IEEE Open Journal of
Intelligent Transportation
Systems

Received 28 March 2025; revised 10 October 2025; accepted 24 November 2025. Date of publication 28 November 2025; date of current version 3 December 2025.

Digital Object Identifier 10.1109/OJITS.2025.3638166

Verifying Safety of Safety-Critical Systems With Rare Events via Optimistic Optimization

TABEA HENNING-GÜNTHER, DANIEL GRUJIC, TINO WERNER, LARS WEBER, BIRTE NEUROHR, AND EIKE MÖHLMANN

Institute of Systems Engineering for Future Mobility, German Aerospace Center (DLR), 26121 Oldenburg, Germany

CORRESPONDING AUTHOR: T. WERNER (e-mail: tino.werner@dlr.de)

This work was supported in part by the German Federal Ministry for Economic Affairs and Climate Action within the Projects "KI Delta Learning", "KI Wissen" and "SET Level—Simulation-Based Development and Testing of Automated Driving", based on a decision by the Parliament of the Federal Republic of Germany, and in part by the German Federal Ministry of Education and Research (BMBF) within the Project "ASIMOV-D", based on a decision of the German Bundestag.

ABSTRACT Failures of safety-critical systems such as highly automated cars may result in loss of life, significant property damage, or environmental harm. Their trustworthiness and acceptance by society relies on safe operation, i.e., they have to be safer than their human-controlled counterparts, which is called the positive risk balance and which is a prerequisite for the operation in the EU. Hence, guaranteeing sufficient safety is a crucial task that requires rigorous examination. However, critical events such as severe accidents are assumed to occur with probabilities of order 10^{-6} or less. For this, automated simulation-based approaches for the purpose of statistical model checking contribute significantly to quantitative safety assessment. Common methods such as pure Monte Carlo simulation are inadequate to estimate the probability of these rare critical events due to excessively high simulation budget required. To overcome this, we provide a mathematical framework for combining an optimization algorithm, here from the family of optimistic optimization algorithms, with importance sampling in order to assess the safety of these systems quantitatively. Our methodology relies on a given criticality function that assesses each state of the underlying deterministic system regarding prescribed safety requirements. Applying the approach to a common test function and a simulated braking scenario using the software SILAB showcases that our method significantly reduces the required effort to quantify acceptable risk levels, compared to pure Monte Carlo simulation.

INDEX TERMS Safety verification, safety-critical systems, optimistic optimization, statistical model checking, rare events.

I. INTRODUCTION

SAFETY-CRITICAL systems (SCSs) are by definition systems whose failure can lead to severe consequences: loss of life, significant property damage, or environmental harm. Hence, guaranteeing that they are sufficiently safe is a crucial task that requires rigorous examination. For complex systems like robotic assistance (e.g., in surgeries) or even more complex for automated mobility (e.g., autonomous cars), that interact with a complex environment, this is difficult as **i)** hand-crafted analytical proofs require abstractions that are often too pessimistic, **ii)** automated reasoning, e.g., model checking, needs a complete, accurate, and formal description of the system and its environment, **iii)** exhaustive examination (comparison with expectation) is impossible due to its uncountably many instantiations, and **iv)** real-world testing is very time- and cost-intensive, but also far too risky. This can be partly overcome by an (1) automated (to address **i**,**iv**) (2) simulation-based approach (to address **ii**,**iv**)[1] involving (3) statistical methods for inferring adequate safety arguments (to address **i**,**ii**,**iii**). The latter is due to the fact that statistical methods can provide evidence although only finitely many instances are inspected while each instance allows for individual simplifications, and

The review of this article was arranged by Associate Editor Johannes Betz.

[1]This relies on the validity of the models used for simulation.

quantitative statements can take a level of significance and accuracy into account.

As SCSs are typically carefully constructed already from the beginning, we assume the probability of fatal failures to be of small order of magnitude, say 10^{-6} or less. To name an example, highly automated driving is introduced (among other reasons) to decrease the number of accidents, fatal or not. Therefore, automated cars have to be safer than human-controlled ones, i.e., they achieve a positive risk balance, which is a necessary requirement for the licensing process, as prescribed by the German Ethics Commission ([7]) and ethics experts of the European Commission ([6]), and recognized as acceptance criterion for automated road in the ISO/TS 5083:2025 norm.[2] In particular, severe accidents with humans caused by automated cars need to occur fewer than about twice per one million miles [12], [21]. Tackling the problem with finite, predefined test catalogs leaves blind spots and makes relevant cases hard to identify due to the SCS being a complex system operating in open context [27]. On the other hand, sampling-based methods such as pure Monte Carlo simulation—where one randomly generates simulations—are inadequate to estimate the probability with a sufficient accuracy as the simulation budget required for this may be excessively high [29].

In this article, we therefore investigate a system under test (SUT) together with its environment by criticality-driven simulation, regarding some imposed safety requirements on the SUT, addressing the following research task: *Efficiently estimating the probability of rare critical events, i.e., when the SUT violates the safety requirements*. To this end, we further assume the existence of a deterministic function—we call it *criticality function (regarding safety requirements)*—that maps each state of the combined system *"SUT+environment"* onto a numerical value representing the satisfaction or violation of the safety requirement. This comes along with a threshold reflecting the safety requirements such that critical events, i.e., violations of the requirements, coincide with the criticality exceeding the threshold.

In this context, we propose applying statistical hypothesis testing to infer whether some generated sampling results give probabilistic evidence that the safety requirements hold in order to obtain a safety argument for the SUT [14]. For efficient hypothesis testing of high quality, we need to derive a low-variance estimator for the critical event probability with tight confidence bounds from the simulation results. For this, we suggest to deploy the common variance reduction technique of *importance sampling (IS)* (see, e.g., [25]) to artificially increase the occurrence of the (actually) rare critical events during simulation. IS draws realizations from a second so-called proposal distribution and accounts for this when estimating the probability of interest. Although other types of model checking approaches such as numerical algorithms exist, the closed box nature of the mapping between the scenario parameters and the criticality values as well as the large number of states in complex scenarios such as driving scenarios necessitates a statistical approach (e.g., [14], [38]), which, as we are interested in checking whether the probability of fatal failures exceeds some threshold, directly amounts to statistical model checking.

For statistical model checking, observing fatal events is not necessary. In particular, if prior knowledge about the failure probability already exists, even without any observed critical event, one may be able to provide accurate estimates using Bayesian methods (e.g., [5], [38]). However, while variance reduction is of course vital in order to keep the model checking costs low, observing fatal events has the practical advantage to inspect the causes in more detail and maybe to even optimize the SUT again. Moreover, when performing model checking of complex systems, the required prior knowledge may not be available. Therefore, we aim at guiding the simulations into critical regions and using IS in order to get unbiased estimates.

We extend the well-established use of IS by accompanying IS with an optimization algorithm for assembling a proposal distribution. Applied to the criticality function, the optimization algorithm guides the system into critical events by incorporating previous simulation results. In this work, we restrict ourselves to the family of *optimistic optimization (OO)* algorithms because they yield a partition of the input space, which we exploit in a mixture importance sampling approach. Ideally, the fineness of this partition correlates with critical regions. Existing alternatives to OO algorithms are cross-entropy techniques in order to find a suitable proposal density. Such approaches often require distributional assumptions (e.g., [36], [37]). In addition, OO algorithms are powerful algorithms that are widely applicable as they assume only minimal smoothness assumptions and knowledge about the objective function [4], which we aim at harnessing for finding a suitable proposal density for IS.

A usual approach to the here considered problem is importance splitting (see, e.g., [30], [32]), which however relies crucially on finding a suitable transition kernel to simulate from a conditioned probability density, which is far from trivial. In addition, as opposed to our work, a commonly faced problem statement in literature is to modify transition probabilities in a Markov decision process in order to see critical situations more frequently (see, e.g., [3], [28]).

