HooVer: A Framework for Verification and Parameter Synthesis in Stochastic Systems using Optimistic Optimization

Negin Musavi[1], Dawei Sun[1], Sayan Mitra[1], Geir Dullerud[1] and Sanjay Shakkottai[2]

***Abstract*— This paper provides a new approach for probabilistic verification of control and dynamical systems in the scenario where there is a finite computational budget that must be used judiciously; it is based on leveraging multi-armed bandits theory from machine learning. We present an algorithm for formal verification and parameter synthesis of continuous state-space Markov chains, introduce our associated computational tool HooVer, and demonstrate their use on example applications. The problem class considered captures the design and analysis of a wide variety of autonomous and cyber-physical systems defined by nonlinear and black-box modules. In order to solve these problems, one has to maximize certain probabilistic objective functions over all choices of initial states and parameters. In this paper, we identify the assumptions that make it possible to view this problem as a multi-armed bandit problem. Based on this fresh perspective, we propose an algorithm *Hierarchical Optimistic Optimization algorithm with Mini-batches (HOO-MB)* for solving the problem that carefully instantiates an existing bandit algorithm—Hierarchical Optimistic Optimization—with appropriate parameters. As a consequence, we obtain theoretical regret bounds on sample efficiency of our solution that depend on key problem parameters like smoothness, near-optimality dimension, and batch size. The batch size parameter enables us to strike a balance between the sample efficiency and the memory usage of the algorithm. Experiments, using our open-source tool HooVer, suggest that the approach scales to realistic-sized problems and is often more sample-efficient compared to PlasmaLab—a leading tool for verification of stochastic systems. Specifically, HooVer has distinct advantages in analyzing models in which the objective function has sharp slopes. In addition, HooVer shows promising behavior in parameter synthesis for a linear quadratic regulator (LQR) example.**

I. Introduction

In this paper, we present a new verification and parameter synthesis algorithm of discrete-time Markov chains (MC) over continuous state-spaces and uncountable sets of initial states or parameters, where the exact knowledge of probabilistic evolution of states is not known. In other words, we want to find what choices of initial states or parameters maximize certain probabilistic objective functions over all choices of initial states or parameters, without having exact knowledge of the system dynamics. For instance, predictive monitoring or runtime verification with worst case state estimation error bounds (e.g., GPS and senor tolerances) naturally lead to this problem. Our approach is based on multi-armed bandit theory, which provides a powerful idealized model for online decision-making in settings where the statistics are unknown. By taking advantage of the fact that multi-armed bandit methods do not rely on exact knowledge of the system dynamics, we gain not only new algorithms for verification and parameter synthesis, but also new types of bounds on the sample efficiency.

A. Main Contributions

The main contributions of our work are as follows:

(i) We formalize the above verification and synthesis problems so that they can be solved using multi-armed bandit algorithms (Propositions 1 and 2).

(ii) We present a tree-based algorithm called *Hierarchical Optimistic Optimization algorithm with Mini-Batches (HOO-MB)* for solving the above problems (Algorithm 1). HOO-MB modifies the *hierarchical optimistic optimization (HOO)* algorithm of [1] by taking advantage of batched simulations and simultaneously reducing the impact of variance from noisy samples and keeping the tree size small.

(iii) We provide theoretical bound (Theorem 1) on the difference between the maximum value $f(\bar{x}_N)$ computed by HOO-MB and the actual maximum $f(x^*)$, as a function of the sampling budget N, the batch size parameter, the smoothness parameters, and the *near-optimality dimension* of f. This bound is fundamentally different from the existing performance bounds in the statistical model checking literature. For example, the bounds relevant for tools like PlasmaLab [2], use Monte Carlo sampling, Chernoff bounds, or sequential hypothesis testing.

(iv) We have implemented HOO-MB in an open-source tool called HooVer and we have created a suite of benchmarks models inspired by typical scenarios used for certification of advanced driving assist systems [3]. The tool and all the benchmarks are available online[1]. The user only has to provide a `Python` class specifying the transition kernel, the unsafe set, parameter and initial set uncertainties to create new examples. We show that HOO-MB can help dramatically reduce the number of nodes in the tree and therefore reduce the running time and memory usage in these benchmarks. As expected, the quality of the verification result (in this case, maximum probability of hitting an unsafe state) improves with the sampling budget N. HooVer scales to reasonably large models: It easily handled models with 18-dimensional state-spaces, and initial uncertainty spanning 8 dimensions, on a standard computer. Running time and memory usage can be controlled with the sampling batch sizes, and HooVer is relatively insensitive to the smoothness parameters.

[1]The authors are with the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, USA

[2]The author is with the Department of Electrical and Computer Engineering at The University of Texas, Austin, TX 78712, USA

[1]https://www.daweisun.me/hoover/.

B. Comparison with Related Approaches

There is a large body of work on model-based approaches for verification and parameter synthesis of stochastic systems [4], [5], [6], [7], [8](and the references therein). These approaches rely on detailed knowledge of the probability transition kernel, which may not always be available.

