Received 15 March 2023; revised 18 June 2023; accepted 10 July 2023. Date of publication 26 July 2023;
date of current version 22 August 2023. Recommended by Senior Editor Majid Zamani.

Digital Object Identifier 10.1109/OJCSYS.2023.3299152

HyHooVer: Verification and Parameter Synthesis in Stochastic Systems With Hybrid State Space Using Optimistic Optimization

NEGIN MUSAVI [1] (Student Member, IEEE), DAWEI SUN [1] (Student Member, IEEE), SAYAN MITRA [1] (Member, IEEE), GEIR E. DULLERUD [1] (Fellow, IEEE), AND SANJAY SHAKKOTTAI [2] (Fellow, IEEE)

(Formal Verification and Synthesis of Cyber-Physical Systems)

[1]Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801 USA
[2]Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA

CORRESPONDING AUTHOR: NEGIN MUSAVI (e-mail: nmusavi2@illinois.edu).

This work was supported by the NSA Science of Security Grant H98230-18-D-0007.

ABSTRACT This article presents a new method for model-free verification of a general class of control systems with unknown nonlinear dynamics, where the state space has both a continuum-based and a discrete component. Specifically, we focus on finding what choices of initial states or parameters maximize a given probabilistic objective function over all choices of initial states or parameters from such hybrid state space, without having exact knowledge of the system dynamics. We introduce the notion of *set initialized Markov chains* to represent such systems. Our method utilizes generalized techniques from multi-armed bandit theory on the continuum, in an attempt to make an efficient use of the available sampling budget. We introduce a new algorithm called the *Hybrid Hierarchical Optimistic Optimization* (HyHOO) algorithm, which is designed to address the problem outlined in this paper. The algorithm combines elements of the existing Hierarchical Optimistic Optimization (HOO) bandit algorithm with carefully chosen parameters to create a fresh perspective on the problem. By viewing the problem as a multi-armed bandit problem, we are able to provide theoretical regret bounds on sample efficiency of our tool, HyHooVer. This is achieved by making assumptions about the smoothness of the underlying system. The results of experiments in formal verification and parameter synthesis of variety of scenarios, indicate that the proposed method is effective and efficient when applied to realistic-sized problems and it performs well compared to other methods, specifically PlasmaLab, BoTorch, and the baseline HOO algorithm. Specifically, it demonstrates better efficiency when employed on models with large state space and when the objective function has sharp slopes in comparison with other tools.

INDEX TERMS Autonomous systems, black-box optimization, monte-carlo tree search, multi-armed bandits, safety verification.

I. INTRODUCTION

Our interest is in verifying safety and synthesizing parameters of autonomous and cyber-physical systems, where the system dynamics are not explicitly known. In other terms, our goal is to identify the choices of initial states or parameters that maximize a given objective function, while not having a detailed knowledge of the underlying system dynamics, and by only using noisy observations of the system. This problem is relevant in applications such as predictive monitoring or runtime verification of safety-critical systems like self-driving vehicles, drones and medical devices that incorporate complex black-box algorithms, where guaranteeing the safety of the system is essential.

Addressing this problem remains challenging because the system dynamics are not fully known and querying the system can be costly. Also typically the state space could be big

and hybrid (with continuum-based and discrete components) which make the problem even harder. This problem can be categorized as a black-box optimization problem that involves expensive system queries. To address this problem, searching for black-box optimization methods that are sample efficient is logical. In the context of finite state-space systems, the theory of *multi-armed bandits* has demonstrated its efficiency in learning about unknown systems through adaptive sampling. Therefore, our approach not only utilizes the multi-armed bandits approach but also extends it to a new setting with a hybrid state space, marking the first time such an extension has been made.

The field of multi-armed bandits [1], [2], [3] has seen significant growth and development in both technique and practical use in the recent years [4], [5]. Especially relevant to our research is $\mathcal{X}$-armed bandit in [5] with its application in black-box optimization, where the goal is to find the maximum of an unknown function f with only noisy queries of the function (i.e. a query of x returns $f(x)+$ noise). One well-known algorithm in this area is the Upper Confidence Bound (UCB) [6] which utilizes the Principle of Optimism for adaptive search. Since the exact value of $f(x)$ is unknown, this principle uses an estimated upper bound on the function – this is constructed from samples, and thus holds with high probability. The x chosen at each time is the one that maximizes this estimated upper bound. Thus, initially different parts of state space are explored, and as more information is gathered, it becomes less optimistic. This principle has been used in the context of black-box optimization by reformulating problems as tree search [4], [5], [7], [8]. These algorithms use a tree structure to search through the unknown function's domain, balancing the exploration of less explored areas with the exploitation of high-value regions that have already been identified, to obtain an approximate solution within a given sampling budget.

These algorithms are assessed based on a measure called cumulative regret, which is the expected accumulated difference between the highest value found by the algorithm and the true maximum value. To ensure theoretical guarantees for the regret, these methods typically require that the objective function satisfy some smoothness properties. While the approach in [5] requires smoothness with respect to a semi-metric, the algorithms in [9], [10] relax the requirement of an exact semi-metric that captures the smoothness of f, and instead use two smoothness parameters to define their notion of smoothness. When estimating these parameters is not practical, [10] provides an algorithm to simultaneously test different parameters, and this approach has been extended to multi-fidelity settings in [9].

Given this, we can utilize the $\mathcal{X}$-armed bandits method to tackle the problem at hand, where the concept of regret can be related to the error in verification and synthesis in our specific context (for more details refer to Section III-D).

In this article, we introduce a new algorithm for statistical verification and parameter synthesis of a new generalized class of discrete-time Markov chains (MCs) with hybrid state space, consisting of a continuum-based and a discrete part, which can be applied in situations where information about the transition probabilities are not known and sampling budget is limited. We carefully introduce the notion of *set initialized Markov chains (SIMC)* that captures the properties of discrete-time MCs with hybrid state space. Our algorithm is called *Hybrid Hierarchical Optimistic Optimization (HyHOO)*, that is built upon a tree-structured method called *hierarchical optimistic optimization (HOO)* [5] and is designed to maximize unknown functions. HyHOO extends our previous work in [11] to systems with hybrid state space and has the following features:

- It expands HOO to cover hybrid systems whose states live in state space that has both continuum-based and discrete components. We use the terminology state to refer to continuous components, and the terminology mode to refer to discrete components. This allows the algorithm to be applied in a wider range of situations where the set of initial states or parameters includes both hyper-rectangles and finite sets.
- It employs the principle of optimism to explore the hybrid state spaces, by strategically allocating the available budget to the regions of the state space that are likely to be optimal.
- It is designed with batched simulations feature which improves its memory usage and running time.

To provide theoretical guarantees for the regret achieved by our algorithm HyHOO, we make assumptions about the smoothness of the function f. These assumptions relate to the smoothness of f in the vicinity of $f(x^*)$ over the regions of the state space that correspond to each mode. This is less restrictive than the smoothness assumptions required for HOO and our previous work [11], which require the smoothness of f in the vicinity of $f(x^*)$ over the entire state space. The theoretical bound on the regret of HyHOO is a function of the smoothness parameters, the sampling budget, the batch size parameter, and a property of the function called the *near-optimality dimension* which captures the steepness of the slope of the function around the optimum. This bound is different from the existing performance bounds found in the literature on statistical model checking. For example, the bounds relevant for tools like PlasmaLab [12] often use techniques like Monte Carlo sampling, Chernoff bounds, or sequential hypothesis testing.

We have built an open-source tool **HyHooVer** that utilizes the HyHOO algorithm, and have established benchmarks using Python to showcase its abilities and gauge its performance in terms of sample efficiency. These benchmarks involve traffic scenarios in a car simulation environment called "highway-env" [13], which offers a minimalist yet realistic framework for simulating cars[1]. Our tool's source code and

[1]Highway-env is a minimal car simulation environment for autonomous driving and tactical decision-making tasks which is available at https://github.com/eleurent/highway-env.

usage instructions, along with the examples, are also available[2].

Through our experiments on the benchmarks, we have observed the following:

- Our tool is capable of performing verification and synthesis in scenarios with hybrid state spaces and it demonstrated better sample efficiency than other methods in situations with higher dimensions and multiple modes.
- The tool performs better in scenarios where the slope of the function is steeper around the maximum. This means that the tool is particularly well-suited to situations where there is a rare event involved.
- The batch size parameter can be adjusted to optimize the tool's performance in terms of running time and memory usage, without compromising the results of the verification and synthesis.

A. COMPARISON WITH RELATED APPROACHES

There is a substantial body of literature on model-based approaches for the verification and parameter synthesis of stochastic systems [14], [15], [16], [17], [18] (and the references therein), which rely on detailed knowledge of the probability transition kernel, which may not always be available. *Statistical model checking (SMC)* also addresses similar verification problems. SMC methods collect samples through system execution and use statistical tests to determine if the constraints have been met or violated [19], [20], [21], [22]. Notable methods in this category include MODEST for probabilistic automata [23], PlasmaLab [12], the learning-based algorithm of [24], [25] in PRISM [26] and UPPAAL, and approaches for Markov Decision Processes with restricted classes of schedulers implemented in [27], [28], [29].

Comparing HyHooVer to other discrete-state SMC tools is challenging because the guarantees are different and it is hard to factor out platform-specific constants. We provide a detailed comparison with PlasmaLab in Section IV-D, which is a tool for model checking of unknown systems. HyHooVer generally have better estimation results with fewer samples than PlasmaLab, particularly for higher dimensional models and models with sharp slopes around the maxima. This suggests that HyHOO may be more effective in finding hard-to-detect bugs with fewer samples. The approach of [30] uses the original HOO algorithm of [5] but we could not find this tool online for running comparative experiments. Our approach differs from [30] in two important ways: (1) we use a search algorithm spawning different smoothness parameters and return the result of the best one; (2) we exploit batched simulations; and (3) we have extended HOO to hybrid state-space settings.

