Evaluating Risk and Confidence in Performance Bounds of Configuration Sampling Strategies

KALLISTOS WEIS, Saarland University, Germany
MARTINA MAGGIO, Saarland University, Germany and Lund University, Sweden
NORBERT SIEGMUND, ScaDS.AI, Germany and Leipzig University, Germany
SVEN APEL, Saarland University, Germany

Modern software usually exposes a large number of configuration options to the user, giving rise to enormous configuration spaces in practice. Appropriate choices for these options dramatically influence the performance of the software (throughput, memory consumption, execution time, etc.). However, due to the sheer size of the configuration space, systematically identifying the worst- or best-performing configurations is computationally infeasible through exhaustive exploration. Instead, practitioners rely on budgeted sampling strategies, such as uniform random sampling or statistical recursive search, to explore the configuration space under fixed measurement budgets in an attempt to find the worst- or best-performing configuration. Even worse, a fundamental limitation of existing sampling strategies is the lack of quantifiable guarantees that the selected configuration truly reflects worst-case (or best-case) performance. In this paper, we define the basic concepts of *posterior risk* and *posterior confidence* and present a probabilistic framework to evaluate how well sampling strategies identify the worst- or best-performing configuration of a software system. We evaluate our framework by comparing five representative sampling strategies on seven real-world configurable software systems. We find that statistical recursive search yields consistently tighter best-case guarantees—higher posterior confidence and lower posterior risk—than the alternatives at the same budget. Our results demonstrate the applicability of our framework as a principled basis for reporting, comparing, and refining sampling strategies, and as a tool for practitioners to select strategies and budgets with quantified guarantees across systems and sample sizes.

CCS Concepts: • **Software and its engineering** → **Software reliability**; **Software performance**; *Software configuration management and version control systems*; • **Theory of computation** → Stochastic approximation.

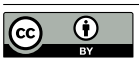
Additional Key Words and Phrases: configurable software systems, performance bounds, sampling strategies, scenario theory, probabilistic guarantees

ACM Reference Format:
Kallistos Weis, Martina Maggio, Norbert Siegmund, and Sven Apel. 2026. Evaluating Risk and Confidence in Performance Bounds of Configuration Sampling Strategies. *Proc. ACM Softw. Eng.* 3, FSE, Article FSE140 (July 2026), 24 pages. https://doi.org/10.1145/3808147

1 Introduction

Modern configurable software systems expose large numbers of configuration options that may substantially influence the systems' performance, such as execution time, throughput, or resource consumption [15, 62]. In practice, however, it is not only the performance of individual configurations that matters, but also the performance characteristics across some or all configurations, in

Authors' Contact Information: Kallistos Weis, Saarland University, Saarbrücken, Germany, kallistos@cs.uni-saarland.de; Martina Maggio, Saarland University, Saarbrücken, Germany and Lund University, Lund, Sweden, maggio@cs.uni-saarland.de; Norbert Siegmund, ScaDS.AI, Leipzig / Dresden, Germany and Leipzig University, Leipzig, Germany, norbert.siegmund@cs.uni-leipzig.de; Sven Apel, Saarland University, Saarbrücken, Germany, apel@cs.uni-saarland.de.



ACM 2994-970X/2026/7-ARTFSE140
https://doi.org/10.1145/3808147

particular the worst-case and best-case behavior. To ensure that performance requirements are met for the whole configuration space, it is crucial to identify reliable worst-case and best-case performance bounds. For instance, safety-critical systems often require guarantees on worst-case execution times to ensure timely responses [64], while cloud-based services may seek best-case throughput bounds to optimize resource utilization and cost efficiency [33]. Exhaustive performance measurement to determine bounds is prohibitive in practice even for moderately sized systems [2, 3], as the space of valid configurations grows exponentially with the number of configuration options. Consequently, engineers rely on sampling a limited set of configurations and use the observed worst-case or best-case performance (or statistical estimates thereof) as proxies for the system's true performance bounds [10, 50, 51]. However, such sample-based bounds may be violated by unseen configurations [10, 15, 31, 51], raising a fundamental question: *How reliable are performance bounds obtained from a finite sample?*