In this work, however, we modify the static environment parameters of the SUT in the sense that we search for parameter configurations that lead to critical events. Here, we restrict our attention to the case of SUTs with deterministic behavior w.r.t. static parameters. That is, if we run a simulation of the combined system with the same parameters again, we get exactly the same results in terms of criticality. In other words, our goal is to assess the safety of the SUT by estimating the probability that scenarios exhibiting a critical behavior of the SUT occur.

[2]https://www.iso.org/standard/81920.html

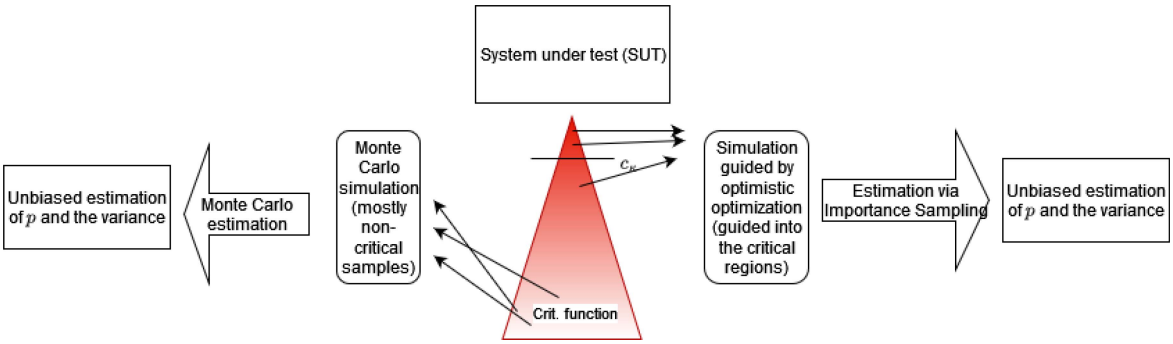

FIGURE 1. A graphical representation of the workflow. The idea is to use optimistic optimization in order to increase the fraction of critical samples. By applying importance sampling, an unbiased estimator for the probability of a critical event can be computed.

Our proposed methodology should be seen as supplement to, rather than substitution of, expert-driven safety assessment. Apart from that, we also require a certain degree of expert knowledge within the scope of our methodology, e.g., for modeling a parameter space that includes relevant phenomena and a well-defined criticality function (from criticality analysis) that reflects the safety requirements adequately.

1) Contribution: We contribute to the safety verification of SCSs by providing a mathematical framework for combining algorithms from the field of optimistic optimization with IS. Our framework assesses the safety quantitatively by estimating bounds on the occurrence probability of critical events w.r.t. some given safety requirements. Experimental results with a common test function and, in particular, on a driving scenario simulated via the software SILAB,[3] confirm the notably improved efficacy compared to pure Monte Carlo simulation. An overview of our methodology is depicted in Fig. 1.

2) Outline: In Section II, we briefly recapitulate the idea of importance sampling and discuss related work. In Section III, we outline the proposed methodology. Section IV describes an exemplary application of the methodology, on the test function Mishra's Bird and on simulated driving scenarios using Silab. In Section V, our findings are reflected and future work is identified.

II. BACKGROUND AND RELATED WORK

We start with a brief description of importance sampling and its required ingredients.

Let F_X be an absolutely continuous distribution on some space $\mathcal{X}$ with corresponding density p_X. Let further $g\colon \mathcal{X} \to \mathbb{R}$ be some function. Consider the problem to approximate the expectation $\mathbb{E}[g(\boldsymbol{X})]$ for $\boldsymbol{X} \sim F_X$ empirically. The *pure Monte Carlo (MC)* method draws independent and identically distributed (i.i.d.) realizations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and approximates $\mathbb{E}[g(\boldsymbol{X})] \approx \sum_i g(\boldsymbol{x}_i)/N$. If $p := P(\kappa(\boldsymbol{X}) \geq c_\kappa)$ should be estimated, where in our setting $\kappa\colon \mathcal{X} \to \mathbb{R}$ is a criticality function and c_κ some threshold (see Section III), we set $g(\boldsymbol{X}) := I(\kappa(\boldsymbol{X}) \geq c_\kappa)$ where $I(\cdot)$ denotes the indicator function, taking the value 1 if the logical statement in the brackets is true, 0 otherwise. MC, however, requires a huge number of samples to reliably estimate small p, rendering it computationally intractable. More precisely, in order to obtain a reasonable relative error, the number N of samples must be of order p^{-1}, e.g., [18], which would require 10^6 samples in order to reasonably estimate a probability of order 10^{-6}.

The idea of *importance sampling (IS)* (see, e.g., [25]) is to draw the realizations from another distribution, a so-called *proposal distribution* with a corresponding *proposal density*, which should put more mass on the region of interest. However, since just sampling from the proposal distribution and averaging the function values $g(\boldsymbol{x}_i)$ would clearly impose a bias, a change of measure has to be executed which requires that the proposal density q dominates the density p_X. The motivation behind IS is that a clever choice of q can significantly decrease the variance of the IS estimator $\hat{p}_q$, which is defined as $\hat{p}_q = 1/N \sum_i I(\kappa(\boldsymbol{x}_i) \geq c_\kappa) p_X(\boldsymbol{x}_i)/q(\boldsymbol{x}_i)$ in our case. In other words, a change of measure from F_X to the distribution corresponding to q is performed, leading to the importance weights $p_X(\boldsymbol{x}_i)/q(\boldsymbol{x}_i)$. Setting $q(\boldsymbol{x}) = p_X(\boldsymbol{x}) I(\kappa(\boldsymbol{x}) \geq c_\kappa)/p$ would result in an unbiased, zero-variance estimator, however, it is intangible due to the unknown value p that has to be estimated in the first place. The crucial question therefore is how the proposal density has to be chosen in practice since an inappropriate one may even increase the variance of the estimator compared to the variance of the MC estimator. For the variance $\hat{\sigma}_q$ of the IS estimator, an approximate $(1-\alpha)$ confidence interval is given by $[\hat{p}_q \pm t_{N-1}^{-1}(1-\alpha/2)\hat{\sigma}_q]$, where $t_{N-1}^{-1}(1-\alpha/2)$ denotes the $(1-\alpha/2)$-quantile of a t-distribution with $N-1$ degrees of freedom.

Approaches based on IS for statistical model checking in the context of SCSs and hybrid systems have already been suggested in the literature. There are methods using, e.g., IS with a cross entropy criterion [37], IS combined with guided simulation by a variant of reinforcement learning [28], mixture IS with a Rényi distance minimization criterion [31], adaptive IS techniques [26], and adaptive IS combined

[3] https://wivw.de/de/silab

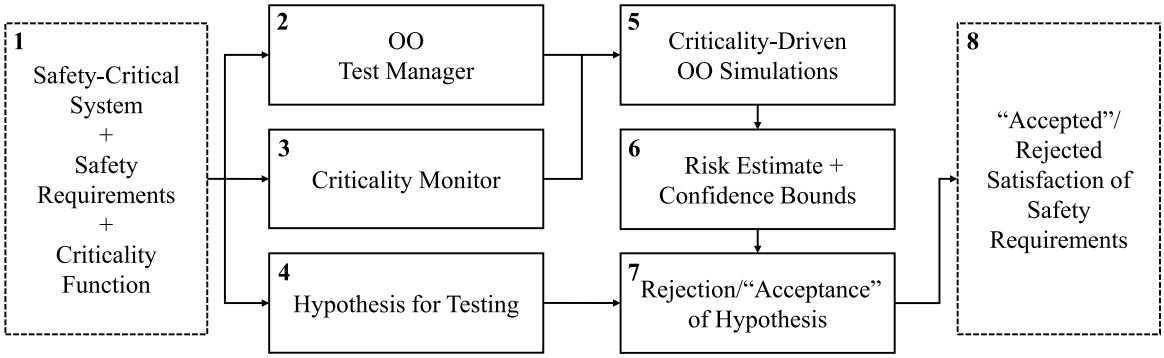

FIGURE 2. A graphical representation of the methodology. Boxes depict elements of the methodology (inputs and output in dashed lines), arrows depict dependencies or data flow. The numbering (1 – 8) is provided as a reference in the textual description of the methodology.

with Kriging in the sense that the unknown function is approximated by Kriging to detect a suitable region where to sample from next [1], [2].

