Sampling-based Reactive Synthesis for Nondeterministic Hybrid Systems

Qi Heng Ho, Zachary N. Sunberg, and Morteza Lahijanian

***Abstract*— This paper introduces a sampling-based strategy synthesis algorithm for nondeterministic hybrid systems with complex continuous dynamics under temporal and reachability constraints. We model the evolution of the hybrid system as a two-player game, where the nondeterminism is an adversarial player whose objective is to prevent achieving temporal and reachability goals. The aim is to synthesize a winning strategy – a reactive (robust) strategy that guarantees the satisfaction of the goals under all possible moves of the adversarial player. Our proposed approach involves growing a (search) game-tree in the hybrid space by combining sampling-based motion planning with a novel bandit-based technique to select and improve on partial strategies. We show that the algorithm is probabilistically complete, i.e., the algorithm will asymptotically almost surely find a winning strategy, if one exists. The case studies and benchmark results show that our algorithm is general and effective, and consistently outperforms state of the art algorithms.**

I. Introduction

Reactive synthesis is the problem of generating a control strategy that enables a robot to *react* to uncertainties on the fly to guarantee satisfaction of complex requirements. The requirements are often expressed in *temporal logic* (TL) such as *linear TL* (LTL) [1] for specification on the sequence of events and *metric interval TL* (MITL) and *signal TL* (STL) [2] for dense-time specifications. Although reactive synthesis is known to be hard, it is an active area of research due to its applications in *safety-critical* and *time-critical* systems such as autonomous driving, search-and-rescue, and surgical robotics [3]. Reactive synthesis is often studied in the discrete setting, where the dynamics are abstracted to a finite model [4]–[7]. For complex and uncertain dynamics with dense-time requirements, however, such abstractions are either unavailable or too coarse (in both space and time), preventing accurate analysis and completeness guarantees. This work focuses on reactive synthesis for such systems and aims to develop an algorithm with correctness and completeness guarantees.

A powerful and expressive model that represents complex robotic systems under uncertainty with TL specifications is *Nondeterministic Hybrid Systems* (NHS) [8]. NHS allows both continuous and discrete dynamics via discrete modes that contain continuous dynamics and discrete switching between the modes. An NHS can be viewed as the composition of a robot's continuous dynamics with its environment and TL requirements, where the continuous dynamics and dense-time requirements are captured within each discrete mode, and switching between modes either represents changes in the dynamics or environment, or capture the requirements on the sequence of events. In this view, the satisfaction of the requirements reduces to a reachability objective for the NHS, and hence, the problem becomes synthesis of a strategy that guarantees reachability under all uncertainties.

Fig. 1: Search-and-rescue scenario. Left: a robot is navigating a building with unknown room blockage and human states. Right: computed robot strategy that reacts to all possibilities of room blockages and human presence in each room. Each colored trajectory is a possible evolution of the reactive strategy.

Example 1. *Consider a search-and-rescue scenario, where a building is on fire, in which there may be a trapped human that needs to be rescued, as depicted in Fig. 1. To aid the search for the human, a rover with second-order car dynamics is tasked with searching and mapping every room of the building within* 2 *minutes. If a room is unblocked, the rover must search it in* 10*s. If the human is found, the robot must protect the human until the rescue team arrives. If all rooms are found to be blocked or no human is found, the rover must go to the exit zone in green within* 20*s to report the map to the rescue team. In this example, the uncertainty is in the environment, where the rooms may or may not be blocked or contain a human.*

Our approach is based on a game-theoretic interpretation of the problem. We model nondeterminism as an adversarial player that attempts to prevent the robot from achieving its temporal and reachability objectives. This game is in the hybrid space, which is infinite and uncountable. Therefore, finite game techniques that are common in abstraction-based approaches are not applicable here. Instead, we aim to synthesize a strategy directly in the hybrid state space by iteratively constructing a game tree and exploring "promising" strategies. This however poses two main challenges: (i) construction of the game tree with nonlinear continuous dynamics and (ii) the *exploration-exploitation* dilemma. To deal with challenge (i), we take inspirations from the tremendous success of sampling-based techniques in motion planning. To

This work was supported by Strategic University Research Partnership (SURP) grants from the NASA Jet Propulsion Laboratory (JPL) (RSA 1688009 and 1704147).

The authors are with the Department of Aerospace Engineering Sciences, University of Colorado Boulder, CO, USA {firstname.lastname}@colorado.edu

overcome challenge (ii), we adapt multi-armed bandit methods developed for planning under uncertainty problems. We devise an algorithm, called *Sampling-based Bandit-guided Reactive Synthesis* (SaBRS), that uses a novel bandit-based method to select a strategy in the game tree for expansion and employs random sampling to grow this strategy. We show that the algorithm is probabilistically complete, i.e., the algorithm asymptotically almost surely finds a strategy that guarantees satisfaction of the objectives if one exists.

The contributions of this paper are threefold: (i) SaBRS, a novel sampling-based reactive synthesis algorithm for NHS, (ii) proof of probabilistic completeness of SaBRS, (iii) a series of benchmarks and case studies, including real robot demonstrations, that illustrate the generality and efficacy of SaBRS. Results show that SaBRS consistently finds solutions up to an order of magnitude faster than the state of the art. To the best of our knowledge, this is the *first* probabilistically-complete reactive synthesis algorithm for NHS.

A. Related Work

Sampling-based algorithms have emerged as powerful tools for kinodynamic motion planning for robotic systems with complex temporal goals and nonlinear or hybrid dynamics [9]–[12]. These techniques are typically used for deterministic systems and, only recently, extended to stochastic models [13]–[17]. Nondeterminism, where no probability distributions are available, is often not considered. In this work, we utilize sampling-based methods to achieve reachability objectives for nonlinear hybrid systems with nondeterminism.

