Submitted to IEEE Conference on Decision and Control (CDC) 2023.

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 view 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. The approach is based on growing a (search) game-tree in the hybrid space by combining a sampling-based planning method with a novel bandit-based technique to select and improve on partial strategies. We provide conditions under which the algorithm is probabilistically complete, i.e., if a winning strategy exists, the algorithm will almost surely find it. The case studies and benchmark results show that the algorithm is general and consistently outperforms the state of the art.**

I. Introduction

Reactive synthesis is the problem of generating a control strategy that enables a system 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 *mectric 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 syntehsis is often studied in the discrete setting, where the dynamics are abstracted to a finite model. For complex and uncertain dynamics with dense-time requirements, however, such abstractions are either unavailable or so coarse (in both space and time) that prevent accurate analysis and completeness guarantees. This work focuses on the problem of reactive synthesis for such systems and aims to develop an algorithm with correctness (sythensized strategies satisfy requirements) and completeness guarantees.

A powerful and expressive model that best represents complex systems under uncertain with TL specifications is *Nondeterministic Hybrid Systems* (NHS) [4]. 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 continuous system 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 unceratinties.

Authors are with the department of Aerospace Engineering Sciences at the University of Colorado Boulder, CO, USA {firstname.lastname}@colorado.edu

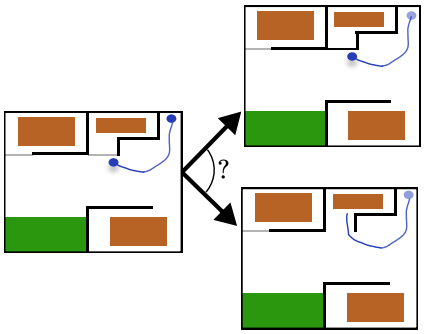


Fig. 1: Search and rescue mission scenario. This figure captures a single nondeterminism due to unknown door state.

Example 1 (Running example)**.** *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 possible to enter (unblocked), the rover must search it in* 10 *seconds. 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 *seconds 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 view nondetermism as an adversarial player that attempts to prevent the system 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 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 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 almost surely finds a strategy that guarantees satisfaction of the objectives if one exists.

The contributions of this paper are threefold: (i) a novel sampling-based reactive synthesis algorithm for nondeterministic hybrid systems, (ii) proof of probabilistic completeness of the algorithm, (iii) a series of benchmarks and case studies that illustrate the generality and effectiveness of the algorithm. The 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* reactive synthesis algorithm with probabilistic completeness for NHS.

A. Related Work

Sampling-based algorithms [5]–[7] have emerged as powerful tools for kinodynamic motion planning for complex nonlinear dynamics and hybrid systems [8], [9]. These techniques are typically used for deterministic systems and, only recently, extended to stochastic systems. Nondeterminism is often not considered. In this work, we utilize sampling-based techniques to deal with nondeterminism and achieve reachability objectives.

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], [10]. When applied to continuous systems, however, they require finite abstraction, which is difficult to obtain for complex systems. In the continuous domain, techniques based on Hamilton-Jacobi analysis and contraction theory have been employed to provide robust controllers with guarantees on system behavior [11]–[13]. However, most of these methods are designed for bounded disturbances and are not able to 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 is closely relates to ours is [14]. It considers the same problem and proposes a two-phase sampling-based strategy planning algorithm that performs exploration in the first phase and strategy improvement in the second phase. The algorithm is very dependent on the quality of strategies that are found in the first phase, and since it cannot return to the first phase, it is incomplete. In this paper, we aim to develop a probabilistically complete algorithm that continually improves promising strategies.

II. Problem

In this work, we consider complex control systems under uncertainty with temporal and reachability objectives. Specifically, we focus on uncertanties of nondeterministic or discrete disturbance type. A general modeling framework that allows for accurate representation of such systems and objectives simultaneously is nondeterministic hybrid systems (NHS).