Statistical model checking (SMC) also solves the same family of verification problems. SMC approaches collect empirical samples (through executions) and use statistical tests to determine if there is evidence to determine if the constraints have/have-not been violated [9], [10], [11], [12]. The notable methods related to this class include MODEST for probabilistic automata [13], PlasmaLab [2], the learning-based algorithm of [14], [15] in PRISM [16] and UPPAAL, and approaches for Markov Decision Processes with restricted classes of schedulers implemented in [17], [18], [19].

Empirical comparison of HooVer with other discrete-state SMC tools is complicated because the guarantees are different and platform specific constants are difficult to factor out. We present a careful comparison with PlasmaLab in Section IV-C. HooVer generally gets closer to the correct answer with fewer samples than PlasmaLab. For models with sharp slopes around the maxima, HooVer is more sample efficient. This suggests that HOO-MB may work with fewer samples in verification problems around hard to find bugs. The approach of [20] uses the original HOO algorithm of [1] but we could not find this tool online for running comparative experiments. Our approach differs from [20] in two important ways: (1) we use a search algorithm spawning different smoothness parameters and return the result of the best one; and (2) we exploit batched simulations. Our preliminary results were presented in a workshop [21][2]. We have also evaluated the performance of HooVer for parameter synthesis for a linear quadratic regulator (LQR) example, and the results show the efficiency of HooVer in tuning controller parameters.

Multi-armed bandits are a class of algorithms that learn unknown systems through adaptive sampling. These algorithms have a rich history [22], [23], [24], [25], with significant advances in both algorithmic and application aspects over the last decade [26], [1]. Most relevant to our work is their application to blackbox optimization, where the goal is to maximize an unknown function $f(\cdot)$ to which we only have noisy query access (i.e. a query of x returns $f(x)+$ noise). A prominent bandit algorithm is the well-known Upper Confidence Bound (UCB) [27] that popularized the Principle of Optimism for adaptive search. This principle has been successfully applied to blackbox optimization by recasting the problem as a tree search in [28], [29], [1], [26]. These algorithms use tree-structured queries to adaptively search over the unknown function's domain, and can optimize for the trade-off between exploiting the previously known high-value regions in the function's domain and exploring less sampled regions, to determine close approximations to the optimal solution for a given sampling budget. To provide theoretical guarantees, these works require some smoothness of the objective function. While [1] requires smoothness with respect to a semi-metric, [30], [31] relax the requirement of an exact semi-metric that captures the smoothness of f, but instead work with two smoothness parameters. For the cases where estimating these parameters is not practical, [31] provides an algorithm to search for different parameters in parallel, and [30] extends this to multi-fidelity settings.

[2]No workshop proceedings were published.

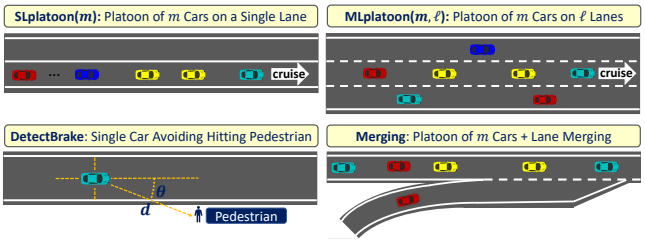

Fig. 1. Benchmark scenarios which can be instantiated with different number of vehicles and initialization.

C. Motivating Examples

We consider several scenarios with platoons of vehicles on the highway (Figure 1). At each time-step, each vehicle probabilistically decides to either cruise, brake, or accelerate, based on the distance to neighboring cars. For example, if the distance to the leading car falls below a certain threshold then the following car brakes after a certain "reaction time" period. In the scenario we call SLplatoon we have a single lane with m vehicles; MLplatoon has multiple lanes. The state space of the system includes the positions and the velocities of all the vehicles. The uncertainty in the initial state comes from the published standard GPS and IMU errors. We would like to find the most unsafe initial position configuration of these cars, where unsafety is defined as collision. Significant fraction of highway collisions occur because of rearend accidents [32], a significant fraction of highway accidents and certification of Automatic emergency braking (AEB) and adaptive cruise control (ACC) systems require analyzing precisely these types of scenarios [3], [33], [34]. Detailed descriptions and Python simulators for our models are available from the HooVer webpage.

II. Model and Problem Statement

Background and Notations: 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 (i) for all $x \in \mathcal{X}$, $\mathbb{P}(x, \cdot)$ is a probability measure on $\mathcal{F}_{\mathcal{X}}$; and (ii) 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 $\beta \in \mathbb{R}^m$. 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* Nondeterministic Markov chain (NMC) $\mathcal{M}$ *is defined by a quadruplet* $((\mathcal{X}, \mathcal{F}_\mathcal{X}), \mathbb{P}_\beta, \mathcal{B}, \Theta)$*, with: (i)* $(\mathcal{X}, \mathcal{F}_\mathcal{X})$*, a measurable space over the state space* $\mathcal{X}$*; (ii)* $\mathbb{P}_\beta : \mathcal{X} \times \mathcal{F}_\mathcal{X} \to [0, 1]$*, a Markovian* transition kernel *depending on parameter* $\beta \in \mathcal{B}$*; (iii)* $\mathcal{B} \subseteq \mathbb{R}^m$*, a set of parameters; and (iv)* $\Theta \subseteq \mathcal{X}$*, the set of possible initial states.*