A common approach to handle nondeterminism is to model it as an adversarial player in a game setting. Reactive synthesis is based on this view and typically studied in discrete games [3]–[5]. When applied to continuous systems, however, they require finite abstraction, which is difficult to obtain for complex dynamics. In the continuous domain, techniques based on Hamilton-Jacobi analysis, contraction theory, and counterexample-guided synthesis have been employed to provide robust controllers with guarantees on system behavior [18]–[22]. However, these methods are designed for continuous disturbance and cannot handle discrete nondeterminism. In this work, we formulate the problem as a two-player minimax game directly in the hybrid space and propose an algorithm for efficient reactive synthesis with formal guarantees.

The work that most closely relates to ours is [23]. It considers the same problem and proposes a two-phase sampling-based strategy planner that performs exploration in the first phase and strategy improvement in the second. The algorithm is highly dependent on the quality of strategies found in the first phase, and since it cannot return to the first phase, it is incomplete. In this work, we develop a probabilistically complete algorithm that continually improves strategies.

II. Problem Formulation

In this work, we consider complex robotic systems under uncertainty with temporal and reachability objectives. Specifically, we focus on uncertanties due to nondeterministism or discrete disturbances. A general modeling framework that allows for accurate representation of such systems and objectives simultaneously is nondeterministic hybrid systems.

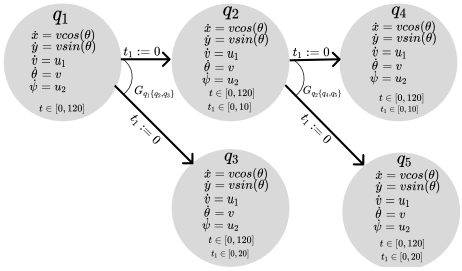


Fig. 2: NHS for single room version of Example 2.

Definition 1 (NHS)**.** *A* nondeterministic hybrid system *(NHS) is a tuple* $H = (S, s_0, U, I, F, E, G, J, S_{goal}, R)$*, where*

- $S = Q \times X$ *is the hybrid state space which is the Cartesian product of a finite set of discrete modes,* $Q = \{q_1, q_2, \dots, q_m\}$ *for* $m \in \mathbb{N}$*, with a set of mode-dependent continuous state spaces* $X = \{X_q \subseteq \mathbb{R}^{n_q} \mid q \in Q\}$*, where* $n_q \in \mathbb{N}$*,*
- $s_0 \in S$ *is the initial state,*
- $U = \{U_q \subset \mathbb{R}^{m_q} \mid q \in Q\}$*, where* $m_q \in \mathbb{N}$*, is the set of mode-dependent control spaces,*
- $I = \{I_q : X_q \to \{\top, \bot\} \mid q \in Q\}$ *is the set of invariant functions,*
- $F = \{F_q : X_q \times U_q \to X_q \mid q \in Q\}$ *is the set of flow functions that describe the continuous dynamics of the robot in each mode,*
- $E \subseteq Q \times Q$ *is the discrete transitions between modes,*
- $G = \{G_{qQ'} : X_q \to \{\top, \bot\} \mid Q' \in 2^Q, \forall q' \in Q', (q, q') \in E\}$ *is a set of guard functions that, given hybrid state* (q, x)*,* $G_{qQ'}(x) = \top$ *triggers a transition from mode* q *to a mode in* Q'*. If* $|Q'| = 1$*, the transition is deterministic; otherwise, it is nondeterministic,*
- $J = \{J_{qq'} : X_q \to X_{q'} \mid (q, q') \in E\}$ *is a set of jump functions that, once a transition* (q, q') *is triggered by a guard at state* $x \in X_q$*,* $J_{qq'}(x) \in X_{q'}$ *resets the continuous state in mode* q'*,*
- $S_{goal} \subseteq S$ *is a set of goal states that define the reachability objective, and*
- $R : S \to \{\top, \bot\}$ *is the reachability indicator function, where* $R(s) = \top$ *if* $s \in S_{goal}$*, and* $R(s) = \bot$ *otherwise.*

In this definition of NHS, nondeterminism is in the discrete transition. The temporal constraints are typically encoded in the invariant I and guard G functions, and the reachability objective is explicitly defined by set S_{goal} and its indicator function R. The evolution of the NHS is determined by a control strategy, which picks control actions for the system.

Definition 2 (Control Strategy)**.** *A control strategy* $\pi : S \to \cup_{q \in Q} U_q$ *is a function that, for hybrid state* $s = (q, x)$*, chooses an input control* $u \in U_q$*.*

Under control strategy π, the evolution of H is as follows. From initial state $s_0 = (q_0, x_0)$, the continuous component, x_t, of hybrid state $s_t = (q_0, x_t)$ evolves according to dynamics $\dot{x} = F_q(x_t, \pi(q_0, x_t))$ until a guard in mode q_0

is triggered. Let τ denote the time that the system first hits the guard, i.e., $G_{q_0Q'}(x_\tau^-) = \top$. Then, the system's discrete dynamics (mode) makes a transition to $q' \in Q'$ nondeterministically, and the continuous state is updated according to the jump function, i.e., $x_\tau^+ = J_{q_0q'}(x_\tau^-)$. In mode q', the system's continuous component evolves according to flow $F_{q'}$ from x_τ^+. This process continues until either the invariant function I becomes false, which indicates that temporal constraints are violated, or the reachability indicator function R becomes true, which indicates that the reachability objective is satisfied.

Note that the NHS in Def. 1 assumes deterministic continuous dynamics in every mode. It is the discrete dynamics (switching between the modes) that is subject to uncertainty, i.e., nondeterministic transitions.

Example 2. *The NHS that models a simplified version of Example 1 (Fig. 1) is shown in Fig. 2. Time parameters t (global clock) and t_1 (local clock) are added to the continuous state x. Note that t_1 resets at every discrete transition; hence, the temporal constraints on the system are captured in the invariant $I_{q.}$ in each mode. The positions that enable the robot to observe the status of the room door in mode q_1 represents a guard region that triggers a nondeterministic transition for closed (mode q_3) or open (mode q_2) door. Searching the room represents a guard region that transitions the system to the next mode with a trapped human (mode q_4) or no human found (mode q_5). These are nondeterministic guards because the status of the door/room is unknown. By reaching the green region in mode q_3 and q_5 or finding a human in mode q_4, R becomes true, which satisfies the timed reachability objective.*

To guarantee existence and uniqueness of solution (trajectory) in each mode and enable completeness analysis, we assume the flow and jump functions are Lipschitz continuous.