Further, there are approaches invoking stochastic optimization algorithms to some extent like [36] where the optimal proposal density for IS is computed via stochastic optimization. Ernst et al. [8] provide a tree search algorithm that iteratively finds critical inputs. In contrast to our work, they do not intend to estimate the probability of critical events. Gerwinn et al. [9] use statistical model checking for the safety verification in nested optimization problem setting to which they apply a randomized maximization algorithm to compute confidence bounds for the probability of satisfaction.

The application of a multi-armed bandit strategy for statistical model checking has been done by Musavi et al. [20], but with a different goal, i.e., approximating the probability of attaining a critical state after at most k time steps, starting from a non-critical initial state. They do not perform statistical tests as they restrict themselves to approximating these probabilities of failure.

Reference [15] propose the `Daisee` algorithm which combines stratified IS and multi-armed bandits by starting with an initial partition of the underlying space and defining a proposal distribution on each region which leads to a mixture proposal distribution. They iteratively select a region in a multi-armed bandit style, i.e., taking the region for which the respective upper confidence bound is largest, and adaptively update the weights while letting the partition fixed. They also propose the `HiDaisee` algorithm which iteratively learns a finer partition in the regions where the density fluctuates strongly.

Currently, there are several approaches in the literature where the cross-entropy, the Kullback-Leibler divergence or some specific criterion is optimized. We are not aware of an approach where an optimization algorithm is applied first and the results in form of a partition of the input space or the sampled points are used for finding a suitable proposal density, which is done in this work.

III. METHODOLOGY

The general structure of our proposed methodology to derive quantitative safety statements for SCSs is depicted in Fig. 2, along with involved elements, dependencies and data flows. Following the numbering (**1** – **8**) given therein, we lead through the process in detail.

A. INPUTS AND OUTPUT (ELEMENTS 1 AND 8)

The proposed methodology is intended to contribute to the safety verification of a **safety-critical system (SCS)** w.r.t. imposed **safety requirements**, both of which we assume are given beforehand (El. **1**). More precisely, we consider the SUT together with the environment in which it is intended to function (operational design domain), also referred to as combined system *"SUT+environment"*. Our methodology ultimately aims at **testing the system's compliance with the safety requirements** (El. **8**).

In order to ensure the applicability of the methodology, we assume a d-dimensional space of relevant parameters $\mathcal{X} \subseteq \mathbb{R}^d, d \in \mathbb{N}$, reflecting the environment of the SUT to be given (e.g., specified as a logical scenario [16], [22]). A fixed parameter configuration is denoted by $\boldsymbol{x} \in \mathcal{X}$. Attached to the parameter space, we further assume that a probability density function p_X is given, where $\boldsymbol{X}$ denotes the random variable whose realizations correspond to single parameter configurations.

As a further crucial prerequisite for applying our methodology, we require the user to be able to determine a numerical, bounded criticality value for every realized parameter sample $\boldsymbol{x} \in \mathcal{X}$ with regard to the imposed safety requirements.[4] That is, a bounded **criticality function** κ defined on the realizations of $\boldsymbol{X}$ is required reflecting the degree of violation of the safety requirements (El. **1**). In addition, the user has to determine a suitable **threshold** c_κ so that the event $\{\kappa(\boldsymbol{X}) < c_\kappa\}$ coincides with the safety

[4]For instance, if a model of the combined system *"SUT+environment"* is at hand that can be simulated with a simulator, the criticality may be obtained by a criticality monitor observing the criticality of the simulation runs. For the sake of simplicity, we assume this case for the following considerations.

requirements being satisfied. Thus, the objective we aim for with the methodology is to obtain a confidence statement about the probability that a critical event w.r.t. the pair (κ, c_κ) occurs:

$$p = P(\kappa(\boldsymbol{X}) \geq c_\kappa). \tag{1}$$

In this work, we do not consider uncertainties that might arise from, e.g., non-deterministic behavior of the SUT, i.e., we assume that the system behavior can be considered deterministic in the sense that the same parameters defining the scenario lead to the same criticality value. In our simulations, we start with a logical scenario w.r.t. the pair (κ, c_κ) (see Section IV-B) from which random concrete scenarios are realized. We assume that simulating the concrete scenario does not add further uncertainties to p (cf. [22]).

B. STATISTICAL HYPOTHESIS TESTING (ELEMENTS 4 AND 7)

We propose employing *statistical hypothesis testing* in order to infer whether the probability of critical events occurring for the SCS is below a prescribed threshold θ [14]. This translates to the following **null hypothesis for testing** (El. **4**):

$$H_0 : p \geq \theta, \quad H_1 : p < \theta, \quad \text{for} \quad p = P(\kappa(\boldsymbol{X}) \geq c_\kappa) \tag{2}$$

with level α.[5]

C. CRITICALITY-DRIVEN OO SIMULATIONS (ELEMENTS 5 AND 3)

As explained earlier, we expect the critical event probability p w.r.t. (c_κ, κ) as given in Eq. (1) to be of order of 10^{-6} or less if κ and c_κ are chosen suitably to reflect the safety requirements. To overcome the ineptitude of pure MC for estimating p,[6] we propose a two-phase approach that combines IS with OO. With the ideal proposal density $q \propto p_X I(\kappa(\cdot) \geq c_\kappa)$ in mind, a natural strategy is to try to detect the critical regions, i.e., where $\kappa \geq c_\kappa$, and to significantly increase the mass of those regions. In other words, we draw **criticality-driven OO samples** in the first phase of our approach, guiding into critical regions, while **resampling** from the resulting proposal distribution in the second phase (El. **5**; see Section III-E).

The core idea of OO is to optimize a given function (here the criticality function κ) over a measurable space $\mathcal{X}$ by iteratively finding a gradually finer *partition* of $\mathcal{X}$, aiming at obtaining the finest resolutions in regions with high function values (see, e.g., [4], [19]). More precisely, OO methods incrementally build a hierarchically structured K-ary tree $\{\mathcal{P}_{h,i}\}$ that forms a disjoint partition of the parameter space $\mathcal{X}$ at every depth $h \geq 0$, where each node (h, i) (h: depth, i: index) is associated with a specific subregion of $\mathcal{X}$. Whenever a node is split, meaning that K child nodes are created, the corresponding subset $\mathcal{P}_{h,i}$ will be divided among its children, which leads to smaller subsets with increasing depth h. Based on the respective assumptions, some OO approaches compute optimistic upper bounds on the maximal function value on each of the subregions to indicate the profitableness of choosing the corresponding nodes.

As to obtain criticality values $\kappa(\boldsymbol{x}_i)$ for the IS proposal distribution, the following iteration loop is carried out in the first phase (**criticality-driven OO simulation**, El. **5**). According to the OO algorithm at hand, single parameter configurations $\boldsymbol{x}_i \in \mathcal{X}$ are generated iteratively and passed as input, e.g., to a simulation engine to simulate the combined system *SUT+environment* for the parameter configuration $\boldsymbol{x}_i$. The corresponding criticality values $\kappa(\boldsymbol{x}_i)$ are determined. For instance, a **criticality monitor** (El. **3**) observes κ for the simulation run induced by $\boldsymbol{x}_i$. The criticality monitor might also observe whether a critical event occurs (i.e., $\kappa \geq c_\kappa$). Then, the corresponding criticality value is passed back to the OO test manager (see Section III-D) to derive the next configuration $\boldsymbol{x}_{i+1}$ according to the chosen OO algorithm. This realization $\boldsymbol{x}_{i+1}$ serves as next input for the simulation, etc., until some stopping criterion becomes valid.