In the examples in Section I-C, the state-dependent probabilistic choices are modeled by the Markov transition kernel $\mathbb{P}_\beta$. We reiterate that our analysis 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 $x_0 \in \Theta$, for all i, $x_i \in \mathcal{X}$ and any $\beta \in \mathcal{B}$. 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 [35][36]. We address two classes of problems:

Verification: Given an NMC $\mathcal{M}$ 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 Θ. 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 examples in Section I-C, x_0 would correspond to an initial configuration of the cars, and Θ would be the set of all possible initial configurations. We say that an execution α of length k hits the unsafe set $\mathcal{U}$ if there exists $i \in \{0, \ldots, k\}$, 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:

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

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 of (2) and (3) by developing the *Hierarchical Optimistic Optimization algorithm with Mini-batches (HOO-MB)*. This can be regarded as a variant of the *Hierarchical Optimistic Optimization (HOO)* algorithm [1] from the *multi-armed bandits* literature [26], [37], [30]. The setup is as follows: suppose we have a sampling budget of N and want to maximize the function $f : \mathcal{X} \to \mathbb{R}$, which is assumed to have a unique global maximum that achieves the value $f^* = \sup_{x \in \mathcal{X}} f(x)$. The algorithm 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:

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

We are interested in algorithms that have two key properties: (I) The algorithm should be *adaptive* in the sense that the $(j+1)^{\text{st}}$ sample x_{j+1} should depend on the previous samples and outputs; and (II) The algorithm should not rely on detailed knowledge of f but *only* on the sampled noisy outputs. These algorithms 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 observation* y_j *is distributed according to a* σ^2*-sub-Gaussian distribution* M_{x_j} *satisfying* $\int u dM_{x_j}(u) = f(x_j)$*.*

Next, we present the HOO-MB algorithm. In Section III-B we present its analysis leading to the regret bound and discuss smoothness parameters. In Section III-C we discuss how HOO-MB can be used for solving the optimization problems of (2) and (3). In Section III-D, we discuss the choice of the *batch size* parameter b as well as its implications.

A. Hierarchical tree optimization with mini-batches

HOO-MB (Algorithm 1) is a batched-sampling variant of HOO [1]. HOO-MB selects the next sample x_{j+1} by building a binary[3] tree in which each height (or level)

[3]As we go down the tree the partition is refined via bisection along the dimension of the coarsest subdivision (ties are broken arbitrarily).

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. HOO-MB 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-D, we discuss the implications of the choice of batch size parameter b.

First, we discuss the tree data-structure. This construction is the same as that in noisy tree-search algorithms (HOO and variants) [1], and is discussed here for clarity of exposition. Each node in the tree is labeled by a pair of integers (h, i), where $h \geq 0$ is the height, and $i \in \{1, \ldots, 2^h\}$, is its position within level h. The root is labeled $(0, 1)$. Each node (h, i) can have two children $(h + 1, 2i - 1)$ and $(h + 1, 2i)$. Node (h, i) is associated with the region $\mathcal{P}_{h,i} \subseteq \mathcal{X}$, where $\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$, and for each h these disjoint regions satisfy $\cup_{i=1}^{2^h} \mathcal{P}_{h,i} = \mathcal{X}$. Thus, larger values of h represent finer partitions of $\mathcal{X}$. For each node (h, i) in the tree, HOO-MB computes the following quantities: (i) $t_{h,i}$ is the number of times the node is chosen or considered for re-partitioning. (ii) $\hat{f}_{h,i}$ is the empirical mean of observations over points sampled in $\mathcal{P}_{h,i}$. (iii) $U_{h,i}$ is an initial estimate of the upper-bound of f over $\mathcal{P}_{h,i}$ based on the smoothness parameters. (iv) $B_{h,i}$ is a tighter and optimistic upper bound for the same. The *tree* starts with a single root $(0, 1)$, with B-values of its two children $B_{1,1}$ and $B_{1,2}$ 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 $(hnew, inew)$ 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 $x_1, x_2, ..., x_N$ chosen uniformly at random. The details are provided in Algorithm 1.

Algorithm 1 HOO-MB with parameters: sampling budget N, noise parameter σ, smoothness parameters $\nu > 0$, $\rho \in (0, 1)$, batch size b.

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

B. Analysis of Regret Bound

The notation and analysis of the regret bounds for HOO-MB closely follows that in [1] (and followups in [31], [30]).

Let $\Delta_{h,i}$ denote the *sub-optimality gap* of node (h, i), that is, $\Delta_{h,i} = f^* - \sup_{x \in \mathcal{P}_{h,i}} f(x)$. A node (h, i) is optimal if $\Delta_{h,i} = 0$ and it is sub-optimal if $\Delta_{h,i} > 0$. We say that a node (h, i) is ϵ-optimal if $\Delta_{h,i} \leq \epsilon$. We will use two parameters ν and $\rho \in (0, 1)$ to characterize the *smoothness* of f relative to the partitions (see Assumption 2). Roughly, these parameters restrict how quickly f can drop-off near the optimal x^* within a $\mathcal{P}_{h,i}$ at each node (h, i). We define $\mathcal{N}_h(\epsilon)$ as in [31] as the number of *ϵ-optimal cells at depth* h, that is, the number of nodes with $\Delta_{h,i} \leq \epsilon$.