Assumption 1 (Lipschitz Continuity)**.** *For every mode $q \in Q$, flow function F_q is Lipschitz continuous in both continuous state and control, i.e., there exists constants $L_x, L_u > 0$ such that $\forall x_1, x_2 \in X_q$ and $\forall u_1, u_2 \in U_q$,*

$$\|F_q(x_1,u_1) - F_q(x_2,u_2)\| \le L_x\|x_1 - x_2\| + L_u\|u_1 - u_2\|.$$

Further, for every transition $(q, q') \in E$, jump function $J_{qq'}$ is Lipschitz continuous in the continuous state, i.e., $\forall x_1, x_2 \in X_q$, there exists a constant $K_x > 0$ such that

$$\|J_{qq'}(x_1) - J_{qq'}(x_2)\| \le K_x\|x_1 - x_2\|.$$

While Assumption 1 guarantees that the continuous state trajectories are unique in each mode given a control strategy π, multiple hybrid state trajectories are still possible due to nondeterminism in the guards, i.e., nondeterminism in the discrete dynamics. In this work, we seek control strategies that are robust to these nondeterministic possibilities. That is, the control strategy guarantees the completion of reachability and temporal objectives by considering all possible outcomes. Such strategies are called winning.

Definition 3 (Winning Control Strategy)**.** *For NHS H, control strategy π^* is* winning *if every hybrid state trajectory induced by π^* terminates in S_{goal}.*

In this work, our goal is to find a winning control strategy.

Problem 1 (Reactive Synthesis)**.** *Given a nondeterministic hybrid system H as in Def. 1 with initial state s_0 and a goal set $S_{goal} \subseteq S$, synthesize a winning control strategy π^* that guarantees reaching S_{goal} from s_0.*

Remark 1. *The (robust) hybrid system reachability problem formulated in Problem 1 captures a large class of uncertain systems with (finite) TL (e.g., LTLf [24], co-safe LTL [25], and MITL and STL [2]) objectives, where the hybrid system is the Cartesian product of an uncertain continuous system with the automaton that represents the TL objectives.*

III. Background

A. Game Trees, AND/OR trees, and Strategies

To approach Problem 1, we use the concept of *game trees*. A game tree is a tree $\mathcal{T}$ whose nodes and edges represent game venue positions and game moves, respectively [26]. At each node, a set of inputs are available. Each node-input pair results in a set of children in the tree. For a tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ with a set of nodes $\mathcal{N}$ and edges $\mathcal{E}$, we denote by $\mathcal{E}(n)$ the set of child nodes of $n \in \mathcal{N}$.

An AND/OR tree $\mathcal{T}$ models a game tree as a two-player min-max game [26]. The players are MIN and MAX. The position resulting from MIN and MAX moves are represented in the tree by OR and AND nodes, respectively. Moves of the game proceed in strict alternation between MIN and MAX until no further moves are allowed by the rules of the game. After the last move, MIN receives a cost which is a function of the final position. The objective of MIN is to minimize the cost, while MAX's goal is to maximize the cost.

Definition 4 (Subtree)**.** *$\mathcal{T}_{sub} = (\mathcal{N}_s, \mathcal{E}_s)$ is a* subtree *of an AND/OR tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ if the following conditions hold:*

- *$\mathcal{N}_s \subseteq \mathcal{N}$ and $\mathcal{E}_s \subseteq \mathcal{E}$,*
- *$n \in \mathcal{N}_s$ is the root of $\mathcal{T}_{sub}$ if n is the root of $\mathcal{T}$,*
- *$|\mathcal{E}_s(n)| = 1$ if n is an OR node and $\mathcal{E}(n) \neq \emptyset$, i.e., only one move of the MIN player is available in $\mathcal{T}_{sub}$,*
- *$\mathcal{E}_s(n) = \mathcal{E}(n)$ if n is an AND node, i.e., all the moves of the MAX player are available in $\mathcal{T}_{sub}$.*

Definition 5 (Strategy)**.** *A strategy over a game tree is a mapping from a node to an element of the input set available at the node. A strategy can be represented as a subtree of an AND/OR tree. We refer to this as a* strategy subtree*.*

In a *reachability* game, the objective is to reach a set of target positions $T \subseteq \mathcal{N}$. Then, after the last move, the MIN player (root node of the AND/OR tree) is penalized for having leaf nodes outside T through the cost function. For a given strategy subtree, if all the leaf nodes are in T, the root gets a zero cost; otherwise, the cost is strictly positive.

Definition 6 (Winning Strategy)**.** *A strategy over a game tree is called* winning *if the root node of its AND/OR tree representation has a cost of zero.*

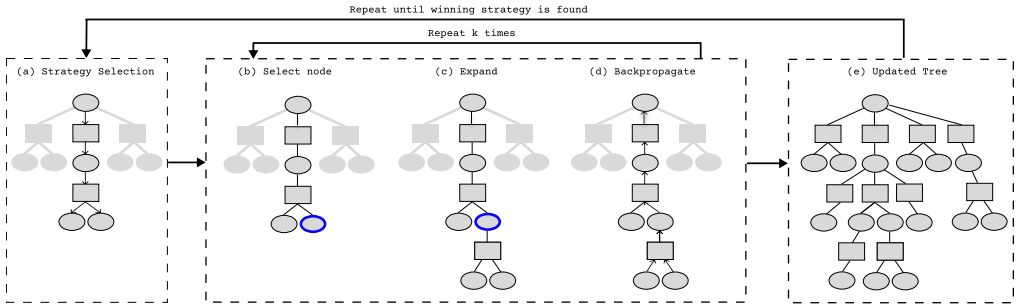


Fig. 3: Illustration of SaBRS algorithm. Circles represent OR (MIN player) nodes, while squares represent AND (MAX player) nodes.