Answering this question is non-trivial because existing sampling strategies do not provide explicit guarantees or statistical approximations about their ability to capture the true worst-case or best-case performance bounds of the system. A key reason is that existing sampling strategies have been developed with different objectives in mind, such as uniform random sampling [22, 28], coverage-based sampling [34, 39], solver-based approaches [65, 66], and learning-guided or adaptive strategies [7, 56]. Furthermore, they are typically evaluated empirically based on performance values or their utility for downstream tasks such as performance modeling or configuration optimization [5]. Hence, with the absence of explicit guarantees on the ability to capture the true worst-case or best-case performance bound, there is no systematic way to *quantify the probability that an unseen configuration performs worse than the worst-case configuration observed in the sample*, nor to compare different sampling strategies systematically with respect to such guarantees. As a result, practitioners may observe identical worst-case performance bounds from different sampling strategies while facing substantially *different and unknown levels of risk*.

To reason rigorously about sampled performance bounds, two probabilistic dimensions are essential: (1) the *risk* level, that is the probability that the true performance bound lies outside the performance bound observed in the sample and (2) the *confidence* level, that is the probability that the risk estimate reflects the true risk associated with the observed bound. Considering either dimension in isolation is insufficient: a low risk estimate without sufficient confidence may be unreliable, while high confidence in a weak risk estimate offers little practical value. Despite their central role in performance-related decision making, these dimensions are largely absent from the evaluation and comparison of sampling strategies for configurable software systems [5, 7, 50]. This gap motivates our endeavour for devising a novel probabilistic framework that *makes the risk and confidence of sampled worst-case and best-case performance bounds explicit and quantifiable*.

We build on an established theory from constrained optimization: Scenario theory [6, 37] provides *a priori* probabilistic guarantees for optimization problems under uncertainty based on finite samples. In our setting, each measured configuration–performance pair can be interpreted as a constraint instance in an optimization problem whose goal is to identify a configuration achieving an extreme (worst- or best-case) performance bound. Scenario theory is able to bound the risk that sampled performance bounds are violated by unseen configurations, together with an associated confidence level [37]. However, these guarantees hinge on strong assumptions, most notably, uniform random sampling from the configuration space, that are, by design, rarely met by practical sampling strategies for configurable software systems [22, 54]. In fact, coverage-based, solver-based, and adaptive sampling strategies intentionally bias the selection of configurations to achieve specific objectives (e.g., covering all pairs of configuration options), thereby violating the assumptions required for scenario-theoretic guarantees. Consequently, existing a priori guarantees cannot assess

the reliability of performance bounds produced by these widely used sampling strategies, leaving a gap between available theory and practical application.

To bridge this gap, we propose the notions of *posterior risk* and *posterior confidence* for performance bounds obtained from finite samples in configurable software systems. In contrast to scenario-theoretic a priori guarantees, these posterior guarantees are computed with respect to a known reference (e.g., ground truth or fixed baseline) and quantify how well an observed worst- or best-case performance bound approximates the corresponding true bound of the configuration space. *Posterior risk* captures the fraction of configurations that violate the observed performance bound. *Posterior confidence* quantifies how consistently this risk recurs across repeated samples of the same size. Crucially, these measures make no assumptions about how the sample was generated and therefore apply to arbitrary sampling strategies. This posterior perspective enables, for the first time, a principled assessment of the probabilistic guarantees offered by non-uniform and adaptive sampling methods. Importantly, the framework is not restricted to deterministic measurements or fully known configuration spaces as we demonstrate in Section 5.4.

We instantiate these concepts in a probabilistic framework that relates posterior risk and posterior confidence to the a priori guarantees known from scenario theory, thereby enabling direct comparison between practical sampling strategies and theoretical baselines. To demonstrate the expressiveness and practical relevance of this framework, we evaluate it on multiple real-world configurable software systems using representative sampling strategies from different categories. Note that, rather than aiming to identify a universally superior sampling strategy, our evaluation illustrates how posterior guarantees can vary substantially across strategies, systems, and sample sizes, even when strategies are designed for similar objectives. Our results highlight that sampling strategies commonly used in practice differ significantly in the level of reliability they provide, and that such differences become particularly pronounced when only limited sample budgets are available. These findings are not intended to generalize to all existing or even future sampling strategies (there are more data on eight further sampling strategies in the supplementary material, but a complete account is out of scope), but rather to demonstrate how the proposed framework empowers researchers and practitioners to gather reliability-oriented insights that remain inaccessible to traditional performance-based evaluations and can inform the evaluation, selection, and development of sampling strategies. In summary, we make the following contributions:

