Optimistic Optimization for Statistical Model Checking with Regret Bounds

Negin Musavi
University of Illinois at Urbana Champaign
nmusavi2@illinois.edu

Dawei Sun
University of Illinois at Urbana Champaign
daweis2@illinois.edu

Sayan Mitra
University of Illinois at Urbana Champaign
mitras@illinois.edu

Geir Dullerud
University of Illinois at Urbana Champaign
dullerud@illinois.edu

Sanjay Shakkottai
University of Texas at Austin
sanjay.shakkottai@utexas.edu

We present an algorithm and a tool for statistical model checking (SMC) of continuous state space Markov chains initialized to a set of states. This model checking problem requires maximization of probabilities of sets of executions over all choices of initial states. We observe that it can be formulated as a multi-armed bandit problem, and therefore, can be solved using hierarchical optimistic optimization. We propose a new algorithm (HOO-MB) and provide a regret bound on its sample efficiency which relies on the smoothness and the near-optimality dimension of the objective function as well as the sampling batch size. The batch size parameter enables us to strike a balance between the sample efficiency and the memory usage of the algorithm. Our experiments, using the tool HooVer, suggest that the approach scales to realistic-sized problems and is often more sample-efficient compared to PlasmaLab. Specifically, it has distinct advantages in analyzing models in which the function that indicates the probability of being unsafe has sharp slope around its worst initial conditions.

1 Introduction

The multi-armed bandit problem is an idealized model for sequential decision making in unknown random environments. It is frequently used to study exploration-exploitation trade-offs [47, 46, 40].

Significant advances have been made in the last decade, and connections have been drawn with optimization, auctions, and online learning [36, 6, 8]. Recently developed bandit algorithms for *black-box optimization* [36] can strike a balance between exploiting the most promising parts of the function's domain, and exploring the uncertain parts, and find nearest to optimal solutions for a given sampling budget. These are promising developments from the perspective of quantitative evaluation of systems. Quantitative verification can be viewed as optimization of a function that maps system's executions to values. Black-box optimization makes the algorithms applicable to black-box systems, for which executions can be sampled, but detailed internal workings are unavailable. Further, minimizing execution sampling budget is attractive where the set of behaviors is large and individual tests are expensive.

In this paper, we present a new statistical model checking algorithm and a supporting tool for discrete time continuous state-space Markov chains. Model checking for stochastic system can be solved by calculating the exact measure of satisfying executions [10, 21, 25, 41]. *Statistical model checking (SMC)* is an alternative that uses hypothesis testing with sampled executions to infer whether the samples provide statistical evidence for the satisfaction (or violation) of the requirement [51, 43, 50]. A

Submitted to:
SOS 2007

number of SMC tools have been developed for different types of requirements (for example, Ymer [52], VESTA [44], MultiVesta [42], PlasmaLab [4], Modes [19], UPPAAL [33], and MRMC [1]), and they have been used to verify many systems [12, 3, 26, 28, 38, 53, 35, 34]. We refer to [2] and [32] for recent surveys of the state of the art. In this paper, we focus on discrete time Markov chains over continuous state spaces and uncountable sets of initial states. In other words, these are continuous state Markov Decision Processes (MDPs) where the adversary gets to only initialize[1]. This class of models is standard for bounded time horizon verification and monitoring of systems such as, autonomous vehicles, driver assist control systems, and manufacturing plants. Stochastic transitions capture the dynamic evolution of the system, often inferred from system identification and past data, while the the non-deterministic initialization captures sensing and estimation errors. The notable SMC tools related to this class include MODEST for networks of probabilistic automata [18], PlasmaLab [4], the reinforcement learning-based algorithm of [20, 11] implemented in PRISM [22] and UPPAAL. Approaches for verifying MDP with restricted types of schedulers are presented in [30, 19, 9]. A recent SMC approach for Markov chains using negative correlation is presented in [49].

We start with the the observation that SMC for bounded time requirements for a model $\mathscr{M}$ of this class can be seen as the multi-armed bandit problem of maximizing a function $f(x)$ over the space $\mathscr{X}_{\mathscr{M}}$ of executions of $\mathscr{M}$, by using a sequence of noisy samples of $f(x_1),\ldots,f(x_N)$. By building this connection with the Bandit literature, we not only aim to gain new algorithms for SMC, but also new types of bounds on the sample efficiency. We propose a tree-based algorithm for solving this SMC problem using a modification of *hierarchical optimistic optimization (HOO)* [7]. The basic HOO algorithm relies on the well-known *upper confidence bound (UCB)* approach for searching the domain of a black-box function [29]. Our algorithm, called HOO-MB (Algorithm 1), improves HOO in two significant ways: (1) it relaxes the requirement of an exact semi-metric that captures the smoothness of f, but instead, it works with two smoothness parameters [17, 45]. (2) It takes advantage of batched samples to reduce the impact of the variance from the noisy samples while keeping the tree size small. We obtain regret bounds (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 smoothness parameters, and the *near-optimality dimension* of f. This bound is fundamentally different from the existing performance bounds in the SMC literature. For example, the bounds relevant for tools like PlasmaLab, use Monte Carlo sampling, Chernoff bounds, or sequential hypothesis testing. We also show empirically that batched simulations can help dramatically reduce the number of nodes in the tree and therefore reduce the running time and memory usage.

We have implemented HOO-MB in a open source tool called HooVer[2]. The user has to provide a `Python` class specifying the transition kernel, the unsafe set, and the initial uncertainty. There is no requirement for learning a new language. We have created a suite of benchmarks models capturing scenarios that are used for certification of autonomous vehicles and advanced driving assist systems [24] and evaluated HooVer on these benchmarks. Our evaluations suggest that (1) as expected, the quality of the verification result (in this case, maximum probability of hitting an unsafe state) improves with the sampling budget N, (2) HooVer scales to reasonably large models. In our experiments, the tool easily handled models with 18-dimensional state spaces, and initial uncertainty spanning 8 dimensions, on a standard computer. (3) Running time and memory usage can be controlled with the sampling batch sizes, and (4) HooVer it is relatively insensitive to the smoothness parameters. A fair comparison of HooVer with other discrete state SMC approaches is complicated because the guarantees are different and

[1]A finite number non-deterministic action choices can be encoded in the choice of the initial state.

[2]The tool and all the benchmarks are available from `https://www.daweisun.me/hoover/`.

platform specific constants are difficult to factor out. We present a careful comparison with PlasmaLab in Section 4.3. 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 SMC problems around hard to find bugs. A related approach [15] uses the original HOO algorithm of [7] for model checking over continuous domains[3]. We believe that requiring upper bounds on the smoothness parameters of f, as required in HOO-MB, is a less stringent requirement than the semi-metric of f used in [15].

2 Model and problem statement

We start by introducing basic mathematical objects needed to define our model. Given a set $\mathscr{X}$, a *σ-algebra* is a collection $\mathscr{F}_{\mathscr{X}}$ of subsets of $\mathscr{X}$ which is closed under complementation and countable union. A *measurable space* is a pair $(\mathscr{X}, \mathscr{F}_{\mathscr{X}})$, where $\mathscr{F}_{\mathscr{X}}$ is a σ-algebra over $\mathscr{X}$ and the elements of $\mathscr{F}_{\mathscr{X}}$ are referred to as *measurable sets*. Given two measurable spaces $(\mathscr{X}, \mathscr{F}_{\mathscr{X}})$ and $(\mathscr{Y}, \mathscr{F}_{\mathscr{Y}})$, a function $f : \mathscr{X} \to \mathscr{Y}$ is *measurable* if the pre-image of every measurable set is measurable. A *probability measure* on $(\mathscr{X}, \mathscr{F}_{\mathscr{X}})$ is a function $\mu : \mathscr{F}_{\mathscr{X}} \to [0,1]$ such that $\mu(\mathscr{X}) = 1$ and $\mu(\cup_{i\in I} \mathscr{X}_i) = \sum_{i\in I} \mu(\mathscr{X}_i)$ for any countable pairwise-disjoint family of sets $\{\mathscr{X}_i\}_{i\in I} \subset \mathscr{F}_{\mathscr{X}}$.

A *Markovian transition kernel* on a measurable space $(\mathscr{X}, \mathscr{F}_{\mathscr{X}})$ is a mapping $\mathbb{P} : \mathscr{X} \times \mathscr{F}_{\mathscr{X}} \to [0,1]$, such that (i) for all $x \in \mathscr{X}$, $\mathbb{P}(x, \cdot)$ is a probability measure on $\mathscr{F}_{\mathscr{X}}$; and (ii) for all $\mathscr{A} \in \mathscr{F}_{\mathscr{X}}$, $\mathbb{P}(\cdot, \mathscr{A})$ is a $\mathscr{F}_{\mathscr{X}}$-measurable function. A real-valued random variable[4] X is *σ^2-sub-Gaussian*, if for all $s \in \mathbb{R}$, $\mathbb{E}[\exp(s(X - \mathbb{E}X))] \le \exp(\sigma^2 s^2/2)$ holds, where $\mathbb{E}$ denotes the expectation operation.

Definition 2.1 *A* Nondeterministically initialized Markov chain (NiMC) $\mathscr{M}$ *is defined by a triplet* $((\mathscr{X}, \mathscr{F}_{\mathscr{X}}), \mathbb{P}, \Theta)$*, where: (i)* $(\mathscr{X}, \mathscr{F}_{\mathscr{X}})$ *is a measurable space and* $\mathscr{X}$ *will serve as the state space;*
(ii) $\mathbb{P} : \mathscr{X} \times \mathscr{F}_{\mathscr{X}} \to [0,1]$ *is a Markovian* transition kernel*; and (iii)* $\Theta \subseteq \mathscr{X}$ *is the set of possible initial states.*

In other words, the uncertainty in the initial state is modeled as a nondeterministic choice over a set Θ and the uncertainty in the transitions is modeled as probabilistic choices by the transition kernel $\mathbb{P}$. From the theoretical standpoint, NiMCs are interesting as a special subclass of MDPs, and they are a natural model for adversarially initialized probabilistic systems (e.g., an autonomous vehicle model that updates periodically based on sensor measurements with worst case bounds).

