos, "Distributed optimal control synthesis for multi-robot systems under global temporal tasks," in *9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, Porto, Portugal, April 2018, pp. 162–173.

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

[9] 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.