(i) We introduce *posterior risk* and *posterior confidence* as novel probabilistic measures that quantify the reliability of worst-case and best-case performance bounds derived from finite sample sets in configurable software systems.
(ii) We formalize the problem for the first time within a general probabilistic framework that enables the assessment of such posterior guarantees for arbitrary sampling strategies, relates them to scenario-theoretic a priori guarantees, and extends to stochastic performance measurements and incomplete configuration spaces.
(iii) We showcase our probabilistic framework's capability of providing valuable insights for selecting and evaluating sampling strategies by means of an empirical study on seven configurable software systems and five representative sampling strategies.
(iv) We support reproducibility by providing all data, code and extended results, including additional sampling strategies and per-system results, as supplement (see Section 7).

2 Background and Related Work

In many configurable software systems, the configuration space is large, rendering exhaustive measurements impractical [39]. Two principal approaches are used to determine configurations with extreme (i.e., worst-case or best-case) performance: *model learning* [13–15, 18, 56, 58, 59] and

search-based optimization [16, 20, 25, 48, 49, 57, 60]. Both rely on an effective strategy of sampling the configuration space. Given that performance is an emergent property that highly depends on configuration decisions, a wide variety of sampling strategies have been developed for this purpose—each with its own set of strengths and weaknesses, which we discuss in Section 2.1. To determine the configuration that exhibits the worst-case (or best-case) performance, the state of the art follows procedures that consists of three distinct steps: sampling, model learning, and search-based optimization, whereby sampling is the foundation of the other two and therefore in the focus of interest. More generally, identifying configurations with extreme performance can be framed as an optimization problem. Such optimization problems involving complex, interdependent constraints have been extensively explored in prior work. In particular, *scenario theory* provides a robust framework for addressing them [6, 37], as detailed in Section 2.4.

2.1 Sampling Strategies

Sampling is the process of creating a *sample set* $\mathcal{S}$ as a subset of the valid configurations $\mathcal{V}$, $\mathcal{S} \subseteq \mathcal{V}$ for a given configurable software system. A sampling strategy τ selects configurations from $\mathcal{V}$ according to a given algorithm, such that $\tau(\mathcal{V}) = \mathcal{S}$. In the following, we review different classes of sampling strategies that can be used to find configurations with worst-case performance.

Uniform Random Sampling. Random sampling strategies attempt to *uniformly* sample the configuration space. That is, the probability of selecting a configuration $\mathbf{c}$ shall be the same for all valid configurations: $\Pr(\mathbf{c} \in \mathcal{V}) = 1/|\mathcal{V}|$. Ensuring uniform randomness is highly non-trivial in practice [21]. Enumerating all configurations and selecting from them according to a uniform distribution is usually intractable [45]. Just randomly selecting configuration options often yields invalid configurations, since constraints among configuration options often drastically restrict valid combinations [23]. As a consequence, different strategies have been proposed to circumvent these scaling and validating issues [2, 22, 28]. Most notably, Oh et al. [45] proposed to use counting binary decision diagrams to transfer the constraints of a configurable software system into a tree representation, which can be used to enumerate vast configuration spaces.

Solver-based Sampling. Several strategies leverage SAT solvers or BDDs for sampling to ensure the validity of configurations [9, 40, 46]. By encoding configuration options as Boolean variables and expressing constraints as propositional formulas, SAT solvers can produce valid assignments to these variables, though not necessarily uniformly distributed. To improve diversity, Henard et al. [20] proposed shuffling literals and variables in the formula for each SAT solver invocation. UniGen [61] and QuickSampler [9] both strive for a compromise on sampling efficiency and uniformity. However, empirical analyses have shown that solver-based sampling strategies often struggle to achieve true uniformity [54]. Newer approaches use #SAT solvers to count valid configurations, enabling uniform sampling over subspaces via a divide-and-conquer strategy [40].