Given an *NiMC* $\mathscr{M}$ and a measurable unsafe set $\mathscr{U} \in \mathscr{F}_{\mathscr{X}}$, we are interested in evaluating the *worst case* probability of $\mathscr{M}$ hitting $\mathscr{U}$ 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 defined in the standard way. The details of the construction of the measure space over executions is not relevant for our work, and therefore, we give an abridged overview below.

An *execution* of $\mathscr{M}$ of length k is a sequence of states $\alpha = x_0 x_1 \cdots x_k$, where $x_0 \in \Theta$ and for all i, $x_i \in \mathscr{X}$. The i^{th} state of an execution α is denoted by $\alpha_i = x_i$. Given x_0 and a sequence of measurable sets of states $A_1, \ldots, A_k \in \mathscr{F}_{\mathscr{X}}$, 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 :

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

[3]We were unable to get this tool and run experiments.

[4]Namely, a function $X : \mathscr{X} \to \mathbb{R}$ for which the set $\{x \in \mathscr{X} : X(x) \le r\}$ is measurable, for each $r \in \mathbb{R}$, with respect to a fixed designated measurable space $(X, \mathscr{F}_{\mathscr{X}})$ equipped with a probability measure.

which is a standard result and follows from the Ionescu Tulceă theorem [23][39].

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

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

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

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

We note here that our verification approach solves the optimization problem of Equation (2) using samples of individual states from Θ and does not rely on explicitly calculating the probability defined by Equation (1). More specifically, our approach relies on noisy observations about whether or not a sampled execution hits $\mathscr{U}$. Thus, the user has to provide a *simulator* for the *NiMC* $\mathscr{M}$ (i.e., the transition kernel $\mathbb{P}$), and the specifications for the initial set Θ, and the unsafe set $\mathscr{U}$. For theoretical bounds, we will assume that the function $p_{k,\mathscr{U}}(x_0)$ is continuous in $x_0 \in \Theta$. Next, we will give a sufficient condition for continuity. First, notice that Eq. (1) can be written in a recursive form $p_{k,\mathscr{U}}(x_0) = 1 - \int_S \left(1 - p_{k-1,\mathscr{U}}(x_1)\right)\cdot\mathbb{P}(x_0, dx_1)$. If $\mathbb{P}(x_0, dx_1)$ is continuous w.r.t. x_0, then the integral is continuous. Thus, a sufficient condition is that the kernel-based function $\mathbb{P}(\cdot, A)$ be continuous in $x_0 \in \Theta$ for each $A \in \mathscr{F}_{\mathscr{X}}$. For models where $\mathbb{P}(\cdot, A)$ is piece-wise continuous, one can easily partition Θ into subsets such that in each subset this condition is satisfied. Then, the algorithm can be applied to each subset of Θ individually.

2.1 A simple example

Figure 1 shows a NiMC input for HooVer specifying a particle moving randomly on a plane. The model `RandomMotion` is specified as a Python class which is a subclass of `NiMC`. The initial set $\Theta \subseteq \mathbb{R}^2$, specified by `set_Theta()`, defines $\Theta = \{(x_1, x_2) \mid x_1 \in [1,2], x_2 \in [2,3]\}$. The unsafe set $\mathscr{U}$, specified by the member function `is_unsafe()`, defines $\mathscr{U} = \{(x_1, x_2) \mid x_1^2 + x_2^2 > 4\}$. The `transition()` function describes the transition kernel $\mathbb{P}$. Given an input (pre)state x, `transition()` returns the post-state x' of the transition by sampling the measure $\mathbb{P}(x, \cdot)$. For this example, x' is computed by adding `inc` to x where `inc` is sampled from a 2-dimensional Gaussian distribution with mean $\mu = (0,0)$ and covariance $\Sigma = \begin{bmatrix}\sigma^2 & 0 \\ 0 & \sigma^2\end{bmatrix}$. We can write the transition kernel explicitly by a density function: $p(x') = \frac{1}{2\pi\sigma^2}\exp(-\frac{1}{2\sigma^2}(x_1' - x_1)^2 + (x_2' - x_2)^2)$. This function is continuous in x, and therefore, $p_{k,\mathscr{U}}(x_0)$ is also continuous over Θ. For more complicated models, for example, the ones studied in Section 4.1, HooVer does not rely on explicit transition kernels, but only the `transition()` function for sampling $\mathbb{P}(x, \cdot)$.

```python
class RandomMotion(NiMC):
  def __init__(self, sigma):
    self.set_Theta([[1,2],[2,3]])
    self.sigma = sigma

  def is_unsafe(self, state):
    # return True if  |state|> 4
    if np.linalg.norm(state) > 4:
      return True
    return False


  def transition(self, state):
    # inc 2-d standard normal(0,sigma^2)
    inc=self.sigma*np.random.randn(2)
    state += inc
    return state # return the new state


# end of class
```

Figure 1: HooVer model description file for a simple random motion model.

3 SMC with Hierarchical Optimistic Optimization

Overview. We will solve the optimization problem of (2) using the *Hierarchical Optimistic Optimization algorithm with Mini-batches (HOO-MB)*. This is a variant of the *Hierarchical Optimistic Optimization (HOO)* algorithm [6] from the *multi-armed bandits* literature [36, 8, 45]. The setup is as follows: suppose we have a sampling budget of N and want to maximize the function $f : \mathscr{X} \to \mathbb{R}$, which is assumed to have a unique global maximum that achieves the value $f^* = \sup_{x \in \mathscr{X}} f(x)$. The algorithm gets to choose a sequence of sample points (arms) $x_1, x_2, \ldots x_N \in \mathscr{X}$, 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 $\bar{x}_N \in \mathscr{X}$ with the aim of minimizing *regret*, which is defined as:

$$S_N = f^* - f(\bar{x}_N). \tag{3}$$

We are interested in algorithms that have two key properties: (I) The algorithm should be *adaptive* in the sense that the $(i+1)^{\text{st}}$ sample x_{i+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 light assumptions on the smoothness of f (see Assumption 2) and on the relationship between $f(x_i)$ and the corresponding observation y_i. Assumption 1 formalizes the latter by stating that y_i is distributed according to some (possibly unknown) distribution with mean $f(x_i)$ and a strong tail-decay property.

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

In the next section, we present the HOO-MB algorithm. In Section 3.2 we present its analysis leading to the regret bound. In Section 3.3 we discuss how HOO-MB can be used for solving the verification problem of Equation (2). In Section 3.4, we discuss the choice of the *batch size* parameter b and smoothness parameters as well as their implications.

3.1 Hierarchical tree optimization with mini-batches

HOO-MB (Algorithm 1) selects the next sample x_{j+1} by building a binary tree in which each height (or level) partitions the state space $\mathscr{X}$ into a number of regions. The algorithm samples states to estimate upper-bounds on $f(x)$ over a region, and based on this estimate, decides to expand certain branches (i.e., re-partition certain regions). The partitioning of nodes presents a tension between (a) sampling the same state x_i multiple times to reduce the variance in the estimate of $f(x_i)$ obtained from the noisy observations y_i, and (b) sampling different states to reduce the region sizes based on the smoothness of f. HOO-MB allows us to tune this choice using the batch size parameter b. In Section 3.4 and 4.3, we will discuss the implications of this choice.

First, we discuss the tree data-structure. 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 $\mathscr{P}_{h,i} \subseteq \mathscr{X}$, where $\mathscr{P}_{h,i} = \mathscr{P}_{h+1,2i-1} \cup \mathscr{P}_{h+1,2i}$, and for each h these disjoint regions satisfy $\cup_{i=1}^{2^h} \mathscr{P}_{h,i} = \mathscr{X}$. Thus, larger values of h represent finer partitions of $\mathscr{X}$.

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) $count_{h,i}$ is the number of times the node is sampled. For batch size b, every time the node (h,i) is chosen, it is sampled b times. (iii) $\hat{f}_{h,i}$ is the empirical mean of observations over points sampled in $\mathscr{P}_{h,i}$. (iv) $U_{h,i}$ is an initial estimate of the upper-bound of f over $\mathscr{P}_{h,i}$ based on the smoothness parameters. (v) $B_{h,i}$ 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 $+\infty$. Each iteration of the **while** loop (line 2), adds a new node $(hnew, inew)$ and updates all of the above quantities for several nodes. First, a *path* from the root to a leaf is found by traversing the child with the higher B-value (with ties broken arbitrarily). Let the child with the higher B-value of the traversed leaf be $(hnew, inew)$. An arbitrary point x in the cell of this node $\mathscr{P}_{hnew,inew}$ is chosen (line 4). Then, this point is simulated in a batch for b times and the observations $y_1, y_2, \ldots, y_b$ are received (line 5). Next, the *tree* is extended by inserting $(hnew, inew)$ in the *tree* (line 6) and for all the nodes (h,i) in *path* including $(hnew, inew)$, the $t_{h,i}$, the $count_{h,i}$, and the empirical mean $\hat{f}_{h,i}$ are updated (line 7). Finally, in line 12, for all nodes (h,i) in *tree*, $U_{h,i}$ and $B_{h,i}$ are updated using the smoothness parameters $\nu > 0$ and $\rho \in (0,1)$ which are discussed below. Once the sampling budget N is exhausted, a leaf with maximum B-value at the maximum depth h_N is returned.

