STyLuS*: A Temporal Logic Optimal Control Synthesis Algorithm for Large-Scale Multi-Robot Systems

Yiannis Kantaros¹, Michael M. Zavlanos¹

Abstract
This paper proposes a new highly scalable and asymptotically optimal control synthesis algorithm from linear temporal logic specifications, called STyLuS* for large-scale optimal Temporal Logic Synthesis, that is designed to solve complex temporal planning problems in large-scale multi-robot systems. Existing planning approaches with temporal logic 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 of magnitude more states than those that existing optimal control synthesis methods or off-the-shelf model checkers can manipulate. We show that STyLuS* is probabilistically complete and asymptotically optimal and has exponential convergence rate. This is the first time that convergence rate results are provided for sampling-based optimal control synthesis methods. We provide simulation results that show that STyLuS* can synthesize optimal motion plans for very large multi-robot systems which is impossible using state-of-the-art methods.

Keywords
Temporal logic, optimal control synthesis, formal methods, sampling-based motion planning, multi-robot systems.

1 Introduction
Control synthesis for mobile robots under complex tasks, captured by Linear Temporal Logic (LTL) formulas, builds upon either bottom-up approaches when independent LTL expressions are assigned to robots (Kress-Gazit et al. 2009, 2007; Bhatia et al. 2010; Ulusoy and Belta 2014) or top-down approaches when a global LTL formula describing a collaborative task is assigned to a team of robots (Chen et al. December 2011, 2012), as in this work. Common in the above works is that they rely on model checking theory (Baier and Katoen 2008; Clarke et al. 1999) to find paths that satisfy LTL-specified tasks, without optimizing task performance. Optimal control synthesis under local and global LTL specifications has been addressed in (Smith et al. 2011; Guo and Dimarogonas 2015) and (Kloetzer and Belta 2010; Ulusoy et al. 2013, 2014), respectively. In top-down approaches (Kloetzer and Belta 2010; Ulusoy et al. 2013, 2014), optimal discrete plans are derived for every robot using the individual transition systems that capture robot mobility and a Non-deterministic Büchi Automaton (NBA) that represents the global LTL specification. Specifically, by taking the synchronous product among the transition systems and the NBA, a synchronous Product Büchi Automaton (PBA) can be constructed. Then, representing the latter automaton as a graph and using graph-search techniques, optimal motion plans can be derived that satisfy the global LTL specification and optimize a cost function. As the number of robots or the size of the NBA increases, the state-space of the product automaton grows exponentially and, as a result, graph-search techniques become intractable. The same holds for recent search-based Λ*-type methods (Khalidi, Danish and Saha, Indranil 2018), although they can solve problems an order of magnitude larger than those that graph-search approaches can handle. Consequently, these motion planning algorithms scale poorly with the number of robots and the complexity of the assigned task. A more tractable approach is presented in (Schillinger et al. November, 2016., 2018) that identifies independent parts of the LTL formula and builds a local product automaton for each agent. Nevertheless, this approach can be applied only to finite LTL missions and does not have optimality guarantees.

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 (Kantaros and Zavlanos 2018a). Specifically, this algorithm builds incrementally directed trees that approximately represent the state-space and transitions among states of the product automaton. The

¹Dept. of Mechanical Engineering and Materials Science, Duke University. This work is supported in part by ONR under agreement #N00014 – 18 – 1 – 2374 and by AFOSR under the award #FA9550 – 19 – 1 – 0169.

Corresponding author:
Yiannis Kantaros, Dept. of Mechanical Engineering and Materials Science, Duke University, Durham, NC, 27708.
Email: ik36@duke.edu
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 propose a new highly scalable optimal control synthesis algorithm from LTL specifications, called STyLuS∗ for large-Scale optimal Temporal Logic Synthesis, that is designed to solve complex temporal planning problems in large-scale multi-robot systems. In fact, STyLuS∗ extends the sampling-based synthesis algorithm proposed in (Kantaros and Zavlanos 2018a) by introducing bias in the sampling process. 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, between 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 (Kantaros and Zavlanos 2018a). We show that introducing bias in the sampling process does not violate the sampling assumptions in (Kantaros and Zavlanos 2018a) so that STyLuS∗ inherits the same probabilistic completeness and asymptotic optimality guarantees. Moreover, we provide exponential convergence bounds; the first in the field of optimal control synthesis methods. Note that such guarantees are not provided for the algorithm in (Kantaros and Zavlanos 2018a). We show that by biasing the sampling process we can synthesize optimal motion plans from product automata with order $10^{600}$ states and beyond, which is hundreds of orders more states than those that any existing optimal control synthesis algorithms (Kantaros and Zavlanos 2015, 2017; Ulusoy et al. 2013, 2014; Kantaros and Zavlanos 2018a) can handle. For example, our algorithm in (Kantaros and Zavlanos 2018a), when implemented with uniform sampling, can optimally solve problems with order $10^{10}$ states, many more than existing methods (Kantaros and Zavlanos 2015, 2017; Ulusoy et al. 2013, 2014) but orders of magnitude fewer than STyLuS∗. Compared to off-the-shelf model-checkers, such as NuSMV (Cimatti et al. 2002) and nuXmv (Cavada et al. 2014), 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 $10^{10}$ states, while nuXmv can handle infinite-state synchronous transition systems but it is slower than STyLuS∗. Note that STyLuS∗ can be implemented in a distributed way, as in our recent work (Kantaros and Zavlanos 2018b), which can further decrease the computational time.

Relevant sampling-based control synthesis methods are also presented in (Karaman and Frazzoli June 2012; Vasile and Belta November 2013). These methods consider continuous state spaces and employ sampling-based methods to build discrete abstractions of the environment that are represented by graphs of arbitrary structure, e.g., as in (Karaman and Frazzoli 2011; Janson et al. 2015) for point-to-point navigation. Once these abstractions become expressive enough to generate motion plans that satisfy the LTL specification, graph search methods are applied to the respective PBA to design a feasible path. However, representing the environment using graphs of arbitrary structure compromises scalability of temporal logic planning methods since, as the size of these graphs increases, more resources are needed to save the associated structure and search for optimal plans using graph search methods. While our proposed sampling-based approach assumes that a discrete abstraction of the environment is available, as in (Conner and Habets 2004; Belta et al. 2005; Kloetzer and Belta 2006; Boskos and Dimarogonas 2015), it build trees, instead of arbitrary graphs, to approximate the product automaton. Therefore, it 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. Instead, it tracks sequences of parent nodes starting from desired accepting states. Combined with the proposed biased sampling approach, our method can handle much more complex planning problems with more robots and LTL tasks that correspond to larger NBAs. A more detailed comparison with (Karaman and Frazzoli June 2012; Vasile and Belta November 2013) can be found in (Kantaros and Zavlanos 2018a).

The advantage of using non-uniform sampling functions for sampling-based motion planning has also been demonstrated before, e.g., in (Van den Berg et al. 2005; Zucker et al. 2008; Ichter et al. 2018). For example, (Ichter et al. 2018) learns sampling distributions from demonstrations that are then used to bias sampling. This results in an order of magnitude improvement in success rates and path costs compared to uniform sampling-based methods. Probabilistically safe corridors that are constructed using a learned approximate probabilistic model of a configuration space have also been recently proposed to enhance scalability of sampling-based motion planning and also minimize collision likelihood (Huh et al. 2019). Nevertheless, all these approaches focus on simple point-to-point navigation tasks, unlike the method proposed here that can solve complex tasks captured by LTL formulas. Related are also Monte Carlo Tree Search (MCTS) algorithms that are used to find optimal decisions in games or planning problems and they also rely on building search trees (Zagoruyko et al. 2019; Best et al. 2019; Browne et al. 2012). The main challenge in MCTS is to balance between exploration and exploitation, which can be achieved using, e.g., the Upper Confidence Bound for Trees (UCT) algorithm (Kocsis, Levente and Szepesvári, Csaba 2006). MCTS methods involve a simulation step that requires to complete one random playout from the child node that is added to the tree. STyLuS∗ completely avoids this step which can be computationally expensive for complex planning problems that require a large horizon until the task is accomplished. Moreover, to the best of our knowledge, MCTS has not been applied to LTL planning problems.

A preliminary version of this work can be found in (Kantaros and Zavlanos 2018c). In (Kantaros and Zavlanos 2018c) it is shown that the proposed biased sampling process satisfies the assumptions in (Kantaros and Zavlanos 2018a) so that the proposed algorithm inherits the same probabilistic completeness and asymptotic optimality. 

...
guarantees. Compared to (Kantaros and Zavlanos 2018c), here we additionally provide exponential convergence rate bounds. To the best of our knowledge, this is the first sampling-based motion planning algorithm with temporal logic specifications that also has convergence-rate guarantees. Moreover, we provide additional extensive simulation studies that show the effect of bias in sampling in the convergence rate of the algorithm, as well as scalability of our method with respect to the number of robots, the size of the transition systems, and the size of the NBA. We also compare our method to relevant state-of-the-art methods. To the best of our knowledge, this is the first optimal control synthesis method for global temporal logic specifications with optimality and convergence guarantees that can be applied to large-scale multi-robot systems.

The rest of the paper is organized as follows. In Section 2 we present the problem formulation. In Section 3 we describe our proposed sampling-based planning algorithm and in Section 4 we examine its correctness, optimality, and convergence rate. Numerical experiments are presented in Section 5.

2 Problem Formulation

Consider \( N \) mobile robots that live in a complex workspace \( W \subset \mathbb{R}^d \). We assume that there are \( W \) disjoint regions of interest in \( W \). The \( j \)-th region is denoted by \( r_j \) and it can be of any arbitrary shape. Given the robot dynamics, robot mobility in the workspace \( W \) can be represented by a weighted Transition System (wTS); see also Figure 1. The wTS for robot \( i \) is defined as follows:

**Definition 1.** wTS. A weighted Transition System (wTS) for robot \( i \), denoted by \( \text{wTS}_i \), is a tuple \( \text{wTS}_i = (\mathcal{Q}_i, \mathcal{Q}_i^0, r_i, w_i, \mathcal{A}_i, L_i) \) where: (a) \( \mathcal{Q}_i = \bigcup_{j=1}^{\mathcal{W}} \{q_i^j\} \) is the set of states, where a state \( q_i^j \) 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) \( r_i \in \{1, \ldots, \mathcal{N}\} \) is the transition relation for robot \( i \). Given the robot dynamics, if there is a control input \( u_i \) that can drive robot \( i \) from location \( r_j \) to \( r_k \), then there is a transition from state \( q_i^j \) to \( q_i^k \) denoted by \( (q_i^j, q_i^k) \in r_i \); (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 \( q_i^j \) to state \( q_i^k \); (e) \( \mathcal{A}_i = \bigcup_{j=1}^{\mathcal{W}} \{\pi_i^j\} \) is the set of atomic propositions, where \( \pi_i^j \) is true if robot \( i \) is inside region \( r_j \) and false otherwise; and (f) \( L_i : \mathcal{Q}_i \rightarrow \mathcal{A}_i \) is an observation/output function defined as \( L_i(q_i^k) = w_i^{\pi_i^k} \) for all \( q_i^k \in \mathcal{Q}_i \).

Given the definition of the wTS, we can define the synchronous Product Transition System (PTS) as follows (Ulusoy et al. 2013):

**Definition 2.** PTS. Given \( N \) transition systems \( \text{wTS}_i = (\mathcal{Q}_i, \mathcal{Q}_i^0, r_i, w_i, \mathcal{A}_i, L_i) \), the product transition system \( \text{PTS} = \text{wTS}_1 \otimes \text{wTS}_2 \otimes \cdots \otimes \text{wTS}_N \) is a tuple \( \text{PTS} = (\mathcal{Q}_{\text{PTS}}, \mathcal{Q}_{\text{PTS}}^0, \rightarrow_{\text{PTS}}, w_{\text{PTS}}, \mathcal{A}_i, L_{\text{PTS}}) \) 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) \( \mathcal{Q}_{\text{PTS}}^0 \equiv \{q_1^0, q_2^0, \ldots, q_N^0\} \in \mathcal{Q}_{\text{PTS}} \) is the initial state, (c) \( \rightarrow_{\text{PTS}} \subseteq \mathcal{Q}_{\text{PTS}} \times \mathcal{Q}_{\text{PTS}} \) is the transition relation defined by the rule \( \mathcal{A}_i(q_i^{\tau(k)} \rightarrow_{\text{PTS}} q_i^{\tau(k+1)}) \) 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 \( q_{\text{PTS}}^k \) is defined accordingly. In words, this transition rule says that there exists a transition from \( q_{\text{PTS}}^k \) to \( q_{\text{PTS}}^{k+1} \) if there exists a transition from \( q_i^{\tau(k)} \) to \( q_i^{\tau(k+1)} \) for all \( i \in \{1, \ldots, N\} \); (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 \( w_{\text{PTS}}(q_{\text{PTS}}, q_{\text{PTS}}') = \sum_{i=1}^{N} w_i(\Pi_{\text{wTS}}(\mathcal{Q}_{\text{PTS}}), q_i^k) \), where \( \mathcal{Q}_{\text{PTS}}, q_{\text{PTS}} \in \mathcal{Q}_{\text{PTS}} \), and \( \Pi_{\text{wTS}} \) stands for the projection of state \( q_{\text{PTS}} \) onto the state space of wTS. The state \( \mathcal{Q}_{\text{PTS}} \subseteq \mathcal{Q}_i \) is obtained by removing all states in \( q_{\text{PTS}} \) that do not belong to \( \mathcal{Q}_i \); (e) \( \mathcal{A}_i = \bigcup_{j=1}^{\mathcal{W}} \mathcal{A}_j \) is the set of atomic propositions; and, (f) \( L_{\text{PTS}} = \bigcup_{i=1}^{N} L_i : \mathcal{Q}_{\text{PTS}} \rightarrow \mathcal{A}_i \) 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 \( \tau \) of a PTS is an infinite sequence of states, \( \tau = \tau(1) \tau(2) \tau(3) \cdots \) such that \( \tau(1) = q_{\text{PTS}}^0 \), \( \tau(k) \in \mathcal{Q}_{\text{PTS}} \), and \( \tau(k+1) \in \tau(\tau(k), \tau(k+1)) \in \mathcal{Q}_{\text{PTS}} \forall k \in \mathbb{N}_+ \), where \( k \) is an index that points to the \( k \)-th entry of \( \tau \) denoted by \( \tau(k) \). The trace of an infinite path \( \tau = \tau(1) \tau(2) \tau(3) \cdots \) of a PTS, denoted by \( \text{trace}(\tau) \in (2^{\mathcal{AP}})^\omega \), where \( \omega \) denotes infinite repetition, is an infinite word that is determined by the sequence of atomic propositions that are true in the states along \( \tau \), i.e., \( \text{trace}(\tau) = L(\tau(1))L(\tau(2)) \cdots \) A finite path of a PTS can be defined accordingly. The only difference with the infinite path is that a finite path is defined as a finite sequence of states of a PTS. Given the definition of the weights \( w_{\text{PTS}} \) in Definition 2, the cost of a finite path \( \tau \), denoted by \( \hat{J}(\tau) \geq 0 \), can be defined as

\[
\hat{J}(\tau) = \sum_{k=1}^{\tau-1} w_{\text{PTS}}(\tau(k), \tau(k+1)),
\]

*For simplicity of notations we consider disjoint regions \( r_j \). However, overlapping regions can also be considered by introducing additional states to wTS, defined in Definition 1 that capture the presence of robot \( i \) in more than one region.*
where, $|\tau|$ stands for the number of states in $\tau$. In words, the cost (1) captures the total cost incurred by all robots during the execution of the finite path $\tau$.

We assume that the robots have to accomplish a complex collaborative task captured by a global LTL statement $\phi$ defined over the set of atomic propositions $AP = \bigcup_{i=1}^{N_{AP}} AP_i$. Due to space limitations, we abstain from formally defining the semantics and syntax of LTL. A detailed overview can be found in (Baier and Katoen 2008). Given an LTL formula $\phi$, we define the language $Words(\phi) = \{\sigma \in (2^{AP})^\omega | \sigma \models \phi\}$, where $\models \subseteq (2^{AP})^\omega \times \phi$ is the satisfaction relation, as the set of infinite words $\sigma \in (2^{AP})^\omega$ that satisfy the LTL formula $\phi$. Any LTL formula $\phi$ can be translated into a Nondeterministic Büchi Automaton (NBA) over $(2^{AP})^\omega$ denoted by $B$ defined as follows (Vardi and Wolper 1986).