From the sampled estimate of $f(x_i)$ at a single point, HOO-MB 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. This Assumption is adopted from [30] and basically ensures that f does not change arbitrarily in a region near the true maximum.

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

Here c is a parameter that relates the variation of f over all $c\nu\rho^h$-optimal cells 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 cells at all $h \geq 0$ is bound by $\nu\rho^h$. 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 [1], [31]. This is a measure of closeness with respect to the number of cells 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 HOO-MB.

Theorem 1. *With the input parameters satisfying Assump-*

tions 1 and 2, a batch size parameter b, and a sampling budget of N, HOO-MB achieves a regret bound of

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

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

Proof: The proof closely follows the approach in [1] (see also [31], [30]), 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 cumulative regret at round N, where $x_i \in \mathcal{X}$ is the point returned by HOO-MB at iteration i. Let $\mathcal{T}$ be the *tree* constructed by HOO-MB at the end of N iterations. Let H be a positive integer, that can be optimized for the best bound. As in [1], we divide $\mathcal{T}$ into three groups: $\mathcal{T}_1$ that includes all $2\nu\rho^h$-optimal arms at $h \geq H$, $\mathcal{T}_2$ that includes all $2\nu\rho^h$-optimal arms at $h \in [0, H-1]$, and $\mathcal{T}_3$ that includes sub-optimal arms 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 $\mathbb{E}[R_N]$

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

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, and we recover the bound in [1]. However, if b is large, we can optimize H to get:

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

This can be easily converted to the required regret bound by randomly returning one of the sampled points. □

We note that the exact knowledge of smoothness parameters in Assumption 2 are not required for implementing the Algorithm 1; these parameters are required for deriving the tightest regret guarantees. As we discuss below, if only bounds are available, these can be used in a parallel search algorithm. We believe that inferring the best smoothness parameters for a given verification problem is challenging and requires further investigations. In general, if only coarse bounds on these smoothness parameters are known—which is often the case for physical processes—then the search for the optimal parameters can be easily parallelized by the Parallel Optimistic Optimization algorithm developed in [31] (see Algorithm 2). Their algorithm adaptively searches for the optimal smoothness parameters by spawning several parallel HOO-MB instances with various (ν,ρ) values. In Section IV-A, we discuss how the choice on upper bound of (ν, ρ) affects the performance of HOO-MB.

Algorithm 2 Parallel search with parameters: sampling budget N, number of instances K, and maximum smoothness parameters ν_{max} and ρ_{max}

1: **for** $i = 1 : K$ **do**
2: Spawn HOO-MB 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 i^{th} HOO-MB instance for $i \in \{1, .., K\}$
4: **return** $\{\bar{x}_i |\ i = 1, .., K\}$

C. Verification and Parameter Tuning with HOO-MB

In order to use HOO-MB for verification, a natural choice for the objective function would be $f(x) := p_{k,\mathcal{U},\beta}(x)$ for any initial state $x \in \Theta$ and a given $\beta \in \mathcal{B}$. To use HOO-MB for parameter tuning a natural choice for the objective function would be $f(\beta) := \mathbb{E}[r(\alpha, \beta)|\beta]$ for any parameter state $\beta \in \mathcal{B}$. Evaluating these functions exactly are infeasible when the transition kernel $\mathbb{P}_\beta$ is unknown. Even if $\mathbb{P}_\beta$ is known, calculating these functions involves integral over the state space (as in (1) and (3)). Instead, we take advantage of the fact that HOO-MB can work with noisy observations.

a) Verification: For any initial state $x \in \Theta$, and an execution α starting from x we define the observation:

$$y = 1 \text{ if } \alpha \text{ hits } \mathcal{U} \text{ within } k \text{ steps, and 0 otherwise.} \quad (5)$$

Thus, given an initial state x, $y = 1$ with probability $p_{k,\mathcal{U},\beta}(x)$, and $y = 0$ with probability $1 - p_{k,\mathcal{U},\beta}(x)$. That is, y is a Bernoulli random variable with mean $p_{k,\mathcal{U},\beta}(x)$. A Bernoulli random variable satisfies the conditions of Assumption 1. Then we have the following proposition.

Proposition 1. *Given smoothness parameters (ν, ρ) satisfying Assumption 2 for the function $f(x) = p_{k,\mathcal{U},\beta}(x)$, if Algorithm 1 returns $\bar{x}_N \in \Theta$, then $p_{k,\mathcal{U},\beta}(x^*) - p_{k,\mathcal{U},\beta}(\bar{x}_N)$ is upper bounded by Theorem 1.*

b) Parameter Synthesis: For a parameter $\beta \in \mathcal{B}$, and an execution α starting from x_0 sampled from a given distribution we receive an observation y. The observation corresponding to this simulation is $y = r(\alpha, \beta)$, with mean $\mathbb{E}[r(\alpha, \beta)|\beta]$. Assume each observation y satisfies the conditions of Assumption 1. Then we have the following proposition.