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

```
1:  $tree = \{(0,1)\}, B_{1,1} = B_{1,2} = \infty$
2:  while $n <= N$ do
3:      $(path, (hnew, inew)) \leftarrow Travrse(tree)$
4:      choose $x \in \mathscr{P}_{H,I}$
5:      query $x$ and get $b$ observations $y_1, y_2, \ldots, y_b$
6:      $tree.Insert((hnew, inew))$
7:      for all $(h,i) \in path$ do
8:          $t_{h,i} \leftarrow t_{h,i} + 1$
9:          $count_{h,i} \leftarrow count_{h,i} + b$
10:         $\hat{f}_{h,i} \leftarrow (1 - \frac{b}{count_{h,i}})\hat{f}_{h,i}$
                $+ \frac{\sum_{j=1}^{b} y_j}{count_{h,i}}$
11:     $m \leftarrow m + 1$
        $n \leftarrow n + b$
        $B_{hnew+1,2inew-1} \leftarrow +\infty$,
        $B_{hnew+1,2inew} \leftarrow +\infty$
12:     for all $(h,i) \in tree$ do leaf up:
13:         $U_{h,i} \leftarrow \hat{f}_{h,i} + \sqrt{\frac{2\sigma^2 \ln m}{bt_{h,i}}} + \nu\rho^h$
14:         $B_{h,i} \leftarrow \min\{U_{h,i},$
                $\max\{B_{h+1,2i+1}, B_{h+1,2i}\}\}$
15: return $\underset{(h,i)\in tree}{\operatorname{argmax}} B_{h,i}$ at depth $h_N$
```

3.2 Analysis of Regret Bound

The analysis of the regret bounds for HOO-MB follows the pattern of analysis in [7, 45] and the details are given in the Appendix B and A. First, we define some notations: Let $\Delta_{h,i}$ denote the *optimality gap* of node (h,i), that is, $\Delta_{h,i} = f^* - \sup_{x \in \mathscr{P}_{h,i}} f(x)$. We say that a node (h,i) is ε-optimal if $\Delta_{h,i} \leq \varepsilon$. A node (h,i) is optimal if $\Delta_{h,i} = 0$ and it is sub-optimal if $\Delta_{h,i} > 0$. Let (h,i^*) be the optimal node at depth h. 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 $f(x)$ can drop-off near the optimal x^* within a $\mathscr{P}_{h,i}$.

We define I_h as the set of all nodes at depth $h \geq 0$ that are $2\nu\rho^h$-optimal, and $\mathscr{N}_h(\varepsilon)$ as the number of *ε-optimal cells at depth h*, that is, the number of cells $\mathscr{P}_{h,i}$ with $\Delta_{h,i} \leq \varepsilon$. That is, $|I_h| = \mathscr{N}_h(2\nu\rho^h)$.

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 $\mathscr{X}$. This is achieved by assuming that f is locally smooth, i.e., there is no sudden large drop in the function around the global 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 $x \in \mathscr{P}_{h,i}$ 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 $c\nu\rho^h$-optimal cells. For $c = 0$ it implies that that there exist smoothness parameters such that the gap between the $f(x^*)$ and the value of f over optimal cells is bound by $\nu\rho^h$; for $c = 2$, it implies that there exist smoothness parameters such that over all $2\nu\rho^h$-optimal cells, the gap between the $f(x^*)$ and value of f over those cells is bounded by $4\nu\rho^h$-optimal, and so on.

We now define the concept of *near-optimality dimension* which plays an important role in the analysis of black-box optimization algorithms [6, 45, 17, 48]. It measures the dimension of sets that are close to optimal.

Definition 3.1 *The near-optimality dimension of f with respect to parameters (ν,ρ) is: $d(\nu,\rho) = \inf\{d' \in \mathbb{R}_{>0} : \exists B > 0,\ s.t.\ \forall h \geq 0,\ \mathscr{N}_h(2\nu\rho^h) \leq B\rho^{-d'h}\}$.*

In other words, the near-optimality dimension is the smallest $d \geq 0$ such that the number of $2\nu\rho^h$-optimal cells at any depth $h \geq 0$ is bounded by $B\rho^{-dh}$, for a constant B. The number $\mathscr{N}_h(2\nu\rho^h)$ of near-optimal cells grow exponentially with h, and the near-optimality dimension gives the exponential rate of this growth. Thus, $|I_h| = \mathscr{N}_h(2\nu\rho^h) \leq B\rho^{-d(\nu,\rho)h}$.

The following lemma bounds the expected number of times the algorithm visits a suboptimal node (h,i) at the end of round n, in terms of the smoothness parameters (ν,ρ), batch size b, and the sub-optimality gap $\Delta_{h,i}$. This bound is used in the regret bound for HOO-MB.

Lemma 1 *For any $n > b+1$, and any sub-optimal node (h,i) with $\Delta_{h,i} > \nu\rho^h$:*

$$\mathbb{E}[t_{h,i}(n)] \leq \frac{8\sigma^2 \log\left(\lfloor \frac{n-1}{b} \rfloor + 1\right)}{b(\Delta_{h,i} - \nu\rho^h)^2} + 4.$$

A sub-optimal node (h,i) might be visited if either the $U_{h,i}$ is greater than f^* or the $U_{h,i*}$ is underestimated. This lemma combines these two cases.

We are now ready to sketch regret bound for HOO-MB.

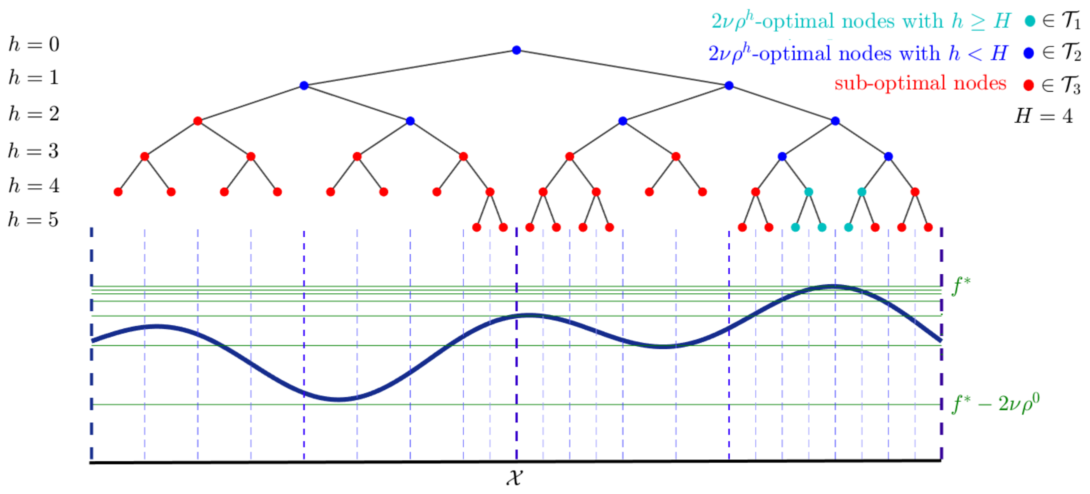
$2\nu\rho^h$-optimal nodes with $h \geq H$ ● $\in \mathcal{T}_1$
$2\nu\rho^h$-optimal nodes with $h < H$ ● $\in \mathcal{T}_2$
sub-optimal nodes ● $\in \mathcal{T}_3$
$H = 4$
$h = 0$
$h = 1$
$h = 2$
$h = 3$
$h = 4$
$h = 5$
f^*
$f^* - 2\nu\rho^0$
$\mathcal{X}$

Figure 2: The *tree* $\mathcal{T}$ constructed by HOO-MB and its decomposition into sub-trees with the parameter $H = 4$. Vertical (dashed) lines represent the $\mathcal{P}_{h,i}$ boundaries constructed by $\mathcal{T}$. Cyan ($\mathcal{T}_1$) and blue ($\mathcal{T}_2$) nodes are $2\nu\rho^h$-optimal at $0 \leq h \leq 5$ as marked by the horizontal lines. The red nodes are sub-optimal and belong to $\mathcal{T}_3$.

Theorem 1 *With the input parameters satisfying Assumptions 1 and 2 and a sampling budget of N, HOO-MB achieves a regret bound of*

$$\mathbb{E}[S_N] = O\Bigg(\Big(\frac{B\log\lfloor\frac{N-1}{b}\rfloor+1) + b}{N}\Big)^{\frac{1}{d+2}}\Bigg),$$

where $d = d(\nu,\rho)$ is the near-optimality dimension and B is the constant appearing in Definition 3.1.

Proof 1 *(sketch) Let $x_i \in \mathcal{X}$ be the point returned by HOO-MB at round i. Let $R_N = \sum_{i=1}^N (f^* - f(x_i))$ be the cumulative regret at round N. Let $\mathcal{T}$ be the tree constructed by HOO-MB at the end of N iterations. Let us fix an arbitrary height $H > 0$ and based on H we partition $\mathcal{T}$ into three sub-trees (See Figure 2): $\mathcal{T}_1$ contains the nodes that are $2\nu\rho^h$-optimal at depth $h \geq H$; $\mathcal{T}_2$ contains the nodes that are $2\nu\rho^h$-optimal at depth $h < H$; and $\mathcal{T}_3$ has all other nodes. For a node that is not $2\nu\rho^h$-optimal at depth h, one of its ancestors at some depth $h_a < h$ is $2\nu\rho^{h_a}$-optimal. The sub-tree $\mathcal{T}_3$ includes all such nodes, i.e. it includes the descendants of any node at depth h that is not $2\nu\rho^h$-optimal but its parent is $2\nu\rho^{h-1}$-optimal.*

Let $R_{N,1}$, $R_{N,2}$ and $R_{N,3}$ be the cumulative regrets for the nodes that belong to the sub-trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, respectively. The sub-tree $\mathcal{T}_1$ contains the nodes (h,i) that are at least $2\nu\rho^H$-optimal. By Assumption 2, we conclude that all the points $x \in \mathcal{P}_{h,i}$ satisfy $f^* - f(x) \leq 4\nu\rho^H$ and summing over $\mathcal{T}_1$ using $|\mathcal{T}_1| \leq N$, we get $\mathbb{E}[R_{N,1}] \leq 4\nu\rho^H N$.