Coverage-based Sampling. The idea of coverage-based sampling is to select samples according to a given *coverage criterion*, for example, such as that all pair-wise combinations of options must be selected in, at least, one configuration in the sample set. There are numerous coverage-based sampling strategies, including option-wise, negative option-wise, and t-wise sampling [4, 24, 34, 36, 39, 59]. In the same vein, other scientific disciplines have proposed various coverage-based sampling strategies, such as Plackett-Burman design [53], Latin hypercube sampling [55], response surface designs [32], and more. In practice, t-wise sampling, where t represents the degree of option combinations one wants to obtain in the sample set, is often used with $t = 2$ [27, 38, 47, 52, 59].

Distance-based Sampling. Distance-based sampling uses a distance metric across the configuration space such that configurations can be classified according to their distance from one another (or with respect to a reference configuration). A sampling strategy can then diversify the set of sampled configurations based on the distance between them. Two strategies, by Kaltenecker et al. [30] and Xiang et al. [65], use internally a solver to obtain valid configurations, but supply an additional distance value as a constraint to the solver to obtain more diverse results. Xiang et al. [66] randomize the probability of selecting an option and encode that into a distance metric that divides the configuration space into parts with similar numbers of selected configuration options. Kaltenecker et al. [30] specify the number of selected options as the distance to an origin point. Then, they uniformly sample configurations across all valid distances using an SMT solver. Spectral learning has been proposed to compute a sample set by recursively dividing the configuration space based on the distance of each configuration to a virtual principal component [42].

Sequential Learning and Sampling. There are hybrid sampling strategies that combine search-based exploration of the configuration space with a statistical or learning-based evaluation of the configurations they have seen so far. Most prominently, Oh et al. [44, 45] proposed an iterative strategy in which random samples are analyzed to determine which options statistically significantly improve or degrade performance with the goal of determining influential options. Influential options are bound (selected or deselected) such that the configuration space gradually shrinks, and the sampling procedure recurses on the shrunk spaces until no influential options are found. In the same vein, Nair et al. [43] build a performance model in multiple iterations. The idea is to compute an acquisition function over performance predictions of unseen configurations and to sample those configurations that are predicted to have the best performance.

2.2 Model Learning

Learning approaches heavily rely on a sample set, but not to use the sampled configurations for choosing which to run in practice, but to train a model that allows for predicting performance of all remaining configurations. Different learning approaches—from linear regression [13, 59] and classification and regression trees [14, 15, 56] to neural networks [12, 18, 58]—have been applied with varying prediction accuracies depending on the sample set used for learning [29, 50, 51]. Interestingly, Sarkar et al. [56] use a combination of sampling and learning. In each iteration, they sample one or more configurations, based on which they build a model and quantify the model error. The key is that, for an increasing number of iterations, the error reduces in a predictable way, which allows them to estimate the number of iterations required to reach a certain prediction accuracy. Recent strategies utilize the sparsity of the sample set by training multiple local models either via Fourier transformation [19] or ensemble methods [11]. An alternative approach proposed by Nair et al. is to estimate the rank of a configuration with respect to their performance, instead of an actual performance estimate [41].

Most related to our work are learning approaches that provide a confidence about their performance predictions. Dorn et al. [7] use Bayesian modeling to learn a probability distribution over the influences of each configuration option (and interaction), from which they sample to predict a configuration's performance distribution. That is, rather than having a scalar value that estimates the performance of a given configuration, they provide a probability density distribution of performance values. A similar approach uses Bayesian neural networks for learning uncertainty-aware models [17]. But, while these approaches have in common that they make estimates about individual configurations, they cannot be used on their own to make statements regarding performance bounds and corresponding confidence measures.

2.3 Searching

Search-based approaches aim at finding optimal configurations given certain constraints. Most approaches rely on meta-heuristics, such as evolutionary [48] and genetic algorithms [60], particle swarm optimization [1], prompting LLMs [62], or multi-objective optimization [16, 20, 49] such as IBEA [25] and NSGA-II [57]. Since many configurations have to be assessed regarding their performance in the process, these approaches rely on surrogate models (e.g., obtained using model-based techniques) [25]. By design, meta-heuristic search algorithms do not provide guarantees of global optimality [35].