One of the prominent techniques for optimizing black-box functions in stochastic systems is through Bayesian optimization-based approaches such as BoTorch [31]. BoTorch is developed using PyTorch and is appropriate for systems with costly queries. We built upon our previous work [11] by providing a comparison of our tool with BoTorch in Section IV-D. Our results indicate that, overall, HyHooVer outperforms BoTorch in terms of running time, especially when a large number of queries are required to verify a model. This difference is particularly noticeable in cases where there are difficult-to-detect bugs or high-dimensional scenarios.

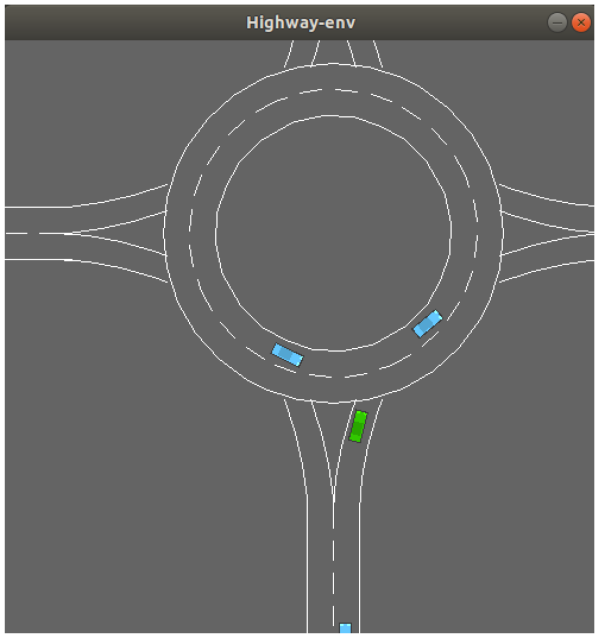


FIGURE 1. Roundabout scenario which can be instantiated with different number of vehicles and initialization.

B. MOTIVATING EXAMPLE, TRAFFIC ROUNDABOUT

Consider a roundabout scenario of multiple cars which is simulated in highway-env and is depicted in Fig. 1. Each vehicle starts from an initial position which is randomly distributed from a distribution $\mathcal{D}$ and moves toward a target destination while meeting a target speed. Suppose the blue vehicles carry their motion according to Intelligent Driver Model (IDM) and Minimizing Overall Braking Induced by Lane Changes (MOBIL) decision policies (built-in policies in highway-env) while avoiding other cars. The green car, on the other hand is able to track its lane while meeting a target speed, however cannot detect rear-end hazards. Define collision as when distance between the green car and any of the blue cars become less than a threshold. We would like to find the most unsafe situation over a set of target destinations and intervals of initial speeds and target speeds of the cars. In this scenario the state space $\mathcal{X}$ can be constructed such that it consists of a discrete and a continuum-based component. Let us explain this with an example. In this scenario, let the letters N, E, W, and S represent the north, east, west, and south exits, respectively. Then let, for instance, the tuple (N, E) represent the north exit and east exit as the target destination for the green car and one of the blue cars. Additionally, let v_0^l, v_0^u denote

[2]The source code is available at https://github.com/NeginMusavi/HyHooVer.git.

lower and upper bounds for the initial speed of the green car, and let v_t^l, v_t^u denote lower and upper bounds for the target speed of the green car. Then an example of a set that consists of a discrete and a continuum-based component would be $\{(E, E), (N, E), (W, E), (S, N)\} \times [[v_0^l, v_0^u], [v_t^l, v_t^u]]$. Detailed descriptions and Python simulators for our models are available from the HyHooVer source page[3].

II. PROBLEM STATEMENT

Background and Notation: Let $\mathcal{Z}$ be a measurable space in $\mathbb{R}^m$ and let $\mathcal{X} := [L] \times \mathcal{Z}$, where $[L] = k_1, k_2, \ldots, k_L$ for some integer $L \geq 1$. Let the pair $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ be a *measurable space*, where $\mathcal{F}_\mathcal{X}$ is a σ-algebra over $\mathcal{X}$ and the elements of $\mathcal{F}_\mathcal{X}$ are referred to as *measurable sets*. Let $\mathbb{P} : \mathcal{X} \times \mathcal{F}_\mathcal{X} \to [0, 1]$ be a *Markovian transition kernel* on a measurable space $(\mathcal{X}, \mathcal{F}_\mathcal{X})$, such that for all $x \in \mathcal{X}$, $\mathbb{P}(x, \cdot)$ is a probability measure on $\mathcal{F}_\mathcal{X}$; and for all $\mathcal{A} \in \mathcal{F}_\mathcal{X}$, $\mathbb{P}(\cdot, \mathcal{A})$ is a $\mathcal{F}_\mathcal{X}$-measurable function. Also let $\mathbb{P}_\beta$ be a Markovian transition kernel that depends on parameter $\beta \in \mathbb{R}^n$. For a $\sigma > 0$, a real-valued random variable X is σ^2-*sub-Gaussian*, if for all $s \in \mathbb{R}$, $\mathbb{E}[\exp(s(X - \mathbb{E}X))] \leq \exp(\sigma^2 s^2/2)$ holds, where $\mathbb{E}$ denotes expectation. For a matrix $z \in \mathbb{R}^{n \times m}$, $\|z\|_F$ denotes its Frobenius norm.

Definition 1: A *set initialized Markov chain (SIMC)* $\mathcal{M}$ is defined by a tuple $((\mathcal{X}, \mathcal{F}_\mathcal{X}), \mathbb{P}_\beta, \mathcal{B}, \Theta)$, with:

- $(\mathcal{X}, \mathcal{F}_\mathcal{X})$, a measurable space over the state space $\mathcal{X}$;
- $\mathbb{P}_\beta : \mathcal{X} \times \mathcal{F}_\mathcal{X} \to [0, 1]$, a Markovian *transition kernel* depending on parameter $\beta \in \mathcal{B}$;
- $\mathcal{B} \subseteq \mathcal{X}$, a set of parameters; and
- $\Theta \subseteq \mathcal{X}$, a set of possible initial states.

In the examples in Section II-B, the state-dependent probabilistic choices are modeled by the Markov transition kernel $\mathbb{P}_\beta$. We reiterate that our algorithm will not rely on the knowledge of this kernel. Let α, a sequence of states $\alpha = x_0 x_1 \cdots x_k$, be an *execution* of $\mathcal{M}$ of length k for any $x_0 \in \Theta$ and any $\beta \in \mathcal{B}$, where x_is $\in \mathcal{X}$. Given x_0, β and a sequence of measurable sets of states $A_1, \ldots, A_k \in \mathcal{F}_\mathcal{X}$, the measure of the set of executions $\{\alpha \mid \alpha_0 = x_0 \text{ and } \alpha_i \in A_i, \forall\, i = 1, \ldots, k\}$ is given by:

$$\Pr(\{\alpha \mid \alpha_0 = x_0 \text{ and } \alpha_i \in A_i, \forall\, i = 1, \ldots, k\}) = \int_{A_1 \times \cdots \times A_k} \mathbb{P}_\beta(x_0, dx_1) \cdots \mathbb{P}_\beta(x_{k-1}, dx_k),$$

which is a standard result from the Ionescu Tulceă theorem [32], [33]. We address two classes of problems:

A. VERIFICATION

Given an SIMC$\mathcal{M}$ and a measurable unsafe set $\mathcal{U} \in \mathcal{F}_\mathcal{X}$, we are interested in evaluating the *worst-case* probability of $\mathcal{M}$ hitting $\mathcal{U}$ over all possible nondeterministic choices of an initial state $x_0 \in \Theta$. Once an initial state $x_0 \in \Theta$ is fixed, the probability of a set of paths is determined by the Markovian transition kernel in the model $\mathcal{M}$, as described above. Circling back to our motivating example in Section II-B, the set Θ would correspond to a set of target destinations and intervals of initial speeds and target speeds of the cars, and x_0 would be an element in this set.

We say that an execution α of length k hits the unsafe set $\mathcal{U}$ if there exists an integer $i \in \{0, \ldots, k\}$, such that $\alpha_i \in \mathcal{U}$. The complement of $\mathcal{U}$, the *safe subset* of $\mathcal{X}$, is denoted by $\mathcal{S}$. The safe set is also a member of the σ-algebra $\mathcal{F}_\mathcal{X}$ since σ-algebras are closed under complementation. From a given initial state $x_0 \in \Theta$ and a given parameter $\beta \in \mathcal{B}$, the probability of $\mathcal{M}$ hitting $\mathcal{U}$ within k steps is denoted by $p_{k,\mathcal{U},\beta}(x_0)$. By definition, $p_{k,\mathcal{U},\beta}(x_0) = 1$, if $x_0 \in \mathcal{U}$. For $x_0 \notin \mathcal{U}$ and $k \geq 1$,

$$p_{k,\mathcal{U},\beta}(x_0) = 1 - \int_{S \times \cdots \times S} \mathbb{P}_\beta(x_0, dx_1) \cdots \mathbb{P}_\beta(x_{k-1}, dx_k). \tag{1}$$

We are interested in finding the *worst-case* probability of hitting unsafe states over all possible initial states of the model $\mathcal{M}$. This can be regarded as solving, for some k and some $\beta \in \mathcal{B}$, the following optimization problem:

$$\sup_{x_0 \in \Theta} p_{k,\mathcal{U},\beta}(x_0). \tag{2}$$

B. PARAMETER SYNTHESIS

Given an execution α of length k and a $\beta \in \mathcal{B}$, let $r(\alpha, \beta)$ be a real-valued objective function. Then, we are interested in evaluating the maximum of expected objective function over all possible nondeterministic choices of the parameter $\beta \in \mathcal{B}$. This can be regarded as solving, the following related optimization problem:

$$\sup_{\beta \in \mathcal{B}} \mathbb{E}[r(\alpha, \beta) | \beta], \tag{3}$$

where the expectation is over the randomness of the transition and the initial state (drawn from a given distribution).