B. Multi-Armed Bandits and Upper Confidence Bounds

In the multi-armed bandit decision-making problem, an agent is presented with multiple arms (actions). When the agent chooses an arm (action), it receives a cost according to a function that is hidden from the agent. The agent's goal is to minimize cost. Since the agent does not know how cost is generated, it needs to trade-off between *exploration* of actions and *exploitation* of the action that seems the least costly.

The agent adopts a policy on how to choose actions. The regret of a policy is the loss caused by not always choosing the best action. As the agent pulls more arms over time, it gains more information; hence, changing the regret of its policy over time. A policy is said to *resolve* the exploration-exploitation tradeoff if its regret growth rate is within a constant factor of a theoretical lower bound [27].

A theoretically sound algorithm whose finite-time regret is well-studied is the *Upper Confidence Bound* (UCB1) algorithm [27]. UCB1 resolves the exploration-explotation tradeoff by using a bias term. Specifically, the UCB1 defines a cost for choosing action a according to

$$\text{UCB}(a) = \hat{C}(a) - e\sqrt{2\ln(N)/N_a}, \tag{1}$$

where $\hat{C}(a)$ is the current estimated cost of taking action a, N is the number of trials, and N_a is the number of times action a has been chosen. Exploration constant e is a non-negative constant that determines the relative ratio of exploration to exploitation, with $e = 1$ in [27]. Term $\sqrt{2\ln(N)/N_a}$ is known as a *bias* sequence that controls the probability that a is chosen as N increases. For real valued $\hat{C}(\cdot)$, as N increases, every action a is eventually taken since the bias term becomes a very large value.

IV. Sampling-based Bandit-guided Reactive Synthesis (SaBRS) Algorithm

In this section, we present our novel approach to Problem 1. Our method is based on modeling the NHS as a two-player game, where the nondeterminism is an adversarial player whose objective is to prevent the robot player from achieving the temporal and reachability goals. The game venue is the hybrid state space. We explore this game venue by combining sampling-based techniques with game-theoretic approaches. Specifically, we use a bandit-guided strategy selection method to decide on which parts of the game venue to explore, and use sampling for exploration.

Our algorithm iteratively grows a search tree in the hybrid state space of H. This is done by growing a search tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ rooted at initial state s_0 and extending the tree based on the semantics of H (details in Sec. IV-B). Each node $n \in \mathcal{N}$ of the tree is a tuple $n = \langle s, N, \mathbf{u} \rangle$, where s is a hybrid state, N is the number of times that node n has been visited, and $\mathbf{u}$ is the set of piecewise constant control inputs previously selected at n. We model this tree as an AND/OR tree, where the robot is the MIN player which chooses a control $(u,t) \in n.\mathbf{u}$ at node n, followed by the MAX player, which chooses a child of the node-control pair $(n,(u,t))$ with the highest cost. Intuitively, the MAX player models the nondeterminism in H. Then, a strategy subtree on the AND/OR tree is a (piecewise constant) control strategy in the hybrid space. The goal of MIN is to find a winning strategy subtree, which has all of its leaf nodes in S_{goal}. Computing a winning strategy on the game tree is therefore equivalent to synthesizing a (piecewise constant) winning control strategy, solving Problem 1. A major challenge is how to effectively extend the game tree in the game venue (exploration) such that it turns a "promising" strategy subtree to a winning strategy (exploitation), if one exists.

To construct this game tree efficiently, we propose a reactive synthesis algorithm that consists of two main components: strategy subtree selection and strategy expansion/improvement. The algorithm, called *Sampling-based Bandit-guided Reactive Synthesis* (SaBRS), is shown in Alg. 1 and depicted in Fig. 3. SaBRS uses a novel bandit-based action selection methodology to select a strategy in the AND/OR tree for expansion. Then, the strategy expansion component uses random sampling to grow this selected subtree. This alternation of selecting promising strategy subtrees and expanding on them combines the exploration-exploitation properties of bandit-based techniques at the strategy-level with the effectiveness of sampling-based motion planners, resulting in an efficient and probabilistically complete algorithm (see Sec. V). The algorithm terminates when a winning strategy is found.

A. Strategy Tree Selection with Upper Confidence Bounds

Since the search tree can become extremely large, we wish to bias our search towards the strategies that are promising, i.e., close to a winning strategy. Hence, in each iteration of planning, the algorithm evaluates and selects a strategy subtree π of the search tree as depicted in the Frame (a) of Fig. 3. The evaluation of a strategy subtree π is done based on a cost function that assigns to node n the cost

$$C^{\pi}(n) = 1 - \frac{|\mathcal{C}^{\pi}_{goal}(n)|}{|\mathcal{C}^{\pi}_{all}(n)|}, \tag{2}$$

Algorithm 1: SaBRS Algorithm

Input : NHS H, Initial state s_0, Expansion ratio k, Exploration constant e, Time limit T_{max}
Output: Winning Reactive Strategy π^* from s_0

1 $n_0 \leftarrow \langle s_0,0,\emptyset\rangle$
2 $\mathcal{T} = (\mathcal{N} \leftarrow \{n_0\}, \mathcal{E} \leftarrow \emptyset)$
3 **while** $\min_{(u,t)\in n.u} Q(n_0,(u,t)) > 0$ *and time* $< T_{max}$ **do**
4 $\quad \pi \leftarrow \texttt{UCB-ST}(n_0,e)$
5 $\quad$ **for** $j = 1 \rightarrow k$ **do**
6 $\quad\quad \pi \leftarrow \texttt{Explore}(\pi)$
7 $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \pi$
8 $\pi^* \leftarrow \texttt{UCB-ST}(n_0,0)$
9 **return** π^*

where $\mathcal{C}^{\pi}_{all}(n)$ is the set of all leaf nodes of the subtree π rooted at n, and $\mathcal{C}^{\pi}_{goal}(n) \subseteq \mathcal{C}^{\pi}_{all}(n)$ is the subset of nodes that are in S_{goal}. Intuitively, the cost of n under strategy π is the portion of leaf nodes that are not in S_{goal}. When $C^{\pi}(n) = 1$, no branches of subtree π from n end in S_{goal}, and when $C^{\pi}(n) = 0$, all the branches of the subtree end in S_{goal} from n, i.e., π is a winning strategy for n.