The sub-tree $\mathcal{T}_2$ contains the nodes (h,i) that are $2\nu\rho^h$-optimal. Again, using Assumption 2 we can conclude that all the points $x \in \mathcal{P}_{h,i}$ satisfy $f^* - f(x) \leq 4\nu\rho^h$. In addition, since $\mathcal{T}_2$ is the union of sets I_h for $h = 0,\dots,H-1$, using the Definition 3.1, the size of sub-tree $\mathcal{T}_2$ can be bounded. Combining these we can get $\mathbb{E}[R_{N,2}] \leq b\sum_{h=0}^{H-1} 4\nu\rho^h|I_h| \leq 4b\nu B\sum_{h=0}^{H-1}\rho^{h(1-d)}$.

The sub-tree $\mathcal{T}_3$ contains sub-optimal nodes and we can upper bound the expected number of visits to sub-optimal nodes using the Lemma 1 and get $\mathbb{E}[R_{N,3}] \leq 8b\nu B\sum_{h=1}^{H}\rho^{(h-1)(1-d)}\Big(\frac{8\sigma^2\log(\lfloor\frac{N-1}{b}\rfloor+1)}{b\nu^2\rho^{2h}} +$ $4\Big)$.

Combining the three upper bounds for $R_{N,1}$, $R_{N,2}$ and $R_{N,3}$ we get an expression of an upper bound on $\mathbb{E}[R_N]$ involving H. We minimize this over H to get $\mathbb{E}[R_N]$ as $O\left(BN^{\frac{d+1}{d+2}}\left(\log(\lfloor\frac{N-1}{b}\rfloor+1)+b\right)^{\frac{1}{d+2}}\right)$. Note that if one of the points sampled by the algorithm were chosen uniformly at random as the final output, then, $\mathbb{E}(S_N) \leq \frac{\mathbb{E}[R_N]}{N}$ [7, 5]. Since, at round N, our algorithm returns the best sampled point, this relation also applies for our algorithm. ■

From Theorem 1, we observe that the regret is minimized if the near-optimality dimension $d(\nu,\rho)$ is minimized. If the smoothness parameters (ν,ρ) for the function f that minimize the near-optimlaity dimension $d(\nu,\rho)$ are known, then HooVer achieves this minimum regret. In general, we believe that inferring the minimizing smoothness parameters for a given verification problem will be challenging and requires further investigations. However, if bounds on these smoothness parameters are known—which is often the case for physical processes—then the search for the optimal parameters can be parallelized with similar regret bounds as given by Theorem 1 [17]. The pseudocode for this parallel search is given in Section 3.4.

3.3 Statistical Model Checking with HOO-MB

In order to use HOO-MB for statistical model checking (SMC), a natural choice for the objective function would be to define $f(x) := p_{k,\mathscr{U}}(x)$, for any initial state $x \in \Theta$. Evaluating this function $p_{k,\mathscr{U}}(x)$ exactly is infeasible when the transition kernel $\mathbb{P}$ is unknown. Even if $\mathbb{P}$ is known, calculating $p_{k,\mathscr{U}}(x)$ involves integral over the state space (as in (1)). Instead, we take advantage of the fact that HOO-MB can work with noisy observations. For any initial state $x \in \Theta$, and an execution α starting from x we define the observation:

$$y = 1 \text{ if } \alpha \text{ hits } \mathscr{U} \text{ within } k \text{ steps, and } 0 \text{ otherwise.} \tag{4}$$

Thus, given an initial state x, $y = 1$ with probability $p_{k,\mathscr{U}}(x)$, and $y = 0$ with probability $1 - p_{k,\mathscr{U}}(x)$. That is, y is a Bernoulli random variable with mean $p_{k,\mathscr{U}}(x)$. In HOO-MB, once an initial state $x \in \mathscr{P}_{hnew,inew}$ is chosen (line 4), we simulate $\mathscr{M}$ upto k steps b times starting from x and calculate the empirical mean of Y, which serves as the noisy observation y.

The noise parameter σ has to be specified according to Assumption 1. That is, σ should be such that each observation y is distributed as a σ^2-sub-Gaussian. As the reward y of each initial state (arm in the bandits language) is a Bernoulli random variable, we have that y is sub-Gaussian with parameter $\sigma = 0.5$. Thus, choosing $\sigma = 0.5$ is a valid parameter for any instances of this optimization problem, independent of the actual underlying model.

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

Proof 2 *For any $x \in \Theta$, $p_{k,\mathscr{U}}(x)$ is the mean of the Bernoulli distribution defined above, and the corresponding observation y satisfies Assumption 1 with $\sigma = 0.5$. Therefore, the point $\bar{x}_N \in \Theta$ returned by HOO-MB is such that expectation of the gap between its probability $p_{k,\mathscr{U}}(\bar{x}_N)$ of hitting $\mathscr{U}$ and the true maximum probability $p_{k,\mathscr{U}}(x^*)$ is bounded by Theorem 1.* ■

3.4 Discussions on choice of parameter values

Batch size b: The main difference between HOO-MB and the original HOO [7] 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. This in turn, required the update rules for $m, U_{h,i}$, and $B_{h,i}$ to be generalized. Indeed, by setting $b = 1$ in HOO-MB we recover the original HOO algorithm and the corresponding simple regret bound $\mathbb{E}[S_N] = O\left((\frac{B \log N}{N})^{\frac{1}{d+2}} \right)$. Comparing the regret bound in HOO-MB and HOO we observe that HOO-MB gets worse in terms of regret bound, however, the number of nodes in the tree is reduced by a factor of b. This reduces the running time and makes HOO-MB more efficient in terms of memory usage with respect to the HOO algorithm. Experiments in Section 4.2 show that the actual regret of the algorithm doesn't increase much when using some reasonable batch sizes (e.g. $b = 100$).[5]

Optimal smoothness parameters (ν, ρ): Recall from Section 2, $p_{k,\mathscr{U}}(x)$ is continuous in $x \in \Theta$, if the transition kernel $\mathbb{P}$ is continuous in $x \in \Theta$. This gives us a sufficient condition for guaranteeing the existence of smoothness parameters ν and ρ satisfying Assumption 2. However, as mentioned in Section 3.2, the optimal smoothness parameters (ν, ρ) minimizing the near-optimality dimension $d(\nu, \rho)$ of $p_{k,\mathscr{U}}(\cdot)$ are generally not known. Finding bounds on the optimal values of these parameters based on partial knowledge of the NiMC model would be a direction for future investigations. The following simple algorithm adaptively searches for the optimal smoothness parameters by spawning several parallel HOO-MB instances with various ν and ρ values. This is a standard approach for parameter search in the bandits literature (see for example [45, 17]).