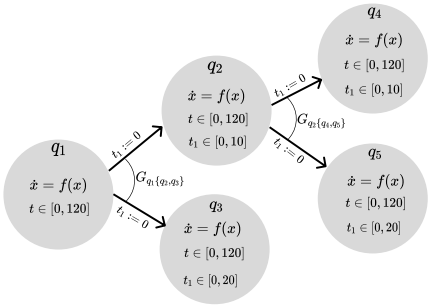


Fig. 2: NHS for single room version of Example in Fig. 1.

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,* $Q = \{q_1, q_2, \ldots, q_m\}$ *for* $m \in \mathbb{N}$*, with a set of mode-dependent continuous state spaces* $X = \{X_q \subseteq \mathbb{R}^{n_q} : q \in Q \wedge n_q \in \mathbb{N}\}$*,*
- $s_0 \in S$ *is the initial state,*
- $U = \{U_q \subset \mathbb{R}^{m_q} \mid q \in Q \wedge m_q \in \mathbb{N}\}$ *is the set of mode-dependent control spaces,*
- $I = \{I_q : q \in Q\}$ *is the set of invariants, where* $I_q : X_q \to \{\top, \bot\}$*,*
- $F = \{F_q : q \in Q\}$ *is the set of flow functions* $F_q : X_q \times U_q \to X_q$ *that describes the continuous dynamics of the system in each mode* q*,*
- $E \subseteq Q \times Q$ *is the discrete transitions between modes in* Q*,*
- $G = \{G_{qQ'} : Q' \in 2^Q \wedge (q, q') \in E \ \forall q' \in Q'\}$ *is a set of guard functions* $G_{qQ'} : X_q \to \{\top, \bot\}$ *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,*
- $J = \{J_{qq'} : (q, q') \in E\}$*, is a set of jump functions* $J_{qq'} : X_q \to X_{q'}$ *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 satisfaction 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 and guard functions in I and G, respectively, 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 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 guard $G_{q_0Q'}$. 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 as long as the invariant function I remains true. The system terminates when the invariant becomes false, which is an indication that temporal constraints are violated, or the reachability indicator function R becomes true, which is an indication that the reachability objective is satisfied.

Example 2. *The NHS that models a simplified version of Example 1 in Fig. 1 is shown in Fig. 2. Time clocks are added as states of the system, and 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 continuous state, i.e., $\forall x_1, x_2 \in X_q$

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

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 nondetermisim in the guards, i.e., nondeterministic transitions between discrete modes. 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}.*

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 [15] and co-safe LTL [16], MITL, and STL [2]) objectives, where the hybrid system is the Cartesian product of an uncertain continuous system with the automaton that represent the temporal logic objectives.*

III. Background

A. Game Trees, AND/OR tree. 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 [17]. 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. 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)**.** *Tree $\mathcal{T}_{sub} = (\mathcal{N}_s, \mathcal{E}_s)$ 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 representation 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 cost function penalizes the MIN player (root node of the AND/OR tree) for having leaf nodes outside T. For a given strategy