**Definition 3.** NBA. A Nondeterministic Büchi Automaton (NBA) $B$ over $2^{AP}$ is defined as a tuple $B = (Q_B, \Phi_B, \Sigma_B, \rightarrow_B, Q_B^F)$, where $Q_B$ is the set of states, $\Phi_B \subseteq Q_B$ is a set of initial states, $\Sigma_B = 2^{AP}$ is an alphabet, $\rightarrow_B \subseteq Q_B \times \Sigma_B \times Q_B$ is the transition relation, and $Q_B^F \subseteq Q_B$ is a set of accepting final states.

Given the PTS and the NBA $B$ that corresponds to the LTL $\phi$, we can now define the Product Büchi Automaton (PBA) $P = PTS \otimes B$, as follows (Baier and Katoen 2008):

**Definition 4.** PBA. Given the product transition system $PTS = (PTS, P_B, \rightarrow_{PTS}, w_{PTS}, AP, L_{PTS})$ and the NBA $B = (Q_B, \Phi_B, \Sigma_B, \rightarrow_B, Q_B^F)$, we can define the Product Büchi Automaton $P = PTS \otimes B$ as a tuple $P = (P_Q, \Phi_Q, \Sigma_Q, \rightarrow_P, \Phi_P, Q_P^F)$, where (a) $P_Q = PTS \times Q_B$ is the set of states; (b) $\Phi_Q = \Phi_{PTS} \times \Phi_B$ is a set of initial states; (c) $\rightarrow_P \subseteq P_Q \times 2^{AP} \times P_Q$ is the transition relation defined by the rule:

$$\begin{align*}
\quad & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \frac{\hat{\omega}(q_P)}{\omega} = \hat{\omega}(q_P) \mid q_P \in \mathcal{L}_{PTS}(\hat{\omega}), \quad \hat{\omega}(q_P) = \omega(q_P) = \max\left\{\frac{\hat{\omega}(q_P)}{\omega(q_P)} | q_P \in Q_P\right\}.
\end{align*}$$

Transition from state $\omega(q_P)$ in $PTS$ to $\omega(q_P)$ in $PTS$ is denoted by $(q_P, q_P) \rightarrow_P (q_P, q_P)$, (d) $\omega(q_P, q_B) = \omega(q_P, q_B)$, (e) $\omega(q_P, q_B) = \omega(q_P, q_B)$, (f) $\omega(q_P, q_B) = \omega(q_P, q_B)$.

Given $\phi$ and the PBA an infinite path $\tau$ of a PBA satisfies $\phi$ if and only if $\text{trace}(\tau) \in \text{Words}(\phi)$, which is equivalently denoted by $\tau \models \phi$. Specifically, if there is a path satisfying $\phi$, then there exists a path $\tau \models \phi$ that can be written in a finite representation, called prefix-suffix structure, i.e., $\tau = (\tau^{pre}|\tau^{suf})$, where the prefix part $\tau^{pre}$ is executed only once followed by the indefinite execution of the suffix part $\tau^{suf}$. The prefix part $\tau^{pre}$ is the projection of a finite path $\tau^{pre}$ that lives in $PTS$ onto $PTS$, the path $\tau^{pre}$ starts from an initial state $q_0^{pre} \in Q_P$, and ends at a final state $q_F^{pre} \in Q_P$, i.e., it has the following structure $\tau^{pre} = (q_0^{pre}, q_B^0) \rightarrow (q_S^{pre}, q_B^S) \rightarrow (q_{F}^{pre}, q_B^F)$, with $(q_0^{pre}, q_B^0) \in Q_P^F$.

The suffix part $\tau^{suf}$ is the projection of a finite path $\tau^{suf}$ that lives in $PTS$ onto $PTS$, the path $\tau^{suf}$ is a cycle around the final state $q_F^{pre}$, and it has the following structure $\tau^{suf} = (q_F^{pre}, q_B^F) \rightarrow (q_F^{pre}, q_B^F) \rightarrow (q_F^{pre}, q_B^F) \rightarrow \ldots$, where $(q_F^{pre}, q_B^F) \in Q_P^F$. Then our goal is to compute a plan $\tau = \tau^{pre}|\tau^{suf}|\tau^{pre}$. Finally, using the constructed prefix and suffix parts, the optimal plan $\tau = \tau^{pre}|\tau^{suf}|\tau^{pre}$ is synthesized in lines 21-22.
Algorithm 1: STyLuS*: large-Scale optimal Temporal Logic Synthesis

**Input:** LTL formula $\phi$, $\{wTS_i\}_{i=1}^N \subseteq \mathbb{Q}_{PTS}$, maximum numbers of iterations $n_{\text{pre}}^{\text{max}}, n_{\text{inf}}^{\text{max}}$

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

1. Convert $\phi$ to an NBA $B = (Q_B, \mathcal{E}_B, \rightarrow_B, Q_B^F)$
2. $\{\mathcal{F}_{\text{pre}}^{\text{feas}}, \mathcal{F}_{\text{inf}}^{\text{pre}}\}_{q_B, q_B^F \in Q_B}$ = FeasibleWords($\{\mathcal{F}_{\text{pre}}^{\text{feas}}, \mathcal{F}_{\text{inf}}^{\text{pre}}\}_{q_B, q_B^F \in Q_B}$)
3. Construct graph $G_B$ and $d(q_B, q_B^F)$
4. Define goal set: $\mathcal{X}_{\text{goal}}$
5. for $b_0 = 1 : |Q_B|$
   6. Initial NBA state: $q_B^0 = Q_B^0(b_0)$
   7. Root of the tree: $q_B^P = (q_B^0, q_B^0)$
   8. $[\mathcal{G}, \mathcal{P}] = \text{ConstructTree}(\mathcal{X}_{\text{goal}}, \{wTS_i\}_{i=1}^N, B, q_B^0, n_{\text{max}}^{\text{max}})$
   9. for $a = 1 : |\mathcal{P}|$
      10. $q_{\text{pre}}^{\text{a}} = \text{FindPath}(\mathcal{G}, q_B^P, \mathcal{P}(a))$
      11. $a^* = \text{argmin}_{a} (\mathcal{R}_{\text{goal}}(q_{\text{inf}}^{\text{a}}) + \mathcal{R}_{\text{pre}}^{\text{a}})$
   12. Define goal set: $\mathcal{X}_{\text{goal}}(q_{\text{pre}}^{\text{a}})$
   13. if ($q_B^P \in \mathcal{X}_{\text{goal}}(q_{\text{pre}}^{\text{a}})$) \& ($q_B^F = (q_B^P, q_B^P, 0)$) then
      14. $\mathcal{G} = (q_B^P, q_B^P, 0)$
      15. $\mathcal{S} = \{q_B^F\}$
   16. else
      17. $[\mathcal{G}, \mathcal{S}] = \text{ConstructTree}(\mathcal{X}_{\text{goal}}(q_{\text{pre}}^{\text{a}}), \{wTS_i\}_{i=1}^N, B, q_B^0, n_{\text{max}}^{\text{max}})$
      18. Compute $q_{\text{inf}}^{\text{a}}(\text{see Kantaros and Zavlanos 2018a})$
      19. $a_{\text{opt}} = \text{argmin}_{a} (\mathcal{J}(q_{\text{pre}}^{\text{a}}) + \mathcal{J}(q_{\text{inf}}^{\text{a}}))$
      20. $a^* = \text{argmin}_{a} (\mathcal{J}(q_{\text{pre}}^{\text{a}}) + \mathcal{J}(q_{\text{inf}}^{\text{a}}))$
      21. Optimal Plan: $\tau = q_{\text{pre}}^{\text{a}} [a^*] w$

3.1 Feasible Symbols

In this section, given the NBA $B$ that corresponds to the assigned LTL formula $\phi$, we define a function $d : Q_B \times Q_B \rightarrow \mathbb{N}$ that returns the minimum number of feasible NBA transitions that are required to reach a state $q_B^0 \in Q_B$ starting from a state $q_B \in Q_B$ [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; see also Figure 2.

**Definition 5.** Feasible NBA transitions. A transition $(q_B, \sigma, q_B^F) \in \rightarrow_B$ is feasible if the finite symbol $\sigma \in \Sigma = 2^{AP}$ is a feasible symbol, i.e., if $\sigma$ can be generated by the PTS defined in Definition 2.

To characterize the symbols $\sigma \in \Sigma$ that are feasible, we first need to define the symbols $\sigma_i \in \Sigma_i = 2^{AP_i}$ that are feasible, i.e., the symbols that can be generated by $wTS_i$ defined in Definition 1.

**Definition 6.** Feasible symbols $\sigma_i \in \Sigma_i$. A symbol $\sigma_i \in \Sigma_i$ is feasible if and only if $\bar{\sigma}_i \not\in b_i^{\text{inf}}$, where $b_i^{\text{inf}}$ is a Boolean formula defined as

$$b_i^{\text{inf}} = \bigvee_{r_j} (\bigvee_{r_j} (r_j^\prime \wedge \pi_i^\prime))$$

(3)

Note that the Boolean formula $b_i^{\text{inf}}$ is satisfied by any finite symbol $\sigma_i \in \Sigma_i$ that requires robot $i$ to be present in two or more disjoint regions, simultaneously. For instance, the symbol $\bar{\sigma}_i = \pi_i^\prime \pi_i^\prime$ satisfies $b_i^{\text{inf}}$. Next, we define the feasible symbols $\bar{\sigma} \in \Sigma$.

**Definition 7.** Feasible symbols $\bar{\sigma} \in \Sigma$. A symbol $\bar{\sigma} \in \Sigma$ is feasible if and only if $\bar{\sigma} \not\in b_i^{\text{inf}}$, for all robots $i$, where $\bar{\sigma}_i = \Pi [\Sigma], \bar{\sigma}$, $b_i^{\text{inf}}$ is defined in (3), and $\Pi [\Sigma]$ stands for the projection of the symbol $\sigma$ onto $\Sigma_i = 2^{AP_i}$

To define the proposed function $d : Q_B \times Q_B \rightarrow \mathbb{N}$, we first construct sets $\Sigma_{\text{feas}}(q_B, q_B^F) \subseteq \Sigma$ that collect feasible finite symbols $\bar{\sigma}$ that enable a transition from a state $q_B \in Q_B$ to $q_B^0 \in Q_B$ according to $\rightarrow_B$ [line 2, Alg. 1]. To construct these sets, sets $\Sigma_{\text{feas}}(q_B, q_B^F) \subseteq \Sigma$ that collect all finite (feasible or infeasible) symbols $\bar{\sigma} \in \Sigma$ that enable a transition from $q_B \in Q_B$ to $q_B^0 \in Q_B$, for all $q_B, q_B^F \in Q_B$, are required. Then, the sets $\Sigma_{\text{feas}}(q_B, q_B^F) \subseteq \Sigma_{\text{feas}}(q_B, q_B^F)$ can be constructed by removing from $\Sigma_{\text{feas}}(q_B, q_B^F)$ all symbols $\bar{\sigma}$ that are not feasible, for all $q_B, q_B^F \in Q_B$.

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 $q_B^F$ denoted by $(q_B, q_B^F)$, where $(q_B, q_B^F)$ exists if $\Sigma_{\text{feas}}(q_B, q_B^F) \not\emptyset$ [line 3, Alg. 1]. Assigning weights equal to one to all edges in the set $\mathcal{E}_B$, we define the function $d : Q_B \times Q_B \rightarrow \mathbb{N}$ as

$$d(q_B, q_B^F) = \begin{cases} |\mathcal{S}_{\text{feas}}(q_B, q_B^F)|, & \text{if } \mathcal{S}_{\text{feas}}(q_B, q_B^F) \text{ exists}, \\ \infty, & \text{otherwise,} \end{cases}$$

(4)

1Note that if we consider regions $r_j$ that are not necessarily disjoint, then the Boolean formula $b_i^{\text{inf}}$ is defined as $b_i^{\text{inf}} = \bigvee_{r_j}(\bigvee_{\sigma_i \in \Sigma} (\sigma_i^r) \wedge \pi_i^r)$. Note also that definition of infeasible symbols depends on the problem at hand, i.e., the definition of atomic propositions included in the sets $AP_i$.

2For instance, $\Pi [\Sigma_i] (\pi_i^{\text{inf}}) = \pi_i^{\prime}$.

3Note that the sets $\Sigma_{\text{feas}}(q_B, q_B^F)$ can be computed during the translation of an LTL formula $\phi$ to an NBA; see e.g., the software package (Ulusoy et al. 2011) that relies on (Gastin and Oddoux 2001) for the construction of the NBA.

Figure 2. Graphical depiction of infeasible transitions of the NBA that corresponds to the LTL formula $\phi = \Box (\pi_i^\prime) \wedge \Box (\pi_i^\prime)$. The states ‘0’ and ‘2’ correspond to the initial and final state, respectively. The transition from the initial to the final state and the self-loop around the final state are infeasible transitions, since they are activated only by the symbol $\bar{\sigma} = \pi_i^r \pi_i^r$, which can never be generated by $wTS_i$, assuming disjoint regions $r_j$ and $r_e$. 

---

Prepared using sagej.cls
where \(S_P(q, q')\) denotes the shortest path in \(G_B\) from \(q\in V_B\) to \(q'\in V_B\) and \(|S_P(q, q')|\) stands for its cost, i.e., the number of transitions/edges in \(S_P(q, q')\).

### 3.2 Construction of Optimal Prefix Parts

In this section we describe the construction of the tree \(G_{T^*} = \{V_T, E_T, \text{Cost}\}\) that will be used for the synthesis of the prefix part [lines 4-10, Alg. 1]. Since the prefix part connects an initial state \(q_p^0 = (q_p^{\text{PTS}}, q_B^0)\) to an accepting state \(q = (q_{\text{PTS}}, q_B)\in Q_B\), with \(q_B\in Q_B\), we can define the goal region for the tree \(G_T\), as [line 4, Alg. 1]:

\[
\mathcal{X}_{\text{goal}} = \{q = (q_{\text{PTS}}, q_B) \in Q_P \mid q_B \in Q_B^F\}.
\]

(5)

The root \(q_p^0\) of the tree is an initial state \(q_p^0 = (q_p^{\text{PTS}}, q_B^0)\) of the PBA and the following process is repeated for each initial state \(q_p^0\) of \(Q_P\), in parallel [line 5-7, Alg. 1]. The construction of the tree is described in Algorithm 2 [line 8, Alg. 1]. In line 6 of Algorithm 1, \(Q_B^0(b_0)\) stands for the \(b_0\)-th initial state assuming an arbitrary enumeration of the elements of the set \(Q_B^0\). The set \(V_T\) initially contains only the root \(q_p^0\), i.e., an initial state of the PBA [line 1, Alg. 2] and, therefore, the set of edges is initialized as \(E_T = \emptyset\) [line 2, Alg. 2]. By convention, we assume that the cost of \(q_p\) is zero [line 3, Alg. 2]. Given the root \(q_p^0\), we select a feasible final state \(q_B^F \in Q_B\) such that \((i)\ d(q_p^0, q_B^F) \neq \infty\) and \((ii)\ d(q_p^0, q_B^F) \neq \infty\). Among all final states that satisfy both \((i)\) and \((ii)\), we select one randomly denoted by \(q_B^{F, \text{feas}}\) [line 5, 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 \(q_p^0\). In this case, the construction of the tree stops without having detected any final states around which a loop exists [lines 6-7, 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 \(D_{\text{min}}\) that collects the nodes \(q = (q_{\text{PTS}}, q_B)\in V_T\) that have the minimum distance \(d(q_B, q_B^{F, \text{feas}})\) among all nodes in \(V_T\), i.e.,

\[
D_{\text{min}} = \{q = (q_{\text{PTS}}, q_B) \in V_T \mid d(q_B, q_B^{F, \text{feas}}) = d_{\text{min}}\},
\]

(6)

where \(d_{\text{min}} = \min \cup\{d(q_B, q_B^{F, \text{feas}})\}\), \(\forall q_B \in E_T\) and \(\Pi_B V_T \subseteq Q_B\) stands for the projection of all states \(q_B \in V_T\) onto \(Q_B\). The set \(D_{\text{min}}\) initially collects only the root [line 8, Alg. 2].