III. VERIFICATION AND PARAMETER SYNTHESIS WITH HIERARCHICAL OPTIMISTIC OPTIMIZATION

We will solve the optimization problems in (2) and (3) by developing the HyHOO algorithm with mini batches. This can be regarded as a variant of the HOO algorithm [5]. The setup is as follows: suppose we have a sampling budget of N and want to maximize the function $f : \mathcal{X} \to \mathbb{R}$ based on receiving noisy observations of f, i.e., f + noise. It is assumed that f has a unique global maximum that achieves the value $f^* = \sup_{x \in \mathcal{X}} f(x)$, where $\mathcal{X} = [L] \times \mathcal{Z}$ is as introduced in Section II. Our approach gets to choose a sequence of sample points (arms) $x_1, x_2, \ldots x_N \in \mathcal{X}$, for which it receives the corresponding sequence of noisy *observations* (or *rewards*) $y_1, y_2, \ldots, y_N$. When the sampling budget N is exhausted, the algorithm has to decide the optimal point $\bar{x}_N \in \mathcal{X}$ with the aim of minimizing *regret*, which is defined as:

$$S_N = f^* - f(\bar{x}_N). \tag{4}$$

Our approach has two key properties:

[3] https://github.com/NeginMusavi/HyHooVer.git.

- It is *adaptive* in the sense that the $(j+1)^{\text{st}}$ sample x_{j+1} should depend on the previous samples and their corresponding noisy observations; and
- It does not rely on detailed knowledge of f but *only* on the sampled noisy observations.

Algorithms with these properties are called *black-box* or *zeroth-order* algorithms. In order to derive rigorous bounds on the regret, however, we will need to make some assumptions on the smoothness of f (see Assumption 2) and on the relationship between $f(x_j)$ and the corresponding observation y_j. Assumption 1 formalizes the latter by stating that y_j is distributed according to some (possibly unknown) distribution with mean $f(x_j)$ and a strong tail-decay property.

Assumption 1: There exists a constant $\sigma > 0$ such that for each sampled x_j, the corresponding noisy observation y_j is distributed according to a σ^2-sub-Gaussian distribution M_{x_j} satisfying $\int u dM_{x_j}(u) = f(x_j)$.

Hence any random variable that is bounded is a sub-Gaussian random variable and thus there exists a $\sigma > 0$ satisfies the conditions of this assumption. Next, we present the HyHOO algorithm. In Sections III-B and III-C we present its analysis leading to the regret bound and discuss the choice of its parameters as well as their implications. In Section III-D we discuss how HyHOO can be used for solving the optimization problems in (2) and (3).

A. HYBRID HIERARCHICAL TREE OPTIMIZATION

HyHOO (Algorithm 1) is a batched-sampling variant of HOO [5], which also takes into account a hybrid state space. HyHOO selects the next sample x_{j+1} by building a tree in which each height (or level) partitions the state space $\mathcal{X}$ into a number of regions. The algorithm samples states to estimate upper bounds on f over a region, and based on this estimate, decides to expand certain branches (i.e., re-partition certain regions) to reduce the region sizes based on the smoothness of f. HyHOO allows us to execute batch simulations of size b to reduce the variance in the estimate of $f(x_i)$ obtained from the noisy observations y_is for any state x_i and, more importantly, maintain a lighter data structure. In Section III-C, we discuss the implications of the choice of batch size parameter b. The tree construction is based on the noisy tree-search algorithms (HOO and variants) [5]. The root of the tree has L children, where the l-th child is the root of the l-th binary[4] sub-tree constructed over $k_l \times \mathcal{Z}$. Each node in the tree except for its root is labeled by a triple of integers (l, h, i), where $l \in [L]$ is the sub-tree index that it belongs to, $h \geq 1$ is the height, and $i \in \{1, \ldots, 2^{h-1}\}$, is its position within level h. Each node (l, h, i) can have two children $(l, h+1, 2i-1)$ and $(l, h+1, 2i)$. Node (l, h, i) is associated with the region $\mathcal{P}_{l,h,i} \subseteq k_l \times \mathcal{Z}$, where $\mathcal{P}_{l,h,i} = \mathcal{P}_{l,h+1,2i-1} \cup \mathcal{P}_{l,h+1,2i}$, and for each h these disjoint regions satisfy $\cup_{i=1}^{2^{h-1}} \mathcal{P}_{l,h,i} = k_l \times \mathcal{Z}$. Thus, larger values of h represent finer partitions of $k_l \times \mathcal{Z}$. It is noted that the root of tree is associated with the whole state space $\{k_1, \ldots, k_L\} \times \mathcal{Z}$. For each node (l, h, i) in the tree, HyHOO computes the following quantities:

- $t_{l,h,i}$ is the number of times the node is chosen or considered for re-partitioning.
- $\hat{f}_{l,h,i}$ is the empirical mean of observations over points sampled in $\mathcal{P}_{l,h,i}$.
- $U_{l,h,i}$ is an initial estimate of the upper-bound of f over $\mathcal{P}_{l,h,i}$ based on the smoothness parameters.
- $B_{l,h,i}$ is a tighter and optimistic upper bound for the same.

The *tree* starts with a single root $(-, 0, 1)$, with B-values of its L children $B_{1,1,1}, B_{2,1,1}, \ldots, B_{L,1,1}$ initialized to $+\infty$. At each iteration a *path* from the root to a leaf is found by traversing the child with the higher B-value (with ties broken arbitrarily), then a new node $(lnew, hnew, inew)$ is added and all of the above quantities are updated. The partitioning continues until the sampling budget N is exhausted. After that the algorithm returns a point among $x_1, x_2, \ldots, x_N$ chosen uniformly at random. The details are provided in Algorithm 1. Hence, Algorithm 1 can be additionally adjusted to return the best-scoring input instead of a randomly chosen one, in order to enhance its performance in practice.

Algorithm 1: HyHOO with parameters: sampling budget N, noise parameter σ, smoothness parameters (ν, ρ), batch size parameter b, number of sub-trees L.

$tree = \{(-, 0, 1)\}$, $B_{1,1,1} = \cdots = B_{L,1,1} = +\infty$, $n = 0$.
while $n \leq N$ **do**
  $(path, (lnew, hnew, inew)) \leftarrow Traverse(tree)$
  choose $x \in \mathcal{P}_{lnew,hnew,inew}$
  query x and get b observations $y_1, y_2, \ldots, y_b$
  $tree.Insert((lnew, hnew, inew))$
  for all $(l, h, i) \in path$ **do**
    $t_{l,h,i} \leftarrow t_{l,h,i} + 1$
    $\hat{f}_{l,h,i} \leftarrow (1 - \frac{1}{t_{l,h,i}})\hat{f}_{l,h,i} + \frac{\sum_{j=1}^{b} y_j}{b \times t_{l,h,i}}$
  $n \leftarrow n + b$,
  $B_{lnew,hnew+1,2inew-1} \leftarrow +\infty$,
  $B_{lnew,hnew+1,2inew} \leftarrow +\infty$
  for all $(l, h, i) \in tree$ **do** leaf up:
    $U_{l,h,i} \leftarrow \hat{f}_{l,h,i} + \sqrt{\frac{2\sigma^2 \ln(n/b)}{b \times t_{l,h,i}}} + \nu\rho^h$
    $B_{l,h,i} \leftarrow \min\left\{U_{l,h,i}, \max\{B_{l,h+1,2i-1}, B_{l,h+1,2i}\}\right\}$
return a point among $x_1, x_2, \ldots, x_N$ chosen uniformly at random.

[4] As we go down the tree the partition is refined via bisection along the dimension of the coarsest subdivision (ties are broken arbitrarily). We note that a binary tree is discussed only for ease of exposition; the construction allows a general tree structure generated by other partitionings (e.g. a p-ary tree).

B. ANALYSIS OF REGRET BOUND

The notation and analysis of the regret bound for HyHOO closely follows that in [5] (and followups in [9], [10]).

Let $\Delta_{l,h,i}$ denote the *sub-optimality gap* of node (l, h, i), that is, $\Delta_{l,h,i} = f^* - \sup_{x\in\mathcal{P}_{l,h,i}} f(x)$. A node (l, h, i) is optimal if $\Delta_{l,h,i} = 0$ and it is sub-optimal if $\Delta_{l,h,i} > 0$. We say that a node (l, h, i) is ϵ-optimal if $\Delta_{l,h,i} \leq \epsilon$. These nodes are also called near-optimal nodes. Let Δ_l denote the *sub-optimality gap* of the node associated with region $k_1 \times \mathcal{Z}$, that is, $\Delta_l = f^* - \sup_{x\in k_1\times\mathcal{Z}} f(x)$. We will use two parameters $\nu > 0$ and $\rho \in (0, 1)$ to characterize the *smoothness* of f relative to the partitions (see Assumption 2). We define $\mathcal{N}_h(\epsilon)$ as in [10] as the number of ϵ-optimal nodes at depth h, that is, the number of nodes with $\Delta_{l,h,i} \leq \epsilon$.

From the sampled estimate of $f(x_i)$ at a single point x_i, HyHOO attempts to estimate the maximum possible value that f^* can take over $\mathcal{X}$. This is achieved by assuming that f is locally smooth around x^*, which is formalized in Assumption 2. It basically ensures that f does not change arbitrarily in the regions that are near-optimal. Based on the fact that the hierarchical partitioning is done over the region associated with each mode, this assumption relaxes the smoothness assumption in [9], which restricts the function to satisfy the smoothness assumption for all near-optimal regions across the entire state space.

Assumption 2: There exist $\nu > 0$ and $\rho \in (0, 1)$ such that for all (l, h, i) satisfying $\Delta_{l,h,i} \leq c\nu\rho^h$ (for a constant $c \geq 0$), we have $f^* - f(x) \leq \max\{2c, c + 1\}\nu\rho^h$ for all $x \in \mathcal{P}_{l,h,i}$.

Here c is a parameter that relates the variation of f over all $c\nu\rho^h$-optimal nodes at all $h \geq 0$. For instance, for $c = 0$, it implies that there exist smoothness parameters such that the gap between the f^* and the value of f over all optimal nodes at all $h \geq 0$ is bounded by $\nu\rho^h$. For instance, for $c = 2$, it implies that there exist smoothness parameters such that over all $2\nu\rho^h$-optimal nodes, the gap between the $f(x^*)$ and value of f over those nodes is bounded by $4\nu\rho^h$-optimal, and so on.