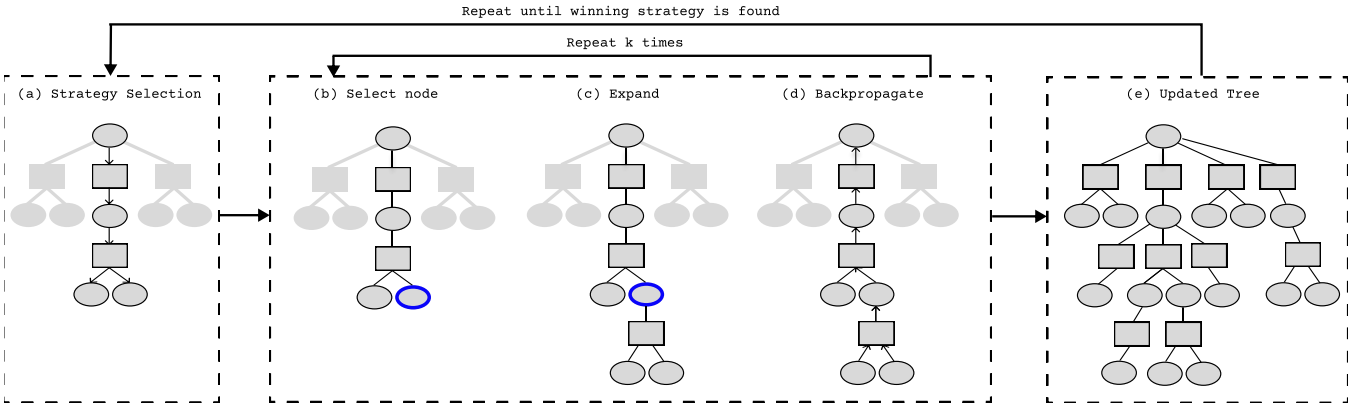


Fig. 3: Illustration of SaBRS algorithm.

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* 0*.*

B. Multi-Armed Bandits and Upper Confidence Bounds

In the classical decision-making problem of the multi-armed bandit problem, an agent is presented with multiple arms (actions). Every time the agent chooses an arm (action), it receives a penalty 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 the 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 [18].

A theoretically sound algorithm whose finite-time regret is well-studied is the *Upper Confidence Bound* (UCB1) algorithm [18]. UCB1 resolves the exploration-explotation tradeoff by using a bias term. Specifically, the UCB1 defines a cost for choosing action a according to

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

where $\hat{C}(a)$ is the current estimated cost of taking action a, N is the number of trials, N_a is the number of times action a is previously chosen, and e is a constant that determines the relative ratio of exploration to exploitation. 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 system 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 the node has been visited, and $\mathbf{u}$ is the set of control inputs previously selected at n. We model this tree as an AND/OR tree, where the system is the MIN player which chooses a control $u \in n.\mathbf{u}$ at node n, followed by the MAX player, which chooses a child of the node-control pair (n, u). Intuitively, the MAX player models the nondeterminism in H. Then, a strategy subtree on the AND/OR tree becomes 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 which solves Problem 1. A major challenge in this approach 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 efficient and probabilistically complete algorithm (see Sec. V). The algorithm terminates when a winning strategy is found.

Algorithm 1: SaBRS Algorithm

Input : H, s_0, k
Output: Winning Strategy π^*
1 $n_0 \leftarrow \langle s_0, 0, \emptyset \rangle$
2 $\mathcal{T} = (\mathcal{N} \leftarrow \{n_0\}, \mathcal{E} \leftarrow \emptyset)$
3 **while** $cost(\mathcal{T}(s_0)) > 0$ *or time limit not reached* **do**
4 $\quad \pi \leftarrow \{n_0\}$
5 $\quad \pi \leftarrow$ UCB-ST(π, n_0)
6 $\quad$ **for** $j = 1 \rightarrow k$ **do**
7 $\quad\quad \pi \leftarrow Explore(\pi)$
8 $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \pi$
9 $\pi^*(s) \leftarrow \text{argmax}_{u \in n.\mathbf{u}}\ \mathcal{Q}(n,u) \forall n \in \mathcal{T}$
10 **return** π^*

A. Strategy Tree Selection with Upper Confidence Bounds

Since search trees are usually large, we wish to bias our search towards the strategies that are promising (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 strategy subtree π is done based on a cost function that assigns to node n the cost

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

where $\mathcal{C}^\pi_{all}(n)$ is the set of all leaf nodes of the subtree π rooted at n, and $\mathcal{C}^\pi_{goal}(n)$ is the subset of $\mathcal{C}^\pi_{all}(n)$ that is in S_{goal}. Intuitively, the cost at n under strategy π is the portion of leaf nodes that are not in S_{goal}. When $C(n,\pi) = 1$, it means no branches of subtree π from n end in S_{goal}, and when $C^\pi(n) = 0$, it means 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 our 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 for n, otherwise $C^\pi(n) > 0$.*