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 $\{\bar{x}_i | \ i = 1, .., K\}$
```

4 HooVer tool and experimental evaluation

The components of HooVer are shown in Figure 3. HooVer generates random trajectories from initial states using the transition kernel simulator and runs K instances of HOO-MB with automatically calculated smoothness parameters. Each instance returns the corresponding most unsafe initial state $\bar{x}_i$, then HooVer estimates the hitting probability $p_{k,\mathscr{U}}(\bar{x}_i)$ using Monte-Carlo simulations[6], and outputs the $\bar{x}$ with the highest hitting probability. Source files and instructions for reproducing the results are available from the HooVer web page[7].

[5] In fact, we observed that the actual regret first decreases and then increases as the batch size increases starting from 1. This phenomena does not contradict the theoretical regret bound we have derived, since it is only an upper bound and may not be tight in some cases. More sophisticated theory has to be built to understand this phenomena, which we leave for future work.

[6] Number of simulations used in this step is included in "#queries" in Fig.5.

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

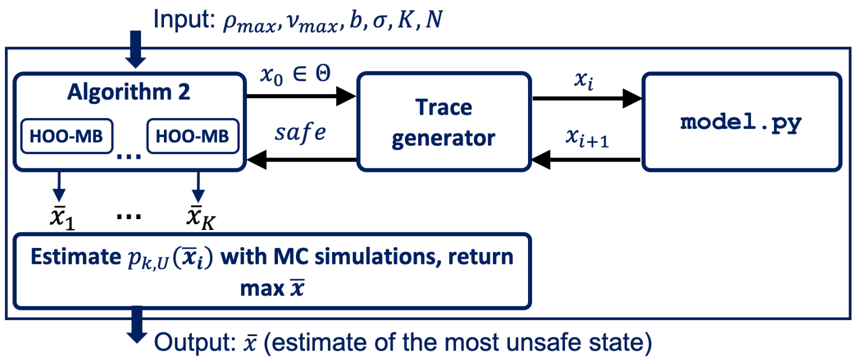
Input: $\rho_{max}, v_{max}, b, \sigma, K, N$
Algorithm 2
HOO-MB
...
HOO-MB
$x_0 \in \Theta$
$safe$
Trace generator
x_i
x_{i+1}
model.py
$\bar{x}_1$ ... $\bar{x}_K$
Estimate $p_{k,U}(\bar{x}_i)$ with MC simulations, return max $\bar{x}$
Output: $\bar{x}$ (estimate of the most unsafe state)

Figure 3: HooVer tool. The parameters: ρ_{max}, v_{max} are used to calculate smoothness parameters. We fix $v_{max} = 1.0$ and results are not sensitive to ρ_{max} (Section 4.2). The impact of batch size b is discussed in Section 4.2. The noise parameter $\sigma = 0.5$ is a valid choice for all models. The number of HOO-MB instances K is fixed to 4.

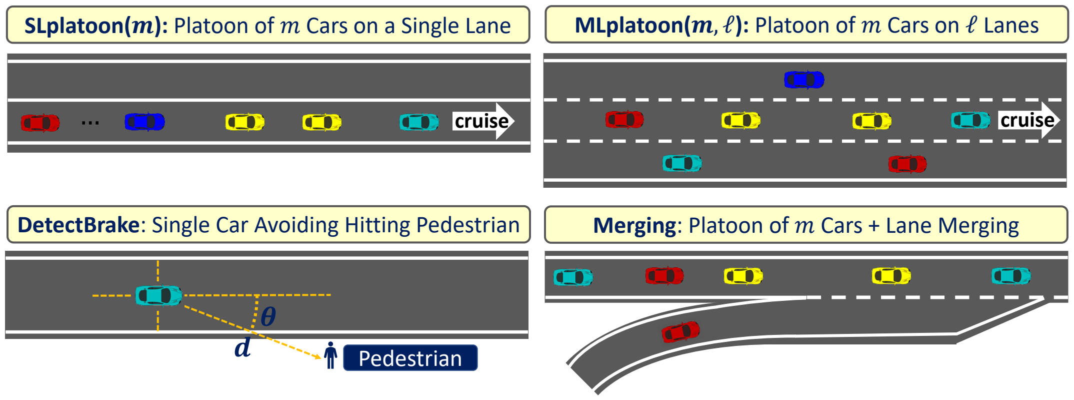
SLplatoon(m): Platoon of m Cars on a Single Lane
cruise
MLplatoon(m, ℓ): Platoon of m Cars on ℓ Lanes
cruise
DetectBrake: Single Car Avoiding Hitting Pedestrian
θ
d
Pedestrian
Merging: Platoon of m Cars + Lane Merging

Figure 4: Illustration of benchmark scenarios which can be instantiated with different number of vehicles and initial conditions.

4.1 Modelling for HooVer and benchmarks

As we saw in Section 2.1, a HooVer user can define an SMC problem by simply writing a Python class. Using this standard programming language provides the model creator with significant expressive power, and reduces the burden of learning a new language. In contrast, for example, the modeling language for the leading model checking tool PRISM [22], does not support sampling continuous probability distributions.

We have created several example models (Figure 4) that capture scenarios involving autonomous vehicles and driving assist systems. More than 25% of all highway accidents are rear-end crashes [27]. Automatic braking and collision warning systems are believed to improve safety, however, their testing and certification remain challenging [16]. Our examples capture typical scenarios used for certification of these control systems.

SLplatoon models m cars on a single lane where each car probabilistically decides to *speed up*, *cruise*, or *brake* based on its current gap with the predecessor. MLplatoon is a similar model with $m \times \ell$ cars on ℓ lanes where cars can also choose to change lanes. Merging models a car on the ramp trying to merge to m cars on the left lane. DetectBrake models a pedestrian crossing the street in front of an approaching car

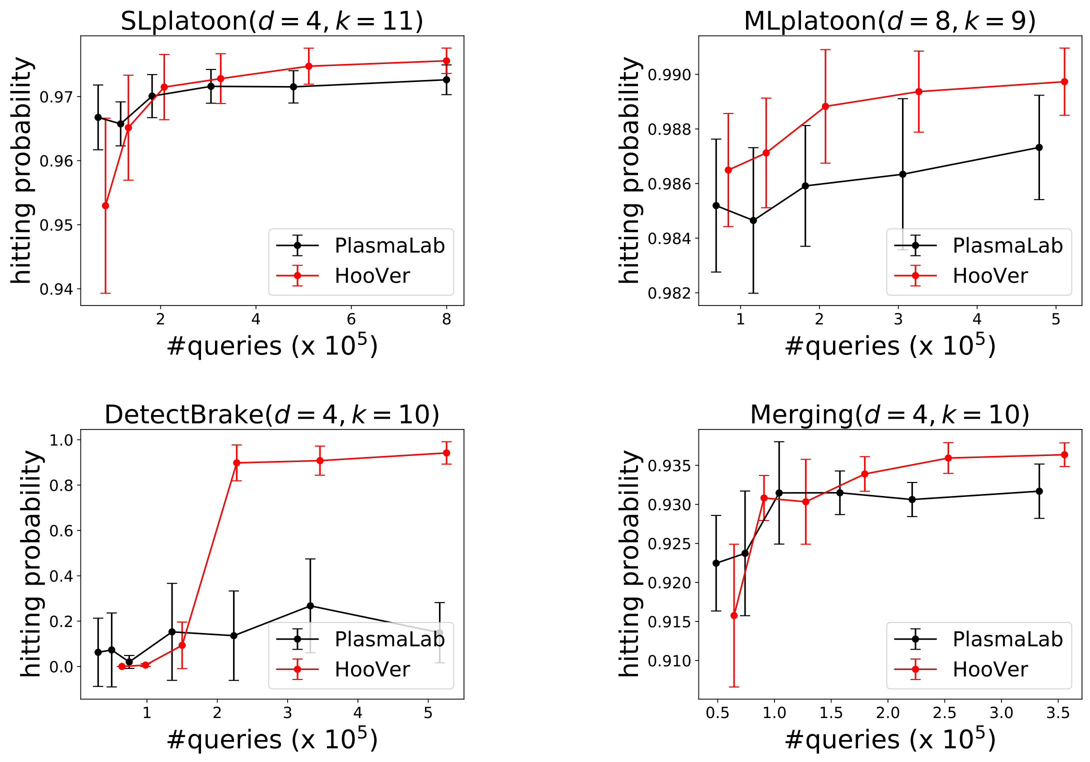

Figure 5: Results on the benchmarks, where d denotes the number of dimensions of Θ, and k denotes the time bound in $p_{k,\mathscr{U}}(\bar{x}_N)$. Mean and standard deviations are averaged over 10 runs. Hitting probability $p_{k,\mathscr{U}}(\bar{x}_N)$ is estimated using Monte Carlo method once $\bar{x}_N$ is returned from the tool.

which brakes only if its sensor detects it. In all these models, the initial state uncertainties (Θ) are defined based on the localization error (e.g., GPS error in position and velocity) and the unsafe sets ($\mathscr{U}$) capture collisions. In our experiments, we use up to 18 vehicles in 2 lanes (in MLplatoon). These give NiMC instances with up to 18-dimensional state spaces with an initial uncertainty (Θ) that spans 8 continuous dimensions. More detailed description of the scenarios are given at the HooVer web page.

4.2 HooVer **performance on benchmarks**

All our experiments were conducted on a Linux workstation with two Xeon Silver 4110 CPUs and 32 GB RAM. Figure 5 shows the performance of HooVer on above benchmarks. For each benchmark, the plots show the evolution of the estimated worst safety violation probability $p_{k,\mathscr{U}}(\bar{x}_N)$ against sampling (query) budget (N) for a fixed time horizon k. Since the actual maximum $\sup_{x\in\Theta} p_{k,\mathscr{U}}(x)$ is not known, we can not evaluate the simple regret $\sup_{x\in\Theta} p_{k,\mathscr{U}}(x) - p_{k,\mathscr{U}}(\bar{x}_N)$. However, comparing $p_{k,\mathscr{U}}(\bar{x}_N)$ directly is enough as the maximum is constant.

We make two main observations: First, as expected, the output $p_{k,\mathscr{U}}(\bar{x}_N)$ improves with the budget. Second, although our implementation is not remotely optimized for running time, the running times are reasonable. For example, for SLplatoon, the running time is less than 1 millisecond per simulation, i.e., about 10 minutes for 800K queries.

Impact of batch size. As discussed earlier in Section 3.4, the batch size parameter b of HOO-MB can help improve the running time and memory usage without significantly sacrificing the quality of the final answer. Table 1 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 $p_{k,\mathscr{U}}()$ 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. Using a very large batch sizes (e.g. $b \geq 6400$) starts to affect the quality of the result, which is consistent with Theorem 1.

Table 1: Impact of batch size b on size of tree, final result, running time, and memory usage, for sampling budget of 800K on SLplatoon model. Results are averaged over 10 runs.

b	10	20	50	100	400	1600	6400	25600
#Nodes	75996	37996	15196	7596	1900	468	116	28
Running Time (s)	2568	1006	567	506	512	573	678	614
Memory (Mb)	67.16	33.67	13.49	6.62	1.64	0.38	0.09	0.03
Result $p_{k,\mathscr{U}}()$	0.9745	0.9740	0.9739	0.9756	0.9741	0.9667	0.8942	0.8699

Impact of smoothness parameter. Table 2 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 forces 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 2: Impact of smoothness parameter ρ_{max} on tree and final result. Results are averaged over 10 runs.

$\rho_{\mathbf{max}}$	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 $p_{k,\mathscr{U}}()$	0.9644	0.9647	0.9740	0.9756	0.9754	0.9728	0.9722

4.3 Comparison with PlasmaLab

Model checking tools such as Storm [13] and PRISM [22] do not support MDPs defined on continuous state spaces. It is possible to compare HooVer with these tools on discrete versions of these examples, but the comparison would not be fair as the guarantees given these tools are different.[8] The SMC approach for stochastic hybrid systems presented in [15] is closely related, but we could not find an implementation to compare against. Furthermore, HooVer uses the HOO-MB which does not rely on a semi-metric on the state space as required by the algorithm used in [15]. Among the tools that are currently available we found PlasmaLab [31] to be closest in several ways, and therefore, we decided to perform a deeper comparison with it.

[8] We refer the curious reader to an earlier manuscript [37] with comparisons with Storm and HooVer.

PlasmaLab uses a smart sampling algorithm [14] 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 $N_{max} > \ln(2/\delta)/(2\varepsilon^2)$, 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 $\varepsilon = \sqrt{\ln(2/\delta)/(2\times 0.8\times N_{max})}$. 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.