Hence, for a finite sampling budget N the final constructed *tree* has a maximum height $h_{\max}$. Therefore it would be sufficient for f to satisfy the conditions of Assumption 2, for all $h \in [0, h_{\max}]$. This would allow f to have finite little jumps around x^*. We now define a modified version of the *near-optimality dimension* which plays an important role in the analysis of black-box optimization algorithms [5], [10]. This is a measure of closeness with respect to the number of nodes that have function values that are "close" to that of the optimum.

Definition 2: $h_{\max}$-bounded near-optimality dimension of f with respect to (ν, ρ) is: $d_m(\nu, \rho) = \inf\{d' \in \mathbb{R}_{>0} : \exists B > 0,\ \forall h \in [0, h_{\max}], \mathcal{N}_h(2\nu\rho^h) \leq B\rho^{-d'h}\}$.

In other words, $\mathcal{N}_h(2\nu\rho^h)$ grow exponentially with h, and the near-optimality dimension gives the exponential rate of this growth. Thus, $\mathcal{N}_h(2\nu\rho^h) \leq B\rho^{-d_m(\nu,\rho)h}$. The modified version of near-optimality dimension is adapted for a finite sampling budget case, and would allow the theoretical guarantees in Theorem 1 to hold for any f that satisfies Assumption 2 with a finite sampling budget. We are now ready to sketch a regret bound for HyHOO.

Theorem 1: With the input parameters satisfying Assumptions 1 and 2, a batch size parameter b, and a sampling budget of N, HyHOO achieves a regret bound of

$$\mathbb{E}[S_N] = O\left(\left(\frac{B\left(\sigma^2 \log(N/b) + b\right)}{N}\right)^{\frac{1}{d_m+2}}\right),$$

where $d_m = d_m(\nu, \rho)$ is the $h_{\max}$-bounded near-optimality dimension and B is the constant appearing in Definition 2.

This theorem suggests that when the near-optimality dimension d_m and/or the constant B decrease, indicating a decrease in the number of near-optimal regions and steeper slopes around the function's maxima, it theoretically leads to a decrease in the regret achieved by HyHOO. This aspect will be elaborated in the experiments. The proof of this theorem is presented below.

Proof: The proof closely follows the approach in [5] (see also [9], [10]), however, attention is needed when batched simulations are used (especially for large batches). Let $R_N = \sum_{i=1}^{N}(f^* - f(x_i))$ be the cumulative regret at the end of iteration N, where $x_i \in \mathcal{X}$ is the point returned by HyHOO at iteration i. Let $\mathcal{T}$ be the *tree* constructed by HyHOO at the end of N iterations. Let H be a positive integer, that can be optimized for the best bound. As in [5], we divide $\mathcal{T}$ into three groups: $\mathcal{T}_1$ that includes all $2\nu\rho^h$-optimal nodes at $h \geq H$, $\mathcal{T}_2$ that includes all $2\nu\rho^h$-optimal nodes at $h \in [0, H - 1]$, and $\mathcal{T}_3$ that includes sub-optimal nodes with sub-optimality gaps greater than $2\nu\rho^h$ for $h \geq 0$. All nodes belonging to these groups contribute to the cumulative regret as $\mathbb{E}[R_N]$

$$\leq O\left(\rho^H N + bB\rho^{-H(d_m+1)} + B\sigma^2 \log(N/b)\rho^{-H(d_m+1)}\right),$$

where the first, second, and the third terms are the contributions of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ to the cumulative regret, respectively. If $b = 1$, the third term dominates the second term, and we recover the bound in [5]. However, if b is large, we can optimize H to get:

$$\mathbb{E}[R_N] \leq O\left(N^{\frac{d_m+1}{d_m+2}}\left(B\left(\sigma^2 \log(N/b) + b\right)\right)^{\frac{1}{d_m+2}}\right),$$

This can be easily converted to the desired simple regret bound of the algorithm (for more details refer to Remark 1 [5]). To be more precise, the connection between simple regret and cumulative regret can be represented in the following manner: $\mathbb{E}[S_N] \leq \frac{\mathbb{E}[R_N]}{N}$, which serves as the concluding statement of the proof. ■

A common alternative to Algorithm 1 where $L > 1$ is to equally distribute the sampling budget N among L modes and use HOO to address the following optimization problem

$$\sup_{k_l\in[L]} \sup_{z\in k_l\times\mathcal{Z}} f(z).$$

We call such alternative approach the "baseline HOO" approach. With no loss of generality suppose that the optimum belongs to the region of the state space associated with the first

Algorithm 2: parallel search with parameters: sampling budget N, number of instances K, maximum smoothness parameters $(\nu_{\max}, \rho_{\max})$.

1: **for** $i = 1 : K$ **do**
2: Spawn HyHOO with $(\nu = \nu_{\max}, \rho = \rho_{\max}^{K/(K-i+1)})$ with budget N/K
3: Let $\bar{x}_i$ be the point returned by the ith HyHOO instance for $i \in \{1, .., K\}$
4: **return** $\{\bar{x}_i | \ i = 1, .., K\}$

mode, i.e. $\Delta_1 = 0$. This algorithm achieves the regret bound of:

$$\mathbb{E}[S_N] = O\left(\left(\frac{B\left(\sigma^2 \log(N/bL) + b\right)}{N/L}\right)^{\frac{1}{d_m+2}}\right) + \sum_{l=2}^{L} \Delta_l O\left(\left(\frac{B\left(\sigma^2 \log(N/bL) + b\right)}{N/L}\right)^{\frac{1}{d_m+2}}\right).$$

Comparing this bound with HyHOO's regret bound reveals that the larger L and Δ_l for $l \neq 1$ get, the more HyHOO gains advantage over the baseline HOO. This is because the baseline HOO would spend a good portion of budget (i.e. $\frac{(L-1)N}{L}$) on searching for the optimum over the parts of state space that do not include the optimum. This is captured in the second term of the regret bound for this algorithm.

1) KNOWLEDGE OF SMOOTHNESS PARAMETERS:

It is worth noting that the exact values of the smoothness parameters in Assumption 2 are not crucial for the implementation of Algorithm 1. They are only necessary to achieve the most precise regret guarantees. If only upper bounds of these parameters are known, they can still be used in a parallel search algorithm. It is noted that determining the optimal smoothness parameters for a given verification problem is a difficult task and requires further research. When only approximate bounds for these parameters are available, which is often the case for physical processes, the search for the optimal parameters can be efficiently performed using the parallel optimistic optimization algorithm developed in [10] (Algorithm 2). This algorithm adaptively searches for the optimal smoothness parameters by creating multiple parallel instances of the HyHOO algorithm with different (ν, ρ) values. In the Section IV, the impact of the upper bound choice of (ν, ρ) on the performance of HyHOO will be discussed.

C. DISCUSSIONS ON CHOICE OF BATCH SIZE *b*

In the HyHOO algorithm, each node (l, h, i) is sampled multiple times (b times) as opposed to the original non-batched version where each node is only sampled once. This change in sampling affects the update rules for the values $U_{l,h,i}$ and $B_{l,h,i}$. Indeed, by setting $b = 1$, the original HOO algorithm can be recovered and the simple regret bound is $\mathbb{E}[S_N] = O\left(\left(\frac{B\sigma^2 \log N}{N}\right)^{\frac{1}{d_m+2}}\right)$.

When comparing the regret bound of the batched version of HyHOO to the non-batched version, it is observed that the regret bound gets worse with larger b, but it has the benefit of reducing the number of nodes in the tree by a factor of b. This leads to a reduction in running time and makes the batched version more memory-efficient compared to the non-batched version. Therefore, the parameter b holds significant importance as a hyperparameter and it is important to ensure that it does not compromise the accuracy of the verification or synthesis. This will be further elaborated in the experiment section.

D. VERIFICATION AND PARAMETER SYNTHESIS WITH HYHOO

Our objective now is to employ the HyHOO algorithm presented in Algorithm 1 to address both the verification problem in (2) and the synthesis problem in (3). To achieve this, we establish a connection between these problems and the optimization problem introduced earlier in Section III. Specifically, we need to identify the function f, the states x, and the noisy observations y in both problems.

- Verification: When using HyHOO for verification, we can choose the objective function as $f(x) := p_{k,\mathcal{U},\beta}(x)$ for any initial state $x \in \Theta$ and a given $\beta \in \mathcal{B}$. Here, $p_{k,\mathcal{U},\beta}(x)$ represents the probability of hitting the unsafe set $\mathcal{U}$ within k steps, starting from the initial state x and following the execution α. We define the noisy observation y as follows:

  $$y = 1 \text{ if } \alpha \text{ hits } \mathcal{U} \text{ within } k \text{ steps, and 0 otherwise.}$$

  Thus, for a given initial state $x \in \Theta$, $y = 1$ with probability $p_{k,\mathcal{U},\beta}(x)$, and $y = 0$ with probability $1 - p_{k,\mathcal{U},\beta}(x)$. In other words, y is a Bernoulli random variable with a mean of $p_{k,\mathcal{U},\beta}(x)$.
- Parameter Synthesis: When using HyHOO for parameter synthesis, we can choose the objective function as $f(\beta) := \mathbb{E}[r(\alpha, \beta)|\beta]$ for any parameter state $\beta \in \mathcal{B}$. Here, $r(\alpha, \beta)$ represents the outcome of the execution α starting from an initial state x_0 sampled from a given distribution, with the parameter set as β. The noisy observation y is defined as:

  $$y = r(\alpha, \beta),$$

  where the mean of y is $\mathbb{E}[r(\alpha, \beta)|\beta]$.

By establishing this connection between the verification and synthesis problems and the optimization problem addressed by Algorithm 1, we can utilize HyHOO with a sampling budget of N to tackle these problems. Furthermore, we can present the following propositions that provide theoretical guarantees regarding the optimization error of HyHOO when applied to the verification and synthesis problems.