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

$$\mathcal{Q}^\pi(n,u) = 1 - \frac{\sum_{n' \in child(n,u)} |\mathcal{C}^\pi_{goal}(n')|}{\sum_{n' \in child(n,u)} |\mathcal{C}^\pi_{all}(n')|}. \tag{3}$$

This $\mathcal{Q}$-cost allows us to evaluate the relative cost of each input u at node n. Note that when $\pi(n) = u$, the $\mathcal{Q}$-cost of 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 because a strategy may have a low cost but it may not be able to extend to 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 *Upper Confidence bounds for Strategy Tree selection* (UCB-ST). The algorithm is shown in Alg. 2. It first initialize the chosen strategy tree π with the root node n_0 (Line 4 of Alg. 1). From n_0, it selects control inputs in the AND/OR tree according to the UCB1 criterion (Lines 5-6 in Alg. 2) adapted from (1):

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

where $\mathcal{Q}^*(n,u) = \min_\pi \mathcal{Q}^\pi(n,u)$ is the optimal $\mathcal{Q}$-cost, and $n.N_u$ is the number of times the control u has been selected at node n. All children nodes of (n,u) are added to π, and control inputs for each children are again selected according to (4) (Lines 7-10). 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 (Lines 3-4). UCB-ST allows the tree to grow in the more promising parts, while still allowing for exploration of parts of the tree that seem less promising but might eventually lead to a winning strategy.

Algorithm 2: UCB-ST(π, n)

1 $n.N = n.N + 1$
2 **if** n *is a leaf node* **then**
3 $\quad$ Return $\{n\}$;
4 $u^* \leftarrow \underset{u \in \mathbb{E}(n,\cdot)}{\text{argmax}}\ \text{UCB}(n,u)$ using (4)
5 **for** $n' \in children(n,u)$ **do**
6 $\quad \pi \leftarrow \pi \cup SelectStrategy(\pi, n')$
7 **return** $\{n\}$;

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

Algorithm 3: Explore(π)

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

control $u \in U_q$ and time duration Δt are randomly sampled, and the node's continuous state x is propagated by F_q. Any tree-based sampling-based motion planning technique that supports kinodynamic constraints (e.g., RRT [5] and EST [6]) 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 Δt, only one new node is created. Then, the control u is added to the set of sampled controls $n.\mathbf{u}$, and the created 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 sufficient exploration is performed for a strategy subtree.

Remark 3. *SaBRS algorithm also works in an anytime fashion, i.e., when given a time limit, SaBRS returns a control strategy that minimizes the cost at the root node.*

V. Analysis

In this section, we prove probabilistic completness of our algorithm. Specifically, we consider the case that kinodynamic RRT [5] 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 it almost surely finds a winning control strategy π^* if one exists, i.e., as the number of algorithm iterations $K \to \infty$, the probability of finding a winning control strategy, if one exists, goes to* 1.

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

Lemma 1. *Given a finite search tree $\mathcal{T}$ and exploration constant $e > 0$, UCB-ST in Alg. 2 always eventually selects every strategy subtree, i.e., as number of iterations $K \to \infty$, the number of times every strategy subtree of $\mathcal{T}$ is selected also goes to infinity.*

Proof. Consider a node $n \in \mathcal{T}$. An input $u \in n.\mathbf{u}$ is selected according to the UCB1 criterion in (4), which weighs the current $\mathcal{Q}$-cost with the exploration term $2e\sqrt{2\ln(n.N)/n.N_u}$. This exploration term increases if u is not selected. Hence, as N increases, the only case for an input $u' \neq u$ to always be selected is if the cost continually decreases at a rate faster than the increase in the exploration term. However, our cost function is defined such that the minimum cost is 0, and therefore, for $e > 0$, the exploration term for any input u eventually dominates the cost term. Since the search tree is finite, $|n.\mathbf{u}|$ is finite, and given sufficient iterations, an $u \in n.\mathbf{u}$ is always eventually chosen. By induction, every strategy subtree is always eventually selected. □

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.*

Proof. Suppose that there exists a winning strategy π_{win} from s_0 to S_{goal}, with clearance $\delta_{clear} > 0$. 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 $p_i = s_0^{g_0} \dots s_k^{g_1} \dots s_l^{g_0} \dots s_m^{g_1} \dots s_{goal,p_i}$ 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 [19, Theorem 2], we are guaranteed that RRT 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, we know that 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. Given enough expansion iterations, π will almost surely find 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 improve efficiency of SaBRS without affecting its probabilistic completeness.