#### 3.2.1 Sampling a state \(q_{\text{rand}}^{\text{new}} \in Q_P\)

The first step in the construction of the graph \(G_T\) is to sample a state \(q_{\text{rand}}^{\text{new}}\) from the state-space of the PBA. This is achieved by a sampling function \(\text{Sample}\); see Algorithm 3. Specifically, we first create a state \(q_{\text{rand}}^{\text{PTS}} = \Pi_B q_{\text{rand}}\), where \(q_{\text{rand}}\) is sampled from a given discrete distribution \(f_{\text{rand}}(q_P | V_T)\) such that \(0 \leq f_{\text{rand}}(q_P | V_T) \leq 1\) and \(\Pi_B V_T \subseteq Q_B\) stands for the projection of \(q_{\text{rand}}^{\text{PTS}}\) onto the state-space of the PTS [line 1, Alg. 3]. The probability density function \(f_{\text{rand}}(q_P | V_T)\) defines the probability of selecting the state \(q_P\) in \(V_T\) as the state \(q_{\text{rand}}^{\text{PTS}}\) at iteration \(n\) of Algorithm 2 given the set \(V_T\). The distribution \(f_{\text{rand}}\) is defined as follows:

\[
f_{\text{rand}}(q_P | V_T, D_{\text{min}}) = \begin{cases} 
\frac{1}{|V_T|} & \text{if } q_p \in D_{\text{min}} \\
\frac{1}{|V_T^*|} & \text{otherwise}
\end{cases}
\]

(7)

where \(p_{\text{rand}} \in (0, 1)\) stands for the probability of selecting any node \(q_P \in D_{\text{min}}\) to be sampled. Note that \(p_{\text{rand}}\) can change with iterations \(n\) but it should always satisfy \(p_{\text{rand}} \in (0, 1)\) so that states \(q = (q_{\text{PTS}}, q_B) \in D_{\text{min}} \subseteq V_T\) are selected more often as \(q_{\text{rand}}^{\text{PTS}}\) [line 1, Alg. 3].

**Remark 8.** Density function \(f_{\text{rand}}\). Observe that the discrete density function \(f_{\text{rand}}\) in (7) is defined in a uniform-like way, since all states in the set \(D_{\text{min}}\) are selected with probability \(\frac{1}{|V_T^*|}\) while the states in \(V_T\setminus D_{\text{min}}\) are selected with probability \(\frac{1-p_{\text{rand}}}{|V_T\setminus D_{\text{min}}|}\), where \(p_{\text{rand}} \in (0, 1)\). However, alternative definitions for \(f_{\text{rand}}\) are possible as long as \(f_{\text{rand}}(a)\) satisfies Assumptions 4.1(ii)-(iii) made in (Kantaros and Zavlanos 2018a); and (b) is biased to generate states that belong to \(D_{\text{min}}\) more often than states that belong to \(V_T\setminus D_{\text{min}}\). Assumptions 4.1(i) and 4.1(ii) made in (Kantaros and Zavlanos 2018a) are required to guarantee that the proposed algorithm is probabilistically complete while Assumptions 4.1(i)-(iii) are required for the asymptotic optimality of the proposed method; see Section 4. The fact that the density function (7) satisfies Assumption 4.1 in (Kantaros and Zavlanos 2018a) is shown in Section 4. Finally, bias in the sampling increases scalability of the proposed control synthesis method; see Section 5. Note that, as it will be discussed later, \(f_{\text{rand}}\) can also change with iterations \(n\) of Algorithm 2.

Given \(q_{\text{rand}}^{\text{PTS}}\), in our previous work (Kantaros and Zavlanos 2018a), we sample a state \(q_{\text{rand}}^{\text{new}}\) from a discrete distribution \(f_{\text{rand}}(q_P | q_{\text{rand}}^{\text{PTS}})\) so that \(q_{\text{rand}}^{\text{new}}\) is reachable from \(q_{\text{rand}}^{\text{PTS}}\). Here, we sample a state \(q_{\text{rand}}^{\text{new}}\) so that it is both reachable from \(q_{\text{rand}}^{\text{PTS}}\) and also it can lead to a final state \(q_B^{F, \text{feas}} = (q_{\text{PTS}}, q_B^{F, \text{feas}}) \in Q_B\) by following the shortest path in \(G_B\) that connects \(q_p^0\) to \(q_B^{F, \text{feas}}\).
The proposed biased sampling process is illustrated in Figure 3.

Selection of $q_{\text{rand}}^{\text{min}} \in Q_B$. First, given $q_{\text{rand}}^{\text{rand}} = (q_{\text{rand}}^{\text{PTS}}, q_{\text{rand}}^{\text{rand}})$, we first construct the reachable set $\mathcal{R}_B(q_{\text{rand}}^{\text{rand}})$ that collects all states $q_B \in Q_B$ that can be reached in one hop in $B$ from $q_{\text{rand}}^{\text{rand}}$ given the observation $L_{\text{PTS}}(q_{\text{rand}}^{\text{PTS}})$ (see also Figure 3) [line 2, Alg. 3], i.e.,

\[ \mathcal{R}_B(q_{\text{rand}}^{\text{rand}}) = \{q_B \in Q_B | (q_{\text{rand}}^{\text{rand}}, L_{\text{PTS}}(q_{\text{rand}}^{\text{PTS}}), q_B) \in B\}. \]  

(8)

Given $\mathcal{R}_B(q_{\text{rand}}^{\text{rand}})$ we construct the set $\mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}})$ that collects the states $q_B \in \mathcal{R}_B(q_{\text{rand}}^{\text{rand}})$ that have the minimum distance from $q_{B,R}^{F,\text{feas}}$ among all other nodes $q_B \in \mathcal{R}_B(q_{\text{rand}}^{\text{rand}})$, i.e.,

\[ \mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}}) = \{q_B \in \mathcal{R}_B(q_{\text{rand}}^{\text{rand}}) | \]  

\[ d(q_B, q_{B,R}^{F,\text{feas}}) = \min_{q_B \in \mathcal{R}_B(q_{\text{rand}}^{\text{rand}})} d(q_B, q_{B,R}^{F,\text{feas}}) \} \subseteq \mathcal{R}_B(q_{\text{rand}}^{\text{rand}}). \]  

(9)

In what follows, we denote by $q_{\text{rand, min}}$ the states that belong to $\mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}})$. For every state $q_{\text{rand, min}} \in \mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}})$, we construct the set $\mathcal{R}_{\text{dec}}(q_{\text{rand, min}})$ that collects all states $q_B \in Q_B$, for which (i) there exists a feasible symbol $\sigma \in \Sigma_{\text{feas}}^{q_{\text{rand, min}}} \rightarrow q_B$, and (ii) $q_B$ is closer to $q_{B,R}^{F,\text{feas}}$ than $q_{\text{rand, min}}$ is, i.e.,

\[ \mathcal{R}_{\text{dec}}(q_{\text{rand, min}}) = \{q_B \in Q_B | (q_{\text{rand, min}}, \mathcal{R}_{\text{dec}}^{q_{\text{rand, min}}}, q_B) \neq 0 \} \]  

\[ \wedge d(q_B, q_{B,R}^{F,\text{feas}}) = d(q_{\text{rand, min}}, q_{B,R}^{F,\text{feas}}) - 1 \} \subseteq Q_B. \]  

(10)

We collect all states $q_{\text{rand, min}} \in \mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}})$ that satisfy $\mathcal{R}_{\text{dec}}(q_{\text{rand, min}}) \neq 0$ in the set $\mathcal{M}(q_{\text{rand}}^{\text{rand}})$ [line 4, Alg. 3] defined as

\[ \mathcal{M}(q_{\text{rand}}^{\text{rand}}) = \{q_{\text{rand, min}} | \mathcal{R}_{\text{dec}}(q_{\text{rand, min}}) \neq 0 \} \subseteq \mathcal{R}_{\text{min}}(q_{\text{rand}}^{\text{rand}}). \]  

Next, given the set $\mathcal{M}(q_{\text{rand}}^{\text{rand}})$, we sample a state $q_{\text{rand}}^{\text{rand}} \in \mathcal{M}(q_{\text{rand}}^{\text{rand}})$ from a discrete distribution $f_{\text{min}}(q_B | \mathcal{M}(q_{\text{rand}}^{\text{rand}}))$:

\[ \mathcal{M}(q_{\text{rand}}^{\text{rand}}) \rightarrow [0,1] \]  

(see also Figure 3) [line 6, Alg. 3] defined as

\[ f_{\text{min}}(q_B | \mathcal{M}(q_{\text{rand}}^{\text{rand}})) = \frac{1}{|\mathcal{M}(q_{\text{rand}}^{\text{rand}})|}. \]

(see also Figure 3) Note that if $\mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}}) = \emptyset$, then no sample will be taken at this iteration [line 12, Alg. 3].

Selection of $q_{\text{dec}}^{\text{rand}} \in \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})$. Given the state $q_{\text{rand}}^{\text{rand}}$ and its respective set $\mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})$ from a given discrete distribution $f_{\text{dec}}(q_B | \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})) : \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}}) \rightarrow [0,1]$, we sample a state $q_{\text{dec}}^{\text{rand}} \in \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})$ [line 7, Alg. 3] defined as

\[ f_{\text{dec}}(q_B | \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})) = \frac{1}{|\mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})|} \]

for all $q_B \in \mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}})$; see also Figure 3. Note that if $\mathcal{R}_{\text{dec}}(q_{\text{rand}}^{\text{rand}}) = \emptyset$, then no sample $q_{\text{new}}^{\text{PTS}}$ is generated [line 10, Alg. 3].

Given $q_{\text{rand}}^{\text{rand}}$ and $q_{\text{dec}}^{\text{rand}}$, we select a symbol $\sigma$ from $\Sigma_{\text{pre}}^{q_{\text{dec}}^{\text{rand}}} \neq \emptyset$ [line 8, Alg. 3]. Given the symbol $\sigma$, 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 symbol $\sigma$ can be generated. For instance, the set $\mathcal{L}$ corresponding to the symbol $\sigma = \pi^r_1 x^r_2$ is constructed so that $L(i) = r_1$, $L(z) = r_2$ and $L(h) = \emptyset$, for all robots $h \neq i, z$. Then, we sample a state $q_{\text{new}}^{\text{new}}$, for all robots $i$, from a discrete distribution $f_{\text{new}}(q_{\text{new}}(|q_i^{\text{rand}}) : \mathcal{L}_{\text{TS}}(q_i^{\text{rand}}) \rightarrow [0,1]$ [line 14, Alg. 3] defined as

\[ f_{\text{new}}(q_{\text{new}}(q_i^{\text{rand}})) = \begin{cases} \frac{1}{|\mathcal{L}_{\text{TS}}(q_i^{\text{rand}})|}, & \text{if } L(i) = \emptyset, \\ \frac{1}{|\mathcal{L}_{\text{TS}}(q_i^{\text{rand}})| - 1}, & \text{if } (L(i) \neq \emptyset) \wedge (q_i \in \mathcal{R}_{\text{TS}}(q_i^{\text{rand}})) \wedge (q_i \in \mathcal{R}_{\text{TS}}(q_i^{\text{rand}})) \wedge (q_i \in \mathcal{R}_{\text{TS}}(q_i^{\text{rand}})) \wedge (q_i \in \mathcal{R}_{\text{TS}}(q_i^{\text{rand}})), \end{cases} \]

\[ \mathcal{R}_{\text{TS}}(q_i^{\text{rand}}) = \{q_i \in Q_i | q_i^{\text{rand}} \rightarrow q_i, q_i \}. \]

where $\mathcal{R}_{\text{TS}}(q_i^{\text{rand}})$ collects all states in $Q_i$ that can be reached in one hop from $q_i^{\text{rand}} = \Pi_{\text{TS}}^{q_i^{\text{rand}}}$, i.e.,

\[ \Pi_{\text{TS}}(q_i^{\text{rand}}) = \{q_i \in Q_i | q_i^{\text{rand}} \rightarrow q_i, q_i \}. \]

To speed up the detection of final states, we always select the same symbol $\sigma \in \Sigma_{\text{pre}}^{q_{\text{dec}}^{\text{rand}}}$ for a given pair of states $q_{\text{rand}}^{\text{rand}}$ and $q_{\text{dec}}^{\text{rand}}$. Also, by construction of $q_{\text{dec}}^{\text{rand}}$, it holds that $\Sigma_{\text{pre}}^{q_{\text{dec}}^{\text{rand}}} \neq 0$.\footnote{To speed up the detection of final states, we always select the same symbol $\sigma \in \Sigma_{\text{pre}}^{q_{\text{dec}}^{\text{rand}}}$ for a given pair of states $q_{\text{rand}}^{\text{rand}}$ and $q_{\text{dec}}^{\text{rand}}$. Also, by construction of $q_{\text{dec}}^{\text{rand}}$, it holds that $\Sigma_{\text{pre}}^{q_{\text{dec}}^{\text{rand}}} \neq 0$.}
In order to build incrementally a graph whose set of nodes approximates the state-space $Q_P$ we need to append to $q_{new}^\text{PTS}$ a state from the state-space $Q_B$ of the NBA. Let $q_{new}^\text{PTS} = Q_B(b)$ [line 15, Alg. 2] be the candidate Büchi state that will be attached to $q_{new}^\text{PTS}$, where $Q_B(b)$ stands for the $b$-th state in the set $Q_B$ assuming an arbitrary enumeration of the elements of the set $Q_B$. The following procedure is repeated for all $q_{new}^\text{PTS} = Q_B(b)$ with $b \in \{1, \ldots, |Q_B|\}$. First, we construct the state $q_{new}^\text{new} = (q_{new}^\text{PTS}, Q_B(b))$ [line 16, Alg. 2] and then we check if this state can be added to the tree $T$ [lines 17-18, Alg. 2]. If the state $q_{new}^\text{new}$ does not already belong to the tree from a previous iteration of Algorithm 2, i.e., if $q_{new}^\text{new} \notin V_T$ [line 13, Alg. 2], we check which node in $V_T$ (if there is any) can be the parent of $q_{new}^\text{new}$ in the tree $T$. If there exist candidate parents for $q_{new}^\text{new}$ then the tree is extended towards $q_{new}^\text{new}$ and the set $D_{\text{min}}$ is updated [line 19, Alg. 2]. If $q_{new}^\text{new} \in V_T$, then the rewiring step follows [lines 20-21, Alg. 2] that aims to reduce the cost of nodes $q_{new}^\text{new}$ in $V_T$. A detailed description of the ‘extend’ and ‘rewire’ steps can be found in (Kantaros and Zavlanos 2018a).

Remark 9. Density function $f_{\text{pre}}^{\text{new},i}$. Observe that the discrete density function $f_{\text{pre}}^{\text{new},i}$ in (11) is defined in a uniform-like way, similar to $f_{\text{rand}}$ in (7). However, alternative definitions for $f_{\text{pre}}^{\text{new},i}$ are possible as long as $f_{\text{pre}}^{\text{new},i}(q_{\text{rand}}^{\text{new}},i)$ satisfies Assumptions 4.2(iii) (in Kantaros and Zavlanos 2018a); and (b) is biased to generate the state $SP_{Q_{\text{goal}}^{\text{new}},i}^{\text{goal}}(2)$ (if defined) more often than the other states that belong to $R_{\text{WTS}}(q_{\text{rand}}^{\text{new}})$. Assumptions 4.2(i) and 4.2(ii) in (Kantaros and Zavlanos 2018a) are required to guarantee that the proposed algorithm is probabilistically complete while Assumptions 4.2(iii) are required to prove the asymptotic optimality of the proposed method; see Section 4. Finally, bias in the sampling increases scalability of the proposed control synthesis method; see Section 5. The fact that the density function (11) satisfies Assumption 4.2 of (Kantaros and Zavlanos 2018a) is shown in Section 4. Note that, as it will be discussed later, $f_{\text{pre}}^{\text{new}}$ can also change with iterations $n$ of Algorithm 2. The same remark also holds for the density functions $f_{\text{new}}^{\text{new},i}$ that will be introduced in Section 3.3 for the construction of the suffix parts.

3.2 Construction of Paths The construction of the tree $G_T$ ends after $n_{\text{max}}$ iterations, where $n_{\text{max}}$ is user specified [line 9, Alg. 2]. Then, we construct the set $P = V_T \cap X_{\text{goal}}$ [line 22, Alg. 2] that collects all the states $q_P \in V_T$ that belong to the goal region $X_{\text{goal}}$. Given the tree $G_T$ and the set $P$ [line 8, Alg. 1] that collects all states $q_P \in X_{\text{goal}} \cap V_T$, we can compute the prefix plans [lines 9-10, Alg. 1]. In particular, the path that connects the $\alpha$-th state in the set $P$, denoted by $P(a)$, to the root $q^*_P$ constitutes the $\alpha$-th prefix plan and is denoted by $\tau_{\text{pre},a}$ [line 10, Algorithm 1]. Specifically, the prefix part $\tau_{\text{pre},a}$ is constructed by tracing the sequence of parent nodes starting from the node that represents the accepting state $P(a)$ and ending at the root of the tree.