Proposition 1: Under the assumption that the noisy observations $y_1, \ldots, y_N$ in the verification problem satisfy Assumption 1, and the smoothness parameters (ν, ρ) satisfy Assumption 2 for the function $f(x) = p_{k,\mathcal{U},\beta}(x)$, if Algorithm 1 returns $\bar{x}_N \in \Theta$, then the expected value of the

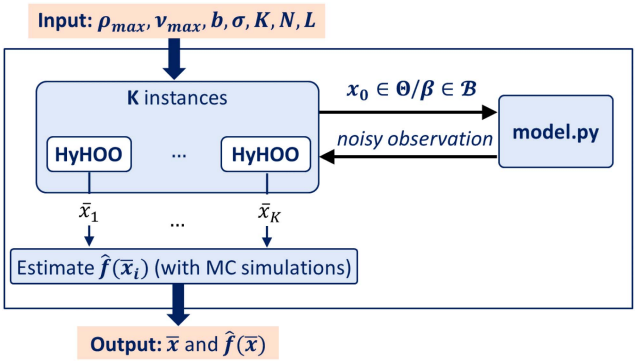


FIGURE 2. HyHooVer **components. Input parameters N, K, ($\rho_{\max}$, $\nu_{\max}$), b, and σ stand for sampling budget, number of HyHOO instances, upper bounds for smoothness parameters, batch size parameter, and noise parameter, respectively. We fix $K = 1$ and $\nu_{\max} = 1.0$ throughout all experiments in Section IV. We will discuss the choice of $\rho_{\max}$ and b later in this section.**

simple regret $S_N = p_{k,\mathcal{U},\beta}(x^*) - p_{k,\mathcal{U},\beta}(\bar{x}_N)$ is upper bounded by Theorem 1.

Proposition 2: Under the assumption that the noisy observations $y_1, \ldots, y_N$ in the synthesis problem satisfy Assumption 1, and the smoothness parameters (ν, ρ) satisfy Assumption 2 for the function $f(\beta) = \mathbb{E}[r(\alpha, \beta)|\beta]$, if Algorithm 1 returns $\bar{x}_N \in \mathcal{B}$, then the expected value of the simple regret $S_N = \mathbb{E}[r(\alpha, \beta)|\beta^*] - \mathbb{E}[r(\alpha, \beta)|\bar{x}_N]$ is bounded from above by Theorem 1.

The following remark discuss how Assumption 2 is satisfied by the verification and synthesis problems:

Remark 1: Assumption 2 asserts that choices of initial states that are near x^* (the initial states that make the system most unsafe) also lead to unsafe states in a verification problem. This requires the probability transition kernel to have a local Lipschitz property at $x^* \in \Theta$. For the safety verification scenarios considered in this article, this assumption implies that initial configurations of the cars that are close to the most unsafe configuration are also unsafe, which aligns with our understanding of the physical dynamics involved. A similar argument holds for the parameter synthesis as well. If the probability transition kernel has a local Lipschitz property at $x^* \in \mathcal{B}$, then Assumption 2 holds for parameter synthesis problems.

IV. HyHooVer TOOL, EVALUATION, AND DISCUSSION

We devote this section to explain the structure of HyHooVer, to introduce several benchmark scenarios and to carefully evaluate its performance over instances of these scenarios. It is noted that our experiments were conducted on a Linux workstation with two Xeon Silver 4110 CPUs and 32 GB RAM.

HyHooVer consists of several components, as shown in Fig. 2. To use HyHooVer, the user needs to provide a Python code for the model that needs to be verified or synthesized (model.py), along with some input parameters. These input parameters include the sampling budget N, the upper bounds for the smoothness parameters ($\nu_{\max}$, $\rho_{\max}$), the number of HyHOO instances K, the noise parameter σ, the batch size parameter b, and the number of modes L. Once the input is provided, HyHooVer runs K instances of HyHOO with automatically calculated smoothness parameters (see Algorithm 2). For each instance i, it generates random trajectories from the model, receives noisy observations (rewards) (see Algorithm 1, line 5), and returns an estimate $\bar{x}_i$ of the optimum. Once the sampling budget N is exhausted, HyHooVer returns the best $\bar{x}_i$ by comparing the mean reward for each $\bar{x}_i$ using Monte-Carlo simulations. For each $\bar{x}_i$, this is done by querying the model for n times (for some n determined by the user) and taking the average of the noisy observations returned by the model $\hat{f}(\bar{x}_i)$. Finally, HyHooVer outputs the $\bar{x}$ that gives the highest mean reward and its corresponding $\hat{f}(\bar{x})$.[5]

In the following subsection, we describe the benchmarks used to evaluate the performance of HyHooVer.

A. BENCHMARKS

We evaluate the performance of HyHooVer using a variety of scenarios including a synthetic example, an LQR example and autonomous driving scenarios. It is noted that each of these scenarios presents unique characteristics. The synthetic example and the LQR example are designed to show the application of HyHooVer in parameter synthesis. The synthetic example, in addition, allows us to evaluate the performance of HyHooVer in state spaces with higher dimensions and modes. The purpose of the driving scenarios including BrokenLidar and Roundabout is to demonstrate how HyHooVer performs in safety verification. These scenarios are created specifically to address the challenges of testing the safety of automatic braking and collision warning systems, which are becoming standard features in vehicles. According to statistics, over 25% of accidents involve rear-end collisions, and about 85% of these occur on straight roads [34]. The BrokenLidar and Roundabout scenarios are designed to capture these features. The BrokenLidar scenario assesses the performance of HyHooVer in rare event scenarios, while the Roundabout scenario is created using a car simulator called Highway-env. This simulator is equipped with motion planning algorithms and controllers that enable the creation of autonomous driving scenarios. The goal of the Roundabout scenario is to demonstrate the application and performance of HyHooVer in decision-making tasks related to autonomous driving. Examples of these scenarios are described in detail below.

1) Synthetic EXAMPLE

Let $\mathcal{Z} \subset \mathbb{R}^m$, and let $f : [L] \times \mathcal{Z} \to \mathbb{R}$ be defined as

$$f(x) = g(z) - a_l \times \mathbb{1}\{l \neq 1\}, \tag{5}$$

[5]The source code is available at https://github.com/NeginMusavi/HyHooVer.git.

where

$$g(z) = 0.5\sin(13z_1)\sin(27z_1) - \sum_{i=2}^{m} z_i^2,$$

with z_i the i-th component of z. Suppose we have access to noisy observations $y(x) = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ for some $\sigma_\epsilon > 0$. Our goal is to solve the optimization problem

$$\sup_{x\in\mathcal{B}} f(x)$$

for a given set $\mathcal{B} \subset [L] \times \mathcal{Z}$. This problem is similar to the one presented in (3) in Section II. The function $\sin(13z_1)\sin(27z_1)$ is an example of a function with a near-optimality dimension $d_m = 0$, as introduced in [5]. This means that the near-optimal nodes of the function can be bounded by a constant B as defined in Definition 2. As a consequence, the near-optimality dimension of both g and f is also equal to zero.

2) LQR EXAMPLE

Consider the following linear system in discrete time:

$$x_{t+1} = Ax_t + Bu_t,$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are unknown matrices, and we are interested in finding a state-feedback gain matrix W that minimizes the cost function $c(W)$ for the control policy $u_t = -Wx_t$. The cost function is defined as follows:

$$c(W) = \mathbb{E}_{x_0\sim\mathcal{D}}\left[\sum_{t=0}^{\infty}(x_t^T Q x_t + u_t^T R u_t)\right],$$

where $\mathcal{D}$ is a distribution that initial state x_0 is randomly drawn from $\mathcal{N}(\bar{x}_0, \sigma_\epsilon^2 I)$ for some $\bar{x}_0$ and σ_ϵ, and matrices $Q \in \mathbb{R}^{n\times n} \succ 0$ and $R \in \mathbb{R}^{m\times m} \succ 0$ that parameterize the cost. The optimization problem we want to solve is

$$\sup_{W\in\mathcal{B}} -c(W),$$

where $\mathcal{B}$ is a given set and we only have access to noisy observations $\sum_{t=0}^{\infty}(x_t^T Q x_t + u_t^T R u_t)$ with $x_0 \sim \mathcal{D}$. This optimization problem is similar to (3) in Section II.

3) BrokenLidar SCENARIO

This scenario models a car running on a single-lane road and a pedestrian crossing the road in front of the car. The car is equipped with a broken Lidar device that somehow cannot detect obstacles in a specific direction. If the car detects the pedestrian, it starts braking. The probability of detecting the pedestrian is a function of the relative angle and distance (θ, d) between the car and the pedestrian is given by

$$p(\theta, d) = \left(1 - \exp\left(-\frac{(\theta-\theta_b)^2}{s}\right)\right)\left(\max\left\{0, \frac{d_b - d}{d_b}\right\}\right)^2,$$

where θ_b is a constant angle along which the pedestrians cannot be detected, and d_b is the maximum distance that the Lidar can cover. Here s is a parameter controlling how fast the probability decreases when $\theta \to \theta_b$. The detecting probability also declines as the distance d increases. This model is almost always safe unless $\theta = \theta_b$ holds constantly for every step. This rarely happens when the speed and initial position of the car and the pedestrian satisfy some constraints. This is a realistic verification problem in the sense that the system is almost always safe unless it triggers some bugs. The goal of verification is to find such rare unsafe cases before deployment.

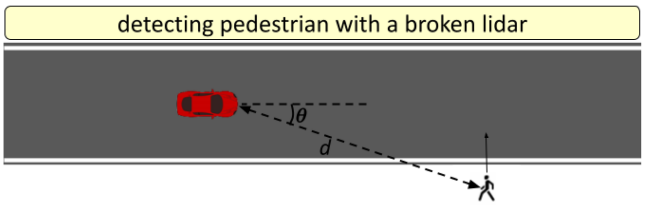


FIGURE 3. BrokenLidar **scenario.**

Let unsafety be defined as where the distance between the car and pedestrian falls below a certain threshold. We aim to solve the optimization problem for a given $k, \mathcal{U}$ and s

$$\sup_{x_0\in\Theta} p_{k,\mathcal{U},s}(x_0),$$

where Θ is a given set of possible initial position and speed of the car and the pedestrian. This optimization problem is similar to (2) in Section II.