2.4 Scenario Theory

The scenario theory (or scenario approach) [6] provides a principled method for reasoning about optimization problems that have large (and possibly infinite) sets of constraints. Let Ω denote the full set of constraint parameters. Each $\omega_i \in \Omega$ defines a constraint through a function $g : \mathcal{X} \times \Omega \to \mathbb{R}$, where the decision variable x must satisfy $g_i(x, \omega_i) \leq 0$. When Ω is large or infinite, solving the original problem directly is generally intractable. Scenario theory renders such problems tractable by selecting a finite subset of N constraint parameters $\{\omega_1, \omega_2, ..., \omega_N\} \subseteq \Omega$ sampled randomly from Ω. The resulting scenario-based problem is

$$\begin{aligned} x^* \;=\; & \arg\max_{x \in \mathcal{X}} \; f(x), \\ & \text{subject to} \;\; g_i(x, \omega_i) \leq 0, \quad \forall \omega_i \in \{\omega_1, \omega_2, ..., \omega_N\}. \end{aligned} \tag{1}$$

In this formulation, $x \in \mathcal{X} \subseteq \mathbb{R}^p$ denotes a p-dimensional decision variable, where $\mathcal{X}$ is the set of admissible values and $f : \mathcal{X} \to \mathbb{R}$ is the objective function to be optimized (in this case, maximized). The solution x^* denotes the optimizer of the scenario-based problem, that is, the decision variable maximizing f while satisfying all constraints in $\{\omega_1, \omega_2, ..., \omega_N\}$.

By using a finite number of randomly sampled constraints, scenario theory finds a solution x^* that is likely near-optimal for the majority of possible scenarios. Moreover, the theory quantifies the uncertainty introduced by ignoring the non-sampled constraints by estimating the *risk* ε that x^* violates them and the corresponding *confidence* γ in this estimate.

Here, the risk $\varepsilon \in (0, 1)$ denotes the probability that x^* violates constraints in non-sampled scenarios—with lower values indicating a smaller chance of non-compliance. The confidence level $\gamma \in (0, 1)$ quantifies the probabilistic assurance that the calculated value of ε reflects the true risk over all constraints. Clearly, the choice of the number of scenarios N, risk ε, and confidence γ are related to one another—the exact mathematical relationship between these variables depends on the specific form of the scenario problem. For problems with a convex and compact feasible set, under certain regularity conditions, this relationship is given by[1]

$$\gamma(\varepsilon, N) = 1 - (1 - \varepsilon)^N. \tag{2}$$

Scenario theory has found many applications in various fields, including finance and economics. The practical application of scenario theory is straightforward. As long as it is possible to randomly sample from the set of alternatives (in our case, software configurations), the notions of risk and confidence apply directly [37]. However, sampling strategies for configurable systems typically violate these assumptions, limiting the direct applicability of scenario theoretic guarantees.

2.5 Research Gap

Existing approaches to configuration–performance analysis address complementary aspects of the problem, yet they do not provide probabilistic guarantees for performance bounds induced by

[1]With limited reworking, the formula leads to the typical advice for practitioners to choose $N \geq \frac{\log(1-\gamma)}{\log(1-\varepsilon)}$ [6].

arbitrary sample sets [37, 44]. Research in this area has established structural properties of sample sets, such as coverage, diversity, or approximate uniformity [22, 30, 46], without relating them to guarantees on extreme performance values. Learning approaches model predictive uncertainty for individual configurations [7, 17] but do not certify worst-case or best-case bounds over the full configuration space. Search-based optimization identifies high-performing configurations but offers no guarantees of global optimality. Scenario theory, by contrast, provides rigorous *a priori* guarantees on risk and confidence, but only for solutions obtained from uniformly random samples [26, 37]. These guarantees do not extend to sampling strategies that intentionally bias selection, such as coverage-based, solver-based, or adaptive approaches. As a result, for such strategies, there is currently no probabilistic characterization of how reliable an observed worst-case or best-case performance bound is with respect to the full configuration space.