D. OO TEST MANAGER (ELEMENT 2)

For criticality-driven OO simulation, we need to instantiate the **OO test manager** (El. **2**). That is, a specific OO algorithm is chosen and set up with the corresponding hyperparameters (if any), along with the simulation budget N. In this work, we consider the following three OO algorithms.

The `Deterministic Optimistic Optimization (DOO)` algorithm, introduced by Munos [19], exploits certain knowledge of the smoothness of the criticality function to approximate its global optimum sequentially, quantified by a so-called semi-metric which is assumed to be *known*. The algorithm hierarchically partitions the parameter space $\mathcal{X}$ into regions whose diameters decrease with the depth (hierarchy level) h. In the present work, we restrict ourselves to the setting $\delta(h) = \nu\rho^h$ with $\nu > 0$, $\rho \in (0, 1)$, as suggested in [19, Sec. 3.2].

In the same setting, Munos proposes a second approach: the `Simultaneous Optimistic Optimization (SOO)` algorithm [19]. It allows for the semi-metric to be *unknown*, which makes `SOO` applicable in a broader sense. A user-given function $h_{\max}(t)$ is intended to control the algorithm's behaviour regarding the trade-off between exploration and exploitation, and thus has to be chosen carefully. Large values for $h_{\max}(t)$ can lead to deep exploiting trees such that regions with high rewards in the past are sampled from more often, while small values force the algorithm to sample more in less explored regions.

Bartlett et al. [4] provide the `Sequential Online Optimization (SequOOL)` algorithm that only requires

[5]Note that for a given SCS and given safety requirements, both θ and α will be of individual nature as they may be domain- and stakeholder-specific, although they might be derived from generally accepted values [12].

[6]Note that we would have to provide a number of samples in the reciprocal order of p, i.e., 10^6 or more, to ensure that the variance of the pure MC estimator is reasonably low. In general, this is far from feasible in terms of available simulation budget.

a local smoothness of the function around (all of) its global optima However, SequOOL can be applied without knowing the corresponding smoothness parameters as emphasized in [4].

As for drawing samples during an OO algorithm, we suggest to sample realizations *uniformly* from the regions (instead of defining a fixed, representative point for each region as suggested in [4], e.g.,) for our purposes. By this, each region is armed with a uniform proposal density due to the fact that OO algorithms aim at finding the global optimum whose occurrence probability is clearly zero (unless the distribution has a point mass there). Note that sampling from a non-uniform density, e.g., the conditional density of $\mathcal{X}$ on the given region, would potentially lead to the conflictive situation that the interesting subregion is disadvantaged if the region-specific proposal density is low there, which can be expected as critical cases are assumed to be rare, hence decelerating the respective algorithm.

E. RESAMPLING AND CONFIDENCE ESTIMATE (ELEMENTS 5 AND 6)

Now, deriving a suitable **estimate** along with an upper **confidence bound** for the critical event probability p (El. **6**) would typically require i.i.d. realizations $\boldsymbol{x}_i$. However, this is not the case when applying an OO algorithm since we have realizations from different distributions, where the realization $\boldsymbol{x}_i$ in a given iteration determines which density is sampled from in the subsequent iteration. This dependence and an unknown dependence structure due to κ being a unknown function (i.e., the distribution of $\kappa(\boldsymbol{X})$ is intangible) do not allow for the application of standard estimation techniques.

As our criticality function is unknown, we first sample OO-simulations $\boldsymbol{x}_k^{(1)}, k = 1, \ldots, N_1 < N$, to identify critical regions and apply a *mixture IS* technique afterwards (**resampling**, El. **5**). Since OO algorithms are designed to (asymptotically) find a global optimum, we can expect them to provide a much finer resolution of the partition where the criticality function takes large values than in regions with small function values.

Let J be the number of final leaves of the partitioning tree that the applied algorithm has computed, corresponding to disjoint regions. In order to maintain this partition, our idea is to apply mixture IS, i.e., our proposal density is a mixture of densities in the sense

$$q_w = \sum_{j=1}^{J} w_j q_j, \tag{3}$$

where q_j is the uniform density on the region $\mathcal{P}_j$ corresponding to cell j. Such uniform region-specific densities also have been suggested in [15]. It remains to define suitable mixture weights $w_1, \ldots, w_J$ such that $\sum_j w_j = 1$, $w_j > 0$ for all j.

We use criticality-based weights, based on the mean criticality of all samples found in a particular partition. For this, we first normalize all criticality values sampled from the OO-simulations to $[0, 1]$, using a standard linear transformation that maps the lowest observed value to 0 and the highest to 1. Based on these normalized evaluations $\check{\kappa}(\boldsymbol{x}_k^{(1)})$ we define the weight w_j of cell j as

$$\tilde{w}_j = c + \frac{1}{N_j} \sum_{k=1}^{N_1} \check{\kappa}\left(\boldsymbol{x}_k^{(1)}\right) I_j\left(\boldsymbol{x}_k^{(1)}\right) \tag{4}$$

where

$$I_j(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \mathcal{P}_j \\ 0, & \boldsymbol{x} \notin \mathcal{P}_j \end{cases} \tag{5}$$

and $N_j \leq N_1$ denotes the number of observations from this particular cell. We add an additional constant $c = 1.0$ to the weights to prevent regions with zero weight (due to the lack of corresponding observations). As the weights from Eq. (4) are contained in the interval $[1.0, 2.0]$, the final weights are computed by the standardization

$$\hat{w}_j = \frac{\tilde{w}_j}{\sum_{i=1}^{J} \tilde{w}_i} \tag{6}$$

As for the drawing process for the subsequent $(N - N_1)$ samples, note that $\hat{w}_j$ can be interpreted as the probability of drawing a realization according to the respective q_j. This can be easily realized with a multinomial distribution on the set $\{1, \ldots, J\}$ with the probability vector $(\hat{w}_1, \ldots, \hat{w}_J)$. However, it has been pointed out (see, e.g., [24]) that deterministic mixture sampling, i.e., generating around $\hat{w}_j(N - N_1)$ samples according to q_j for all j, leads to an estimator with even lower variance. Hence, this variant may be recommended which can be realized by standard stratified sampling.

Once the mixture proposal density is selected, we can draw independent realizations $\boldsymbol{x}_k^{(2)}$, $k = 1, \ldots, N - N_1$, from it and compute the approximate confidence interval for p. The mixture IS estimator is then given by

$$\hat{p}_{q_{\hat{w}}} = \frac{1}{N - N_1} \sum_{k=1}^{N-N_1} \frac{I\left(\kappa(\boldsymbol{x}_k^{(2)}) \geq c_\kappa\right) p_X\left(\boldsymbol{x}_k^{(2)}\right)}{\hat{w}_{j_k} \frac{1}{\mathrm{vol}(\mathcal{P}_{j_k})}}, \tag{7}$$

where j_k indicates the region in which $\boldsymbol{x}_k^{(2)}$ falls and where $\mathrm{vol}(\mathcal{P}_j) = \int_{\mathcal{P}_j} 1 \, dx$. As for the variance of the mixture IS estimator $\hat{p}_{q_{\hat{w}}}$ (see, e.g., [11]), we can only use the evaluations $\kappa(\boldsymbol{x}_k^{(2)})$ to empirically approximate it since the actual shape of the function κ is not known, which leads to

$$\hat{\sigma}^2_{q_{\hat{w}}} = \frac{1}{N - N_1} \sum_{k=1}^{N-N_1} \left(\frac{I\left(\kappa(\boldsymbol{x}_k^{(2)}) \geq c_\kappa\right) p_X\left(\boldsymbol{x}_k^{(2)}\right)}{\hat{w}_{j_k} \frac{1}{\mathrm{vol}(\mathcal{P}_{j_k})}} - \hat{p}_{q_{\hat{w}}} \right)^2, \tag{8}$$

F. ALGORITHM

Our overall algorithm is given in Algorithm 1.