In out experiments, HooVer gets better than PlasmaLab, as the sampling budget increases (see Figure 5). It is not surprising that for small budgets, before a threshold depth in the tree is reached, HooVer cannot give an accurate answer. Once number of queries exceeds the threshold, the tree-based sampling of HooVer works more efficiently than PlasmaLab fro the given examples.

A conceptual example. To illustrate the above behavior of the tools, we consider a conceptual example with hitting probability given by $p_{k,\mathscr{U}}(x)$ directly without specifying $\mathbb{P}$, k and $\mathscr{U}$. Given an initial state $x = (x_1, x_2)$, $p_{k,\mathscr{U}}(x_1,x_2) = p_{max}\cdot\exp\left(-\frac{(x_1-0.5)^2+(x_2-0.5)^2}{s}\right)$, where the parameter s controls the slope of $p_{k,\mathscr{U}}(\cdot)$ around the maximum $p_{k,\mathscr{U}}(\frac{1}{2},\frac{1}{2}) = p_{max}$. 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. 6. With small query budget, PlasmaLab beats HooVer, however, as the budget increases HooVer improves swiftly and becomes better (see Figure 6). 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 much more sample efficient. This suggests that HooVer might perform better than PlasmaLab in SMC problems with hard-to-find bugs or unsafe conditions.

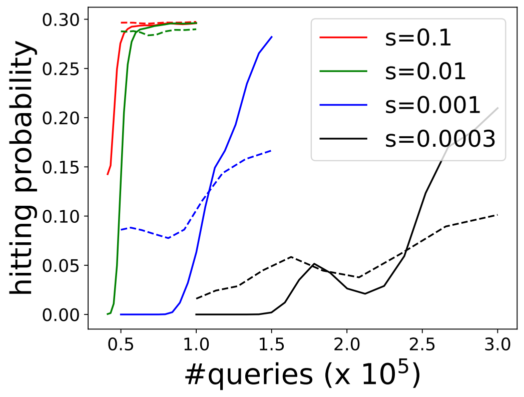

Figure 6: HooVer (solid lines) and PlasmaLab (dashed) output in optimizing sharp functions.

5 Conclusions

We presented a new tree-based algorithm, HOO-MB, for statistical model checking of nondeterministically initialized, continuous state, Markov chains (MC) by building a connection with the multi-armed bandits literature. HOO-MB sequentially samples executions of the MC in batches and relies on a lightweight assumption about the smoothness of the objective function, to find near-optimal solutions violating the given safety requirement. We provide theoretical regret bounds on the optimality gap in terms of the sampling budget, smoothness, near-optimality dimension, and sampling batch size. 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. HooVer easily handles models with 18-dimensional state spaces, and initial uncertainty spanning up to 8 dimensions. Running time and memory usage can be strongly controlled with batch sizes. Detailed comparison with other SMC tools and exploration of general MDPs with this approach would be directions for further investigation.

References

[1] *MRMC. 2011. MRMC tool.* Retrieved from `http://www.mrmc-tool.org`.

[2] Gul Agha & Karl Palmskog (2018): *A Survey of Statistical Model Checking*. *ACM Trans. Model. Comput. Simul.* 28(1), pp. 6.1–6.39, doi:10.1145/3158668. Available at `http://doi.acm.org/10.1145/3158668`.

[3] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns & Joost-Pieter Katoen (2002): *Automated Performance and Dependability Evaluation Using Model Checking*. In: *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, pp. 261–289.

[4] Benoît Boyer, Kevin Corre, Axel Legay & Sean Sedwards (2013): *PLASMA-lab: A Flexible, Distributable Statistical Model Checking Library*. In: *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems*, QEST'13, Springer-Verlag, Berlin, Heidelberg, pp. 160–164.

[5] Sébastien Bubeck, Rémi Munos & Gilles Stoltz (2011): *Pure exploration in finitely-armed and continuous-armed bandits*. *Theoretical Computer Science* 412(19), pp. 1832–1852.

[6] Sébastien Bubeck, Rémi Munos, Gilles Stoltz & Csaba Szepesvári (2011): *X-Armed Bandits*. *J. Mach. Learn. Res.* 12, pp. 1655–1695. Available at `http://dl.acm.org/citation.cfm?id=1953048.2021053`.

[7] Sébastien Bubeck, Rémi Munos, Gilles Stoltz & Csaba Szepesvári (2011): *X-armed bandits*. *Journal of Machine Learning Research* 12(May), pp. 1655–1695.

[8] Sébastien Bubeck & Nicolò Cesa-Bianchi (2012): *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. *Foundations and Trends in Machine Learning* 5(1), pp. 1–122, doi:10.1561/2200000024. Available at `http://dx.doi.org/10.1561/2200000024`.

[9] Carlos E Budde, Pedro R D'Argenio, Arnd Hartmanns & Sean Sedwards (2018): *A statistical model checker for nondeterminism and rare events*. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 340–358.

[10] Doron Bustan, Sasha Rubin & Moshe Y. Vardi (2004): *Verifying omega-Regular Properties of Markov Chains*. In: *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, pp. 189–201.

[11] Alexandre David, Peter G Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen & Jakob H Taankvist (2014): *On time with minimal expected cost!* In: *International Symposium on Automated Technology for Verification and Analysis*, Springer, pp. 129–145.

[12] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, Danny Bøgsted Poulsen, Jonas van Vliet & Zheng Wang (2011): *Statistical Model Checking for Networks of Priced Timed Automata*. In: *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, pp. 80–96.

[13] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen & Matthias Volk (2017): *A storm is coming: A modern probabilistic model checker*. In: *International Conference on Computer Aided Verification*, Springer, pp. 592–600.

[14] Pedro D'Argenio, Axel Legay, Sean Sedwards & Louis-Marie Traonouez (2015): *Smart sampling for lightweight verification of Markov decision processes*. *International Journal on Software Tools for Technology Transfer* 17(4), pp. 469–484.

[15] Christian Ellen, Sebastian Gerwinn & Martin Fränzle (2015): *Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains*. *International Journal on Software Tools for Technology Transfer* 17(4), pp. 485–504.

[16] Simone Fabris (2012): *Method for hazard severity assessment for the case of undemanded deceleration*. *TRW Automotive, Berlin.*

[17] Jean-Bastien Grill, Michal Valko & Rémi Munos (2015): *Black-box optimization of noisy functions with unknown smoothness*. In: *Advances in Neural Information Processing Systems*, pp. 667–675.

[18] Arnd Hartmanns & Holger Hermanns (2009): *A Modest approach to checking probabilistic timed automata*. In: *2009 Sixth International Conference on the Quantitative Evaluation of Systems*, IEEE, pp. 187–196.

[19] Arnd Hartmanns & Holger Hermanns (2014): *The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification*. In: *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pp. 593–598, doi:10.1007/978-3-642-54862-8_51. Available at `https://doi.org/10.1007/978-3-642-54862-8_51`.

[20] David Henriques, Joao G Martins, Paolo Zuliani, André Platzer & Edmund M Clarke (2012): *Statistical model checking for Markov decision processes*. In: *2012 Ninth International Conference on Quantitative Evaluation of Systems*, IEEE, pp. 84–93.

[21] Holger Hermanns, Björn Wachter & Lijun Zhang (2008): *Probabilistic CEGAR*. In: *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, pp. 162–175.

[22] A. Hinton, M. Kwiatkowska, G. Norman & D. Parker (2006): *PRISM: A Tool for Automatic Verification of Probabilistic Systems*. In H. Hermanns & J. Palsberg, editors: *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, *LNCS* 3920, Springer, pp. 441–444.

[23] CT Ionescu Tulcea (1949): *Mesures dans les espaces produits*. *Atti Accad. Naz. Lincei Rend* 7, pp. 208–211.

[24] ISO (2011): *ISO 26262:Road vehicles – Functional safety*. Norm ISO 26262.

[25] David N. Jansen, Joost-Pieter Katoen, Marcel Oldenkamp, Mariëlle Stoelinga & Ivan S. Zapreev (2007): *How Fast and Fat Is Your Probabilistic Model Checker? An Experimental Performance Comparison*. In: *Hardware and Software: Verification and Testing, Third International Haifa Verification Conference, HVC 2007, Haifa, Israel, October 23-25, 2007, Proceedings*, pp. 69–85.

[26] Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer & Paolo Zuliani (2009): *A Bayesian Approach to Model Checking Biological Systems*. In: *Computational Methods in Systems Biology, 7th International Conference, CMSB 2009, Bologna, Italy, August 31-September 1, 2009. Proceedings*, pp. 218–234.

[27] Kenji Kodaka, Makoto Otabe, Yoshihiro Urai & Hiroyuki Koike (2003): *Rear-end collision velocity reduction system*. Technical Report 2003-01-0503, SAE International.