3 A Probabilistic Framework

We introduce the notions of *a posteriori risk* and *a posteriori confidence* to quantify the reliability of identifying the worst-case performing configuration of a configurable software system, henceforth referred to as *performance bound*. Note that, without loss of generality, we restrict the formal development to worst-case performance bounds. The best-case case follows analogously by reversing the optimization direction and is omitted for clarity. Using scenario theory, we can directly analyze random sampling, as its assumptions are satisfied (each sample is drawn uniformly from the set of valid configurations). However, scenario theory is not applicable for alternative sampling strategies that select configurations not uniformly at random. Therefore, we propose a probabilistic framework around a posteriori risk and confidence to evaluate the probabilistic guarantees that sampling strategies offer for estimating worst-case performance bounds—without requiring uniform randomness. This requires linking the *a priori* guarantees provided by scenario theory with the *a posteriori* information obtained from sampled configuration–performance pairs.

First, we formulate the problem of identifying the configuration that yields the worst-case performance (i.e., a performance bound) as an optimization problem. In other words, we seek the configuration $x_{\max} = \mathbf{c}_{\max}$ within the valid configuration set $\mathcal{V}$ that maximizes the performance function $\Pi(\mathbf{c})$, that is, $\mathcal{V} \ni \mathbf{c}_{\max} = \arg\max_{\mathbf{c}\in\mathcal{V}} \Pi(\mathbf{c})$. In this setting, the decision variable $\mathbf{c}$ represents the software configuration, and the objective function $\Pi(\mathbf{c})$ denotes the measured performance when the system runs under configuration $\mathbf{c}$. We denote by $\mathbf{c}_{\max}$ the true worst-case configuration over the valid configuration space $\mathcal{V}$, and by $\mathbf{c}^*$ the worst-case configuration identified by the optimization problem, which may have explored only a subset of $\mathcal{V}$. We can then write

$$\begin{aligned} \mathbf{c}^*(\mathcal{S}) \;=\; & \arg\max_{\mathbf{c}\in\mathcal{V}} \;\; \Pi\,(\mathbf{c}), \\ & \text{subject to } \;\; \Pi_k\,(\mathbf{c}) \le \omega_i, \quad \forall \omega_i \in \{\omega_1, \omega_2, \ldots, \omega_N\}\,. \end{aligned} \tag{3}$$

The constraints $\omega_i \in \{\omega_1, \omega_2, \ldots, \omega_N\}$ capture all measurements obtained for the sampled configurations (i.e., the software performance under configuration $\mathbf{c}$ given specific hardware and operating conditions). Our goal is to approximate an upper performance bound for the whole valid configuration space while acknowledging the risk ε that omitting certain configurations (due to sampling from the whole space) may be above the bound. This optimization problem is analogous to the scenario theory problem described in Section 2.4.

In a controlled setting—where the performance of all configurations is known, as in our experiments in Section 4—we can determine the a posteriori risk (i.e., the risk computed from the complete data set). We define the *posterior risk* $\hat{\varepsilon}$ as the empirical probability that a solution fails to meet the constraints *in non-sampled* configurations. Let $\mathcal{V}$ be the set of valid configurations with $|\mathcal{V}| = n$, and let $\mathcal{S}$ be a sample set of size N from which we identify $\mathbf{c}^*$ as the worst-case configuration. If we measured the performance of all n configurations of a software system, the actual (posterior)

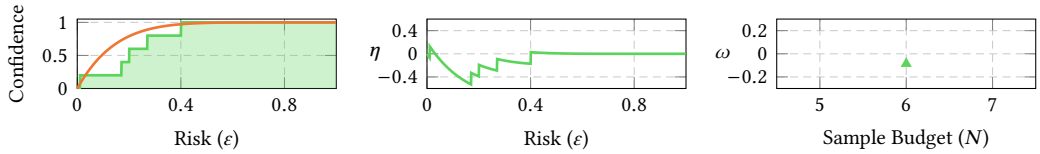


Fig. 1. Examples of the relation between prior and posterior risk and confidence levels, the confidence discrepancy (η), and the integrated confidence discrepancy (ω) for our running example with sample size $N = 6$.

risk $\hat{\varepsilon}$ —representing the probability that the solution fails to satisfy the constraints in unseen scenarios—is given by

$$\hat{\varepsilon}(\mathcal{S}) = \frac{1}{n}\sum_{i=1}^{n}\left[\Pi(\mathbf{c}_i) > \Pi(\mathbf{c}^*(\mathcal{S}))\right]. \tag{4}$$

Here, $[p]$ denotes the Iverson bracket, which equals 1 if the proposition p is true and 0 otherwise. Note that Equation (4) computes the posterior risk $\hat{\varepsilon}$ using empirical data from all n configurations. It relates the configurations $\mathbf{c}_i$ whose performance exceeds that of $\mathbf{c}^*$ to the total number of configurations.