Remark 10. Bias in Prefix Parts. During the construction of the prefix parts, the sampling density functions $f_{\text{rand}}$ and $f_{\text{pre},a}$ are biased towards detecting states $q_P = (q_{\text{PTS}}, q_{\text{B}}^f) \in Q_P$, where $q_{\text{PTS}}$ can be any state in $Q_{\text{PTS}}$ and $q_{\text{B}}^f$ is given feasible final state of the NBA. Once such a state $q_P = (q_{\text{PTS}}, q_{\text{B}}^f)$ is detected, we can switch the bias towards a different feasible final state. Also, once all such feasible final states $q_{\text{B}}^f$ are detected or after a pre-determined number of iterations $n$, we can switch from biased density functions to unbiased (uniform) density functions $f_{\text{rand}}$ and $f_{\text{pre},a}$, that favor exploration of $Q_P$ towards all directions, by selecting $p_{\text{rand}} = |D_{\text{min}}|/|V_T|$ and $p_{\text{new}} = 1/|R_{\text{WTS}}(q_{\text{rand}}^{\text{new}})|$, where recall that $q_{\text{rand}}^{\text{new}} = \Pi_{\alpha \in T}(q_{\text{rand}}^{\text{new},i})$. 3.3 Construction of Optimal Suffix Parts Once the prefix plans $\tau_{\text{pre},a}$ for all $a \in \{1, \ldots, |P|\}$ are constructed, the corresponding suffix plans $\tau_{\text{Suf},a}$ are constructed [lines 11-19, Alg. 1]. Specifically, every suffix part $\tau_{\text{Suf},a}$ is a sequence of states in $Q_P$ that starts from the state $P(a)$ and ends at the same state $P(a)$. To construct the suffix part $\tau_{\text{Suf},a}$ we build a tree $G_T = (V_T, E_T, \text{Cost})$ that approximates the PBA $P$, in a similar way as in Section 3.2, 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 $q^*_P = P(a)$ [line 12, Alg. 1] detected during the construction of the prefix plans, (ii) the goal region corresponding to the root $q^*_P = P(a)$, is defined as $X_{\text{goal}}(q^*_P) = \{q_P = (q_{\text{PTS}}, q_B) \in Q_P | (q_P, (q_{\text{PTS}}, q_B), q_P) \in E_T \}$, (12) (iii) we first check if $q_P \in X_{\text{goal}}^{\text{goal}}$ [line 14, Alg. 1], and (iv) a probability density function $f_{\text{Suf},a}$ that is different from (11) is employed. As for (iii), if $q_P \notin X_{\text{goal}}^{\text{goal}}$, the construction of the suffix part $\tau_{\text{Suf}}$ is trivial, as it consists of only the root, and a loop around it with zero cost [line 15, Alg. 1]. If $q_P \notin X_{\text{goal}}^{\text{goal}}$, then the tree $G_T$ is constructed by Algorithm 2 [line 18, Alg. 1]. As for (iv), $f_{\text{Suf}}^{\text{new},i}(q_{\text{rand}}^{\text{new}},i) : R_{\text{WTS}}(q_{\text{rand}}^{\text{new},i}) \rightarrow [0, 1]$ [line 14, Alg. 3] that generates $q_{\text{new}}^{\text{Suf}}$ is defined as $f_{\text{Suf}}^{\text{new},i}(q_{\text{rand}}^{\text{new},i}) = f_{\text{pre}}^{\text{new},i}(q_{\text{rand}}^{\text{new},i})$, (13)
If \( q_B^r \notin \text{Beech} \), where \( q_B^r = \Pi_B q_B^P \in \text{Beech} \), then this means that the NBA part of the root (final state), i.e., \( q_B^r \in \text{Beech} \), can be reached in one hop from \( q_B^\text{min} \). Then in this case, to steer the tree towards the root \( q_B^r = (q_{\text{BTS}},q_B^r) \), we select the following probability density function \( f_{\text{new},i}(q_i|q_{\text{rand}}) \) to generate the samples \( q_{\text{new}} \).

\[
f_{\text{new},i}(q_i|q_{\text{rand}}) = \begin{cases} \frac{1}{|\text{BTS}(q_{\text{rand}})|} & \text{if } (L(i) = \varnothing) \wedge (q_i \neq SP_{q_{\text{rand}},q_i}^{\text{rand}}(2)) \\ 1 - \frac{1}{|\text{BTS}(q_{\text{rand}})|} & \text{if } (L(i) \neq \varnothing) \wedge (q_i = SP_{q_{\text{rand}},q_i}^{\text{rand}}(2)) \\ \frac{1}{|\text{BTS}(q_{\text{rand}})|} & \text{if } (L(i) = \varnothing) \wedge (q_i \in \text{BTS}, (q_{\text{rand}}) \setminus SP_{q_{\text{rand}},q_i}^{\text{rand}}(2)) \\ \end{cases}
\]

where \( q_i^P = \Pi_{\text{wTS}} q_{\text{BTS}} \). Note that when (13) is employed, the sampling process is biased towards any state \( q_P = (q_{\text{BTS}},q_B^P) \) (and not \( q_P = (q_{\text{BTS}},q_B^{\text{rand}}) \) as in the prefix part); see also [line 13, Alg. 2]. On the other hand, when (14) is employed the sampling process is biased towards the root \( q_B^r = (q_{\text{BTS}},q_B^r) \). Once a tree rooted at \( q_B^r = P(a) \) is constructed, a set \( S_T \subseteq V_T \) is formed that collects all states \( q_j \in V_T \setminus V_{\text{new}}(q_{\text{rand}}) \) [lines 16, 18, Alg. 1]. Given the set \( S_T \), we compute the best suffix plan \( \tau_{\text{pre},a} \) associated with \( q_j^P = P(a) \in \text{Beech} \), as in (Kantaros and Zavlanos 2018a), [line 19, Alg. 1]. This process is repeated for all \( a \in \{1,\ldots,|P|\} \), in parallel [line 11, Alg. 1]. In this way, for each prefix plan \( \tau_{\text{pre},a} \) we construct its corresponding suffix plan \( \tau_{\text{suf},a} \) if it exists.

Remark 11. Bias in Suffix Parts. During the construction of the suffix parts, the sampling density functions \( f_{\text{rand}} \) and \( f_{\text{new},i} \) are biased towards detecting the root \( q_B^P \), so that a loop around \( q_B^P \), i.e., a suffix part is detected. Once this happens, we can switch from biased density functions to unbiased (uniform) density functions \( f_{\text{rand}} \) and \( f_{\text{new},i} \) that favor exploration of \( Q_P \) towards all directions, by selecting \( P_{\text{rand}} = \text{Rand}_{\text{min}}(|V_T|) \) and \( P_{\text{new}} = 1/|\text{wTS}, (q_{\text{rand}})| \), where recall that \( q_{\text{rand}} = \Pi_{\text{wTS}} q_{\text{BTS}} \).

3.4 Construction of Optimal Discrete Plans
By construction, any motion plan \( \tau^a = \tau_{\text{pre},a} \cdot \tau_{\text{suf},a} \cdot \omega \), with \( S_a \neq \emptyset \), and \( a \in \{1,\ldots,|P|\} \) satisfies the LTL task \( \phi \). The cost \( J(\tau^a) \) of each plan \( \tau^a \) is defined in (2). Given an initial state \( q_B^B \in \text{Beech} \), among all the motion plans \( \tau^a \models \phi \), we select the one with the smallest cost \( J(\tau^a) \) [line 20, Alg. 1]. The plan with the smallest cost given an initial state \( q_B^B \) is denoted by \( \tau_{q_B^B} \). Then, among all plans \( \tau_{q_B^B} \), we select again the one with smallest cost \( J(\tau_{q_B^B}) \), i.e., \( \tau = \tau_{a^*,a^*} \), where \( a^* = \text{argmin}_{a} J(\tau_{q_B^B}) \) [lines 21-22, Alg. 1].

3.5 Complexity Analysis
The memory resources needed to store the PBA as a graph structure \( \mathcal{G} = \{V_P,E_P,w_P\} \), defined in Section 2, using its adjacency list is \( O(|V_P| + |E_P|) \) (Sedgewick and Wayne 2011). On the other hand, the memory needed to store a tree, constructed by Algorithm 2, that approximates the PBA is \( O(|V_T|) \), since \( |E_T| = |V_T| - 1 \). Due to the incremental construction of the tree we get that \( |V_T| \leq |V_P| < |V_T| + |E_P| \), which shows that our proposed algorithm requires fewer memory resources compared to existing optimal control synthesis algorithms that rely on the construction of the PBA (Ulusoy et al. 2013, 2014).

Moreover, the time complexity of sampling the state \( q_{\text{new}} \) in Algorithm 3 is \( O(|Q_B| + \max(|Q_i|) + \max(|Q_i| \log(|Q_i|))) \), where \( \rightarrow \) denotes the total number of edges in the wTS, viewing it as a graph. The terms \( |Q_B| \) and \( \max(|Q_i|) \) capture the computational time to construct the reachable sets \( R_B(q_{\text{rand}}) \) and \( R_{\text{wTS}}(q_{\text{rand}}) \) for the largest state-space \( Q_i \) among all robots, respectively; note that the sets \( R_{\text{wTS}}(q_{\text{rand}}) \) can be constructed simultaneously, i.e., in parallel, across the robots, which explains the max operator. Also, these reachable sets do not need to be computed on-the-fly; instead they can be pre-computed making the sampling process more computationally efficient. Similarly, the term \( \max(|Q_i| \log(|Q_i|)) \) is due to the computation of the shortest paths \( SP_{q_{\text{rand}},q_i}^{\text{rand}} \) using the Dijkstra algorithm. Moreover, the time complexity of ‘extend’ and the ‘rewire’ step is \( O(|V_T| (N + 1)) \), as shown in Section IV-D in (Kantaros and Zavlanos 2017).

Remark 12. Implementation. Note that in practice the prefix and suffix parts can be constructed in parallel. Specifically, as soon as a new final state is detected during the construction of the prefix parts, the construction of a new tree rooted at this final state can be triggered immediately so that the respective suffix part is detected. Moreover, observe that the for-loop over all the initial states of the PBA in line 5 of Algorithm 1 can also be executed in parallel. A distributed implementation of the ‘sample’, ‘extend’, and ‘rewire’ step is also presented in (Kantaros and Zavlanos 2018b).

4 Convergence Analysis
In this section, we examine the correctness, optimality, and convergence rate of STyLuS* described in Algorithm 1. Specifically, in Section 4.1, we show that STyLuS* is probabilistically complete and asymptotically optimal. Then, in Section 4.2, we show that Algorithm 2 converges exponentially fast to the optimal solution of Problem 1. In what follows, we denote by \( \mathcal{G}^P = \{V_P,E_P,Cost\} \) the tree that has been built by Algorithm 2 at the \( n \)-th iteration for the construction of either a prefix or suffix part. We also denote the nodes \( q_{\text{rand}}^P \) and \( q_{\text{new}}^P \) at iteration \( n \) by \( q_{\text{rand}}^{P,n} \) and \( q_{\text{new}}^{P,n} \), respectively. The same notation also extends to \( f_{\text{new},i}, f_{\text{rand}}, P_{\text{rand}}, P_{\text{new}} \). Also, in what follows we define the reachable set of state \( q_P \in Q_P \) as follows

\[
R_P^P(q_P) = \{ q_P \in Q_P \mid q_P \rightarrow_P q_P \} \subseteq Q_P,
\]

i.e., \( R_P^P(q_P) \) collects all states \( q_P \in Q_P \) that can be reached in one hop from \( q_P \in Q_P \).
defined in (7) is (i) bounded away from zero on \( V^n_P \) (ii) bounded below by a sequence \( g^n(q_P|V^n_P) \), such that
\[
\sum_{i=1}^{\infty} g^n(q_P|V^n_P) = \infty,
\]
for any given node \( q_P \in V^n_P \), and (iii) can generate independent samples \( q^{rand,n}_i \).

Proposition 13 implies that \( f^n_{new} \) defined in (7) satisfies Assumption 4.1 (i) and (ii) in (Kantaros and Zavlanos 2018a) and the relaxed version of Assumption 4.1 (ii) discussed in Remark A.1 in (Kantaros and Zavlanos 2018a). Note that to ensure that the sampling process is biased towards a final state, \( \epsilon \) should satisfy \( \epsilon > 0.5 \). Note that, since \( p^n_{new} \) can change with \( n \), as long as \( p^n_{new} > \epsilon \) for all \( n \geq 1 \), we can switch from biased to unbiased (uniform) sampling process by selecting \( p^n_{new} = |P^n_{PTS}||V^n_P| \), where in this case \( \epsilon = 1/|Q_P| \). Finally, due to Proposition 13, we can show that there exists an infinite sequence \( K \), so that for all \( k \in K \) it holds \( q^{rand,n+k}_i = q^{PTS,n}_i \); see Lemma 5.4 in (Kantaros and Zavlanos 2018a).

**Proposition 14.** Biased Probability Density \( f^n_{new} \). Assume that \( p^n_{new} > \epsilon \), for all \( n \geq 1 \) and for any \( \epsilon > 0 \). Then, the probability density functions \( f^n_{new}(q_{PTS}|q^{rand,n}) = \Pi_{i=1}^{N}f_{new}(q_i|q^{rand,n}) \) defined in (11), (13), and (14) (i) are bounded away from zero on \( R_{PTS}(q^{rand,n}) \); (ii) for any fixed and given node \( q^{rand,n} \in V^n_P \), there exists an infinite sequence \( h^{n+k}(q_{PTS}|q^{rand,n+k}) \) so that the distributions \( f^{n+k}(q_{PTS}|q^{rand,n+k}) \) for all \( k \in K \), satisfy \( f^{n+k}(q_{PTS}|q^{rand,n+k}) \geq h^{n+k}(q_{PTS}|q^{rand,n+k}) \); and (iii) given a state \( q^{rand,n-k} \), independent samples \( q^{new,n}_i \) can be drawn from the probability density function \( f^n_{new} \).

Proposition 14 implies that the functions \( f^n_{new} \) defined in (11), (13), and (14) satisfy Assumption 4.2 (i) and (iii) in (Kantaros and Zavlanos 2018a) and the relaxed version of Assumption 4.2 (ii) discussed in Remark A.2 in (Kantaros and Zavlanos 2018a). Note that to ensure that the sampling process is biased towards a final state, \( \epsilon \) should satisfy \( \epsilon > 0.5 \). Moreover, note that Proposition 14 holds even if the state towards which \( f^n_{new} \) is biased at iteration \( n \) changes. This means that during the construction of the prefix part we can switch the bias towards a different state \( Q^{bias}_{bias} \) at any iteration \( n \). As before, the fact that \( p^n_{new} \) can change with \( n \), as long as \( p^n_{new} > \epsilon \) for all \( n \geq 1 \), allows us to switch from biased to unbiased (uniform) sampling process by selecting \( p^n_{new} = 1/|R_{PTS}(q^{rand,n})| \), where in this case \( \epsilon = 1/|Q_i| \).

Since the probability density functions \( f^n_{rand} \) and \( f^n_{new} \) satisfy Assumptions 4.1 and 4.2 in (Kantaros and Zavlanos 2018a), we can show, following the same steps as in (Kantaros and Zavlanos 2018a), that STyLuS* is probabilistically complete and asymptotically optimal.

**Theorem 15.** Completeness and Optimality. Assume that there exists a solution to Problem 1. Then, STyLuS* is probabilistically complete, i.e., it will find with probability 1 a motion plan \( \tau \vdash \phi \), as \( v^{max}_{max} \to \infty \) and \( v^{max}_{inf} \to \infty \) and asymptotically optimal, i.e., it will find with probability 1 the optimal motion plan \( \tau \vdash \phi \), as \( v^{max}_{max} \to \infty \) and \( v^{max}_{inf} \to \infty \), that minimizes the cost function \( J(\tau) \) defined in (2).

### 4.2 Rate of Convergence

In this section, we show that STyLuS* converges exponentially fast to the optimal solution of Problem 1. To show this, we first show that it converges exponentially fast to a feasible solution of Problem 1.