[28] David Kyle, Jeffery P. Hansen & Sagar Chaki (2015): *Statistical Model Checking of Distributed Adaptive Real-Time Software*. In: *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, pp. 269–274.

[29] Tze Leung Lai & Herbert Robbins (1985): *Asymptotically efficient adaptive allocation rules*. *Advances in applied mathematics* 6(1), pp. 4–22.

[30] Richard Lassaigne & Sylvain Peyronnet (2015): *Approximate planning and verification for large Markov decision processes*. *International Journal on Software Tools for Technology Transfer* 17(4), pp. 457–467.

[31] Axel Legay, Sean Sedwards & Louis-Marie Traonouez (2016): *Plasma Lab: a modular statistical model checking platform*. In: *International Symposium on Leveraging Applications of Formal Methods*, Springer, pp. 77–93.

[32] Axel Legay & Mahesh Viswanathan (2015): *Statistical Model Checking: Challenges and Perspectives*. *Int. J. Softw. Tools Technol. Transf.* 17(4), pp. 369–376, doi:10.1007/s10009-015-0384-z. Available at `http://dx.doi.org/10.1007/s10009-015-0384-z`.

[33] Fabio Martinelli, Francesco Mercaldo, Antonella Santone, Christina Tavolato-Wötzl & Paul Tavolato: *Timed Automata Networks for SCADA Attacks Real-Time Mitigation*.

[34] João Martins, André Platzer & João Leite (2011): *Statistical Model Checking for Distributed Probabilistic-control Hybrid Automata with Smart Grid Applications*. In: *Proceedings of the 13th International Conference on Formal Methods and Software Engineering*, ICFEM'11, Springer-Verlag, Berlin, Heidelberg, pp. 131–146.

[35] José Meseguer & Raman Sharykin (2006): *Specification and Analysis of Distributed Object-based Stochastic Hybrid Systems*. In: *Proceedings of the 9th International Conference on Hybrid Systems: Computation and Control*, HSCC'06, Springer-Verlag, Berlin, Heidelberg, pp. 460–475.

[36] Rémi Munos (2014): *From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*. Technical Report. Available at https://hal.archives-ouvertes.fr/hal-00747575. 130 pages.

[37] Negin Musavi, Dawei Sun, Sayan Mitra, Geir Dullerud & Sanjay Shakkottai (2019): *Optimistic Optimization for Statistical Model Checking with Regret Bounds*. *arXiv preprint arXiv:1911.01537*.

[38] Sucheendra K. Palaniappan, Benjamin M. Gyori, Bing Liu, David Hsu & P. S. Thiagarajan (2013): *Statistical Model Checking Based Calibration and Analysis of Bio-pathway Models*. In: *Computational Methods in Systems Biology - 11th International Conference, CMSB 2013, Klosterneuburg, Austria, September 22-24, 2013. Proceedings*, pp. 120–134.

[39] Dimitri Petritis (2012): *Markov chains on measurable spaces*. Retrieved from https://perso.univ-rennes1.fr/dimitri.petritis/ps/markov.pdf.

[40] Herbert Robbins (1952): *Some aspects of the sequential design of experiments*. *Bulletin of the American Mathematical Society* 58(5), pp. 527–535.

[41] Jan J. M. M. Rutten, Marta Z. Kwiatkowska, Gethin Norman, David Parker & Prakash Panangaden (2004): *Mathematical techniques for analyzing concurrent and probabilistic systems*. *CRM monograph series* 23, American Mathematical Society. Available at http://www.ams.org/publications/authors/books/postpub/crmm-23.

[42] Stefano Sebastio & Andrea Vandin (2013): *MultiVeStA: Statistical model checking for discrete event simulators*. In: *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ICST (Institute for Computer Sciences, Social-Informatics and ..., pp. 310–315.

[43] Koushik Sen, Mahesh Viswanathan & Gul Agha (2005): *On Statistical Model Checking of Stochastic Systems*. In: *Proceedings of the 17th International Conference on Computer Aided Verification*, CAV'05, Springer-Verlag, Berlin, Heidelberg, pp. 266–280.

[44] Koushik Sen, Mahesh Viswanathan & Gul A. Agha (2005): *VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems*. In: *QEST*, pp. 251–252.

[45] Rajat Sen, Kirthevasan Kandasamy & Sanjay Shakkottai (2019): *Noisy Blackbox Optimization using Multi-fidelity Queries: A Tree Search Approach*. In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2096–2105.

[46] William R Thompson (1933): *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*. *Biometrika* 25(3/4), pp. 285–294.

[47] William R Thompson (1935): *On the theory of apportionment*. *American Journal of Mathematics* 57(2), pp. 450–456.

[48] Michal Valko, Alexandra Carpentier & Rémi Munos (2013): *Stochastic simultaneous optimistic optimization*. In: *International Conference on Machine Learning*, pp. 19–27.

[49] Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan & Geir E Dullerud (2019): *Statistical verification of PCTL using antithetic and stratified samples*. *Formal Methods in System Design* 54(2), pp. 145–163.

[50] Håkan L. S. Younes (2005): *Probabilistic Verification for "Black-Box" Systems*. In: *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, pp. 253–265.

[51] Håkan L. S. Younes & Reid G. Simmons (2002): *Probabilistic verification of discrete event systems using acceptance sampling*. In: *International Conference on Computer Aided Verification (CAV2002)*, Springer, pp. 223–235.

[52] Håkan LS Younes, Edmund M Clarke & Paolo Zuliani (2010): *Statistical verification of probabilistic properties with unbounded until*. In: *Brazilian Symposium on Formal Methods*, Springer, pp. 144–160.

[53] Paolo Zuliani, André Platzer & Edmund M. Clarke (2013): *Bayesian statistical model checking with application to Stateflow/Simulink verification*. *Formal Methods in System Design* 43(2), pp. 338–367.

A Appendix: Proof of Lemma 1

Let $x_i \in \mathscr{X}$ be the point returned by Algorithm 1 at round i. In Algorithm 1, once a node is chosen, it is queried for b times. Let $L(n) = \left\lfloor \frac{n-1}{b} \right\rfloor$, then at round n, there are $L(n)+1$ mini-batches. Denote by (h_j, i_j) the node chosen by HOO-MB in the j-th mini-batch, where $j = 1, \dots, L(n)+1$. Let z_j denotes the arm played in the j-th mini-batch, where $z_1 = x_1 = \cdots = x_b$, $z_2 = x_{b+1} = \cdots = x_{2b}, \dots$. And $y_{j,k}$ denotes the k-th observation in the j-th mini-batch.

We begin the proof of Lemma 1, with a lemma (Lemma 14 from [7]) which can be adapted to the Algorithm 1 as follows:

Lemma 2 *Let (h,i) be a sub-optimal node, i.e. $\Delta_{h,i} > 0$. Let $0 \le l \le h-1$ be the largest depth such that (l, i_l^*) is on the path from the root $(0,1)$ to (h,i), where i_l^* is the optimal nodes at depth l. Then for all integers $u \ge 0$, we have*

$$\mathbb{E}[t_{h,i}(n)] \le u + \sum_{j=u+1}^{L(n)+1} \mathbb{P}\Big\{ [U_{s,i_s^*}(j) \le f^* \textit{ for some } s \in \{l+1, \dots, j-1\}] \\ \textit{or } [\, t_{h,i}(j) > u \textit{ and } U_{h,i}(j) > f^*] \Big\}.$$

See [7] for proof of Lemma 2.

In order to come up with a bound for $\mathbb{E}[t_{h,i}(n)]$, we will use the following two lemmas (Lemma 3 and Lemma 4). Lemma 3 gives a bound for the first term, and Lemma 4 gives a bound for the second term.

Lemma 3 *Let Assumption 2 hold. Then for all optimal nodes (h,i) and all integers $n \ge b$,*

$$\mathbb{P}\{U_{h,i}(n) \le f^*\} \le (L(n)+1)^{-3}.$$

Proof 3 *For the cases that (h,i) was not chosen in the first n rounds, by convention, $U_{h,i}(n) = +\infty$. Therefore, We focus on the event $\{t_{h,i}(n) \ge 1\}$:*

$$\begin{aligned}
&\mathbb{P}\{U_{h,i}(n) \le f^* \textit{ and } t_{h,i}(n) \ge 1\} \\
&= \mathbb{P}\Big\{\hat{f}_{h,i}(n) + \sqrt{\frac{2\sigma^2 \log(L(n)+1)}{b t_{h,i}(n)}} + \nu\rho^h \le f^* \textit{ and } t_{h,i}(n) \ge 1\Big\} \\
&= \mathbb{P}\Big\{t_{h,i}(n)\hat{f}_{h,i}(n) + t_{h,i}(n)(\nu\rho^h - f^*) \le -\sqrt{\frac{2\sigma^2 t_{h,i}(n) \log(L(n)+1)}{b}} \\
&\qquad \textit{and } t_{h,i}(n) \ge 1\Big\}
\end{aligned}$$

$$= \mathbb{P}\Big\{ \sum_{l=1}^{L(n)+1} \Big(\frac{\sum_{j=1}^{b} y_{l,j}}{b} - f(z_l)\Big)\mathbb{I}_{\{(h_l,i_l)\in\mathscr{C}(h,i)\}} + \sum_{l=1}^{L(n)+1} (f(z_l) + \nu\rho^h - f^*)\mathbb{I}_{\{(h_l,i_l)\in\mathscr{C}(h,i)\}} \leq -\sqrt{\frac{2\sigma^2 t_{h,i}(n)\log(L(n)+1)}{b}} \text{ and } t_{h,i}(n) \geq 1\Big\},$$

where z_l is the point chosen in the l-th mini-batch, and $y_{l,1}, y_{l,2}, \ldots, y_{l,k}$ are k observations at point z_l, and $\mathscr{C}(h,i)$ denotes the descendants of the node (h,i). According to the Assumption 2 with $c = 0$, $(f(z_l) + \nu\rho^h - f^*)\mathbb{I}_{\{(h_l,i_l)\in\mathscr{C}(h,i)\}} \geq 0$. Therefore we can write:

$$\mathbb{P}\{U_{h,i}(n) \leq f^* \text{ and } t_{h,i}(n) \geq 1\} \leq \mathbb{P}\Big\{ \sum_{l=1}^{L(n)+1} \Big(f(z_l) - \frac{\sum_{j=1}^{b} y_{l,j}}{b}\Big)\mathbb{I}_{\{(h_l,i_l)\in\mathscr{C}(h,i)\}} \geq \sqrt{\frac{2\sigma^2 t_{h,i}(n)\log(L(n)+1)}{b}} \text{ and } t_{h,i}(n) \geq 1\Big\}.$$

Since $y_{l,j}$ is assumed to be σ^2-sub-Gaussian, $\frac{\sum_{j=1}^{b} y_{l,j}}{b}$ is $\frac{\sigma^2}{b}$-sub-Gaussian. Similar to the approach in proof of Lemma 15 *in [7], we can take care the last inequality with union bound, optional skipping and the Hoeffding-Azuma inequality and obtain the following:*

$$\mathbb{P}\{U_{h,i}(n) \leq f^* \text{ and } t_{h,i}(n) \geq 1\} \leq (L(n)+1)^{-3},$$

where we conclude the proof of lemma 3. ■

Lemma 4 *For all integers $j \leq n$, all sub-optimal nodes (h,i) such that $\Delta_{h,i} > \nu\rho^h$, and all integers $u \geq 1$ such that*

$$u \geq \frac{8\sigma^2 \log(L(n)+1)}{b(\Delta_{h,i} - \nu\rho^h)^2},$$

one has

$$\mathbb{P}\{U_{h,i}(j) > f^* \text{ and } t_{h,i}(j) > u\} \leq (L(j)+1)(L(n)+1)^{-4}.$$

Proof 4 *The u in the statement satisfies*

$$\frac{\Delta_{h,i} - \nu\rho^h}{2} \geq \sqrt{\frac{2\sigma^2 \log(L(n)+1)}{bu}},$$

thus $\sqrt{\frac{2\sigma^2 \log(L(j)+1)}{bu}} + \nu\rho^h \leq \frac{\Delta_{h,i} + \nu\rho^h}{2}$. Therefore,

$$\begin{aligned} &\mathbb{P}\{U_{h,i}(j) > f^* \text{ and } t_{h,i}(j) > u\} \\ &= \mathbb{P}\Big\{\hat{f}_{h,i}(j) + \sqrt{\frac{2\sigma^2 \log(L(j)+1)}{b\, t_{h,i}(j)}} + \nu\rho^h > f^*_{h,i} + \Delta_{h,i} \text{ and } t_{h,i}(j) > u\Big\} \\ &\leq \mathbb{P}\Big\{t_{h,i}(j)(\hat{f}_{h,i}(j) - f^*_{h,i}) > \frac{\Delta_{h,i} - \nu\rho^h}{2} t_{h,i}(j) \text{ and } t_{h,i}(j) > u\Big\}. \end{aligned}$$

Now similar to the approach in proof of Lemma 16 *in [7] we can take care of the last inequality with union bound, optional skipping and the Hoeffding-Azuma inequality and obtain the following:*

$$\mathbb{P}\{U_{h,i}(j) > f^* \text{ and } t_{h,i}(j) > u\} \leq (L(j)+1)(L(n)+1)^{-4},$$

where we conclude the proof of lemma 4. ∎

Now combining the results of Lemma 2, Lemma 3 and Lemma 4 we obtain the following for sub-optimal nodes with $\Delta_{h,i} > \nu\rho^h$:

$$\mathbb{E}[t_{h,i}(n)] \leq \frac{8\sigma^2 \log(L(n)+1)}{b(\Delta_{h,i} - \nu\rho^h)^2} + 1 + \sum_{j=u+1}^{L(n)+1} \Bigg((L(j)+1)(L(n)+1)^{-4} + \sum_{s=1}^{L(j)} (L(j)+1)^{-3} \Bigg).$$

Now similar to the approach in the proof of the Lemma 16 in [7] and using $L(n) = \lfloor \frac{n-1}{b} \rfloor$, we get the following:

$$\mathbb{E}[t_{h,i}(n)] \leq \frac{8\sigma^2 \log(\lfloor \frac{n-1}{b} \rfloor + 1)}{b(\Delta_{h,i} - \nu\rho^h)^2} + 4,$$

which concludes the proof of Lemma 1.

B Appendix: Proof of Theorem 1

We follow the proof of regret bound from [7, 45]. Let $x_i \in \mathscr{X}$ be the point returned by Algorithm 1 at round i. Define cumulative regret at round N as

$$R_N = \sum_{i=1}^{N} (f^* - f(x_i)).$$

We first provide an upper bound for the cumulative regret and then obtain an upper bound for the simple regret.

In Algorithm 1, once a node is chosen, it is queried for b times. Let $L(N) = \left\lfloor \frac{N-1}{b} \right\rfloor$. Therefore, at round N, there are $L(N)+1$ mini-batches. Denote by (h_j, i_j) the node chosen by HOO-MB in the j-th mini-batch, where $j = 1, \ldots, L(N)+1$. Let z_j denotes the arm played in the j-th mini-batch, where $z_1 = x_1 = \cdots = x_b$, $z_2 = x_{b+1} = \cdots = x_{2b}, \ldots$. And $y_{j,k}$ denotes the k-th observation in the j-th mini-batch. Using these notations, we rewrite the cumulative regret at round N by splitting it into two parts, regret of the first L batches and that of the last batch:

$$\mathbb{E}[R_N] = \sum_{j=1}^{L(N)} b(f^* - f(z_j)) + (N - bL(N)) \cdot (f^* - f(z_{L(N)+1})).$$

Now, let I_h be the set of all nodes at depth h that are $2\nu\rho^h$-optimal, where $h = 0,1,2,\ldots$. Hence, $I_0 = (0,1)$. Now let I be the union of I_h's. In addition, let $\mathscr{J}_h$ be the set of nodes at depth h that are not in I and whose parents are in I_{h-1}. Then denote by $\mathscr{J}$ the union of all $\mathscr{J}_h$'s.

Similar to [7], we partition the tree $\mathscr{T}$ into three subsets, $\mathscr{T} = \mathscr{T}^1 \cup \mathscr{T}^2 \cup \mathscr{T}^3$, as follows. Let H be an integer to be chosen later. The set $\mathscr{T}^1$ contains the descendants of the nodes in I_H (by convention, a node is considered its own descendant); $\mathscr{T}^2 = \cup_{0\le h<H} I_h$; $\mathscr{T}^3$ contains the descendants of the nodes in $\cup_{1\le h\le H}\ \mathscr{J}_h$.

It is easy to see from the algorithm that any node chosen by the algorithm is chosen at most once. Since (h_j, i_j) belongs to either of the $\mathscr{T}^i$'s, where $i = 1,2,3$, we can decompose the cumulative regret according to which of the sets $\mathscr{T}^i$ the node (h_j, i_j) belongs to:

$$\mathbb{E}[R_N] = \mathbb{E}[R_{N,1} + R_{N,2} + R_{N,3}], \tag{5}$$

where $R_{N,i}$ is the part of R_N for all $(h_j, i_j) \in \mathscr{T}^i$. Specifically, we have

$$\begin{aligned}\mathbb{E}[R_{n,i}] = \mathbb{E}\Bigg[&\sum_{j=1}^{L(N)} b(f^* - f(z_j))\mathbb{I}_{\{(h_j,i_j)\in\mathscr{T}^i\}} \\ &+ (N - bL(N))\cdot(f^* - f(z_{L(N)+1}))\mathbb{I}_{\{(h_{L(N)+1},i_{L(N)+1})\in\mathscr{T}^i\}}\Bigg].\end{aligned}$$

First, $\mathbb{E}[R_{N,1}]$ is easy to bound. All nodes in $\mathscr{T}^1$ are $2\nu\rho^H$-optimal. According to Assumption 2, all points in these cells are $4\nu\rho^H$-optimal. Thus we obtain the following

$$\mathbb{E}[R_{N,1}] \le 4\nu\rho^H bL(N) + 4\nu\rho^H(N - bL(N)) = 4\nu\rho^H N.$$

For $h \ge 0$, any node $(h,i) \in \mathscr{T}^2$ belongs to I_h, and thus is $2\nu\rho^h$-optimal. Therefore, all points in any cell $(h,i) \in \mathscr{T}^2$ are at least $4\nu\rho^h$-optimal. Also, by Definition 3.1, we have that $|I_h| \le B(\nu,\rho)\rho^{-d(\nu,\rho)h}$. Therefore, using the fact that each node in $\mathscr{T}^2$ is played at most in one mini-batch, we have the following:

$$\mathbb{E}[R_{N,2}] \le b\sum_{h=0}^{H-1} 4\nu\rho^h|I_h| \le 4b\nu B\sum_{h=0}^{H-1}\rho^{h(1-d)}.$$

For $h \ge 0$, any node $(h,i) \in \mathscr{T}^3$ belongs to $\mathscr{J}_h$. Since the parents of any node $(h,i) \in \mathscr{J}_h$, belong to I_{h-1}, these nodes are at least $2\nu\rho^{h-1}$-optimal. Then by Assumption 2, we have that all points that lie in these cells are at least $4\nu\rho^{h-1}$-optimal. There we can get

$$\mathbb{E}[R_{N,3}] \le b\sum_{h=1}^{H} 4\nu\rho^{h-1}\sum_{i:(h,i)\in\mathscr{J}_h}\mathbb{E}[t_{h,i}(N)], \tag{6}$$

where $t_{h,i}(N)$ is the number of mini-batches where a descendant of node (h,i) is played up to and including round N.

Using Lemma 1, inequality (6) can be written as:

$$\mathbb{E}[R_{N,3}] \le b\sum_{h=1}^{H} 4\nu\rho^{h-1}|\mathscr{J}_h|\Big(\frac{8\sigma^2\log(L(N)+1)}{b\nu^2\rho^{2h}} + 4\Big).$$

Using the fact that $|\mathscr{J}_h| \leq 2|I_{h-1}|$, we have:

$$\mathbb{E}\left[R_{N,3}\right] \leq 8b\nu B\sum_{h=1}^{H}\rho^{(h-1)(1-d)}\big(\frac{8\sigma^2\log\left(L(N)+1\right)}{b\nu^2\rho^{2h}}+4\big).$$

Now, putting the obtained bounds together, we get

$$\begin{aligned}\mathbb{E}\left[R_N\right] &\leq 4\nu\rho^H N + 4b\nu B\sum_{h=0}^{H-1}\rho^{h(1-d)} \\ &\quad + 8b\nu B\sum_{h=1}^{H}\rho^{(h-1)(1-d)}\left(\frac{8\sigma^2\log\left(L(N)+1\right)}{b\nu^2\rho^{2h}}+4\right) \\ &= O\left(\rho^H N + B\big(log(L(N)+1)+b\big)\rho^{-H(1+d)}\right). \qquad (7)\end{aligned}$$

Now choosing H such that $\rho^H = O\left(B\frac{log(L(N)+1)+b}{N}\right)$ and using $L(N) = \lfloor\frac{N-1}{b}\rfloor$, minimizes the bound in (7) as follows:

$$\mathbb{E}\left[R_N\right] = O\left(BN^{\frac{d+1}{d+2}}\left(\log(\lfloor\frac{N-1}{b}\rfloor+1)+b\right)^{\frac{1}{d+2}}\right) \qquad (8)$$

Using the Remark 1 from [7], we can relate the simple regret S_N and the cumulative regret R_N and obtain the following:

$$\mathbb{E}(S_N) = O\left(\left(\frac{B\log(\lfloor\frac{N-1}{b}\rfloor+1)+b}{N}\right)^{\frac{1}{d+2}}\right),$$

which concludes the proof of the Theorem 1.