Warm Starting: The effectiveness of SaBRS relies on selecting promising strategies based on the cost function in (2). However, while 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 can first

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 [14]	78.1 ± 4.6	91	113.1 ± 9.1	82	153.68 ± 15.5	44	222.5 ± 26.3	10
	SaBRS (Ours)	$\mathbf{4.2 \pm 0.5}$	**100**	$\mathbf{8.3 \pm 1.1}$	**100**	$\mathbf{23.04 \pm 3.6}$	**99**	$\mathbf{57.9 \pm 6.5}$	**93**
Case 2	Planner in [14]	98.1 ± 6.5	96	157.1 ± 12.0	68	208.9 ± 18.3	32	251.1 ± 49.0	4
	SaBRS (Ours)	$\mathbf{4.6 \pm 0.5}$	**100**	$\mathbf{11.4 \pm 2.8}$	**99**	$\mathbf{24.4 \pm 4.0}$	**97**	$\mathbf{48.99 \pm 6.8}$	**88**

perform a *warm start* of the algorithm by exploration using the full AND/OR tree for some fixed time, or until a single goal is in the leaf node. In our benchmarks, we observed that warm starting is especially effective in problems with longer horizons, where reaching a goal by a leaf node is difficult.

Strategy Tree Expansion Guidance: During expansion/improvement of strategies, the algorithm uses sampling-based exploration. This exploration is shown to be greatly improved by heuristic guidance mechanisms, such as goal bias and trajectory bias [20]. The exploration step of our algorithm is general and amenable to such heuristics. An example of such heuristic that is unique to Problem 1 is *Guided Path-generation* and introduced in [14]. It uses the search tree branches that end in a goal state to guide the expansion of nodes (see [14] for details). To maintain probabilistic completeness of our algorithm, we use this mode-dependent guided path-generation mechanism with low probability p and the random exploration of Alg 3 with probability $1 - p$. In our evaluations, we find that guided path generation greatly improves the 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-subtree (sub-strategy tree) Selection: At every 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 $\mathcal{T}$. However, note that input actions in a given node are actually continuous, whereas a given $\mathcal{T}$ only has a discrete set of available inputs. During strategy expansion, nodes in π are expanded, 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 the subtree to obtain a smaller strategy, with probability ρ. This effectively chooses 'no action' in 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 such a heuristic 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 state-of-the-art algorithms RRT planner, a continuous space MCTS algorithm (MCTS-PW), and the nondeterministic motion planner in [14] in a series of benchmarks. We also provide several illustrative examples to show the generality of SaBRS. We implement all algorithms in C++ using OMPL [21]. All computations are performed single-threaded on a nominally 2.20 GHz CPU wih 32 GB RAM.

A. Benchmarking Results

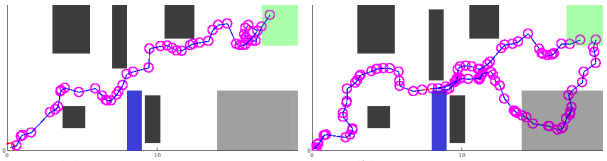
We first evaluate the algorithms on the benchmark problems proposed in [14]. The problems consists of two NHS models of a three-gear second-order car system that is subject to nondeterminism when shifting gears in the four environments considered in [14]. In Case Study 1, the system, when shifting from gear two to three, may mistakenly change to gear one instead of three. In Case Study 2, there is an additional nondeterminism when the car has to shift from gear three to one. In this case, the system may mistakenly change to gear two instead of one. We refer the reader to [14] for details on the dynamics of each gear.