4) Roundabout SCENARIO

This scenario is described in detail in Section I-B and is illustrated in Fig. 1.

It is worth noting that the unsafety probability/objective function in all these scenarios exhibits nonlinearity concerning the states/parameters.

B. PERFORMANCE OF HyHooVer WITH VARIOUS BATCH SIZE AND SMOOTHNESS PARAMETERS

The batch size parameter in the tool serves as a hyperparameter designed to reduce running time and memory usage. However, it requires careful consideration to ensure that it does not compromise the accuracy of the verification or synthesis process. Theoretical analysis, as presented in Theorem 1, shows that the simple regret concerning the batch size b scales as $\mathcal{O}((\frac{b}{N})^{\frac{1}{d_m+2}})$ with a sampling budget of N. In Fig. 4, the convergence rate of the actual regret and the theoretical regret rate is depicted for instances of the Synthetic example with different sampling budgets $N = 50K, 100K, 300K$. The actual regret rate is shown to be upper-bounded by the theoretical rate provided by the theorem. While larger batch sizes can negatively impact the regret, choosing an appropriate batch size parameter b can significantly improve the running time and memory usage of HyHooVer without substantially sacrificing the quality of the final result. This observation is supported by Fig. 5 and Table 1. Fig. 5 illustrates the impact of the batch size parameter b on the performance of HyHOO for the Synthetic example. When the sampling budget exceeds $100K$, the performance of

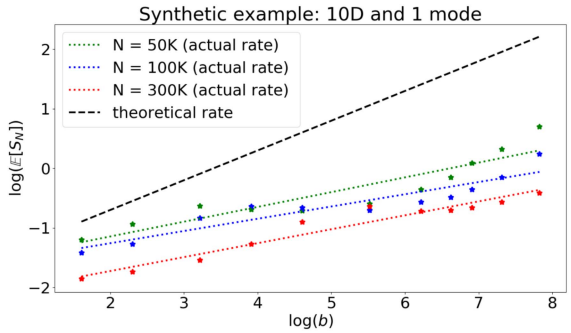


FIGURE 4. **Actual regret rate and theoretical rate versus different batch sizes b on an instance of the** Synthetic **example with $m = 10$, $L = 1$, and $\sigma_\epsilon = 0.1$. Here, $\rho_{\max} = 0.95$, and any data represented by "*" is averaged over 100 runs. It is noted that the lines are fitted to the data using the least square method, and the slopes of the lines are 0.25, 0.21, 0.22 for the three sampling budgets, respectively.**

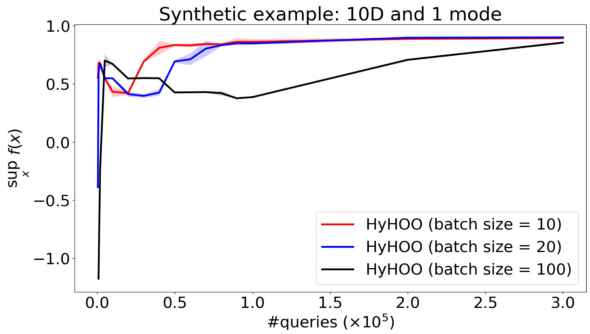


FIGURE 5. **Impact of batch size b on performance of** HyHooVer **on instance of** Synthetic **example with $m = 10$, $L = 1$ and $\sigma_\epsilon = 0.1$. Here $\rho_{\max} = 0.95$ and results are averaged over 10 runs.**

TABLE 1 **Impact of batch size b on tree size, running time, memory usage, and results returned by** HyHooVer **on instance of** Synthetic **example with $m = 10$, $L = 1$ and $\sigma_\epsilon = 0.1$. Here $N = 100$K, $\rho_{\max} = 0.95$ and results are averaged over 10 runs.**

Batch Size b	10	20	100
#Nodes	9988	4994	998
Running Time (s)	1292	300	24
Memory (Mb)	36.52	18.70	3.58
Result	0.86	0.85	0.39

HyHOO remains unaffected for batch sizes $b \leq 20$, but it is negatively influenced when $b = 100$. For batch size $b = 100$, more sampling budget is required to achieve similar performance compared to batch sizes $b \leq 20$.

On the other hand, Table 1 demonstrates that with a sampling budget of $100K$ and batch size $b \leq 20$, we can achieve comparable results with a smaller tree size, reduced memory usage, and faster running time. In essence, finding the optimal value for the hyperparameter b, which minimizes running time while keeping the actual regret below a specific threshold, is challenging to calculate precisely. Nevertheless, based on our observations, we recommend using a batch size ranging from 5 to 20 to achieve faster running times while maintaining satisfactory results.

TABLE 2 **Impact of smoothness parameter $\rho_{\max}$ on the results returned by** HyHooVer **on instance of** Synthetic **example with $m = 10$, $L = 1$ and $\sigma_\epsilon = 0.1$. Here $N = 200$K, $b = 20$ and results are averaged over 10 runs.**

$\rho_{\max}$	0.95	0.9	0.85	0.8	0.75	0.5	0.25
depth	51.0	53.5	53.3	56.0	56.2	56.1	56.7
Result	0.900	0.900	0.901	0.901	0.898	0.901	0.902

As discussed in Section III-B, it is not always practical to determine the exact upper bound for ρ. To investigate the impact of different choices of $\rho_{\max}$ on the performance of HyHooVer, experiments were conducted and the results are recorded in Table 2. According to the results, the performance of the tool is not influenced by different choices of $\rho_{\max}$ for the Synthetic example. However, when $\rho_{\max}$ is higher, the tool tends to perform a more aggressive search, resulting in more thorough exploration of the shallower levels of the tree due to uncertainty in observations over the region of state space associated with each node. Therefore, if the smoothness of the model is unknown, a higher $\rho_{\max}$ can be chosen as a conservative approach, which still provides reasonably good estimates.

C. PERFORMANCE OF HyHooVer IN COMPARISON WITH OTHER METHODS

We found that among the available SMC tools, PlasmaLab [35] is the most similar to HyHooVer in terms of its ability to verify black-box systems with continuous/hybrid state spaces. It should be noted that Storm [36] and PRISM [26] do not have support for continuous state-space models. While it would be possible to compare HyHooVer with these tools using discrete versions of the examples, the comparison would not be appropriate as the guarantees offered by these tools are different. The SMC approach in [30], is related, but we were unable to find an implementation to compare against. Given this we conduct a more in-depth comparison with PlasmaLab. In addition, we conduct experiments with BoTorch [31] which is a Bayesian optimization based tool for black-box optimization of stochastic system and we provide in-depth comparison with HyHooVer's performance. For scenarios involving a hybrid state space, we also compare the performance of our tool with a traditional approach called baseline HOO. More information about these approaches is provided below:

- **PlasmaLab**: It utilizes a smart sampling algorithm [37] to efficiently distribute the simulation budget among the schedulers of an MDP. To use this algorithm, one must set the parameters ϵ and δ in the Chernoff bound, with $N_{\max} > \ln(2/\delta)/(2\epsilon^2)$, where $N_{\max}$ is the per-iteration simulation budget. We set the confidence parameter δ to 0.01, and then, for a given $N_{\max}$, obtained the precision parameter ϵ by $\epsilon = \sqrt{\ln(2/\delta)/(2 \times 0.8 \times N_{\max})}$. To make a fair comparison, we developed a PlasmaLab

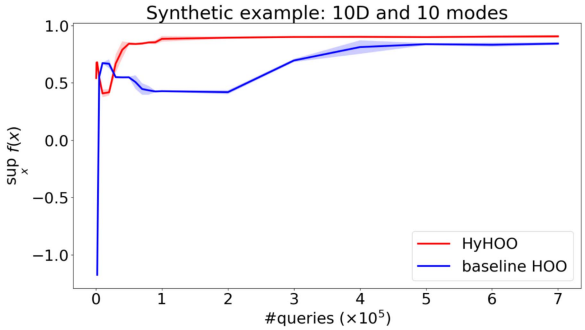


FIGURE 6. Performance of HyHooVer in comparison with baseline HOO on instance of Synthetic example with $m = 10$, $L = 10$, and $\sigma_\epsilon = 0.1$.

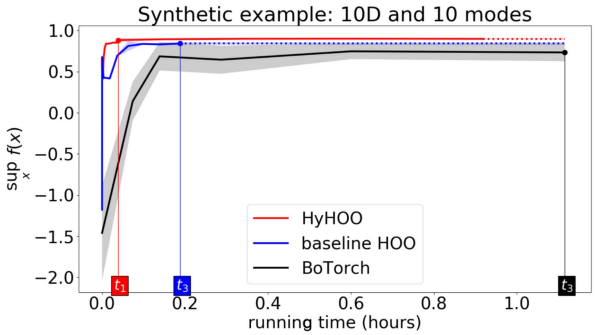


FIGURE 7. Performance of HyHooVer in comparison with baseline HOO and BoTorch on instance of Synthetic example with $m = 10$, $L = 10$ and $\sigma_\epsilon = 0.1$. Note $t_3 \sim 30 \times t_1$ and $t_3 \sim 6 \times t_2$.

plugin that enables PlasmaLab to use the same external Python simulator as HyHooVer. It is noted that the comparisons with PlasmaLab are conducted over safety verification scenarios.

- **BoTorch**: It is a software tool that is designed for black-box optimization of expensive-to-evaluate functions through Bayesian optimization techniques. It is a PyTorch-based library that can be utilized in a range of situations, including hyperparameter optimization for machine learning models, as well as scientific and engineering problems. To utilize BoTorch for our specific scenarios, we employ its Ax module. It is worth mentioning that we compare BoTorch with our tool in both parameter synthesis and safety verification scenarios.
- **Baseline HOO**: A common approach for addressing a problem with multiple modes is to divide the budget equally among the modes and conduct multiple HOOs and then compare the results. The theoretical guarantee of this approach is discussed in Section III-B.