**Theorem 16.** Convergence Rate to a Feasible Solution. Assume that there exists a solution to Problem 1 in prefix-suffix form. Let \( p \) denote either the prefix or suffix part of this solution in the PBA defined as
\[
p = q^1_P, q^2_P, \ldots, q^{K-1}_P, q^K_P,
\]
that is of length \( (\text{number of states}) \) \( K \) and connects the root \( q^1_P = q^0_P \) of the tree to a state \( q^K_P \in \mathcal{X}_{goal} \), where \( q^0_P \rightarrow \cdots \rightarrow q^k_P \), for all \( k \in \{1, 2, \ldots, K-1\} \). Then, there exist parameters \( \alpha_n(p) \in (0, 1) \) for every iteration \( n \) of Algorithm 2, such that the probability \( \Pi_{max}(q^K_P) \) that Algorithm 2 detects the state \( q^K_P \in \mathcal{X}_{goal} \) within \( n_{max} \) iterations satisfies
\[
1 \geq \Pi_{max}(q^K_P) \geq 1 - e^{\frac{\alpha_{n_{max}}(p)}{\max_{n_{max}} + K}}, \text{ if } n_{max} > K.
\]

**Proof.** To prove this result, we model the sampling process discussed in Section 3.2.1 as a Poisson binomial process. Specifically, we define Bernoulli random variables \( Y_n \) at each iteration \( n \) of Algorithm 2 so that \( Y_n = 1 \) only if the state \( q^K_P \) is sampled and added to the tree at iteration \( n_i \), where \( k \) is the smallest element of the set \( \{1, \ldots, K\} \) that satisfies \( q^K_P \notin V^{n-1} \). Then, using the random variables \( Y_n \), we define the random variable \( Y = \sum_{i=1}^{n_{max}} Y_i \) which captures the total number of successes of the random variables \( Y_n \) and we show that it follows a Poisson binomial distribution. Finally, we show that \( \Pi_{max}(q^K_P) \geq \mathbb{P}(Y \geq K) \) which yields (17) by applying the Chernoff bounds (Motwani and Raghavan 2010) to \( Y \).

Let \( X_k^n \) for all \( k \in \{1, \ldots, K\} \) denote a Bernoulli random variable associated with iteration \( n \) of Algorithm 2, that is equal to 1 if the state \( q^K_P = (q^K_P, q^K_B) \) in (16) is added to the tree at iteration \( n_i \) or has already been added to the tree at a previous iteration \( m < n \), and 0 otherwise. Observe that \( X_k^n \) is equal to 1 for all iterations \( n_i \geq 1 \) of Algorithm 2, since the tree is rooted at \( q^1_P \) and, therefore, \( q^K_P \in V^n_P \) for all \( n_i \geq 1 \). By construction of the sampling step in Section 3.2.1 the probability that \( X_k^n = 1 \) is defined as follows
\[
\mathbb{P}_{add}(q^K_P) = \left\{ \begin{array}{ll}
\sum_{q_B \in \mathcal{V}_T \cap R^P(q^K_B, q^K_P)} |f^n_{rand}(q_P)f^n_{new}(q^K_P|q^{PTS})|, & \text{if } q^K_P \notin V^n_P, \\
1, & \text{if } q^K_P \in V^n_P.
\end{array} \right.
\]

where \( R^P(q^K_P) \) is defined in (15), and \( q^K_B = \Pi_{max}q^{PTS} \). Observe that if \( V^n_P \cap R^P(q^K_P) = \emptyset \) then \( \mathbb{P}_{add}(q^K_P) = 0 \), if \( q^K_P \notin V^n_P \). Moreover, note that if \( q^K_P \) already belongs to \( V^n_P \), from a previous iteration \( m < n \) of Algorithm 2, then it trivially holds that \( \mathbb{P}_{add}(q^K_P) = 1 \).

Given the random variables \( X_k^n \), we define the discrete random variable \( Y_n \) initialized as \( Y_1 = X_1 \) and for every subsequent iteration \( n_i > 1 \) defined as
\[
Y_n = \left\{ \begin{array}{ll}
X_k^n, & \text{if } (Y_{n-1} = X_{k-1}^{n-1}) \land (X_{k-1}^{n-1} = 0) \land (k + 1 \leq K) \\
X_{k+1}^{n+1}, & \text{if } (Y_{n-1} = X_{k+1}^{n-1}) \land (X_{k+1}^{n-1} = 1) \land (k + 1 \leq K) \\
X_K^n, & \text{if } (Y_{n-1} = X_K^{n-1}) \land (X_K^{n-1} = 1) \land (k + 1 > K).
\end{array} \right.
\]
In words, $Y_n$ is defined exactly as $Y_{n-1}$, i.e., $Y_n = Y_{n-1} = X^k_{n-1} = X_k^n$, if $Y_{n-1} = X^k_{n-1} = 0$, i.e., if the state $q^k_{P}\mu$ does not exist in the tree at iteration $n - 1$. Thus, in this case, $Y_n = X_k^n = 1$ if the state $q^k_{P}$ in (16) is added to the tree at iteration $n$; see the first case in (19). Also, $Y_n = X^n_{k+1}$, if $Y_{n-1} = X^k_{n-1} = 1$, i.e., if the state $q^k_{P}$ was added to the tree at the previous iteration $n - 1$. In this case, $Y_n = X^n_{k+1} = 1$, if the next state $q^{k+1}_{P}$ in (16) is added to the tree at iteration $n$ (or has already been added at a previous iteration $m < n$); see the second case in (19). If $k + 1 > K$ and $X^k_{n-1} = 1$, then we define $Y_n = X^n_K$; see the last case in (19). Note that in this case, $Y_n$ can be defined arbitrarily, i.e., $Y_n = X^n_k$, for any $k \in \{1, \ldots, K\}$, since if $k + 1 > K$ and $X^k_{n-1} = 1$, then this means that all states that appear in (16) have been added to $\mathcal{Y}$. By convention, in this case we define $Y_n = X^n_k$. Since $Y_n$ is equal to $X^n_k$ for some $k \in \{1, \ldots, K\}$, as per (19), for all $n \geq 1$, we get that $Y_n$ also follows a Bernoulli distribution with parameter (probability of success) $p^{\text{inc}}$ equal to the probability of success of $X^n_k$ defined in (18), i.e.,

$$
p^{\text{inc}}_n = p^{\text{add}}(q^n_{P}),
$$

where the index $k$ is determined as per (19).

Given the random variables $Y_n$, $n \in \{1, \ldots, n_{\text{max}}\}$, we define the discrete random variable $Y$ as

$$
Y = \sum_{n=1}^{n_{\text{max}}} Y_n. \tag{20}
$$

Observe that $Y$ captures the total number of successes of the random variables $Y_n$ after $n_{\text{max}} > 0$ iterations, i.e., if $Y = y$, $y \in \{1, \ldots, n_{\text{max}}\}$, then $Y_n = 1$ for exactly $y$ random variables $Y_n$. Therefore, if $Y \geq K$, then all states that appear in the path $\mathcal{P}$ of length $K$ given in (16) have been added to the tree, by definition of the random variables $Y_n$ and $Y$ in (19) and (20), respectively. Since there may be more than one path in the PBA that connect the goal state $q^K_{P}$ to the root $q^0_{P}$, we conclude that the probability $\mu^{\text{inc}}(q^K_{P})$ of finding the goal state $q^K_{P}$ within $n_{\text{max}}$ iterations is at least $\mathbb{P}(Y \geq K)$, i.e.,

$$
\mu^{\text{inc}}(q^K_{P}) \geq \mathbb{P}(Y \geq K). \tag{21}
$$

In what follows, our goal is to compute the probability $\mathbb{P}(Y \geq K)$. Observe that $Y$ is defined as a sum of Bernoulli random variables $Y_n$ that are not identically distributed as their probabilities of success $p^{\text{inc}}_n$ are not fixed across the iterations $n$, since the definition of $Y_n$ changes at every iteration $n$ as per (19). Therefore, $Y$ follows a Poisson Binomial distribution which has a rather complicated pmf which is valid for small $n$ and numerically unstable for large $n$; see e.g., (Choi and Xia 2002; Hong 2011). Therefore, instead of computing $\mathbb{P}(Y \geq K)$, we compute a lower bound for $\mathbb{P}(Y \geq K)$ by applying the Chernoff bound to $Y$; see e.g., (Motwani and Raghavan 2010).

Specifically, we have that

$$
\mathbb{P}(Y < K) < \mathbb{P}(Y \leq K) = \mathbb{P}(Y \leq K \mu) = \mathbb{P}(Y \leq (1 - (1 - \frac{K}{\mu}) \mu) = \mathbb{P}(Y \leq (1 - \delta) \mu) \leq e^{-\frac{(\delta K)}{\mu}}, \tag{22}
$$

where $\mu$ is the mean value of $Y$ defined as $\mu = \sum_{n=1}^{n_{\text{max}}} p^{\text{inc}}_n$. Also, the last inequality in (22) is due to the Chernoff bound in the lower tail of $Y$ and holds for any $\delta \in (0, 1)$. Observe that the Chernoff bound can be applied to $Y$, as it is defined as the sum of independent Bernoulli random variables $Y_n$. Specifically, the random variables $Y_n$ are independent since independent samples $q^n_{P}$ can be generated by the proposed sampling process, described in Section 3.2.1, by construction of the density functions $f_{\text{rand}}$ and $f_{\text{new}, i}$. Substituting $\delta = 1 - \frac{K}{\mu}$ in (22), we get

$$
\mathbb{P}(Y < K) \leq e^{\frac{-K^2}{\mu} + \frac{K^2}{2}} \leq e^{\frac{-K^2}{\mu} + 1} = e^{\frac{\sum_{n=1}^{n_{\text{max}}} p^{\text{inc}}_n}{\mu} + K}, \tag{23}
$$

where the last inequality is due to $e^{-K^2/\mu} \leq 1$. Recall that (23) holds for any $\delta = 1 - \frac{K}{\mu} \in (0, 1)$, i.e, for any $n_{\text{max}}$ that satisfies

$$
0 < \delta < 1 \implies K < \mu \sum_{n=1}^{n_{\text{max}}} p^{\text{inc}}_n \implies K < n_{\text{max}}, \tag{24}
$$

where the last inequality in (24) is due to $p^{\text{inc}} \leq 1$. Therefore, (23) holds as long as $n_{\text{max}} > K$.

Note also that the inequality $0 < K < \sum_{n=1}^{n_{\text{max}}} p^{\text{inc}}_n$ in (24) is well defined, since $p^{\text{inc}}_n = p^{\text{add}}(q^n_{P})$ is strictly positive for all $n \geq 1$ by definition of $Y_n$. To show that, observe that if $Y_n = X^n_k$, for some $k \in \{1, \ldots, K\}$, then it holds that $q^{k-1}_{P} \in \mathcal{Y}_T$, by construction of $Y_n$ in (19). Then, by definition of the feasible path (16), we have that $q^{k-1}_{P} \in \mathcal{Y}_T = \mathcal{R}_P(q^k_{P})$. Thus, we get that $q^{k-1}_{P} \in \mathcal{Y}_T \cap \mathcal{R}_P(q^k_{P})$. Then, $p^{\text{inc}}_n = p^{\text{add}}(q^n_{P}) > 0$ (see also (18)) holds, since (i) $\mathcal{Y}_T \cap \mathcal{R}_P(q^n_{P}) \neq \emptyset$, as $q^{k-1}_{P} \in \mathcal{Y}_T \cap \mathcal{R}_P(q^k_{P})$; (ii) $f^{\text{rand}}$ is bounded away from zero $\mathcal{Y}_T$ by construction in (7) as long as $p^{\text{rand}}_n > 0$, for all $n \geq 1$, i.e., $f^{\text{rand}}(q^{k-1}_{P}) > 0$, $q^{k-1}_{P} \in \mathcal{Y}_T$; and (iii) $f^{\text{new}, i}$ is bounded away from zero on $\mathcal{R}_{\text{wts}}(q^{k-1}_{P})$, for all $n \geq 1$ and for all $i \in \{1, \ldots, N\}$ by construction in (11), (13), and (14) (as long as $p^{\text{new}}_n > 0$, i.e., $f^{\text{new}, i}(q^{k-1}_{P}) > 0$), where $q^{k}_{P} = \Pi_{\text{wts}}(q^{k}_{P})$.

Thus, we proved that there exist parameters

$$
\alpha_n(\mathcal{P}) = p^{\text{inc}}_n \in (0, 1], \tag{25}
$$

associated with every iteration $n$ of Algorithm 2 such that the probability $\mu^{\text{inc}}(q^K_{P})$ of finding the goal state $q^K_{P}$ within $n_{\text{max}} > K$ iterations satisfies

$$
1 \geq \mu^{\text{inc}}(q^K_{P}) \geq 1 - \mathbb{P}(Y < K) \geq 1 - e^{-\sum_{n=1}^{n_{\text{max}}} \alpha_n(\mathcal{P}) + K}, \tag{26}
$$

completing the proof.

Observe that Theorem 16 holds for any (biased or unbiased) density functions $f^{\text{rand}}$ and $f^{\text{new}, i}$, that can possibly

\[\text{Prepared using sagej.cls} \]
change with iterations \( n \), as long as they (i) are bounded away from zero on \( \mathbb{V}_T^\mathcal{P} \) and \( \mathcal{R}_{\text{TS}}(q) \), respectively; and (ii) can generate independent samples, which are essentially the Assumptions 4.1(i), 4.1(ii), 4.2(i), and 4.2(ii), respectively. Therefore, the probability of finding a feasible plan converges exponentially to 1 even if we switch to an unbiased density function by selecting \( P_{\text{rand}}^n = |\mathcal{D}_{\text{min}}^n|/|\mathbb{V}_T^\mathcal{P}| \) and \( P_{\text{new}}^n = 1/|\mathcal{R}_{\text{TS}}(q_{\text{rand}}^n)| \).

**Remark 17.** Effect of bias. The parameters \( \alpha_n(p) \) in Theorem 16 capture the probability of sampling a state that belongs to a path \( p \) in (16) and adding it to the tree \( \mathcal{G}_T^n \) at iteration \( n \); see also (18). The larger the parameters \( \alpha_n(p) \) are, the larger the probability \( \Pi_{\text{rec}}(q_{\text{rand}}^n) \) is of detecting the goal state \( q_{\text{goal}}^n \) in \( \mathcal{X}_\text{goal} \) given any feasible path \( p \) of length \( K \) that connects \( q_{\text{rand}}^n \) to the root \( q_1^p \) within \( n_{\text{max}} > K \) iterations; see also (17). This also implies that for a given \( q_{\text{rand}}^n \) if the density functions \( f_{\text{rand}}^{\text{new}} \) and \( f_{\text{new},i}^{\text{new}} \) are biased towards feasible states, i.e., towards states \( q_{\text{rand}}^{\text{new},i} \) with \( T_p(q_{\text{rand}}^{\text{new},i}) \neq \emptyset \), then there exists at least one feasible (either prefix or suffix) path \( p \) associated with \( q_{\text{rand}}^n \) so that its parameters \( \alpha_n(p) \) are larger than the parameters of any other feasible path, which can be associated with any NBA final state in case of the prefix part. As a result, this path will be detected faster than any other feasible path in the PBA. If this path \( p \) is a prefix part, then it ends at a PBA final state that is associated with \( q_{\text{rand}}^n \) while if it is a suffix part, its end state is the root. Note that using biased density functions does not mean that all, e.g., prefix paths associated with \( q_{\text{rand}}^n \) will be detected faster than if unbiased density functions were used. Instead, what this means is that if the sampling process is biased towards feasible states \( q_{\text{rand}}^{\text{new},i,n} \), then there exists at least one prefix path associated with \( q_{\text{rand}}^{\text{new},i,n} \) that will be detected much faster. Also, paths which the sampling process is not biased to are expected to be detected slower than if unbiased (uniform) density functions are employed. Finally, recall that the sampling process is biased towards shortest paths, in terms of numbers of hops, in the state-space of both the wTSs and the NBA. Therefore, the paths that the sampling functions are biased to are expected to have a small \( K \), i.e., a small number of transitions/states. The latter is also verified through numerical experiments in Section 5.