Proposition 2. *Given smoothness parameters (ν, ρ) satisfying Assumption 2 for the function $f(\beta) = \mathbb{E}[r(\alpha, \beta)|\beta]$, if Algorithm 1 returns $\bar{x}_N \in \mathcal{B}$, then $\mathbb{E}[r(\alpha, \beta)|\beta^*] - \mathbb{E}[r(\alpha, \beta)|\bar{x}_N]$, is upper bounded by Theorem 1.*

Remark 1. *Qualitatively, Assumption 2 requires that in a safety set verification problem, choices of input that are close to x^* (the input parameter that makes the system most unsafe) also lead to unsafeness. This requires a local Lipschitz property of the probability transition kernel at $x^* \in \Theta$. For autonomous driving safety examples that we consider, this assumption implies that the initial car locations (at the beginning of the simulation) that are close to the most unsafe configuration are also unsafe, which matches our intuition of the physical dynamics.*

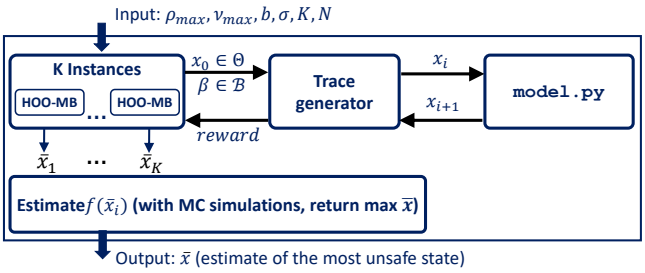

Fig. 2. HooVer. Parameters: ρ_{max}, ν_{max} are used to calculate smoothness parameters. We fix $\nu_{max} = 1.0$ and results are not sensitive to ρ_{max} (Sec IV-B). Impact of batch size b is discussed in Sec IV-B. The number of HOO-MB instances K is fixed to 4.

D. Discussions on choice of Batch size b

The difference between HOO-MB and the original HOO [1] is that each node (h, i) in the HOO-MB is sampled b times. In other words once a node (arm) is chosen, instead of a single observation, $b \geq 1$ observations are received (thus modifying the update rules for $U_{h,i}$ and $B_{h,i}$). Indeed, by setting $b = 1$ in HOO-MB we recover the original HOO algorithm and the corresponding simple regret bound $\mathbb{E}[S_N] = O\left((\frac{B\sigma^2 \log N}{N})^{\frac{1}{d_m+2}}\right)$. Comparing the regret bound in HOO-MB and HOO we observe that HOO-MB gets worse in terms of regret bound, however, it has the advantage that the number of nodes in the tree is reduced by a factor of b. This reduces the running time and makes memory usage more efficient in HOO-MB compared to HOO. Finally, as is common in literature, Algorithm 1 can be modified to return the best-scoring input, instead of a randomly chosen one, to improve empirical performance.

IV. HooVer TOOL, EVALUATION, AND DISCUSSIONS

The components of HooVer are shown in Figure 2. Given the initial state and/or the parameter, HooVer generates random trajectories using the transition kernel simulator and gets rewards. It runs K instances of HOO-MB with automatically calculated smoothness parameters (see Algorithm 2). Each instance returns an estimate $\bar{x}_i$ of the optimum, then HooVer computes the mean of reward for each $\bar{x}_i$ using Monte-Carlo simulations[4], and outputs the $\bar{x}$ that gives the highest mean reward. Source files and instructions for reproducing the results are available from the HooVer web page[5].

Benchmarks: We use instances of examples in Figure 1 to evaluate HooVer performance in verification. Moreover, we evaluate the performance of HooVer on parameter synthesis tasks using a Linear–quadratic regulator (LQR) benchmark. Specifically, we consider the system as $x_{t+1} = Ax_t + Bu_t + w_t$, where $w_t \in \mathcal{N}(0, \epsilon^2 I)$ are independent and identically distributed (i.i.d.) Gaussian noises. We want to search for a state-feedback gain matrix L such that $u_t = Lx_t$ minimizes the cost

[4]Number of simulations used here is included in "#queries" in Fig.3.

[5]https://www.daweisun.me/hoover. The source code is available at https://github.com/sundw2014/HooVer

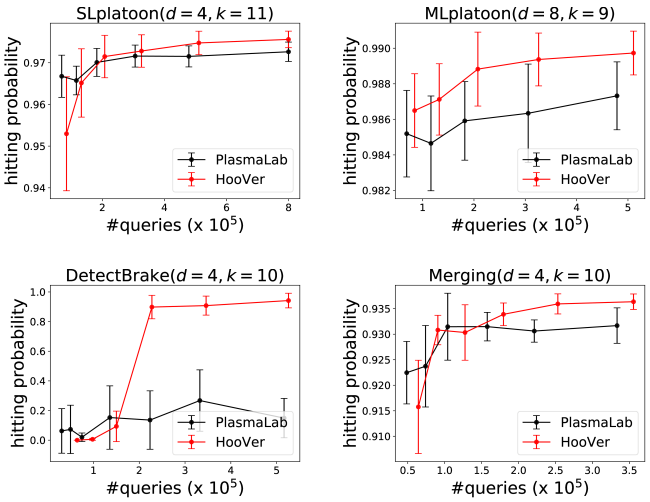

Fig. 3. Results on the verification benchmarks, d:dimensions of Θ, k: time bound in $p_{k,\mathcal{U},\beta}(\bar{x}_N)$. Mean and standard deviations are averaged over 10 runs. Hitting probability $p_{k,\mathcal{U},\beta}(\bar{x}_N)$ is estimated using Monte Carlo method once $\bar{x}_N$ is returned from the tool.