We provided a time limit of 300 seconds to find a solution for 100 trials for each of the four environments and two cases. The results are summarized in Table I. For the first case study, MCTS did not find any solutions. For the second case study, both RRT and MCTS did not find any solutions. It is evident from the poor performance of both RRT and MCTS that neither a purely sampling-based exploration method nor a purely heuristic search method works for finding winning strategies for NHS planning problems. Additionally, we see that SaBRS significantly outperforms the compared methods both in computation time (up to an order of magnitude) and success rate (up to $3\times$) of 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 nondeterministic uncertainty.

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. Online, 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. 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



(a) Spaced Obstacles (b) Cluttered Obstacles

Fig. 4: Robotic charging example with different obstacle (in black) configuration. The blue, grey and green regions represents the puddle, carpet, and charger, respectively. SaBRS synthesizes a strategy that accounts for nondeterminism in traversing the puddle due to motion uncertainty.

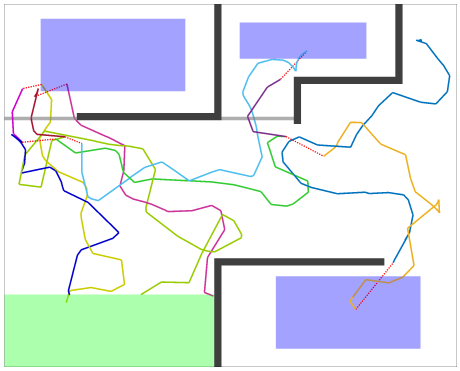


Fig. 5: Example synthesized strategy for building search-and-rescue example.

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

Finally, we show that SaBRS is effective for problems with complex temporal specifications. We consider variants of the search-and-rescue scenario of Example 1. Figure 5 shows an example synthesized strategy (computed within $20s$). Each possible trajectory in the reactive strategy is plotted with a different color, and where a red dotted path segment indicates a nondeterministic mode transition. SaBRS successfully finds a winning strategy that reacts based on if a door is open or closed, and if the human is in each room. Detailed examples can be found in the extended version of our paper [22].

VIII. Conclusion and Future Work

This paper considers the problem of computing reactive strategies for nondeterministic hybrid systems with complex continuous dynamics under temporal and reachability constraints. We propose an algorithm that guarantees the satisfaction of reaching goals under all possible evolution of the nondeterministic hybrid system. This algorithm combines the effectiveness of sampling-based planning with bandit-based exploration-exploitation of promising strategies. We show that the algorithm is probabilistically complete, and benchmarks and case studies demonstrate its effectiveness. For future work, we plan to consider cases where winning strategies are impossible, and extending the work to include uncertainty in continuous dynamics.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[2] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds., 2004.

[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] 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.

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

[6] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *International Conference on Robotics and Automation*, vol. 3, 1997, pp. 2719–2726 vol.3.

[7] J. Cortés and T. Siméon, *Sampling-Based Tree Planners (RRT, EST, and Variations)*, 2020, pp. 1–9.

[8] N. Wang and R. G. Sanfelice, "A rapidly-exploring random trees motion planning algorithm for hybrid dynamical systems," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 2626–2631.

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

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

[11] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," *IEEE Conference on Decision and Control (CDC)*, pp. 1517—-1522, 2017.

[12] S. J. Leudo and R. G. Sanfelice, "Sufficient conditions for optimality in finite-horizon two-player zero-sum hybrid games," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 3268–3273.

[13] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5883–5890.

[14] M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "A sampling-based strategy planner for nondeterministic hybrid systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[15] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," ser. IJCAI '13. AAAI Press, 2013.

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

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

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

[19] 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 Robotics and Automation Letters*, 2019.

[20] C. Urmson and R. Simmons, "Approaches for heuristically biasing rrt growth," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2. IEEE, 2003, pp. 1178–1183.

[21] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[22] Q. H. Ho, , Z. N. Sunberg, and M. Lahijanian, "Sampling-based reactive synthesis for nondeterministic hybrid systems (extended version)." [Online]. Available: https://mortezalahijanian.com/papers/CDC2023.pdf