**Remark 18.** Biasing towards infeasible states. Note that if the density functions \( f_{\text{rand}}^{\text{new},i} \) are biased towards states that violate the LTL formula, i.e., states \( q_{\text{rand}}^{\text{new},i,n} \) with \( \mathcal{R}_p(q_{\text{rand}}^{\text{new},i,n}) = \emptyset \), then the parameters \( \alpha_n(p) \), associated with a feasible path \( p \), will have low values, by construction of \( f_{\text{rand}}^{\text{new},i} \) and \( f_{\text{new},i}^{\text{new}} \). As a result, a large number of iterations is expected until \( p \) is detected due to Theorem 16. Also, note that in general, it is not possible to know a priori whether the states in the PBA which the sampling process is biased to are feasible or not; therefore, in practice, values for \( P_{\text{rand}}^{\text{new},i} \) and \( P_{\text{new},i}^{\text{new}} \) that are arbitrarily close to 1 should be avoided, as they might significantly slow down the synthesis part.

Using Theorem 16, and Propositions 13-14, we establish the exponential convergence rate to the optimal, either prefix or suffix, path that connects a desired goal state to the root of the tree.

**Theorem 19.** Convergence Rate to the Optimal Path. Assume that there exists a solution to Problem 1 in prefix-suffix form. Let \( p^* \) denote either the prefix or suffix part of the optimal solution in the PBA defined as

\[
p^* = q_1^{p_1^*}, q_2^{p_2^*}, \ldots, q_{K-1}^{p_{K-1}^*}, q_K^{p_K^*},
\]

that is of length \( K \) (number of states) and connects the root \( q_1^{p_1^*} \) of the tree to a state \( q_{\text{goal}}^n \) in \( \mathcal{X}_\text{goal} \) where \( q_{\text{goal}}^n \rightarrow p_{\text{goal}}^{p_{\text{goal}}^*+1} \), for all \( k \in \{1, 2, \ldots, K-1\} \). Then, there exist parameters \( \gamma_n(q_{k}^{p_{k}^*}) \in (0, 1] \) for every state \( q_k^{p_k^*} \) in (26) and every iteration of Algorithm 2, as well as an iteration \( n_k \) of Algorithm 2 for every state \( q_k^{p_k^*} \) in (26), so that the probability \( \Pi_{\text{opt}}(p^*) \) that Algorithm 2 detects the optimal path (26) within \( n_{\text{max}} \) iterations satisfies

\[
\Pi_{\text{opt}}(p^*) \geq \prod_{k=1}^{K-1} \left( 1 - e^{-\sum_{\alpha_n(q_{k}^{p_{k}^*})} \gamma_n(q_{k}^{p_{k}^*}+1) - K_n} \right)
\]

if \( n_{\text{max}} > 2K \). In (27), the parameters \( \alpha_n(p^*) \) are defined as in Theorem 16 and \( n \) is any fixed iteration of Algorithm 2 that satisfies \( n \leq n_{\text{max}} - K \).

**Proof.** To show this result, first we show that if all states in (26) exist in the tree \( \mathcal{G}_T^n \) within \( n < n_{\text{max}} \) iterations of Algorithm 2, and if all these states are eventually selected (after iteration \( n \)) to be the nodes \( q_{\text{new},i}^{n} \) in the order that they appear in (26), then the optimal path (26) is detected by Algorithm 2. Second, we compute the probability that the above statement is satisfied within \( n_{\text{max}} \) iterations. Finally, we show that this probability is a lower bound to the probability of finding the optimal path (26), which yields (27).

First, assume that (i) all states \( q_k^p \) exist in the tree at iteration \( n < n_{\text{max}} \), and (ii) there exists a finite sequence \( K \) of iterations of Algorithm 2, where \( n_k+1 > n_k > n \) such that \( q_{k}^{p_k^*} \in \mathcal{X}_{\text{feas},k} \). In what follows, we use induction to show that the optimal path is detected at iteration \( n_{k+1} \). Specifically, by assumptions (i) and (ii), we have that \( q_k^p \) will be selected to be the node \( q_{\text{new},i}^{n_k} \) at iteration \( n_{k+1} \) and that \( q_k^{p_k^*} \in \mathcal{X}_{\text{feas},k} \). Observe that by definition of the optimal path (26), the optimal cost of \( q_k^{p_k^*} \) is obtained when \( q_k^p \) is directly connected to the root \( q_1^p \). Therefore, by construction of the rewiring step, \( q_k^p \in \mathcal{V}_{\text{feas},k} \) will get rewired to \( q_{\text{new},i}^{n_k} \in \mathcal{V}_{\text{feas},k} \) at iteration \( n_k \). Next, the inductive step follows. By assumptions (i) and (ii), it holds that \( q_{k+1}^{p_{k+1}^*} \) is selected to be the node \( q_{\text{new},i}^{n_k} \) at iteration \( n_k+1 > n_k+2 > n \) and that \( q_k^{p_k^*} \in \mathcal{X}_{\text{feas},k+1} \). Assume that \( q_k^{p_k^*} \in \mathcal{V}_{\text{feas},k+1} \) gets rewired to \( q_{\text{new},i}^{n_{k+1}} \) at iteration \( n_{k+1} \), which is connected to the root as in (26) (inductive assumption). We will show that at the subsequent iteration \( n_{k+1} > n_{k+1} \), \( q_k^{p_k^*} \) will get rewired to \( q_{\text{new},i}^{n_{k+1}} \), which is connected to the root as in (26). In particular, by assumptions (i) and (ii), we have that there exists an iteration \( n_k > n_k+1 \), such that \( q_k^{p_k^*} \in \mathcal{V}_{\text{feas},k} \) is selected to be the node \( q_{\text{new},i}^{n_k} \) while \( q_{k+1}^{p_{k+1}^*} \) is connected to the root \( q_1^p \). Then, by induction, we conclude that at iteration \( n_{k+1} \),
the state $q^K_n$ will get rewired to $q^{K-1}_n$ which is connected to the root as in (26), i.e., the optimal path (26) exists in the tree at iteration $n_{K-1}$.

The probability that both assumptions (i) and (ii) will become true within $\tilde{n}$ and $n_{\text{max}}$ iterations, respectively, is $\mathbb{P}(R(n_{\text{max}}) \cap A(\tilde{n})) = \mathbb{P}(R(n_{\text{max}})|A(\tilde{n})) \mathbb{P}(A(\tilde{n}))$, where $A(\tilde{n})$ is the event that all states in (26) have been added to the tree at iteration $\tilde{n} < n_{\text{max}}$ and $R(n_{\text{max}})$ denotes the event that all the states that appear in the optimal path (26) have been selected to be the nodes $q^{\text{new}}_p$ in the order they appear in (26) after the iteration $\tilde{n}$ and before the iteration $n_{\text{max}}$. Then, the probability that the optimal path is detected within $n_{\text{max}}$ iterations satisfies

$$
\Pi_{\text{opt}}(\pi^*) \geq \mathbb{P}(R(n_{\text{max}})|A(\tilde{n})) \mathbb{P}(A(\tilde{n})),
$$

where the inequality in (28) is due to the fact that satisfying assumptions (i) and (ii) is a sufficient, and not necessary, condition to find the optimal path.

In what follows, we compute a lower bound for the probabilities $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n}))$ and $\mathbb{P}(A(\tilde{n}))$ which we will then substitute in (28) to get (27). Recall that $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n}))$ captures the probability that Algorithm 2 will reach all iterations $n_1 < n_2 < \cdots < n_{K-1}$, after the iteration $\tilde{n}$ (i.e., $\tilde{n} < n_1$) and before the iteration $n_{\text{max}}$, given the event $A(\tilde{n})$. Therefore, we first need to show that the sequence $\mathcal{N} = \{n_1, \ldots, n_{K-1}\}$ exists since, otherwise, it trivially holds that $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n})) = 0$ for any $n_{\text{max}}$.†† Assume that such a sequence $\mathcal{N}$ does not exist. Then, this means that there exists an iteration $n_K \in \mathcal{N}$, where $1 \leq K < K$, such that after the iteration $n_K$ the state $q^{K+1}_p$ will never be selected to be the node $q^{\text{new}}_p$. The latter implies that after the iteration $n_K$ the state $q^{K+1}_p$ that is reachable from $q^K_p \in \mathcal{V}^K_p$ as per (26), i.e., $q^K_p \rightarrow_p q^{K+1}_p$, will not be selected infinitely often to be the node $q^{\text{new}}_p$, which contradicts Corollary 5.6 of (Kantaros and Zavlanos 2018a).†† Therefore, we conclude that the sequence $\mathcal{N}$ exists. Note that given $K$ and $n_{\text{max}}$, the sequence $\mathcal{N}$ can only exist if

$$
\tilde{n} < n_{\text{max}} - K,
$$

since otherwise, there will not be enough iterations left after $\tilde{n}$ so that the states $q^K_p, k \in \{1, \ldots, K-1\}$ are selected to be the nodes $q^{\text{new}}_p$.

Next, we compute a lower bound for the probability $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n}))$. To do this, we first define Bernoulli random variables $Z^k_n$ so that $Z^k_n = 1$ if $q^K_p$ is selected to be the node $q^{\text{new}}_p$ at iteration $n$ of Algorithm 2, and 0 otherwise. By construction of the sampling step, $Z^k_n = 1$ occurs with probability

$$
\mathbb{P}^{\text{new}}(q^K_p) = \sum_{q^K_p \in \mathcal{V}^{K-1}_p \cap \mathcal{R}^{K-1}_p(q^K_p)} [f_{\text{rand}}(q^{K-1}_p) f_{\text{new}}^{n}(q^K_p | q^{K-1}_p)],
$$

where $\mathcal{R}^{K-1}_p(q^K_p)$ is defined in (15) and $f_{\text{new}}^{n}(q^K_p | q^{K-1}_p) = \prod_{k=1}^{n} \mathbb{P}_{q^{k-1}_p}(q^K_p | q^{K-1}_p)$.

Given $Z^k_n$, we define the following discrete random variables for all states $q^K_p, k \in \{1, \ldots, K-1\}$

$$
Z_k = \sum_{n=n_k-1}^{n_{\text{max}}} Z^k_n,
$$

where $n = \tilde{n}$. Notice that the sum in (31) is well-defined as long as Algorithm 2 has reached iteration $n_{k-1}$ within $n_{\text{max}}$ iterations. Also, note that $Z_k$ follows a Poisson Binomial distribution, since it is defined as the sum of Bernoulli random variables that are not identically distributed. Moreover, observe that $Z_k$ captures the total number of times $q^K_p$ was selected to be the node $q^{\text{new}}_p$ when $n \in [n_k-1, n_{\text{max}}]$ i.e., after $q^{K-1}_p \in \mathcal{V}^{K-1}_p$ was selected to be the node $q^{\text{new}}_p$ at least once in the interval $[n_k-1, n_{\text{max}}]$. Therefore, $Z_k \geq 1$ means that $q^K_p$ was selected to be $q^{\text{new}}_p$ at least once in the interval $[n_k-1, n_{\text{max}}]$. Thus, we can rewrite the event $R(n_{\text{max}})$ as

$$
R(n_{\text{max}}) = \bigcap_{k=1}^{K-1} (Z_k \geq 1).
$$

Given (32) and using the Bayes rule, $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n}))$ can be written as follows

$$
\mathbb{P}(R(n_{\text{max}})|A(\tilde{n})) = \prod_{k=1}^{K-1} \mathbb{P}(Z_k \geq 1 | B_k, A(\tilde{n})),
$$

where $B_k = \bigcap_{k=1}^{k-1} (Z_k \geq 1)$, for all $k > 1$ and $B_0 = \emptyset$. In words, $\mathbb{P}(Z_k \geq 1 | B_k, A(\tilde{n}))$ captures the probability of reaching iteration $n_k$ within $n_{\text{max}}$ iterations, given that iteration $n_{k-1}$ has already been reached and all states $q^K_p$ have been added to the tree by the iteration $\tilde{n} < n_k$. Also, observe that $Z_k$ is well-defined in (33). Specifically, recall that $Z_k$, defined in (31), is well defined as long as Algorithm 2 reaches iteration $n_{k-1}$ within $n_{\text{max}}$ iterations. This requirement is satisfied in (33) since the probability of $Z_k \geq 1$ is computed given the events $B_k$ and $A(\tilde{n})$.

Following the same logic as in the proof of Theorem 16, we can show that

$$
\mathbb{P}(Z_k \geq 1 | B_k, A(n_{\text{max}})) \geq 1 - e^{-\sum_{n=n_{k-1}}^{n_{\text{max}}-1} \gamma_n(q^K_p) + 1},
$$

as long as $n_{\text{max}} > n_{k-1}$. In (34), $\gamma_n(q^K_p) = \mathbb{P}_{q^{K-1}_p}(q^K_p)$ defined in (30). Note that $\gamma_n(q^K_p) > 0$ since $\mathcal{V}^{K-1}_p \cap \mathcal{R}^{K-1}_p(q^K_p) \neq \emptyset$, for all $n \in [n_{k-1}, n_{\text{max}}]$, since in (34), we assume that the event $B_k$ is true. As a result, $q^{K-1}_p \in \mathcal{V}^{K-1}_p$, for all $n \in [n_{k-1}, n_{\text{max}}]$. Also, by construction of the path (26), it holds that $q^{K-1}_p \in \mathcal{R}^{K-1}_p(q^K_p)$ which means that $\mathcal{V}^{K-1}_p \cap \mathcal{R}^{K-1}_p(q^K_p) \neq \emptyset$.

Moreover, recall from the proof of Theorem 16 that

$$
\mathbb{P}(A(\tilde{n})) = \Pi_{\text{opt}}(\pi^*) \geq 1 - e^{-\sum_{n=1}^{\tilde{n}} \gamma_n(\pi^*) + K},
$$

which holds for $\tilde{n} > K$, where $K$ is the length of any path, including the optimal path $\pi^*$, that connects $q^K_p$ to the root of the tree.

Substituting equations (33), (34), and (35) into (28) yields (27) which holds for $\tilde{n} < n_{\text{max}} - K$ due to (29) and $\tilde{n} > K$ due to (35) or, equivalently, for $\tilde{n} < n_{\text{max}} - K$ and $n_{\text{max}} > 2K$ completing the proof.

††Note that here we prove that such iterations $n_k$ exist and not that they necessarily satisfy $n_k \leq n_{\text{max}}$, for all $k \in \{1, \ldots, K-1\}$. Recall that the probability that all iterations $n_k$ satisfy $\tilde{n} < n_k \leq n_{\text{max}}$ is $\mathbb{P}(R(n_{\text{max}})|A(\tilde{n}))$.

††Recall that Corollary 5.6 of (Kantaros and Zavlanos 2018a) holds since the employed sampling density functions satisfy Assumptions 4.1 and 4.2 of (Kantaros and Zavlanos 2018a), as shown in Propositions 13-14.
Note that Theorem 19 holds for any (biased or unbiased) density functions $f_{\text{rand}}^n$ and $f_{\text{new}}^n$, that can possibly change with iterations $n$, as long as they satisfy Assumptions 4.1 and 4.2 of (Kantaros and Zavlanos 2018a), respectively. Therefore, the probability of finding the optimal plan converges exponentially to 1 even if at any iteration $n$ we switch bias to a different state or switch to an unbiased density function by selecting $p_{\text{rand}}^n = |D_{\text{min}}^n|/|V_T^n|$ and $p_{\text{new}}^n = 1/|\mathcal{R}_n|_{\text{wTS}}, (q_{\text{rand}}, n)$.

5 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, that illustrate the scalability of our proposed algorithm compared to state-of-the-art methods. Simulation studies that show the advantage of the employed reachability-based sampling process, discussed in Section 3.2.1, compared to sampling approaches that generate random states $q_{\text{new}}^n$ that belong to $Q_P$, as e.g., in (Kantaros and Zavlanos 2017), can be found in Section VI in (Kantaros and Zavlanos 2018a); see also Proposition 5.1 in (Kantaros and Zavlanos 2018a).

Given an NBA that corresponds to the assigned LTL specification, the first step in the proposed algorithm is to construct the sets of feasible symbols $\Sigma_{\text{feas}}$. Note that the construction of the sets $\Sigma_{\text{feas}}$ can become computationally expensive as $N$ and $W$ increase since the size of $\mathcal{A}P = \bigcup_{i=1}^N \bigcup_{e=1}^W \{ \pi_i^e \}$ also increases and so does the size of the alphabet $\Sigma = 2^{\mathcal{A}P}$ and the size of $\Sigma_{\text{feas}}$. To mitigate this issue, instead of defining the set of atomic propositions as $\mathcal{A}P = \bigcup_{i=1}^N \bigcup_{e=1}^W \{ \pi_i^e \}$, we define the set $\mathcal{A}P$ as $\mathcal{A}P = \{ \cup_i \{ \xi_e \} \cup \emptyset \}$, where $\emptyset$ stands for the empty symbol and $E$ denotes the total number of Boolean formulas $\xi$ in conjunctive or disjunctive normal form that appear in the LTL formula and are defined over the atomic propositions $\pi_i^e$. This allows us to construct smaller LTL formulas defined over $2^{\{\cup_i \{ \xi_e \} \cup \emptyset\}}$. As a result, the computational cost of creating the NBA and the sets $\Sigma_{\text{feas}}$ decreases. In the following case studies, the sets $\Sigma_{\text{feas}}$ are computed using the software package (Ulusoy et al. 2011) that relies on (Gastin and Oddoux 2001) for the construction of the NBA.