$J(L) = \mathbb{E}_w\left[\sum_{t=0}^{T-1}(x_t^\intercal Q x_t + u_t^\intercal R u_t) + x_T^\intercal Q x\right]$. In the experiments, we consider the case where $x_t \in \mathbb{R}^2$ and $u_t \in \mathbb{R}^2$, and ϵ is set to 0.01. The distribution for the initial state is a Dirac delta function, i.e. the initial state is fixed.

A. HooVer *performance on benchmarks and LQR example*

Our experiments were conducted on a Linux workstation with two Xeon Silver 4110 CPUs and 32 GB RAM. Figure 3 shows the performance of HooVer on the benchmarks. For each benchmark, the plots show the estimated worst safety violation probability $p_{k,\mathcal{U},\beta}(\bar{x}_N)$ against sampling (query) budget (N) for a fixed time horizon k. Since the ground truth for $\sup_{x\in\Theta} p_{k,\mathcal{U},\beta}(x)$ is not known, we cannot evaluate the simple regret $\sup_{x\in\Theta} p_{k,\mathcal{U},\beta}(x) - p_{k,\mathcal{U},\beta}(\bar{x}_N)$. Instead we compare the estimated $p_{k,\mathcal{U},\beta}(\bar{x}_N)$ by HooVer with PlasmaLab's estimation. We make two observations: As expected, the output $p_{k,\mathcal{U},\beta}(\bar{x}_N)$ improves with the budget. Second, though our implementation is not optimized, the running times are reasonable. SLplatoon ran at less than 1 millisecond per simulation, i.e., about 10 minutes for 800K queries.

To evaluate the the performance of HooVer on the parameter synthesis task in the LQR example, we compare the estimated optimal parameter $\hat{L}$ provided by HooVer with different sampling budgets with the ground truth of optimal parameter L^*. It is noted that L^* is obtained by solving the LQR problem with A, B, Q, and R given explicitly. As shown in Table I, the accuracy of estimation by HooVer improves with increasing the sampling budget.

TABLE I
PERFORMANCE OF HooVer ON PARAMETER SYNTHESIS.

#queries	1K	2K	4K	8K	16K	32K
$\|\hat{L} - L^*\|_F$	0.594	0.581	0.222	0.161	0.052	0.022

B. Impact of batch size and smoothness parameter

As discussed in Section III-D, the batch size parameter b of HOO-MB can improve the running time and memory usage without significantly sacrificing the quality of the final answer. Table II shows this for SLplatoon with sampling budget $N = 800$K. The #Nodes refers to the number of the nodes in the final tree generated by HooVer. Notice that the final answer $p_{k,\mathcal{U},\beta}()$ does not change much when using reasonable batch sizes ($b \leq 400$), while the number of nodes in the tree is reduced, which in turn can reduce the running time and the memory usage by orders of magnitude. A very large batch size (e.g. $b \geq 1600$) affect the quality of the results, which is consistent with Theorem 1.

TABLE II
IMPACT OF BATCH SIZE b ON TREE SIZE, FINAL RESULT, RUN TIME, MEMORY USAGE, FOR SAMPLING BUDGET OF 800K ON SLplatoon MODEL. RESULTS ARE AVERAGED OVER 10 RUNS.

b	10	100	400	1600	6400
#Nodes	75996	7596	1900	468	116
Running Time (s)	2568	506	512	573	678
Memory (Mb)	67.16	6.62	1.64	0.38	0.09
Result $p_{k,\mathcal{U},\beta}()$	0.9745	0.9756	0.9741	0.9667	0.8942

Smoothness parameter: Table III shows how smoothness parameter ρ_{max} impacts the performance of HooVer. For large values of ρ_{max} (0.95 and 0.9), the upper confidence bounds (UCB) computed in HOO-MB force the state space exploration to be more aggressive, and the algorithm explores shallower levels of the tree more extensively. As ρ_{max} decreases, HooVer proceeds to deeper levels of the tree. Below a threshold (0.8 in this case), the algorithm becomes insensitive to variation of ρ_{max}. Thus, if the smoothness of the model is unknown, one can select a small ρ_{max}, and obtain a reasonably good estimate for result.

TABLE III
IMPACT OF SMOOTHNESS PARAMETER ρ_{max} ON TREE AND FINAL RESULT. RESULTS ARE AVERAGED OVER 10 RUNS.