Remark 2. *It is important to note that various cost functions are possible in this framework, and the effectiveness of a cost function may be problem dependent. The only requirement of a cost function is that $C^{\pi}(n) = 0$ if π is a winning strategy from n, and $C^{\pi}(n) > 0$ otherwise.*

From (2), we define $\mathcal{Q}$*-cost* to be the cost for choosing an input (u,t) at node n and then following strategy π at subsequent nodes, i.e.,

$$\mathcal{Q}^{\pi}(n,(u,t)) = 1 - \frac{\sum_{n' \in children(n,(u,t))} |\mathcal{C}^{\pi}_{goal}(n')|}{\sum_{n' \in children(n,(u,t))} |\mathcal{C}^{\pi}_{all}(n')|}. \quad (3)$$

This $\mathcal{Q}$-cost allows us to evaluate the relative cost of each input (u,t) at node n. Note that when $\pi(n) = (u,t)$, the $\mathcal{Q}$-cost is equivalent to $C^{\pi}(n)$, i.e., $\mathcal{Q}^{\pi}(n,\pi(n)) = C^{\pi}(n)$. The minimization of $\mathcal{Q}$-cost at each node thus provides us with strategies that are seemingly closer to a winning strategy subtree. However, this leads us to the classical *exploitation-exploration dilemma*, since a strategy subtree with low cost may not always be the best strategy to choose, since a strategy may have a low cost but it may be difficult or impossible to be extended into a winning strategy subtree due to, e.g., one of its leaf nodes being stuck in a "dead end".

Therefore, we choose a strategy subtree by treating the control selection problem at each node as a separate multi-armed bandit problem to solve this exploration-exploitation tradeoff. To this end, we propose an adaptation of the UCB1 algorithm to be used in the context of strategy subtree selection. We call this new algorithm *UCB for Strategy Tree selection* (UCB-ST). The algorithm is shown in Alg. 2. It first initializes the chosen strategy tree π with the root node n_0 (Line 4 Alg. 1). From n_0, it selects control inputs $(u,t)_{sel}$ in the AND/OR tree according to the UCB1 criterion (Line 3 in Alg. 2) adapted from (1):

$$\text{UCB}(n,(u,t)) = \mathcal{Q}^*(n,(u,t)) - e\sqrt{2\ln(n.N)/n.N_{(u,t)}}, \quad (4)$$

where $\mathcal{Q}^*(n,(u,t)) = \min_{\pi} \mathcal{Q}^{\pi}(n,(u,t))$ is the optimal $\mathcal{Q}$-cost, and $n.N_{(u,t)}$ is the number of times the control input (u,t) has been selected at node n. All children nodes of $(n,(u,t)_{sel})$ are added to π, and control inputs for each children are again selected according to (4) (Lines 5-6). This process repeats until a strategy subtree π of $\mathcal{T}$ is obtained, which is when all the leaf nodes of a subtree are reached. UCB-ST allows the tree to grow in more promising parts of the tree, while still allowing for exploration of parts that seem less promising but might eventually lead to a winning strategy.

Algorithm 2: UCB-ST(n,e)

1 $\pi \leftarrow \{n\}$; $n.N = n.N + 1$
2 **if** n *is not a leaf node* **then**
3 $\quad (u,t)_{sel} \leftarrow \text{argmin}_{(u,t)\in n.u} \text{UCB}(n,(u,t))$ using (4)
4 $\quad \pi(n) = (u,t)_{sel}$
5 $\quad$ **for** $n' \in children(n,(u,t)_{sel})$ **do**
6 $\quad\quad \pi \leftarrow \pi \cup \texttt{UCB-ST}(n',e)$
7 **return** π

B. Strategy Improvement with Sampling-based Expansion

A strategy subtree π is extended in a sampling-based tree expansion manner, by growing the tree in the hybrid state space. This sampling-based expansion technique is inspired by motion planning algorithms. Pseudocode for our exploration algorithm is shown in Alg. 3 and depicted in Frames (b)-(d) of Fig. 3. In each iteration of exploration, a node n in π that has non-zero cost is first randomly sampled. Note that zero cost nodes already have a winning subtree, and do not need to be expanded further. Let $n.s = (q,x)$. Then, a control $u \in U_q$ and time duration t are randomly sampled, and the node's continuous state x is propagated by F_q. Any tree-based sampling-based planner that supports kinodynamic constraints (e.g., RRT or EST [9]) can be used in this step.

During propagation, the invariant I_q checks the validity of the generated trajectory, and reachability indicator R checks if the trajectory visits S_{goal}. If a guard $G_{qQ'}$ is enabled during propagation at continuous state x', propagation is terminated and, for every $q' \in Q'$, node $n' = \langle (q', J_{qq'}(x')),0,\emptyset\rangle$ is created. If no guard is triggered and I_q remains true for the entire duration t, only one new node is created. Then, the control (u,t) is added to the set of sampled controls $n.\mathbf{u}$, and the new leaf nodes are added to the tree. Finally, the cost of nodes in the strategy is updated by backpropagation using (2).

This expansion step is repeated k times for each strategy subtree selection iteration to ensure that sufficient exploration is performed for a strategy subtree. Note that k is an input to SaBRS. The choice of k can affect the efficiency of the algorithm. When k is small, the algorithm quickly switches between promising strategy subtrees but may not expand a strategy subtree sufficiently. On the other hand, when k is large, the algorithm expands a strategy subtree extensively, but switching between strategy subtrees becomes slower.