The evaluation of the four approaches (HyHooVer, PlasmaLab, BoTorch, and baseline HOO) involves the use of two metrics: sample complexity and running time. More information on the detail of these comparisons is provided as we go through the scenarios. Unless otherwise stated, we fix the batch size parameter to $b = 10$ and the maximum smoothness parameter to $\rho_{\max} = 0.95$ for all experiments conducted using HyHooVer. Additionally, we present the results of 10 independent runs for each approach.

1) PERFORMANCE OF *HyHooVer* ON PARAMETER SYNTHESIS

We tested HyHooVer on an instance of the Synthetic example with $m = 10$, $L = 10$ to evaluate the performance of HyHOO in high-dimensional state space with multiple modes. The results shown in Fig. 6 indicate that HyHOO needs fewer queries from the system compared to the baseline HOO, which is consistent with the theoretical guarantees provided for both methods in Section III-B.

Examining the plots more closely, it is evident that both methods show a decline in performance after initial improvement. This is mainly due to the non-monotonic behavior of the function $f(x)$ in (5). Specifically, the UCB algorithm initially discovers regions of the state space that is close to optimal, but as it continues to explore, it finds other regions that are also near optimal but separate from the initial ones, and it then chooses to explore further to reduce the uncertainty of its observations.

Remark 2: We did not compare the running time performance of our tool with PlasmaLab, despite noticing a significant lower running time of HyHOO compared to PlasmaLab to reach the same level of performance in the BrokenLidar and Roundabout scenarios. This decision was made because the PlasmaLab plug-in is developed in Java, and we used a simulation bridge between Python and PlasmaLab. The difference in performance could potentially be attributed to communication issues between the two languages, making a comparison between the two tools unfair.

We conducted a comparison between our tool and BoTorch by testing their performance on the Synthetic example in terms of their running time. The results are illustrated in Fig. 7. The graph demonstrates that HyHOO outperforms baseline HOO and BoTorch in terms of running time. Notice HyHOO achieves a result after only 0.04 hours of running time, whereas BoTorch could not achieve this level of performance even after running for 1.12 hours, indicating that HyHOO is approximately 30 times faster than BoTorch. It is noted that we expect the other approaches to achieve similar results as HyHOO achieves in a longer run; however, due to high computational times, we decided not to continue the runs for the other methods. In addition, note that BoTorch needed 1250 queries to achieve this result, whereas HyHOO requires $100K$ queries to achieve these results. Moreover, with 1250 queries, HyHOO attains a performance result of 0.677 ± 0.006, while BoTorch achieves a performance result of 0.734 ± 0.106 with the same number of queries. In summary, while HyHOO performs better in terms of running time, BoTorch performs better in terms of number of queries.

Remark 3: We did not compare the performance of HyHooVer and BoTorch based on the number of queries due to the long running time required for BoTorch to obtain its results.

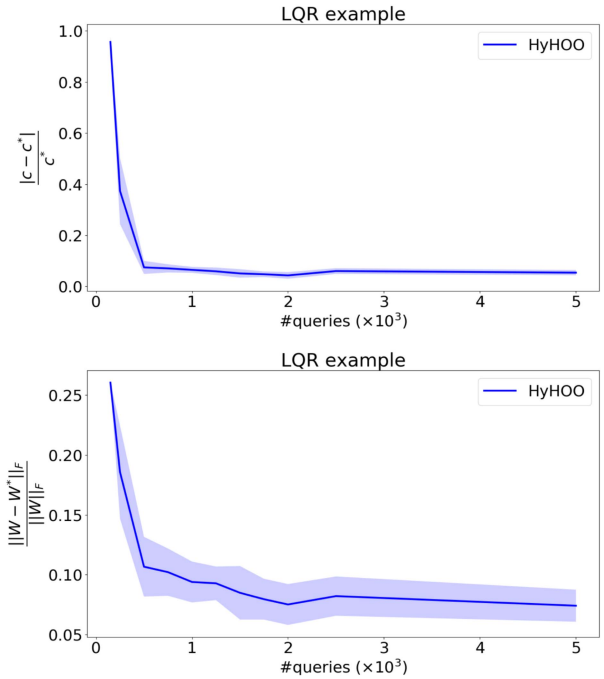


FIGURE 8. **Performance of** HyHooVer **on instance of** LQR **example with** $n = 2, m = 1$, **and** $\sigma_\epsilon^2 = 0.001$. **Here** c **refers to the value of cost evaluated at** W **returned by** HyHooVer. **In addition,** W^* **and** c^* **refers to the optimal state feedback gain and optimal cost computed by solving Algebraic Riccati Equation, respectively.**

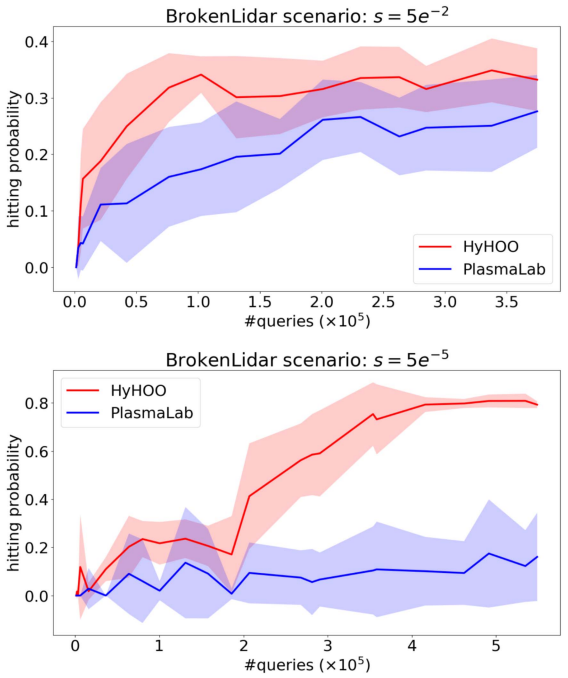


FIGURE 9. **Performance of** HyHooVer **in comparison with PlasmaLab on instance of** BrokenLidar **scenario with** $m = 4$ **and** $L = 1$.

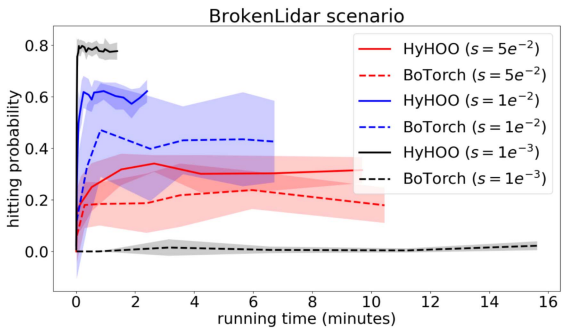


FIGURE 10. **Performance of** HyHooVer **in comparison with BoTorch on instance of** BrokenLidar **scenario with** $m = 4$ **and** $L = 1$.

To explain further, Bayesian optimization is computationally expensive for two main reasons: (1) it requires an iterative numerical optimization step to fit a probabilistic model to the observed data, and (2) it involves optimizing an acquisition function to select the next point for evaluation at each iteration. In contrast, HyHooVer does not require such optimization steps, which is why it has a faster running time compared to BoTorch. Therefore, in scenarios where the cost of evaluating the system at each iteration is high and dominates the optimization process, BoTorch is expected to perform better than HyHooVer. However, in scenarios where the opposite is true, HyHooVer is expected to outperform BoTorch. The Synthetic example is an instance of the latter scenario.

We tested HyHooVer on an instance of the LQR example with $n = 2, m = 1$ to demonstrate its application in parameter synthesis in an optimal control setting. This problem has an analytical solution which can be computed by solving Algebraic Riccati Equation. The results shown in Fig. 8 indicate the accuracy of estimation is improved with increasing the sampling budget.

2) *PERFORMANCE OF HyHooVer ON SAFETY VERIFICATION*

We tested HyHooVer on instances of the BrokenLidar scenario with $m = 4, L = 1$ and different s parameters. Different parameters s result in functions with different slopes around their optimum. The smaller s corresponds to rarer unsafe event. The results shown in Fig. 9 demonstrates comparison between performance of HyHooVer with $s = 5e^{-5}$, and PlasmaLab with $s = 5e^{-2}$. It suggests that as the unsafe event gets rarer (with $s = 5e^{-5}$), HyHOO's performance is superior to PlasmaLab in terms of number of queries. This is because as s decreases, the number of near-optimal regions also decreases and from a theoretical standpoint, this corresponds to the smaller constant B introduced in Definition 2 which also corresponds to a lower simple regret in Theorem 1. Thus HyHOO offers an advantage if the function exhibits a sharper slope around the maxima in comparison with methods that rely on random sampling like PlasmaLab.

Fig. 10 displays the performance of HyHOO and BoTorch in terms of running time with different values of s parameter $s = 5e^{-2}, s = 1e^{-2}$, and $s = 1e^{-3}$. The graph indicates that as the value of s decreases, which implies a rarer event, HyHOO

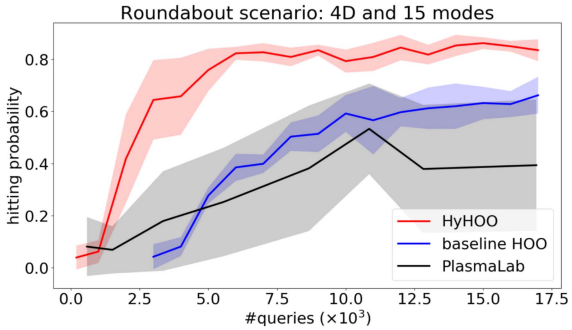


FIGURE 11. Performance of HyHooVer **in comparison with baseline HOO and PlasmaLab on** Roundabout **scenario with $m = 4$ and $L = 15$.**

outperforms BoTorch. In the case $s = 1e^{-3}$, this difference became even more significant. In this case HyHOO achieved its best performance in roughly 2 minutes, whereas BoTorch after 16 minutes still could not achieve one-fifth of HyHOO's result. It is noted that the BoTorch achieves these results with 500 queries. This scenario is another example of the scenarios where the cost of evaluating the system at each iteration is dominated by the cost of the optimization process in BoTorch and as a result HyHooVer performs better than BoTorch.