$\rho_{\max}$	0.95	0.90	0.80	0.60	0.40	0.16	0.01
Tree depth	11.6	14.3	25.3	25.4	25.6	24.2	24.3
Result $p_{k,\mathcal{U},\beta}()$	0.9644	0.9647	0.9740	0.9756	0.9754	0.9728	0.9722

C. Comparison with PlasmaLab

Among the tools that are currently available we found PlasmaLab [38] to be closest to HooVer in terms of being able to model check on a continuous state-space; therefore, we decided to perform a deeper comparison with it. Model checking tools such as Storm [39] and PRISM [16] do not support continuous state-space models. It is possible to compare HooVer with these tools on discrete versions of these examples, but the comparison would not be on par as the guarantees given these tools are different. The SMC approach for stochastic hybrid systems presented in [20] is related, but we could not find an implementation to compare against.

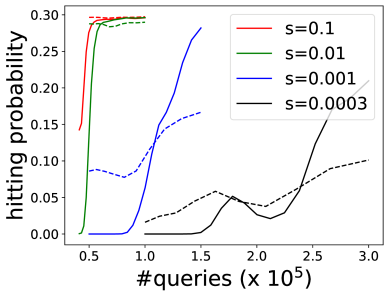

Fig. 4. HooVer (solid lines) and PlasmaLab (dashed) output in optimizing sharp functions.

PlasmaLab uses a smart sampling algorithm [40] to assign the simulation budget efficiently to each scheduler of an MDP. In order to use this algorithm, one has to set parameters ϵ and δ in the Chernoff bound, satisfying $N_{max} > \ln(2/\delta)/(2\epsilon^2)$, where N_{max} is per-iteration simulation budget. We set the confidence parameter δ to 0.01, and given an N_{max}, the precision parameter ϵ is then obtained by $\epsilon = \sqrt{\ln(2/\delta)/(2 \times 0.8 \times N_{max})}$. In order to make a fair comparison, we developed a PlasmaLab plugin which enables PlasmaLab to use exactly the same external Python simulator as HooVer.

HooVer gets better than PlasmaLab as the sampling budget increases (see Figure 3). It is not surprising that for small budgets, before a threshold depth in the tree is reached, HooVer cannot give an accurate answer. Once the threshold is exceeded, the tree-based sampling of HooVer works more efficiently than PlasmaLab for the given examples.

A conceptual example.: To illustrate the above behavior, we consider a conceptual example with hitting probability given by $p_{k,\mathcal{U},\beta}(x)$ directly without specifying $\mathbb{P}$, k and $\mathcal{U}$. Given an initial state $x = (x_1, x_2)$, $p_{k,\mathcal{U},\beta}(x_1, x_2) = p_{max} \cdot \exp\left(-\frac{(x_1-0.5)^2+(x_2-0.5)^2}{s}\right)$, where the parameter s controls the slope of $p_{k,\mathcal{U},\beta}(\cdot)$ around the maximum $p_{k,\mathcal{U},\beta}(\frac{1}{2}, \frac{1}{2}) = p_{max}$. The smaller the value of s, the sharper the slope.

In the experiments, we set $\Theta = \{(x_1, x_2) | 0 < x_1 < 1,\ 0 < x_2 < 1\}$ and $p_{max} = 0.3$. Results are shown in Fig. 4. With small query budget, PlasmaLab beats HooVer, however, as the budget increases HooVer improves swiftly and becomes better (see Figure 4). For "easy" objective functions (where f_s is smooth, $s = 0.1$), both tools perform well. As s decreases, f_s becomes sharper and the most unsafe state become harder to find, the point where HooVer beats PlasmaLab moves to the right. For sharp objective functions (e.g., $s = 0.0003$), HooVer is more sample efficient, suggesting that HooVer might perform better than PlasmaLab in SMC problems with hard-to-find bugs or unsafe conditions.

V. CONCLUSIONS

We presented HOO-MB, an optimistic mini-batched tree search algorithm, for verification and parameter synthesis in class of discrete-time nondeterministic, continuous state,

Markov chains (MC). In this class of problems the uncertainty is in the initial states or parameters. HOO-MB sequentially samples executions of the MC in batches and relies on a mild assumption about the smoothness of the objective function, to find near-optimal solutions violating the given safety requirement. We provided theoretical regret bounds in terms of the sampling budget, smoothness, near-optimality dimension, and sampling batch size; importantly, HOO-MB does not require exact parameters or quantities to run effectively. We created several benchmarks models, implemented a tool (HooVer), and the experiments show that our approach is competitive compared to PlasmaLab in terms of sample efficiency. Detailed comparison with other verification and synthesis tools and exploration of general Markov decision processes with this approach would be directions for further investigation.

Acknowledgements

This work was supported by an ONR Science of Security Grant H98230-18-D-0007.

References

[1] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *Journal of Machine Learning Research*, vol. 12, no. May, pp. 1655–1695, 2011.

[2] B. Boyer, K. Corre, A. Legay, and S. Sedwards, "PLASMA-lab: A flexible, distributable statistical model checking library," in *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems*. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 160–164.

[3] ISO, "ISO 26262:road vehicles–functional safety," Organizacion Internacional de Normalizacion (ISO), Norm ISO 26262, 2011.

[4] 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.