Remark 3. *SaBRS also works in an anytime fashion, i.e., when given a time limit, SaBRS returns a control strategy minimizing root node cost. This also allows us to find partial solutions for problems in which no winning strategy exists.*

Algorithm 3: `Explore`(π)

$n_{select} \leftarrow$ SampleAndSelect(π,s_{rand})
u_{rand},$t_{rand} \leftarrow$ SampleControl&Dur($U_{n.s.q}$,$(0,T_{prop}]$)
$\mathrm{N}_{new} \leftarrow$ Propagate(n_{select},u_{rand},t_{rand})
for $n_{new} \in \mathrm{N}_{new}$ **do**
  if *IsValidTrajectory*(n_{select},n_{new}) **then**
    $n_{select}.\mathbf{u} \leftarrow n_{select}.\mathbf{u} \cup \{(u_{rand},t_{rand})\}$
    Add vertex and edge to π
    Update costs in π by backpropagation
return π

V. Analysis

In this section, we prove probabilistic completness of our algorithm. Specifically, we consider the case that kinodynamic RRT [10] is used as the strategy expansion technique. We begin by defining the notion of probabilistic completeness for algorithms that solve Problem 1.

Definition 7 (Probabilistic Completeness)**.** *Given an NHS H as in Def. 1, an algorithm is probabilistically complete if, as the number of iterations $K \to \infty$, the probability of finding a winning control strategy, if one exists, approaches* 1.

Next, we prove that our strategy selection methodology repeatedly selects every strategy.

Lemma 1. *Given a search tree $\mathcal{T}$ and exploration constant $e > 0$, `UCB-ST` in Alg. 2 selects every strategy subtree infinitely often, i.e., as number of iterations approaches infinity, the number of times every strategy subtree of $\mathcal{T}$ is selected also approaches infinity.*

Proof. Consider a node $n \in \mathcal{T}$. $\mathcal{T}$ has a finite number of nodes for a finite number of SaBRS iterations. Input $(u,t) \in n.\mathbf{u}$ is selected according to UCB1 criterion (4), which weighs the current $\mathcal{Q}$-cost with the exploration term $e\sqrt{2\ln(n.N)/n.N_{(u,t)}}$. The exploration term increases if (u,t) is not selected. As N increases, another input $(u,t)' \neq (u,t)$ is always selected only if $\mathcal{Q}$-cost of $(u,t)'$ decreases at a rate faster than the increase in the exploration term of (u,t). However, the cost function is defined such that the minimum Q-cost is 0, and therefore, for $e > 0$, the exploration term for (u,t) eventually dominates the $\mathcal{Q}$-cost terms. Since the search tree is finite, $|n.\mathbf{u}|$ is finite, and given sufficient iterations, each $(u,t) \in n.\mathbf{u}$ is eventually selected. By induction, every strategy subtree is selected infinitely often. □

Similar to RRT, we require that a solution with a non-zero radius of clearance exists, as defined below.

Definition 8 (Clearance)**.** *For strategy π, let $Traj^{\pi}$ be the trajectories of H induced under π and $S^{\pi} = X^{\pi} \times Q^{\pi} \subset S$ be the set of hybrid states visited by $Traj^{\pi}$. Further, denote by $\mathbb{B}_{\delta}(x)$ the ball centered at point x with radius $\delta \geq 0$. Clearance δ^{π}_{clear} of π is the supremum of radius δ such that for every $s = (x,q) \in S^{\pi}$ and $\forall x' \in \mathbb{B}_{\delta}(x) \subset \mathbb{R}^{n_q}$, it holds that $I_q(x') = I_q(x)$ and $G_{qQ'}(x') = G_{qQ'}(x)$.*

Intuitively, δ^{π}_{clear} defines a tube around each $Traj^{\pi}$ that contains all hybrid trajectories that follow the same sequence of modes and their continuous components remain δ^{π}_{clear} close.

We now formally state the main result of our analysis, which is that SaBRS (Alg. 1) is probabilistically complete.

Theorem 1 (Probabilistic Completeness)**.** *Alg. 1 is probabilistically complete if there exists a winning strategy π^* with clearance $\delta^{\pi^*}_{clear} > 0$.*

Proof. Consider a winning strategy subtree π_{win} from s_0 to S_{goal} with clearance $\delta_{clear} > 0$. Let P be the set of all paths from s_0 to a leaf in S_{goal}. Now, a path $p_i \in P$ can be described by the sequence of nodes with states $p_i = s_0^{g_0} \ldots s_k^{g_1} \ldots s_l^{g_0} \ldots s_m^{g_1} \ldots s_{goal,p_i}$ ending in a node $s_{goal} \in S_{goal}$, where the superscript g_1 denotes that a guard is triggered, and g_0 denotes the guard is not triggered. Cover p_i with a set of balls of radius δ centered at $s_0^{g_0}, s_1, \cdots, s_{goal,p_i}$. We say that a path p_j *follows* another path p_i if each vertex of p_j is within the δ radius ball of p_i.

Since the flow functions F and jump functions J are Lipschitz continuous (Assumption 1), from [28, Theorem 2], we are guaranteed that RRT asymptotically almost surely finds a control trajectory from s_0 to S_{goal} that follows p_i when starting from a tree which contains s_0. From Lemma 1, `UCB-ST` will always eventually select any strategy subtree π_i of our search tree $\mathcal{T}$. Let t be the number of paths in π that uniquely follows a path $p_i \in P$. Assume that at step j, the selected subtree π contains $0 < t < |P|$ paths. As expansion iterations increase, π asymptotically almost surely finds a path from s_0 to S_{goal} that follows a new path $p_l \in P$ which it did not uniquely follow before. Hence, the new π_i^+ expanded from π_i now contains $t+1$ paths that unique follows paths in P. From Lemma 1, π_i^+ will eventually be selected again. By induction, $t \to |P|$ and a winning strategy is found. □

VI. Extensions to Improve Base Algorithm

We present three extensions that can improve efficiency of SaBRS without affecting its probabilistic completeness.