Finally, we ran HyHooVer on instances of the Roundabout scenario with $m = 4, L = 15$ and compared its performance with both the baseline HOO and PlasmaLab. As shown in Fig. 11, HyHOO generally performs better than the baseline HOO, which supports our claim that HyHooVer is effective when the number of modes increases. This is because the baseline HOO approach spends a significant portion of the budget on exploring regions of the state space associated with modes that do not contain the optimum. In addition, HyHooVer is more sample-efficient than PlasmaLab because the later relies on random exploration which is not efficient in state spaces with multiple modes.

V. CONCLUSION AND FUTURE FOCUS

We introduced HyHOO, an optimistic mini-batched tree search algorithm designed for verification and parameter synthesis in a specific class of discrete-time Markov chains. These Markov chains have states consisting of both discrete and continuum-based components. HyHOO operates by sequentially sampling executions of the Markov chain in batches, leveraging a mild assumption about the smoothness of the objective function. Its objective is to find solutions that are near-optimal, minimizing the corresponding regret. We provided theoretical regret bounds that consider factors such as the sampling budget, smoothness, near-optimality dimension, and sampling batch size. Importantly, HyHOO exhibits effective performance without requiring exact parameters or quantities. We created a tool named HyHooVer and assessed its effectiveness by analyzing various benchmark models and we showed that our approach competes favorably with BoTorch (a Bayesian-based tool), PlasmaLab (a tool based on random sampling), and the baseline HOO method, in terms of sample efficiency and/or running time.

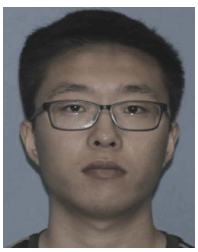
Our current approach focuses on minimizing regret but could be extended to minimizing the sampling budget. This presents an intriguing area for further investigation and exploration.

REFERENCES

[1] W. R. Thompson, "On the theory of apportionment," *Amer. J. Math.*, vol. 57, no. 2, pp. 450–456, 1935.
[2] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. Amer. Math. Soc.*, vol. 58, no. 5, pp. 527–535, 1952.
[3] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.
[4] R. Munos et al., "From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning," *Foundations Trends Mach. Learn.*, vol. 7, no. 1, pp. 1–129, 2014.
[5] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *J. Mach. Learn. Res.*, vol. 12, no. 5, pp. 1655–1695, 2011.
[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, 2002.
[7] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," in *Proc. 23rd Conf. Uncertainty Artif. Intell.*, 2007, pp. 67–74.
[8] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proc. Eur. conf. Mach. Learn.*, 2006, pp. 282–293.
[9] R. Sen, K. Kandasamy, and S. Shakkottai, "Noisy blackbox optimization using multi-fidelity queries: A tree search approach," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2096–2105.
[10] J.-B. Grill, M. Valko, and R. Munos, "Black-box optimization of noisy functions with unknown smoothness," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2015, pp. 667–675.
[11] N. Musavi, D. Sun, S. Mitra, G. Dullerud, and S. Shakkottai, "Hoover: A framework for verification and parameter synthesis in stochastic systems using optimistic optimization," in *Proc. IEEE Conf. Control Technol. Appl.*, 2021, pp. 923–930.
[12] B. Boyer, K. Corre, A. Legay, and S. Sedwards, "PLASMA-lab: A flexible, distributable statistical model checking library," in *Proc. 10th Int. Conf. Quantitative Eval. Syst.*, 2013, pp. 160–164.
[13] E. Leurent, "An environment for autonomous driving decision-making," 2018. [Online]. Available: https://github.com/eleurent/highway-env
[14] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.
[15] I. Tkachev and A. Abate, "On infinite-horizon probabilistic properties and stochastic bisimulation functions," in *Proc. IEEE 50th Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 526–531.
[16] S. E. Z. Soudjani and A. Abate, "Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes," *SIAM J. Appl. Dynamical Syst.*, vol. 12, no. 2, pp. 921–956, 2013.
[17] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. Int. Workshop Hybrid Syst.: Computat. Control*, vol. 2004, pp. 477–492.
[18] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Trans. Autom. Control*, vol. 66, no. 7, pp. 3097–3110, Jul. 2021.
[19] A. Legay and S. Sedwards, "On statistical model checking with PLASMA," in *Proc. 8th Int. Symp. Theor. Aspects Softw. Eng.*, 2014, pp. 139–145.
[20] H. L. S. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *Proc. Int. Conf. Comput. Aided Verification*, 2002, pp. 223–235.
[21] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Proc. 17th Int. Conf. Comput. Aided Verification*, 2005, pp. 266–280.
[22] H. L. S. Younes, "Probabilistic verification for "black-box" systems," in *Proc. Comput. Aided Verification, 17th Int. Conf.*, 2005, pp. 253–265.
[23] A. Hartmanns and H. Hermanns, "A modest approach to checking probabilistic timed automata," in *Proc.6th Int. Conf. Quantitative Eval. Syst.*, 2009, pp. 187–196.

[24] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke, "Statistical model checking for Markov decision processes," in *Proc. IEEE 9th Int. Conf. Quantitative Eval. Syst.*, 2012, pp. 84–93.
[25] A. David et al., "On time with minimal expected cost!," in *Proc. Int. Symp. Automated Technol. Verification Anal.*, 2014, pp. 129–145.
[26] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2006, pp. 441–444.
[27] R. Lassaigne and S. Peyronnet, "Approximate planning and verification for large markov decision processes," *Int. J. Softw. Tools Technol. Transfer*, vol. 17, no. 4, pp. 457–467, 2015.
[28] A. Hartmanns and H. Hermanns, "The modest toolset: An integrated environment for quantitative modelling and verification," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2014, pp. 593–598.
[29] C. E. Budde, P. R. D'Argenio, A. Hartmanns, and S. Sedwards, "A statistical model checker for nondeterminism and rare events," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2018, pp. 340–358.
[30] C. Ellen, S. Gerwinn, and M. Fränzle, "Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains," *Int. J. Softw. Tools Technol. Transfer*, vol. 17, no. 4, pp. 485–504, 2015.
[31] M. Balandat et al., "BoTorch: A framework for efficient monte-carlo bayesian optimization," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21524–21538.
[32] C. Ionescu Tulcea, "Mesures dans les espaces produits," *Atti Accad. Naz. Lincei Rend*, vol. 7, pp. 208–211, 1949.
[33] D. Petritis, "Markov chains on measurable spaces," Apr. 2012. [Online]. Available: https://perso.univ-rennes1.fr/dimitri.petritis/ps/markov.pdf
[34] K. Kodaka, M. Otabe, Y. Urai, and H. Koike, "Rear-end collision velocity reduction system," SAE Int., Warrendale, PA, USA, Tech. Rep. 2003-01-0503, 2003.
[35] A. Legay, S. Sedwards, and L.-M. Traonouez, "Plasma lab: A modular statistical model checking platform," in *Proc. Int. Symp. Leveraging Appl. Formal Methods*, 2016, pp. 77–93.
[36] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Proc. Int. Conf. Comput. Aided Verification*, 2017, pp. 592–600.
[37] P. D'Argenio, A. Legay, S. Sedwards, and L.-M. Traonouez, "Smart sampling for lightweight verification of markov decision processes," *Int. J. Softw. Tools Technol. Transfer*, vol. 17, no. 4, pp. 469–484, 2015.

NEGIN MUSAVI (Student Member, IEEE) received the M.Sc. degree in mechanical engineering from Bilkent University, Ankara, Trkiye, in 2017. She is currently working toward the Ph.D. degree with the Mechanical Science and Engineering Department, University of Illinois Urbana Champaign, Urbana, IL, USA. Her research interests include control, optimization, and machine learning.



DAWEI SUN (Student Member, IEEE) received the undergraduate degree in automation from Tsinghua University, Beijing, China. He is currently a graduate with the University of Illinois at Urbana-Champaign, Urbana, IL, USA. His research is focused on the design and analysis of safety-critical autonomous systems. His research utilizes advances in machine learning, control theory, and formal methods.

SAYAN MITRA (Member, IEEE) received the undergraduation degree in electrical engineering from Jadavpur University, Kolkata, India, the master's degree in computer science from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in EECS from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2007. After one year of postdoctoral work with the Center for Mathematics of Information, California Institute of Technology, Pasadena, CA, USA, he joined the University of Illinois at Urbana-Champaign, Urbana, IL, USA, where he is currently a Professor with the Electrical and Computer Engineering Department. His research interests include formal methods, hybrid systems, autonomous systems, and verification of cyber-physical systems and their applications. His work has been recognized by National Science Foundation's CAREER Award, Air Force Office of Scientific Research Young Investigator Program Award, IEEE-HKN C. Holmes MacDonald Outstanding Teaching Award, and several best article awards. He was the Program Co-Chair of the 20th International Conference on Hybrid Systems.

GEIR E. DULLERUD (Fellow, IEEE) received the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K., in 1994. Since 1998, he has been on faculty with the University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, USA, where he is currently the W. Grafton and Lilian B. Wilkins Professor of Mechanical Engineering, and Director of the Center for Autonomy. Previous to this, he was an Assistant Professor of applied mathematics with the University of Waterloo, Waterloo, ON, Canada, and a Research Fellow and Lecturer with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, USA. His research interests include games and learning for control, robotic vehicles, hybrid dynamical systems, and cyberphysical systems security. Dr. Dullerud was the recipient of the National Science Foundation CAREER Award in 1999 and Xerox Faculty Research Award at UIUC in 2005. He became a Fellow of the American Society of Mechanical Engineers in 2011.

SANJAY SHAKKOTTAI (Fellow, IEEE) received the Ph.D. degree from the ECE Department, University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2002. He is with The University of Texas at Austin, Austin, TX, USA, where he is currently the Temple Foundation Endowed Professor No. 4 and a Professor with the Department of Electrical and Computer Engineering. His research interests lie at the intersection of algorithms for resource allocation, statistical learning and networks, with applications to wireless communication networks, and online platforms. He was the recipient of the NSF CAREER Award in 2004.