We assume that a suitable criticality function κ is available. Of course, the design of such a criticality function is very difficult in practice (cf. [35]). This is however not a problem induced by our methodology but the quantification

TABLE 1. Mean variance of the probability estimators (based on 1000 replications each) using the reward-based weights, compared to pure Monte Carlo (MC) (left column). N_R denotes the number of repetitions in which the algorithm was able to detect at least one critical event. The number in brackets denotes the variance estimated by MC based on these N_R repetitions.

	MC		DOO(2000,0.7)		SOO(0.6)		SequOOL	
c_κ	$\hat{\sigma}^2_{\text{MC}}$	N_R	$\hat{\sigma}^2_{q\hat{w}}$	N_R	$\hat{\sigma}^2_{q\hat{w}}$	N_R	$\hat{\sigma}^2_{q\hat{w}}$	N_R
60	0.02281013	1000	0.00428413	1000	0.00381313	1000	0.00811913	1000
100	0.00250513	1000	0.00016713	1000	0.00004513	1000	0.00015213	1000
106.5	0.00009313 (0.00014713)	634	0.00000513	1000	0.00000013	1000	0.00000013	1000

Algorithm 1 OO-Driven Mixture IS

Require: Total simulation budget N, OO budget N_1, OO algorithm, density p_X, criticality function κ, confidence level α

Output: Confidence interval for crit. event probability p as in Eq. (1)

$k \leftarrow 1$

while $k \leq N_1$ **do**

$\boldsymbol{x}_k^{(1)} \leftarrow$ ApplyOOAlgorithm

$k \leftarrow k + 1$

end while

$(\mathcal{P}_j)_j \leftarrow$ GetRegions(OO algorithm)

$(\tilde{q}_j)_j \leftarrow (1/\text{vol}(\mathcal{P}_j))_j$

$(\hat{w}_j)_j \leftarrow (\hat{w}_j)_j$ as defined in Eq. (6)

for $k = 1, ..., N - N_1$ **do**

$\boldsymbol{X}_k \sim Q_{\hat{w}}$ i.i.d. for a distribution $Q_{\hat{w}}$ with density $q_{\hat{w}}$ as defined in Eq. (3)

$\boldsymbol{x}_k^{(2)} \leftarrow$ DrawRealization($\boldsymbol{X}_k$)

end for

$\hat{p}_{q_{\hat{w}}} \leftarrow \hat{p}_{q_{\hat{w}}}$ as defined in Eq. (7)

$\hat{\sigma}^2_{q_{\hat{w}}} \leftarrow \hat{\sigma}^2_{q_{\hat{w}}}$ as defined in Eq. (8)

return $\left[0, \hat{p}_{q_{\hat{w}}} + \hat{\sigma}_{q_{\hat{w}}} t^{-1}_{N-N_1-1}(1-\alpha)/\sqrt{N-N_1}\right]$.

of criticality is a general requirement for safety verification. It is important to note that the choice of the threshold c_κ is not necessary for the OO-algorithm. Therefore, our methodology is robust against potential changes in the criticality threshold, e.g., by updated safety standards, allowing for re-computing the estimates for a different safety threshold without requiring to repeat the expensive simulations or real-world runs.

IV. EXPERIMENTS AND RESULTS

A. APPLICATION ON A TEST FUNCTION

We apply Algorithm 1 in different variants on a test criticality function to initially validate our proposed probability estimation (El. **6** in Fig. 2). *Mishra's Bird* [17] is a standard test function for optimization and used, among others, in the context of reliability analysis for automated driving [23]:

$$\kappa(x_1, x_2) = -\sin(x_2)e^{\left[(1-\cos(x_1))^2\right]} - \cos(x_1)e^{\left[(1-\sin(x_2))^2\right]} - (x_1 - x_2)^2, \tag{9}$$

here constrained on $\mathcal{X} = [-10, 0] \times [-6.5, 0]$.

TABLE 2. Mean relative absolute error of the probability estimators (based on 1000 replications each) using the reward-based weights. The number in brackets denotes the error estimated by MC based on the N_R repetitions where at least one critical event was detected.

c_κ	MC	DOO(2000,0.7)	SOO(0.6)	SequOOL
60	0.0498	0.0229	0.0217	0.0307
100	0.1621	0.0403	0.0219	0.0341
106.5	0.7243 (0.5652)	0.1941	0.0282	0.0453

Mishra's bird can be seen as a typical criticality function that exhibits multiple local extrema and only few very critical peaks. In this particular case, we further know that the function is Lipschitz continuous and bounded with optimum $\kappa^* \approx 106.764537$ on $\mathcal{X}$.

We consider $c_\kappa \in \{60.0, 100.0, 106.5\}$, corresponding approximately to $p \in \{0.02336, 0.00248, 9.362 \cdot 10^{-5}\}$, respectively.[7] We instantiate Algorithm 1 with $N = 10^4$, $N_1 = 500$, $p_X = 1/\text{vol}(\mathcal{X})$ (uniform dist.), $\alpha = 0.05$, and apply all three OO methods DOO, SOO, and SequOOL with binary trees ($K = 2$). While no hyperparameters are required for SequOOL, we heuristically test several admissible configurations for DOO and SOO w.r.t. the exploration-exploitation trade-off in the OO simulation phase. As a result, we investigate DOO instances with $(\nu, \rho) = (2000, 0.7)$ more closely, whereas for SOO we follow an approach in [19] by choosing $h_{\max}(t) = t^\varepsilon$, and opt for $\varepsilon = 0.6$ ($h_{\max}$ fairly restrictive w.r.t. exploitation). We denote these parameter settings by writing DOO(2000,0.7), SOO(0.6), and SequOOL (since no parameters are required here) in the tables below. The experiment is repeated with 1000 replications for every instantiation of Alg. 1. In Tab. 1, we report the mean variances for the different experiments, comparing standard MC with our approach (Alg. 1), while Tab. 2 shows the mean relative absolute errors, i.e., the average of the $|\hat{p}^{(b)} - p|/p$, for the estimated probabilities $\hat{p}^{(b)}$ on replication b, over all 1000 replications.

We can see from Tab. 2 that our approach reliably estimates the critical event probability with a lower mean absolute relative error than MC in all experiments. At the same time, it improves the variance (see Tab. 1), up to two orders of magnitude, compared to MC.

As for the applied OO algorithms, it seems that SequOOL and SOO are better suited for our methodology than DOO. The reason for that is, that our methodology heavily favours exploration over exploitation. Note that we are not interested in a single global optimum, but in the full area where the

[7] Estimated by 10^8 Monte Carlo samples.