Warm Starting: The effectiveness of SaBRS relies on selecting promising strategies based on (2). However, when no branch of the search tree is in S_{goal} yet, the costs of strategies remain the same, namely 1. To improve the effectiveness of strategy selection, we first perform a *warm start* of SaBRS by performing sampling-based exploration on the full `AND/OR` tree for some fixed time, or until at least one leaf node is in S_{goal}. In our experiments, we observed that warm starting is especially useful in problems with longer horizons, where reaching a goal by a leaf node is difficult.

Strategy Tree Expansion Heuristics: During strategy expansion, SaBRS uses sampling-based exploration. This exploration has been shown to be greatly improved by heuristic guidance mechanisms, such as goal bias and trajectory bias [29]. The exploration step of our algorithm is general and amenable to such heuristics. An example of such a heuristic is *Guided Path-Generation*, introduced in [23] for Problem 1. It uses the search tree branches that end in a goal state to guide the expansion of nodes (see [23] for details). To maintain probabilistic completeness of our algorithm, we use Guided Path-Generation with low probability p and the

TABLE I: Benchmark planner performance results. We report the mean time with standard error, and success rate over 100 simulation trials, with best scores in bold. RRT in case study 2 and MCTS for both cases are excluded from the table since they had 0 success rate.

	Algorithm	Environment 1		Environment 2		Environment 3		Environment 4	
		Time (s)	Success (%)	Time (s)	Success (%)	Time (s)	Success (%)	Time	Success (%)
Case 1	RRT	299.0±0.0	3	299.5±0.0	2	300±0.0	1	–	0
	Planner in [23]	78.1±4.6	91	113.1±9.1	82	153.68±15.5	44	222.5±26.3	10
	SaBRS (Ours)	**4.2±0.5**	**100**	**8.3±1.1**	**100**	**23.04±3.6**	**99**	**57.9±6.5**	**93**
Case 2	Planner in [23]	98.1±6.5	96	157.1±12.0	68	208.9±18.3	32	251.1±49.0	4
	SaBRS (Ours)	**4.6±0.5**	**100**	**11.4±2.8**	**99**	**24.4±4.0**	**97**	**48.99±6.8**	**88**

random exploration of Alg. 3 with probability $1-p$. In our evaluations, we find that guided path generation greatly improves efficiency if the hybrid goal set has the same continuous component in the goal modes, i.e., $S_{goal} = Q_{goal} \times X_{goal}$, where $Q_{goal} \subseteq Q$ and $X_{goal} \subseteq \bigcap_{q \in Q_{goal}} X_q$.

Sub-Strategy Tree Selection: At each strategy selection iteration, Alg. 2 chooses a subtree π of the AND/OR tree $\mathcal{T}$. This allows us to expand promising strategies currently available in $\mathcal{T}$. $\mathcal{T}$ has a discrete set of available inputs, a subset of the continuous input space. During exploration, π is extended, increasing the discrete set of available inputs. When the search tree is very large and deep, it may become difficult to improve *smaller* strategies within a subtree. To ameliorate this issue, we probabilistically prune parts of a subtree to obtain a smaller strategy, with probability ρ. This effectively chooses 'no action' at an *OR* node, leading to a smaller strategy that may be improved to become a winning strategy more easily. In our experiments, we observe that this improves search for longer horizon problems where there are many leaf nodes that lead to "dead ends".

VII. Experiments

We evaluate the performance of SaBRS against 3 state-of-the-art algorithms, namely kinodynamic RRT, continuous space Monte Carlo Tree Search (MCTS-PW), and the motion planner in [23], in a series of benchmarks. We also provide several illustrative examples to show the generality of SaBRS. We implemented all algorithms in C++ using OMPL [30]. All computations were performed single-threaded on a nominally 3.60 GHz CPU with 32 GB RAM. For SaBRS, we used $k=5000$ and $e=0.0005$ for all planning instances.

A. Benchmarking Results

We first evaluate the algorithms on the benchmark problems proposed in [23]. The problems consist of two NHS models of a three-gear second-order car system that is subject to nondeterminism when shifting gears in four environments [23]. In Case Study 1, the system, when shifting from gear two to three, may mistakenly shift to gear one. In Case Study 2, there is an additional nondeterminism when the car shifts from gear three to one. In this case, the system may mistakenly change to gear two instead of one. A solution is a winning strategy that reaches the goal state over all possible gear transitions from the initial state. We refer the reader to [23] for details on the dynamics of each gear.

We conducted 100 trials for each of the four environments and two cases, and set a time limit of 300 seconds to find a solution per trial. The results are summarized in Table I. In Case Study 1, MCTS did not find any solutions. In Case Study 2, both RRT and MCTS did not find any solutions. It is evident from the poor performance of both RRT and MCTS that neither a pure sampling-based exploration method nor a pure heuristic search method is effective for finding winning strategies for NHS planning problems. Further, we see that SaBRS significantly outperforms the compared methods both in computation time (up to an order of magnitude) and success rate (up to 3×) for finding winning strategies. This suggests that the combination of the bandit-based game theoretic approach for strategy selection and sampling-based exploration is important for reactive synthesis under nondeterminism.

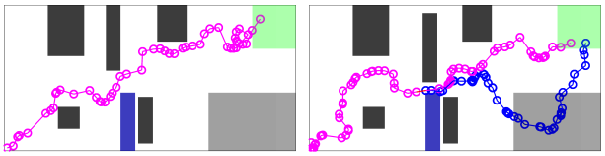
B. Robotic Charging System

Next, we showcase SaBRS's versatility to handle NHS with time constraints. Consider a robot with second-order car dynamics with a bounded motion disturbance at every time step, which is equipped with a closed-loop controller that is able to maintain the robot in a ball of radius r around its nominal plan. The robot is tasked with navigating to the charger within 2 minutes. If it goes over a water puddle, the robot needs to dry off on the carpet for 10 seconds before going to the charger. During online execution, the robot has perfect observation of its state. However, during offline planning, due to the motion disturbance, if the radius r ball around a nominal plan intersects with a water puddle, the robot may traverse the puddle during execution. Hence, the robot motion can be modelled as a NHS.