[5] I. Tkachev and A. Abate, "On infinite-horizon probabilistic properties and stochastic bisimulation functions," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 526–531.

[6] S. Esmaeil Zadeh Soudjani and A. Abate, "Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes," *SIAM Journal on Applied Dynamical Systems*, vol. 12, no. 2, pp. 921–956, 2013.

[7] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates." in *HSCC*, vol. 2993, 2004.

[8] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Trans. Auto. Ctrl.*, 2020.

[9] A. Legay and S. Sedwards, "On statistical model checking with PLASMA," in *The 8th Intl. Symposium on Theoretical Aspects of Software Engineering*, September 2014.

[10] H. L. S. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *International Conference on Computer Aided Verification (CAV2002)*. Springer, 2002, pp. 223–235.

[11] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Proceedings of the 17th International Conference on Computer Aided Verification*, ser. CAV'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 266–280.

[12] H. L. S. Younes, "Probabilistic verification for "black-box" systems," in *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, 2005, pp. 253–265.

[13] A. Hartmanns and H. Hermanns, "A modest approach to checking probabilistic timed automata," in *2009 Sixth Intl. Conference on the Quantitative Evaluation of Systems*. IEEE, 2009, pp. 187–196.

[14] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke, "Statistical model checking for markov decision processes," in *2012 Ninth Intl. Conference on Quantitative Evaluation of Systems*. IEEE, 2012, pp. 84–93.

[15] A. David, P. G. Jensen, K. G. Larsen, A. Legay, D. Lime, M. G. Sørensen, and J. H. Taankvist, "On time with minimal expected cost!" in *Intl. Symposium on Automated Technology for Verification and Analysis*. Springer, 2014, pp. 129–145.

[16] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, ser. LNCS, H. Hermanns and J. Palsberg, Eds., vol. 3920. Springer, 2006, pp. 441–444.

[17] R. Lassaigne and S. Peyronnet, "Approximate planning and verification for large markov decision processes," *Intl. Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 457–467, 2015.

[18] A. Hartmanns and H. Hermanns, "The modest toolset: An integrated environment for quantitative modelling and verification," in *Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014, pp. 593–598.

[19] C. E. Budde, P. R. D'Argenio, A. Hartmanns, and S. Sedwards, "A statistical model checker for nondeterminism and rare events," in *Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2018, pp. 340–358.

[20] C. Ellen, S. Gerwinn, and M. Fränzle, "Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains," *Intl. Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 485–504, 2015.

[21] N. Musavi, D. Sun, S. Mitra, G. Dullerud, and S. Shakkottai, "Optimistic optimization for statistical model checking with regret bounds," *Presented at The 6th Intl. Workshop on Symbolic-Numeric Methods for Reasoning about CPS and IoT (SNR)*, August 2020, Vienna, Austria.

[22] W. R. Thompson, "On the theory of apportionment," *American Journal of Mathematics*, vol. 57, no. 2, pp. 450–456, 1935.

[23] ——, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[24] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Math. Soc.*, vol. 58, no. 5, pp. 527–535, 1952.

[25] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[26] R. Munos, "From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning," 2014.

[27] P. Auer, C.-B. N., and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," vol. 47, 2002.

[28] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," *arXiv preprint cs/0703062*, 2007.

[29] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conf. on machine learning*. Springer, 2006, pp. 282–293.

[30] R. Sen, K. Kandasamy, and S. Shakkottai, "Noisy blackbox optimization using multi-fidelity queries: A tree search approach," in *The 22nd Intl. Conf. on Artificial Intelligence and Statistics*, 2019.

[31] J.-B. Grill, M. Valko, and R. Munos, "Black-box optimization of noisy functions with unknown smoothness," in *Advances in Neural Information Processing Systems*, 2015, pp. 667–675.

[32] K. Kodaka, M. Otabe, Y. Urai, and H. Koike, "Rear-end collision velocity reduction system," SAE Intl., Tech. Rep. 2003-01-0503, 2003.

[33] S. Fabris, "Method for hazard severity assessment for the case of undemanded deceleration," *TRW Automotive, Berlin*, 2012.

[34] C. Fan, B. Qi, and S. Mitra, "Data-driven formal reasoning and their applications in safety analysis of vehicle autonomy features," *IEEE Design and Test*, vol. 35, no. 3, pp. 31–38, 2018.

[35] C. Ionescu Tulcea, "Mesures dans les espaces produits," *Atti Accad. Naz. Lincei Rend*, vol. 7, pp. 208–211, 1949.

[36] D. Petritis, "Markov chains on measurable spaces," April 2012, https://perso.univ-rennes1.fr/dimitri.petritis/ps/markov.pdf.

[37] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[38] A. Legay, S. Sedwards, and L.-M. Traonouez, "Plasma lab: a modular statistical model checking platform," in *Tntl. Symposium on Leveraging Applications of Formal Methods*. Springer, 2016, pp. 77–93.

[39] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Intl. Conference on Computer Aided Verification*. Springer, 2017, pp. 592–600.

[40] P. D'Argenio, A. Legay, S. Sedwards, and L.-M. Traonouez, "Smart sampling for lightweight verification of markov decision processes," *Intl. Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 469–484, 2015.