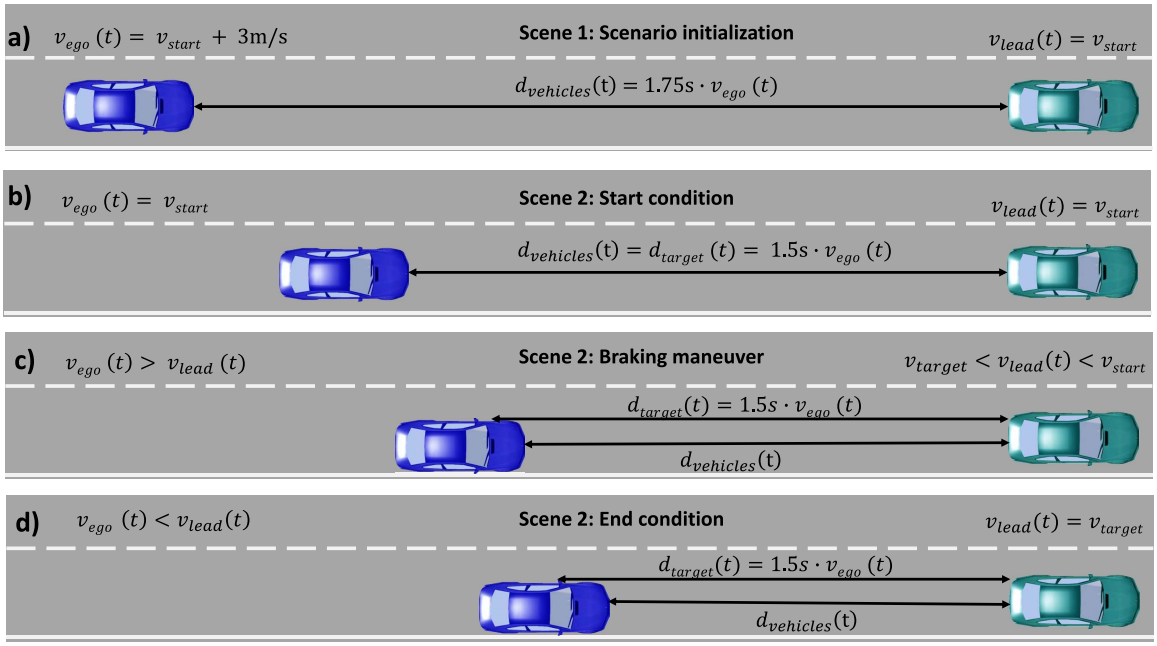

FIGURE 3. Depicted are four different states of the scenario. In a), during the initialization condition, the ego-vehicle is still far away from the lead vehicle. Compared to the lead vehicle, ego is set up with a slightly higher speed ($+3ms^{-1}$) and slowly approaches the leading car until it has reached the time headway of 1.5s in b). Reaching the time headway of 1.5s, the front vehicle will abruptly start braking until it has reached its target velocity v_{target}, which is shown in c). Because the ego-vehicle requires a certain reaction time and has a limited maximum deceleration, its actual time headway distance will fall under the ACC's target time headway distance of 1.5s, which is visualized by the two different arrow lengths between the vehicles. During this phase, the criticality given by Eq. (10) and (11) is measured. The braking maneuver ends as soon as the velocity of the ego-vehicle will fall below the velocity of the lead vehicle. This is the end condition shown in d). The start and target velocities v_{start} and v_{target} of the leading vehicle are the scenario parameters and are held constant for each scenario.

criticality κ can exceed the critical event threshold c_κ. SOO and SequOOL naturally lean more towards exploration, since they do not require any information about the smoothness of the input function. However, DOO leans more towards exploitation as it works on assumptions on how much the criticality can raise in its individual partitions. Due to this, SOO and SequOOL are not only more easily configurable than DOO but also much more naturally-suited towards our methodology.

B. APPLICATION TO A DRIVING SCENARIO IN SILAB

Additionally, we also applied the algorithm to a driving scenario simulated in the commercial software SILAB, v7.2,[8] which we have extended with a specific traffic controller to generate the repeatable traffic conditions required by our scenario.

The scenario for our study is a typical highway braking scenario. Our ego-vehicle is equipped with an Adaptive Cruise Control (ACC) system. The goal is to keep a fixed target time headway of 1.5s to the leading vehicle. The scenario is structured into two scenes. During the initialization scene, the ego-vehicle approaches a slower lead vehicle from a distance of initially 1.75s time headway. When the ego-vehicle has reached the distance of 1.5s time headway, the scenario will enter the second scene, where the leading vehicle suddenly begins to brake. As the ego-vehicle cannot brake fast enough, this will lead to a distance smaller than the intended target distance and thus to possible critical behavior. A more detailed overview of the scenario can be found in Fig. 3.

The internal traffic engine in SILAB allows to mainly control additional vehicles based on virtual waypoints / event locations, e.g., the ego vehicle drives over a certain virtual waypoint and triggers a behavioral change of a certain traffic participant. This procedure is helpful if one wants to repeat the same scenario in a specific road layout and everything can be triggered location-based. For our scenario, we need to re-setup the other traffic participant thousands of times with varying relative distance and speed to the ego vehicle on a highway. Because the speed and behavior of ego changes in every situation, there is no way to define this in a location-based manner. But, SILAB offers the possibility to control up to 20 additional traffic participants through a specific interface which allows to manipulate the 6 degrees of freedom of the 3D objects. A simplified physics model for longitudinal and lateral position, speed and acceleration was implemented and used and can be configured to relocate the vehicles, change speed instantaneously to create new scenario starting conditions and also allows to command acceleration and deceleration maneuvers. Fig. 3 shows that, e.g., during initialization the lead vehicle is set up at a certain relative distance and speed and whenever c) is reached, a braking command is issued with a certain deceleration. It should be mentioned, that the automation system is using

[8]https://wivw.de/en/silab-2/

ground truth data from the simulation, we did not add, e.g., a radar simulation, which would add noise and make the experiments non-repeatable.

The scenario is parameterized using the following two parameters. The first parameter is the start velocity v_{start} of the leading vehicle in m/s at the beginning of the simulation. The second parameter is the target velocity v_{target} of the leading vehicle after the braking process in m/s. The parameter space is constrained to $\mathcal{X} = [24.0, 32.0] \times [15.0, 20.0]$ (uniformly distributed). As metric for the criticality, we measure the largest difference between the intended target distance $d_{\text{target}}(t)$ and the actual distance $d_{\text{vehicles}}(v_{\text{start}}, v_{\text{target}}, t)$ between the two vehicles w.r.t. the scenario parameters v_{start} and v_{target}. More precisely, for each test run, the respective distance is measured at each time step and the criticality value assigned to the whole test run is this largest difference. Let the timestamps t_s and t_e denote the start and end of the measuring time period during the braking maneuver. Formally, the criticality κ of one simulation run is therefore defined by

$$\kappa\left(v_{\text{start}}, v_{\text{target}}\right) = \frac{d_{\text{critical}}\left(v_{\text{start}}, v_{\text{target}}\right)}{c_{\text{norm}}} \tag{10}$$

and

$$d_{\text{critical}}\left(v_{\text{start}}, v_{\text{target}}\right) = \max_{t_s < t < t_e} \left(d_{\text{target}}(t) - d_{\text{vehicles}}(v_{\text{start}}, v_{\text{target}}, t)\right) \tag{11}$$

normalized by a constant c_{norm}. By this definition, the criticality increases proportionally with the decreasing distance below the intended target distance of 1.5s time headway (ego undershoots or cuts into the target distance of the ACC). For the scenario and ACC at hand, we choose $c_{\text{norm}} = 15\text{m}$, because in none of the scenarios the ego vehicle undershot the target distance with a higher value, and $c_\kappa = 0.85$. In the context of certification of SCSs, the manufacturer and the corresponding authorities have to choose values for these constants in dependence of the targeted safety requirements. c_κ should be chosen in a way that all values above c_κ can be considered critical or undesired.