Figs. 4a and 4b show two examples with different obstacle configurations. In both cases, SaBRS finds a solution within 30 seconds. In Fig. 4a, when there is more space for the robot to traverse in the dry regions (in white), SaBRS synthesizes a strategy such that the entire ball stays within the white region without touching the water (in blue) and navigates to the charger (green). When the environment is more cluttered (Fig. 4b), such that the robot cannot reach the charger without guaranteeing that it does not go over the water puddle due to the uncertain ball, SaBRS plans a reactive strategy with two possible trajectories. If it does not get wet, it navigates directly to the charger. If it gets wet, it first goes to and stays on the carpet for 10 seconds before navigating to the charger.

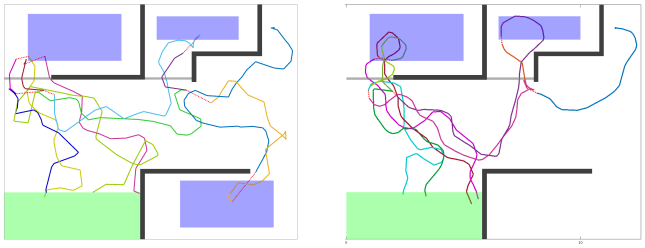
C. Search-and-rescue Scenario

Next, we show that SaBRS is effective for problems with complex temporal specifications. We consider variants of the search-and-rescue scenario of Example 1, in which each room's door state is fixed but unknown at planning time.



(a) Spaced Obstacles (b) Cluttered Obstacles

Fig. 4: SaBRS synthesizes strategies that account for nondeterminism in traversing the puddle due to motion uncertainty in the robotic charging example. The black, blue, grey and green regions represent obstacles, puddle, carpet, and charger, respectively.



(a) Human may be in all rooms. (b) Human in either of 2 rooms.

Fig. 5: SaBRS synthesizes strategies in the search-and-rescue example. Purple regions represent possible human locations.

Fig. 5 show example synthesized strategies (computed within $20s$) for a 3-room variant. Each possible trajectory in the reactive strategies is plotted with a different color, and a red dotted path segment indicates a nondeterministic mode transition. SaBRS successfully finds winning strategies that reacts based on if a door is blocked, and if the human is in each room. Finally, we compute and demonstrate strategies in real world experiments for a 2-room variant of the search-and-rescue scenario on a ClearPath Jackal robot. A demonstration video can be found at `https://youtu.be/Hy1rNGfX6Zg`.

VIII. Conclusion and Future Work

This paper considers the problem of computing reactive strategies for NHS under temporal and reachability constraints. We propose SaBRS, an algorithm that guarantees reachability under all possible NHS evolutions. SaBRS combines the effectiveness of sampling-based motion planning with bandit-based techniques. We show that SaBRS is probabilistically complete, and experiments demonstrate its effectiveness. For future work, we plan to extend SaBRS to explicitly include uncertainty in continuous dynamic, and extend SaBRS to also optimize an objective function.

References

[1] C. Baier and J.-P. Katoen, *Princ. of Mod. Check.* The MIT Press, 2008.

[2] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[3] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Syst.*, vol. 1, pp. 211–236, 2018.

[4] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *Proc. IEEE/RSJ Int. Conf. on Intel. Robots and Systems*, 2017, pp. 5326–5332.

[5] K. Muvvala, P. Amorese, and M. Lahijanian, "Let's collaborate: Regret-based reactive synthesis for robotic manipulation," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2022, pp. 4340–4346.

[6] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 1002–1028, 2020.

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

[8] J. Lygeros, K. Johansson, S. Simic, J. Zhang, and S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[9] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018.

[10] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int'l Journ. of Rob. Research*, vol. 20, no. 5, pp. 378–400, 2001.

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

[12] N. Wang and R. G. Sanfelice, "A rapidly-exploring random trees motion planning algorithm for hybrid dynamical systems," in *IEEE Conference on Decision and Control*, 2022, pp. 2626–2631.

[13] B. Luders, M. Kothari, and J. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," *AIAA guidance, navigation, and control conference*, 2010.

[14] Q. H. Ho, Z. N. Sunberg, and M. Lahijanian, "Gaussian belief trees for chance constrained asymptotically optimal motion planning," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2022, pp. 11 029–11 035.

[15] ——, "Planning with SiMBA: Motion planning under uncertainty for temporal goals using simplified belief guides," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2023, pp. 5723–5729.

[16] A. Theurkauf, Q. H. Ho, R. Ilyes, N. Ahmed, and M. Lahijanian, "Chance-constrained motion planning with event-triggered estimation," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2023, pp. 7944–7950.

[17] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *Int'l Journ. of Rob. Research*, vol. 33, no. 2, pp. 268–304, 2014.

[18] S. Herbert, M. Chen, S. Han, S. Bansal, J. Fisac, and C. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," *IEEE Conference on Decision and Control*, 2017.

[19] S. J. Leudo and R. G. Sanfelice, "Sufficient conditions for optimality in finite-horizon two-player zero-sum hybrid games," in *IEEE Conference on Decision and Control*, 2022, pp. 3268–3273.

[20] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2017, pp. 5883–5890.

[21] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annual Reviews in Control*, vol. 52, pp. 135–169, 2021.

[22] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proc. 18th Int. Conf. on Hybrid Syst.: Comp. and Control*, 2015, p. 239–248.

[23] M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "A sampling-based strategy planner for nondeterministic hybrid systems," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 2014.

[24] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Int'l Joint Conf. on Artif. Intel.*, 2013, pp. 854–860.

[25] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, 1999.

[26] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. USA: Prentice Hall Press, 2009.

[27] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, 05 2002.

[28] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Rob. and Autom. Letters*, 2019.

[29] C. Urmson and R. Simmons, "Approaches for heuristically biasing rrt growth," in *Proc. IEEE/RSJ Int. Conf. on Intel. Robots and Systems*, 2003, pp. 1178–1183.

[30] I. Şucan, M. Moll, and L. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Mag.*, vol. 19, pp. 72–82, 2012.