5.1 Completeness, Optimality, and Scalability

Probabilistic Completeness & Scalability In what follows, we examine the performance of STyLuS$^+$ with respect to the number of robots and the size of the wTSs. The results for all considered planning problems are reported in Table 1. In Table 1, the first, second, and third column show the number $N$ of robots, the size of the state-space $|Q|$, which is the same for all wTSs, and the size of the state-space of the PBA, defined as $|Q_P| = |Q_B| \prod_{i=1}^N |Q_i|$, respectively. Viewing the wTSs as graphs, the average degree of the nodes of the examined wTSs with $|Q| = 9, |Q| = 10^2, |Q| = 10^3, |Q| = 2.5 \times 10^3$, and $|Q| = 10^4$ is 3, 12, 30, 42, and 42, respectively, for all robots $i$ and for each case study in Table 1. The wTSs with $|Q| = 9$ is depicted in Figure 4. All other transition systems are generated randomly. The fourth and fifth column of Table 1 show the total number of iterations $n$ of Algorithm 2 required to find the first prefix and suffix part and the corresponding size of the constructed trees, respectively. The sixth column shows the time required until the first prefix and suffix part are detected, without executing the rewiring step. The runtime of NuSMV (Cimatti et al. 2002) or nuXmv (Cavada et al. 2014) are reported in the last column. The ‘M’ in the last column, means that NuSMV and nuXmv failed to build the model of the multi-robot system and, therefore, we could not proceed to the model-checking process. On the other hand, ‘F’ means that the model was built but the LTL formula is too long, with thousands of logical operators and atomic propositions $\pi_i^e$, to input it to NuSMV or nuXmv in a user-friendly way. Particularly, neither of NuSMV and nuXmv allow to define the LTL specifications in the more compact and symbolic form, discussed before, using the Boolean formulas $\xi$. Instead, the LTL formulas have to be expressed explicitly using the atomic propositions $\pi_i^e$.

For all case studies shown in Table 1, we consider the following LTL task

$$\phi = G(\xi_1 \rightarrow (\neg \xi_5 \land U \xi_2)) \land (\circ \circ \xi_1) \land (\circ \circ \xi_1) \land (\circ \circ \xi_4) \land (\neg \xi_4 \land \xi_2) \lor (\circ \circ \xi_4) \land (\circ \circ \xi_3) \land (\circ \circ \xi_3).$$

The LTL formula (39) is satisfiable if (i) always when $\xi_1$ is true, then at the next step $\xi_5$ should be false until $\xi_2$ becomes true; (ii) $\xi_1$ is true infinitely often; (iii) $\xi_3$ is true infinitely often; (iv) $\xi_4$ is true infinitely often; (v) $\xi_1$ is false until $\xi_5$ becomes true; (vi) $\xi_6$ is always false; (vii) $\xi_5$ is true infinitely often; and (viii) eventually either $\xi_4$ or $\xi_8$ are true. Recall that, in (39), $\xi_5$ is a Boolean formula in conjunctive or disjunctive normal form that is true depending on what atomic propositions $\pi_i^e$ that are true. For instance, for the planning problem with order $10^{21}$ in Table 1, $\xi_1$ is defined as

$$\xi_1 = (\pi_1^{\circ \circ} \lor (\pi_2^{\circ \circ} \land \pi_8^{\circ \circ} \land \pi_9^{\circ \circ}) \land \pi_8^{\circ \circ}) \land (\pi_1^{\circ \circ} \lor (\pi_2^{\circ \circ} \land \pi_8^{\circ \circ} \land \pi_9^{\circ \circ})),$$

which is true if (i) robot 1 is in region $r_{100}$; (ii) robot 8 is in one of the regions $r_{20}, r_{90}$, and $r_{111}$; and (iii) robot 9 is in one of the regions $r_{11}, r_{10}, r_{25}$, and $r_{35}$. All other
Table 1. Feasibility and scalability analysis: $|Q_B| = 21$

| $N$ | $|Q_B|$ | $|Q_P|$ | $n_{Pre1} + n_{Suf1}$ | $|Pre1| + |Suf1|$ | $Pre1 + Suf1$ | NuSMV/nuXmv |
|-----|------|------|----------------|----------------|-------------|-------------|
| 1   | 100  | $10^3$ | 28 + 28 | 180 + 54 | 0.5 + 0.2 (secs) | < 1 sec |
| 1   | 1000 | $10^3$ | 42 + 31 | 338 + 119 | 0.9 + 0.7 (secs) | < 1 sec |
| 1   | 10000| $10^4$ | 71 + 43 | 512 + 131 | 19.2 + 11.2 (secs) | M/M |
| 9   | 9    | $10^{10}$ | 36 + 37 | 373 + 83 | 0.78 + 0.29 (secs) | < 1 sec |
| 10  | 100  | $10^{21}$ | 31+ 31 | 289 + 101 | 0.7 + 0.4 (secs) | ≈ 1.5 sec |
| 10  | 1000 | $10^{21}$ | 34 + 27 | 309 + 82 | 2.1 + 1.7 (secs) | ≈ 50/40 sec |
| 10  | 2500 | $10^{25}$ | 41 + 32 | 367 + 142 | 5.94 + 6.4 (secs) | M/≈ 1860 sec |
| 10  | 10000| $10^{41}$ | 40 + 23 | 357+123 | 48.44 + 38.1 (secs) | M/M |
| 100 | 100  | $10^{200}$ | 49 + 39 | 421 + 81 | 2.5 + 0.8 (secs) | F/F |
| 100 | 1000 | $10^{200}$ | 30 + 38 | 254 + 110 | 12.6 + 24.1 (secs) | M/M |
| 100 | 10000| $10^{200}$ | 24 + 49 | 241 + 55 | 252.1 + 312.1 (secs) | M/M |
| 150 | 10000| $10^{200}$ | 29 + 87 | 382+530 | 365.91 + 1662.6 (secs) | M/M |
| 200 | 10000| $10^{200}$ | 42 + 49 | 453 + 276 | 678.1 + 1410.1 (secs) | M/M |

Boolean formulas $\xi$ are defined similarly. Similar LTL formulas are widely considered in the relevant literature as they can capture a wide range of robot tasks; see e.g., (Guo and Dimarogonas 2015). For instance, the subformula $((\Box \xi_1) \land (\Box \xi_2) \land (\Box \xi_3) \land (\Diamond \xi_4))$ can capture surveillance and data gathering tasks whereby the robots need to visit certain locations of interest infinitely often to collect data (Guo and Zavlanos, Michael M 2017; Smith et al. 2011). Also, it can capture intermittent connectivity tasks where the robots need to meet infinitely often at communication points to exchange their collected information (Kantaros, Yiannis and Zavlanos, Michael M 2017; Kantaros, Yiannis and Guo, Meng and Zavlanos, Michael M 2019). Specifically, in this task, the robots are divided into overlapping robot teams and all robots of each team need to meet at some locations of interest infinitely often at communication points that are optimally selected to exchange the information. Also, the term $\Box \neg \xi_0$ can capture obstacle avoidance (or collision avoidance) constraints that require the robots to avoid obstacles. Moreover, $G(\xi_1 \land (\neg \xi_2 \land U \xi_3))$ can be used to assign priorities to different subtasks. For instance, if $\xi_1$ requires the robots to visit food/water resources and $\xi_2$ requires the robots to deliver those resources to the end users. Then, the formula $G(\xi_1 \land (\neg \xi_2 \land U \xi_3))$ requires the robots to never re-gather/visit the resources without first delivering supplies to the users. The subformula $\neg \xi_2 U \xi_3$ can be interpreted similarly.

The LTL formula (38) corresponds to an NBA with $|Q_B| = 21$, $|Q_P| = 1$, $|Q_E| = 2$, and 125 transitions. Note that both final states of the NBA are feasible. Given the NBA, we construct the sets $\Sigma_{feas}^{\pi_8} \subseteq 2^{AP}$ in 0.1 seconds approximately for all case studies in Table 1. 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. Due to this modification, STyLuS$^*$ can only generate feasible plans; see the proof of Theorem 19.

Observe in Table 1 that STyLuS$^*$, NuSMV, and nuXmv have comparable performances when both a small number of robots and small enough wTSs are considered; see e.g., the planning problems with order $10^3$, $10^{10}$, and $10^{21}$. However, as either the number of robots or the size of the wTSs increases, STyLuS$^*$ outperforms both NuSMV and nuXmv in terms of both runtime and the size of problems that it can handle. For instance, both NuSMV and nuXmv fail to build the model when $N = 1$ and $|Q_B| = 10^4$, with average degree of the wTS equal to 42. As a result, it was impossible to synthesize a feasible motion plan with NuSMV or nuXmv using either symbolic techniques (Binary Decision Diagrams) or bounded model checking options. In fact, for this single-robot planning problem, NuSMV could build the model only for much sparser wTSs with average degree equal to 4. In this case, the model was built in 15 minutes and the verification process finished in almost 1 second. Moreover, SPIN, an explicit state model checker, also failed to build this model due to excessive memory requirements (Holzmann, Gerard J 2004); comparisons and trade-offs between NuSMV and an SPIN can be found in (Fraser, Gordon and Gargantini, Angelo 2009; Choi, Yunja 2007). On the other hand, for such sparse wTSs, STyLuS$^*$ synthesized a feasible prefix and suffix part in 16 and 14 seconds, respectively.

Moreover, observe in Table 1 that STyLuS$^*$ can synthesize feasible motion plans for large-scale multi-robot systems quite fast. For instance, it can synthesize in almost 35 minutes a feasible team plan $T$ for a team of $N = 200$ robots where each robot is modeled as a transition system with $|Q_B| = 10^4$ states and the PBA has $|Q_P| = 21 \times (10^4)^{200} \approx 10^{801}$ states. Finally, note that the runtimes reported in Table 1 can be significantly improved if STyLuS$^*$ is executed in parallel, as in (Kantaros and Zavlanos 2018b), and if the shortest paths required by each robot in the sampling step are constructed simultaneously (and not sequentially) across the robots, as also discussed in Section 3.5. Note also that the LTL formulas for the problems that involve more than 100 robots include thousands of logical operators and atomic propositions $\pi_j$ and, therefore, they are too long to input them to NuSMV and nuXmv in a user-friendly way. Also, recall that NuSMV and nuXmv can only generate feasible, and not optimal, plans; see also Section VI in (Kantaros and Zavlanos 2018a) for relevant comparative simulation studies.

In Figures 5 and 6, we examine the performance of STyLuS$^*$ with respect to the sparsity of the transition systems wTS, viewing them as graphs, for the planning problems of Table 1 with order $10^{31}$ and $10^{300}$, respectively.
Figure 5. Case Study I \(|N = 10\) robots, \(|Q_i| = 1000\) states: Figures 5(a) and 5(b) show the required number of iterations and total runtime required to construct the first feasible prefix-suffix plan, respectively.

Figure 6. Case Study I \(|N = 100\) robots, \(|Q_i| = 1000\) states: Figures 6(a) and 6(b) show the required number of iterations and total runtime required to construct the first feasible prefix-suffix plan, respectively.

Specifically, in Figure 5(a), we observe that as the wTSs become denser, the required number of iterations to find the first feasible prefix-suffix plan decreases, since the length of the shortest paths in the wTSs, that are used in the sampling process, decreases. We also observe in Figure 5(b) that as the wTSs become denser, the required number of iterations to find the first feasible prefix-suffix plan initially increases and eventually starts to decrease. The reason is that as the wTSs become denser, the required shortest paths are more difficult to find and, therefore, the computational cost per iteration of generating the states \(q_i^{\text{feas}}\) increases; see also Section 3.5. However, as the wTSs becomes fully connected the total runtime starts to decrease, since only few iterations are required to find a feasible plan, as discussed before. The same observations apply to Figure 6, as well, that pertains to the planning problem with order \(10^{301}\).

Switching between sampling densities Since optimality depends on the discovery of prefix plans to all feasible final states, bias can be used sequentially to every feasible final state \(q_i^{\text{feas}}\) to discover such plans, as also discussed in Remark 10. Once all feasible final states \(q_i^{\text{feas}}\) 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 \(Q_i\). Instead, biased sampling favors exploration towards the shortest paths that lead to the final states.

Note also that the proposed method finds initial feasible plans with small enough cost \(J(\tau)\), even though the rewiring step is not activated, due to the biased sampling that favors wTS transitions that correspond to the shortest NBA paths towards \(q_i^{\text{feas}}\). For instance, observe in Figure 9, which concerns the planning problem with order \(10^{10}\), that the cost of the best prefix part decreased from 415.48 to 314.58 meters within 5.5 hours. In total, 156 final states were detected and the first one was found after 0.42 seconds. On the other hand, using uniform sampling distributions from the beginning, the cost of the best prefix part decreased from 823.96 to 706.18 meters within 5.5 hours. In this case, 4 final states were detected and the first one was found after 1.34 hours.

Asymptotic Optimality Next, we validate the asymptotic optimality of STyLuS⁺ discussed in Theorem 15. Specifically, we consider a team of \(N = 2\) robots modeled by wTSs with \(|Q_i| = 16\) states and 70 transitions, including self-loops around each state; see Figure 7. The assigned task is expressed in the following temporal logic formula.

\[
\phi = \Box \Diamond (\pi_1^{\text{ref}} \land \Diamond (\pi_2^{14}) \land \Box (\neg \pi_1^n) \land \Box (\neg \pi_1^{11}) \land \Box (\neg \pi_2^{11}) \\
\land \Box (\pi_2^{14} \to \Box (\neg \pi_2^{14} \land \pi_1^n)) \land (\Diamond \neg \pi_2^{13}) \land (\Box \Diamond \pi_2^{10})
\]

(37)

In words, this LTL-based task requires (a) robot 1 to visit location \(r_9\), (b) once (a) is true robot 2 to visit location \(r_{14}\), (c) conditions (a) and (b) to occur infinitely often, (d) robot 1 to always avoid location \(r_9\), (e) once robot 2 visits location \(r_{14}\), it should avoid this area until robot 1 visits location \(r_9\), (f) robot 2 to visit location \(r_{12}\) eventually, (g) robot 2 to visit location \(r_{10}\) infinitely often and (h) both robots should always avoid the obstacle in region \(r_9\). The NBA that corresponds to the LTL specification (37) has \(|Q_B| = 24\) states with \(|Q_B^0| = 1\), \(|Q_B^1| = 4\), and 163 transitions.

Specifically, the set of final states is defined as \(B_F = \{6,9,11,13\}\) and all of them are feasible. Also, the state space of the corresponding PBA consists of \(\Pi_{i=1}^N |Q_i| |Q_B| = 6,144\) states, which is small enough so that the existing optimal control synthesis methods discussed in Section 2.1 can be used to find the optimal plan. Figure 8 shows the evolution of the cost of the best prefix part constructed by STyLuS⁺ with respect to iterations \(n\) and time. Observe in
are true infinitely often. Also, the LTL formula $\xi$ becomes true; (vi) $\xi$ is true until the next step $\xi_1$ becomes true; (vii) $\xi_2$ is true when $\xi_1$ is false. The LTL formula (38) is satisfied if (i) always when $\xi_1$ is true, then at the next step $\xi_1$ should be false until $\xi_2$ becomes true; (ii) $\xi_1$ is true infinitely often; (iii) $\xi_3$ is true infinitely often; (iv) $\xi_4$ is true infinitely often; (v) $\xi_5$ is false until $\xi_5$ becomes true; (vi) $\xi_6$ is always false; and (vii) $\xi_7$, $\xi_8$, and $\xi_9$ are true in this order infinitely often. Also, the LTL formula (38) corresponds to an NBA with $|Q_B|=59$, $|Q_B^0|=1$, $|Q_B^1|=8$, among which 6 final states are feasible, and 884 transitions. Given the NBA, we can construct the sets $F,w_{\text{feas}} \subseteq 2^{AP}$ in 0.14 seconds approximately for all case studies in Table 2.