We initialized Algorithm 1 with $N = 1000$, $N_1 = 150$. Since SOO is easily configurable and achieved the best results in Section IV-A, we only ran the experiments with SOO as subroutine, using the same hyperparameter configuration as in Section IV-A. Due to the long runtime of the individual simulations, we repeated the experiment only 40 times, leading to overall 40.000 simulation runs being performed. The results are compared to Monte Carlo simulations ($N = 1000$) and reported in Tab. 3. In this experiment, most of the MC repetitions fail in the sense that no critical event is observed. On the contrary, our method was able to detect in all 40 repetitions at least one critical event and also decreased the variance by multiple orders of magnitude compared to Monte Carlo. Of course, one would in practice not use the degenerate zero-variance estimate $\hat{p} = 0$ but compute a confidence interval based on other means, for example, a Clopper-Pearson confidence interval (e.g., [34]). However, this interval would, for example for $\alpha = 0.05$, be approximately $[0, 0.00368]$, i.e., with a width of order 10^{-3}, while the corresponding one-sided IS-intervals would have a width of order 10^{-5}.

TABLE 3. Results from the simulation study in SILAB. The experiment was repeated 40 times. The calculated values for mean $\hat{\sigma}^2$ are based on the N_R repetitions for which the algorithm was able to detect at least one critical event.

	Monte Carlo	SOO(0.6)
mean $\hat{p}$	$7.5 \cdot 10^{-5}$	$8.15 \cdot 10^{-5}$
mean $\hat{\sigma}^2$	$1.0 \cdot 10^{-3}$	$3.73 \cdot 10^{-7}$
N_R	3	40

TABLE 4. Results from the simulation study in SILAB. The experiment was repeated 20 times. The calculated values for mean and median $\hat{\sigma}^2$ are based on the N_R repetitions for which the algorithm was able to detect at least one critical event.

	Monte Carlo	SOO(0.6)
mean $\hat{p}$	0.00105	0.00096
mean $\hat{\sigma}^2$	0.00175	0.00183
median $\hat{\sigma}^2$	0.001	$4.83 \cdot 10^{-5}$
N_R	12	17

We performed 40 repetitions of the experiment on the SILAB scenario (and 1000 for the experiment on Mishra's Bird) to show the improvements of our algorithm compared to Monte Carlo. In practice, one would run only a single batch of simulations instead. For this, an exemplary 95%-confidence interval for SOO from one of our 40 batches from the SILAB experiments is $[2.97 \cdot 10^{-5}, 9.81 \cdot 10^{-5}]$. In the context of our methodology, this confidence interval indicates that, with a confidence of 95%, the probability p that a critical event occurs (here: that the ego vehicle is more than $0.85 \cdot 15\text{m} = 12.75\text{m}$ below the targeted safety distance) is lower than 0.00981%. If p is lower than the θ specified for this scenario by the responsible authorities, then the system would be accepted w.r.t. the safety requirements, else it would be returned to the engineer for further modification.

We considered another scenario with a larger search space. In particular, we allowed for the time headway at which the ego vehicle starts to brake as well as the maximum deceleration to vary, leading to the four-dimensional parameter space $\mathcal{X}^{\text{new}} = [24.0, 32.0] \times [15.0, 20.0] \times [1.25, 1.8] \times [-5, -3.2]$. We again consider a uniform distribution, now over $\mathcal{X}^{\text{new}}$. All other settings are as before, we only change the safety threshold to $c_\kappa = 0.99$. We simulated 20 repetitions. The results are presented in Tab. 4.

While the results in Tab. 3 clearly favor our approach, apparently, when considering the mean variance, the results in Tab. 4 favor Monte Carlo, However, the median variance achieved by our approach is considerably smaller than the median variance achieved by Monte Carlo. As explained above, all cases where no critical event has been observed does not correspond to a meaningful estimated variance. We now compute the confidence interval widths and consider the Clopper-Pearson intervals $[0, 0.00368]$ for all repetitions without critical event. The resulting boxplots are shown in Fig. 4. They reveal that while in the simulations over the parameter space $\mathcal{X}$, all interval widths corresponding to our

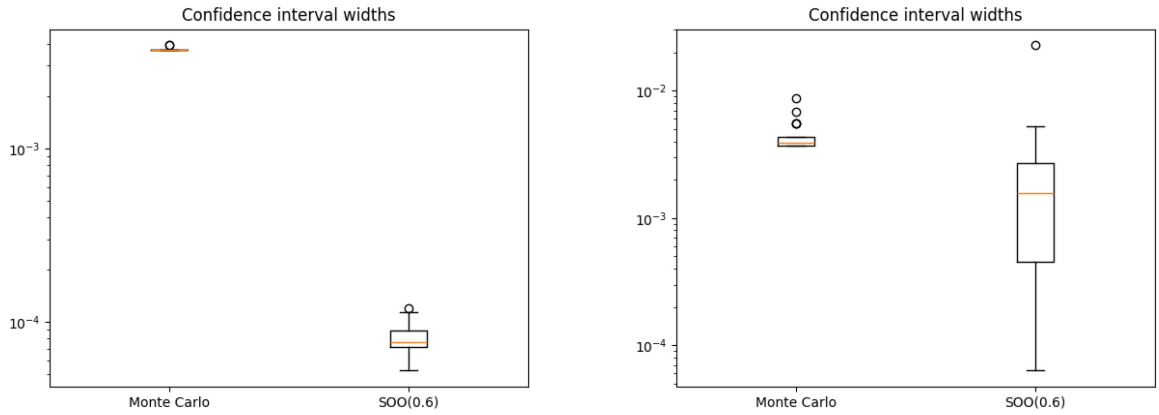

FIGURE 4. Boxplots of the confidence interval lengths corresponding to the simulations with parameter space $\mathcal{X}$ (top) and $\mathcal{X}^{\text{new}}$ (bottom), respectively. The left boxplot always corresponds to Monte Carlo, the right boxplots to our method.

method are smaller than those corresponding to Monte Carlo, there is one outlier in the simulations over the extended parameter space $\mathcal{X}^{\text{new}}$, resulting from a variance outlier that distorts the arithmetic mean in Tab. 4. The effect results from the well-known fact that the proposal distribution is often skewed (e.g., [13], [33]) and can be considered as a general issue of IS, and not of particularly our approach. Apart from this instance, our method clearly dominates Monte Carlo.

V. CONCLUSION AND FUTURE WORK

We presented a methodology for applying optimization algorithms in order to assess the safety of SCSs w.r.t. some given safety requirements quantitatively. Our mathematical framework comprises applying these optimization algorithms on a (deterministic) criticality function reflecting the safety requirements to increase the frequency of rare critical events during simulation, and estimating bounds on the probability of their occurrence. For this, we proposed a strategy that combines optimization and importance sampling (IS). The optimization methods identify critical regions, enabling to suitably select the weights in the resulting mixture proposal density for IS. In the context of this study, we restricted ourselves to optimization algorithms from the field of Optimistic Optimization (OO) (specifically `DOO` [19], `SOO` [19] and `SequOOL` [4]) as input algorithms as they provide some advantages for our methodology like a ready-to-use partitioning of the input space.

Strengths of our approach: **i**) Reliable estimation of rare event probabilities with a lower relative error than MC in all performed experiments. **ii**) A considerable improvement of the variance estimator compared to MC which improves even further with a decreasing critical event probability. **iii**) Reliable detection of rare critical events despite limited simulation budget. **iv**) User-friendly applicability thanks to the absence of hyperparameters for `SequOOL` and a single hyperparameter for `SOO`, respectively.[9]

[9]As for a good hyperparameter selection, our methodology enables the use of parallel approaches such as `POO` [10] with, e.g., `DOO` as base algorithm so that suitable hyperparameters can be chosen adaptively.

Topics for future work: **i**) Considering multiple safety criteria, given by criticality functions κ_r, $r = 1, \ldots, R$, in parallel, may either lead to a weighted aggregation of these functions into a single criticality function or, more informative and avoiding defining the respective weights, to multiple testing. **ii)** Extension of our approach for the case of noisy function evaluations, e.g., induced by non-deterministic behavior of the SUT. This amounts to the application of stochastic OO algorithms and the estimation of p via potentially nested simulations in order to capture the non-determinism for each parameter configuration. **iii)** Lastly, as our methodology is not strictly depending on the use of OO algorithms, it would be interesting to apply other types of optimization algorithms as well (like, e.g., gradient-based methods or particle swarm methods).

REFERENCES

[1] M. Balesdent, J. Morio, and L. Brevault, "Rare event probability estimation in the presence of epistemic uncertainty on input probability distribution parameters," *Methodol. Comput. Appl. Probab.*, vol. 18, no. 1, pp. 197–216, 2016.

[2] M. Balesdent, J. Morio, and J. Marzat, "Recommendations for the tuning of rare event probability estimators," *Rel. Eng. Syst. Safety*, vol. 133, pp. 68–78 Jan. 2015.

[3] B. Barbot, S. Haddad, and C. Picaronny, "Coupling and importance sampling for statistical model checking," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.*, 2012), pp. 331–346.

[4] P. L. Bartlett, V. Gabillon, and M. Valko, "A simple parameter-free and adaptive approach to optimization under a minimal local smoothness assumption," in *Proc. 30th Int. Conf. Algorithmic Learn. Theory*, 2019, pp. 184–206.

[5] L. Belzner and M. Wirsing, "Synthesizing safe policies under probabilistic constraints with reinforcement learning and Bayesian model checking," *Sci. Comput. Program.*, vol. 206, pp. 1–22, Jun. 2021.

[6] J. F. Bonnefon et al., "Ethics of connected and automated vehicles: Recommendations on road safety, privacy, fairness, explainability and responsibility," presented at Horizon 2020 Commission, 2020.

[7] U. Di Fabio et al., "Ethics commission automated and connected driving,"presented at Federal Ministry Transport Digital Infrastruct. Federal Republic Germany, 2017.

[8] G. Ernst, S. Sedwards, Z. Zhang, and I. Hasuo, "Fast falsification of hybrid systems using probabilistically adaptive input," in *Proc. Int. Conf. Quant. Eval. Syst.*, 2019, pp. 165–181.

[9] S. Gerwinn, E. Möhlmann, and A. Sieper, "Statistical model checking for scenario-based verification of ADAS," in *Control Strategies for Advanced Driver Assistance Systems and Autonomous Driving Functions*. Cham, Switzerland: Springer, 2019, pp. 67–87.

[10] J. B. Grill, M. Valko, and R. Munos, "Black-box optimization of noisy functions with unknown smoothness," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 667–675. [Online]. Available: http://papers.nips.cc/paper/5721-black-box-optimization-of-noisy-functions-with-unknown-smoothness.pdf

[11] H. Y. He and A. B. Owen, "Optimal mixture weights in multiple importance sampling," 2014, *arXiv:1411.3954*.

[12] P. Junietz, U. Steininger, and H. Winner, "Macroscopic safety requirements for highly automated driving," *Transport. Res. Record*, vol. 2673, no. 3, pp. 1–10, 2019. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/8656/

[13] W. Kouw and M. Loog, "Effects of sampling skewness of the importance-weighted risk estimator on model selection," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, 2018, pp. 1468–1473.

[14] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification*, H. Barringer et al., Eds. Heidelberg, Germany: Springer, 2010, pp. 122–135.

[15] X. Lu, T. Rainforth, Y. Zhou, J.-W. van de Meent, and Y. Whye Teh, "On exploration, exploitation and learning in adaptive importance sampling," 2018, *arXiv:1810.13296*.

[16] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 1821–1827.

[17] S. K. Mishra, "Some new test functions for global optimization and performance of repulsive particle swarm method," 2006. [Online]. Available: http://dx.doi.org/10.2139/ssrn.926132

[18] J. Morio, R. Pastel, and F. Le Gland, "An overview of importance splitting for rare event simulation," *Eur. J. Phys.*, vol. 31, no. 5, pp. 1295–1303, 2010.

[19] R. Munos, "Optimistic optimization of a deterministic function without the knowledge of its smoothness," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 783–791.

[20] N. Musavi, D. Sun, S. Mitra, G. Dullerud, and S. Shakkottai, "Optimistic optimization for statistical model checking with regret bounds," 2019, *arXiv:1911.01537*.

[21] "Traffic safety facts 2017—A compilation of motor vehicle crash data (annual report)," Nat. Highway Traffic Safety Admin., Washington, DC, USA, Sep. 2019. Accessed: Mar. 22, 2021. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812806

[22] C. Neurohr, L. Westhofen, T. Henning, T. de Graaff, E. Möhlmann, and E. Böde, "Fundamental considerations around scenario-based testing for automated driving," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2020, pp. 121–127.

[23] R. Niemeier, G. Gallee, and B. Dion, "Workflow based exploration of parameter space and reliability analysis for automated driving," presented at Driving Simulat. Conf. Europe 2020 VR, 2020. Accessed: Mar. 29, 2021. [Online]. Available: https://dsc2020.org/Docs/Slides/DSC_2020_Presentation_Ansys_Roland_Niemeier.pdf

[24] A. Owen and Y. Zhou, "Safe and effective importance sampling," *J. Amer. Stat. Assoc.*, vol. 95, no. 449, pp. 135–143, 2000.

[25] A. B. Owen, "Monte Carlo theory, methods and examples," 2013. [Online]. Available: https://artowen.su.domains/mc/

[26] R. Pastel, "Estimation de probabilités d'évènements rares et de quantiles extrêmes: Applications dans le domaine aérospatial," Ph.D. dissertation, École doctorale Mathématiques, télécommunications, informatique, signal, systèmes, électronique (Rennes), Université de Rennes, Rennes, France, 2012.

[27] A. Poddey, T. Brade, J. E. Stellet, and W. Branz, "On the validation of complex systems operating in open contexts," 2019, *arXiv:1902.10517*.

[28] S. Puch, M. Fränzle, and S. Gerwinn, "Quantitative risk assessment of safety-critical systems via guided simulation for rare events," in *Leveraging Applications of Formal Methods, Verification and Validation. Verification*, T. Margaria, B. Steffen, Eds. Cham, Switzerland: Springer Int., 2018, pp. 305–321.

[29] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87456–87477, 2020.

[30] G. Rubino and B. Tuffin, *Rare Event Simulation Using Monte Carlo Methods*. Hoboken, NJ, USA: Wiley, 2009.

[31] E. Schmerling and M. Pavone, "Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning," 2016, *arXiv:1609.05399*.

[32] J. F. Shortle and P. L'Ecuyer, "Introduction to rare-event simulation," in *Wiley Encyclopedia of Operations Research and Management Science*. Hoboken, NJ, USA: Wiley, 2010.

[33] P. J. Smith, M. Shafi, and G. Hongsheng, "Quick simulation: A review of importance sampling techniques in communications systems," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 4, pp. 597–613, May 1997.

[34] M. Thulin, "The cost of using exact confidence intervals for a binomial proportion," *Electron. J. Stat.*, vol. 8, no. 1, pp. 817–840, 2014.

[35] L. Westhofen et al., "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *Arch. Comput. Methods Eng.*, vol. 30, pp. 1–35, Jan. 2023.

[36] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc, "Accelerated evaluation of automated vehicles in car-following maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 733–744, Mar. 2018.

[37] D. Zhao et al., "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 595–607, Mar. 2017.

[38] P. Zuliani, A. Platzer, and E. M. Clarke, "Bayesian statistical model checking with application to stateflow/simulink verification," *Formal Methods Syst. Design*, vol. 43, no. 2, pp. 338–367, 2013.