Observe in Table 2 that the total number of iterations and the total runtime to detect the first feasible plan have increased compared to the corresponding planning problems in Table 1, due to the larger size of the NBA. Moreover, observe that both NuSMV and nuXmv can find the first feasible plan faster than STyLuS* when planning problems with few robots and small wTSs are considered; see, e.g., the problems with order $10^3$, $10^{10}$, $10^{21}$ in Table 2. However, similar to the previous case study, when the number $N$ of robots or the size of the wTSs increases, STyLuS* outperforms both model checkers both in terms of runtime and the size of the state-space that they can handle.

Remark 20. Self-loops in wTSs. Note that all wTSs considered in Tables 1-2 are constructed so that there are self-loops around each state, modeling in this way waiting actions of robots at all regions $r_j$. If self-loops are removed, the number of required iterations and respective runtime increase, as then the length of the shortest loop around every state increases (from its previous value of 1). For instance, for the planning problem with order $10^3$ in Table 1, after removing all self-loops, the total number of iterations required to detect the first feasible prefix and suffix part have increased from $n_{\text{iter}}=33$ and $n_{\text{safe}}=33$ to $n_{\text{iter}}=533$ and $n_{\text{safe}}=991$, respectively. Similarly, the corresponding runtimes have increased from 0.7 and 0.4 seconds to 7.1 and 10.6 seconds. On the other hand, NuSMV and nuXmv can synthesize feasible motion plans in almost 3 seconds. Adding or removing self-loops does not seem to affect much the runtime/scalability of NuSMV and nuXmv.

5.3 Comparison with off-the-shelf model checkers: Summary

The numerical experiments presented in Sections 5.1 and 5.2 show that STyLuS* outperforms both NuSMV and nuXmv for large and dense wTSs, whether they have self-loops or not, and regardless of the size of the NBA. On the other hand, for small and sparse transition systems with or without self-loops, NuSMV and nuXmv become faster than STyLuS*, as the size of the NBA increases. Also, for small and sparse transition systems without self-loops, the off-the-shelf model checkers find feasible paths faster than STyLuS*. Nevertheless, NuSMV and nuXmv can only find feasible paths while STyLuS* can detect the optimal plan with probability that converges to 1 exponentially fast.

Finally, note that NuSMV and nuXmv cannot handle collision avoidance constraints that require all robots to maintain a distance between them that is always at least equal to $R$ units; see e.g., (Kantaros et al. 2018d). The reason is that a cost function cannot be embedded in the transition systems that are provided as an input to them. Instead, STyLuS* can check if such safety properties are satisfied every time a sample is taken. Figure 11 shows the total time required by STyLuS* to detect the first final state for the planning problem with order $10^{31}$ when collision

![Figure 8. Case Study III: Figures 8(a) and 8(b) show the evolution the cost of the best prefix part constructed by STyLuS* with respect to iterations $n$ and runtime, respectively. Red diamonds denote a new final state detected by STyLuS*.
](image)

![Figure 9. Case Study I [N = 9 robots, |Q| = 9 states]: Comparison of the cost of the best prefix part constructed when biased (red line) and unbiased (blue line) from the beginning of STyLuS* are employed with respect to time. The cost of the best prefix part is reported every time a new final state is detected. Red diamonds and blue squares denote a new final state detected by STyLuS*.
](image)

Figure 8(a) that as $n$ increases, STyLuS* finds the optimal prefix part which has cost 5 meters. The required runtime to find the optimal prefix part is illustrated in Figure 8(b). On the other hand, the cost of the prefix part constructed by NuSMV and nuXmv is 8.8824 meters.

5.2 Scalability for larger and denser NBA

In this case study, we examine the performance of STyLuS* with respect to the number of the robots and the size of the wTSs, as in the previous case study, but also for a larger and denser NBA. The results are reported in Table 2 which has the same structure as Table 1. For all case studies shown in Table 2, we consider the following LTL task

$$\phi = G (\xi_1 \rightarrow (\neg \xi_1 U \xi_2) \land (\Box \neg \xi_1) \land (\Box \xi_3) \land (\Box \neg \xi_4) \land \neg (\xi_1 U \xi_5) \land (\Box \neg \xi_6) \land (\Box \neg \xi_7 \land (\Box \xi_8 \land (\Box \xi_9))))$$

(38)

The LTL formula (38) is satisfied if (i) always when $\xi_1$ is true, then at the next step $\xi_1$ should be false until $\xi_2$ becomes true; (ii) $\xi_1$ is true infinitely often; (iii) $\xi_3$ is true infinitely often; (iv) $\xi_4$ is true infinitely often; (v) $\xi_1$ is false until $\xi_5$ becomes true; (vi) $\xi_6$ is always false; and (vii) $\xi_7$, $\xi_8$, and $\xi_9$ are true in this order infinitely often. Also, the LTL formula (38) corresponds to an NBA with $|Q_B|=59$, $|Q_B^0|=1$, $|Q_B^1|=8$, among which 6 final states are feasible, and 884 transitions. Given the NBA, we can construct the sets $F,w_{\text{feas}} \subseteq 2^{AP}$ in 0.14 seconds approximately for all case studies in Table 2.
avoidance constraints are imposed. Specifically, observe that as $\tau$ increases the runtime of STYuLuS increases, since the number of samples that are rejected increases due to the imposed proximity restrictions.

### 5.4 The effect of biased sampling

Next, we illustrate the effect of introducing bias in the sampling process on the control synthesis performance. First, note that (Kantaros and Zavlanos 2018a) using uniform/unbiased sampling functions can synthesize plans for the synthesis problems considered in Tables 1-2 with

| $N$ | $|Q_n|$ | $|Q_r|$ | $n_{Pre1} + n_{Suf1}$ | $|PTS^q| + |\nu_{Out}^q|$ | Pre1+Suf1 | NuSMV/nuXmv |
|-----|--------|--------|----------------|----------------|-----------|--------------|
| 1   | 100    | $10^2$ | 54 $+$ 92      | 533 $+$ 274    | 2.18 $+$ 1.55 secs | $< 1$ sec |
| 1   | 1000   | $10^4$ | 78 $+$ 51      | 326 $+$ 252    | 1.84 $+$ 1.37 secs | $< 1$ sec |
| 1   | 10000  | $10^4$ | 150 $+$ 107    | 769 $+$ 364    | 19.2 $+$ 11.2 secs | M/M       |
| 9   | 9      | $10^{10}$ | 93 $+$ 27      | 400 $+$ 168    | 20.7 $+$ 18.9 secs | $< 1$ sec |
| 10  | 100    | $10^{24}$ | 51 $+$ 39      | 650 $+$ 239    | 2.1 $+$ 0.74 secs | $\approx 3/2$ secs |
| 10  | 25000  | $10^{10}$ | 36 $+$ 154     | 450 $+$ 404    | 3.9 $+$ 6.1 secs | $\approx 80/65$ secs |
| 100 | 10000  | $10^{10}$ | 61 $+$ 98      | 710 $+$ 516    | 10.4 $+$ 11.9 secs | M/M       |
| 100 | 100000 | $10^{10}$ | 39 $+$ 89      | 398 $+$ 1621   | 306.51 $+$ 1698.12 secs | M/M       |
| 200 | 200000 | $10^{10}$ | 39 $+$ 112     | 526 $+$ 1864   | 498.12 $+$ 3606.61 secs | M/M       |

**Figure 10.** Figures 10(a)-10(c) and Figures 10(d)-10(f) refer to the case where sampling is biased towards $q_{fibes}^q = 6$ and $n_{fibes}^q = 9$, respectively. Illustration of the probabilities $P(Y \geq K)$ and $\Pi_{fibes}(q_{fibes}^q)$ when biased and unbiased sampling is employed for various choices of $q_{fibes}^q$ and $n_{fibes}^q$. The probability $P(Y \geq K)$ is approximated as $P(Y \geq K) \approx \epsilon_{fibes}(n_{fibes}^q)/100$, where $\epsilon_{fibes}(n_{fibes}^q)$ is the number of experiments in which all states of the given feasible prefix path $p_j$, $j \in \{1, 2, 3\}$ of length $K$ that connects $q_{fibes}^q$ to the root are added to the tree. The probability $\Pi_{fibes}(q_{fibes}^q)$ is approximated as $\Pi_{fibes}(q_{fibes}^q) \approx \epsilon_{fibes}(n_{fibes}^q)/100$, where $\epsilon_{fibes}(n_{fibes}^q)$ is the number of experiments in which the final state $q_{fibes}^q \in Q_{fibes}$ is added to the tree. The probabilities $P(Y \geq K)$ and $\Pi_{fibes}(q_{fibes}^q)$ were estimated from a total of 100 experiments for every value of $n_{fibes}^q$. To generate the results of Figure 10 the rewiring step is deactivated.
order of states up to $10^{10}$. These are more states than what existing optimal control synthesis algorithms can handle – see (Kantaros and Zavlanos 2018a) – but orders of magnitudes less than the states that Stylus can solve.

In what follows, we compare the performance of Stylus for biased and unbiased sampling; see Figure 10. In this case study, we consider the same task planning problem considered in (37) and Figure 7. Note that the considered planning problem is small enough so that the comparisons with uniform sampling functions can be provided. Particularly, in Figure 10, we show the probabilities $P(Y \geq K)$ and $\Pi_{\text{succ}}(q^K_B)$ defined in Theorem 16 for biased and unbiased sampling for various choices of $n_{\text{max}}$ and final states $q^K_B$. Recall that (i) $P(Y \geq K)$ captures the probability that all states that belong to a given feasible path $p$ of length $K$ that connects $q^p_F$ to the root have been added to the tree within $n_{\text{max}}$ iterations and (ii) $\Pi_{\text{succ}}(q^K_B) \geq P(Y \geq K)$, using either biased or unbiased sampling. Also, notice that as $n_{\text{max}}$ increases, both probabilities $P(Y \geq K)$ and $\Pi_{\text{succ}}(q^K_B)$ converge to 1, using either biased or unbiased (uniform) sampling density functions, as expected due to Theorem 16.

In Figures 10(a)-10(c) and 10(d)-10(f) the density functions are biased to the feasible final states $q^K_B$ $= 6$ and $q^K_B = 9$, respectively. In Figures 10(a) and 10(d), the state $q^K_B$ is selected as $q^K_B = ((13, 10), 6) \in Q^K_F$ that is connected to the root through a feasible prefix part $p_1$ with length $K = 13$. In Figures 10(b) and 10(e) the state $q^K_B$ is selected as $q^K_B = ((6, 10), 9) \in Q^K_F$ that is connected to the root through a feasible prefix part $p_2$ with length $K = 6$. In fact, $p_2$ corresponds to the optimal prefix part. In Figures 10(c) and 10(f) the state $q^K_B$ is selected as $q^K_B = ((13, 15), 13) \in Q^K_F$ that is connected to the root through a feasible prefix part $p_3$ with length $K = 7$. Note that the feasible prefix paths $p_1$, $p_2$, and $p_3$ and the respective final states $q^K_B$ are randomly selected.

Observe first that in Figure 10, as $n_{\text{max}}$ increases, both probabilities $P(Y \geq K)$ and $\Pi_{\text{succ}}(q^K_B)$ converge to 1 for both biased and uniform (unbiased) sampling, as expected by Theorem 16. This shows probabilistic completeness of Stylus. Moreover, observe in Figure 10(a) that for small $n_{\text{max}}$, when sampling is biased towards $q^K_B = 6$, the probability $\Pi_{\text{succ}}(q^K_B)$ associated with the NBA final state 6 is larger compared to the case of uniform sampling, as expected; see also Remark 17. The same also holds for Figure 10(b) although sampling is not biased towards the NBA final state 9. Note, however, that this is a problem-specific behavior that depends on the structure of the NBA, and cannot be generalized. On the other hand, in Figure 10(c), the probability $\Pi_{\text{succ}}(q^K_B)$ associated with NBA final state 13 is larger for uniform sampling compared to sampling that is biased towards $q^K_B = 6$, as expected; see also Remark 17. The same observations are also made in Figures 10(d)-10(f). Observe also that when sampling is biased towards $q^K_B = 6$, the probability $\Pi_{\text{succ}}(q^K_B)$ associated with the NBA final state 6 in Figure 10(a) is higher compared to the one in Figure 10(b). The reason is that the probability $\Pi_{\text{succ}}(q^K_B)$ in Figures 10(a) and 10(b) is associated with NBA final state 6 but sampling is steered towards $q^K_B = 6$ and $q^K_B = 9$, respectively. Moreover, notice in Figures 12(a) and 12(b) that when biased sampling is employed, feasible prefix paths with smaller length $K$ are detected compared to the case of uniform sampling, as also discussed in Remark 17. Finally, note that for a given $n_{\text{max}}$, the size of the resulting trees is larger when uniform sampling is employed. For instance, for $n_{\text{max}} = 1000$, the tree consists of 4000 and 2000 nodes in average using uniform and biased sampling, respectively. The reason is that uniform sampling explores more states in the PBA while biased sampling favors exploration in the vicinity of the shortest paths, in terms of hops, to the final states.

6 Conclusion

This paper proposed a new optimal control synthesis algorithm for multi-robot systems with temporal logic specifications. We showed that the proposed algorithm is probabilistically complete, asymptotically optimal, and converges exponentially fast to the optimal solution. Finally we provided extensive comparative simulation studies showing that the proposed algorithm can synthesize optimal motion plans from product automata with state-spaces hundreds of orders of magnitude larger than those that state-of-the-art methods can manipulate.
Appendix: Proofs of Propositions

A.1 Proof of Proposition 13

Part (i) holds, since \( p_{rand}^n > \epsilon \), by assumption, for all \( n \geq 1 \). Part (iii) trivially holds by definition of \( f_{rand}^n \) in (7). As for part (ii) observe that \( \max(1/|D_{min}|, 1/|P_N^\omega|, |D_{min}|) \geq 1/|\{Q\}_P| \), since \( D_{min}^n \subseteq V_n \subseteq Q_n \). Combining these two observations, we conclude that \( f_{rand}^n(q_P|v_P^n) \geq \min(f_{rand}^{n+k}, 1 - p_{rand}^{n+k})/|\{Q\}_P| \geq \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| > 0 \). Therefore, there exists an infinite sequence \( \left\{ g_{rand}^n(q_P|v_P^n) \right\}_{n=1}^\infty = \{\min(\epsilon, 1 - \epsilon)/|\{Q\}_P|\}_{n=1}^\infty \) such that \( f_{rand}^n(q_P|v_P^n) \geq g_{rand}^n(q_P|v_P^n) \) and \( \sum_{n=1}^\infty g_{rand}^n(q_P|v_P^n) = \infty \), since \( \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| \) is a strictly positive constant term completing the proof.

A.2 Proof of Proposition 14

Part (i) holds since \( f_{new,i}^n(q_i|d_i^{rand,n}) \) is bounded away from zero on \( \{Q\}_{PTS}^i(q_i|d_i^{rand,n}) \), for all robots \( i \) as \( p_{new}^n > \epsilon \), for all \( n \geq 1 \), by assumption. Part (iii) trivially holds by definition of \( f_{new}^n \). As for part (ii), following the same logic as in the proof of Proposition 13, we can show that for the functions \( f_{new,i}^n \) in (11), (13), and (14), it holds that \( f_{new,i}^{n+k}(PTS|PTS^{n+k}) \geq \min(f_{new,i}^{n+k}, 1 - f_{new,i}^{n+k})/|\{Q\}_P| \geq \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| \). Thus, we get that \( f_{new,i}^n(PTS|PTS^{n+k}) = \Pi_{N=1}^\infty f_{new,i}^n(q_i|d_i^{rand,n}) \geq \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| \). For any fixed and given node \( PTS \), \( PTS^{n+k} \) there exists an infinite sequence \( h_{new,i}^{n+k}(PTS|PTS^{n+k}) = \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| > 0 \). Therefore, for any fixed and given node \( PTS \), \( PTS^{n+k} \) there exists an infinite sequence \( h_{new,i}^{n+k}(PTS|PTS^{n+k}) = \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| \). For any fixed and given node \( PTS \), \( PTS^{n+k} \) there exists an infinite sequence \( h_{new,i}^{n+k}(PTS|PTS^{n+k}) = \min(\epsilon, 1 - \epsilon)/|\{Q\}_P| > 0 \) is a strictly positive constant term completing the proof.

References