We can also quantify the *posterior confidence level* $\hat{\gamma}$, which is the probability that the posterior risk $\hat{\varepsilon}$ describes the actual risk in finding a configuration that lies outside the performance bound. The posterior confidence level $\hat{\gamma}$ is defined as a function of both the risk ε and multiple sample sets. We estimate $\hat{\gamma}$ by computing the posterior risk $\hat{\varepsilon}$ for k different random seeds using the same sample size N, which yields different configuration sets and corresponding worst-case performance values. For a fixed risk value ε, we can use different random seeds to obtain a collection $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ and

Example 1 (Risk). Let us consider a configurable software system with $n = 100$ configurations, each with a unique performance value equal to its index in a sorted list of their performance, that is, $\forall i \in \{1, \ldots, 100\}, \Pi(\mathbf{c}_i) = i$. We would like to estimate the worst-case performance value sampling only $N = 6$ different configurations. Let us assume that, using uniform random sampling, we select the set of configurations $\mathcal{S}_1 = \{\mathbf{c}_2, \mathbf{c}_{15}, \mathbf{c}_{38}, \mathbf{c}_{40}, \mathbf{c}_{74}, \mathbf{c}_{99}\}$. The worst-case performance among the sampled configurations is $\Pi(\mathbf{c}_{99}) = 99$, and $\mathbf{c}^*(\mathcal{S}_1) = \mathbf{c}_{99}$. The posterior risk, which is the probability that by sampling another configuration from all the 100 configurations we would have found one that is actually worse than our worst case $\mathbf{c}_{99}$, is $\hat{\varepsilon}(\mathcal{S}_1) = {}^1\!/_{100}\sum_{i=1}^{100}[\Pi(\mathbf{c}_i) > \Pi(\mathbf{c}_{99})] = {}^1\!/_{100}\sum_{i=1}^{100}[i > 99] = 0.01$.

$$\hat{\gamma}(\varepsilon, \mathcal{S}_1, \ldots, \mathcal{S}_k) = \frac{1}{k}\sum_{i=1}^{k}\left[\varepsilon \geq \hat{\varepsilon}(\mathcal{S}_i)\right]. \tag{5}$$

Note, the sample sets used to compute $\hat{\gamma}$ all have size N, and ε itself depends on the sample size. This measure is inspired by the a priori confidence level γ from scenario theory.

Both, prior and posterior, confidence levels converge to 1 as the sample size approaches infinity. For each risk value ε, we can compare the prior confidence γ and the posterior confidence $\hat{\gamma}$, and define a measure termed *confidence discrepancy*.

Definition 1 (Confidence Discrepancy η). Given a sample set size N and k repetitions, the confidence discrepancy is given as $\eta = \hat{\gamma} - \gamma$, i.e., the difference between the priori and posterior confidence values. A zero confidence discrepancy indicates that the sample sets match the probabilistic guarantees of theoretical random sampling, while positive and negative discrepancies suggest superior and inferior guarantees, respectively.

The measure of confidence discrepancy quantifies how a sampling strategy's probabilistic guarantees for identifying the worst performing configuration differ from those of a true random strategy. To evaluate the probabilistic guarantees of different sampling strategies in estimating the performance bound of a configurable software system, we aggregate the posterior and prior confidence levels over all possible risk values (i.e., $0 \leq \varepsilon \leq 1$) into a single measure that reflects the probabilistic guarantees for a given sampling strategy for a given sample set size N and number of repetitions k. We aggregate these confidence levels by computing the *empirical cumulative distribution function* for the posterior risk $\hat{\varepsilon}$ obtained using k sample sets. We then calculate the area $\hat{\alpha}$ under the empirical cumulative distribution function of $\hat{\varepsilon}$, corresponding to the integrated posterior confidence defined in Equation (5). This area represents the integrated confidence level for the sample size N and k repetitions.

Example 2 (Confidence Level)**.** To illustrate the the measure of posterior confidence, we continue Example 1. We repeat the process of sampling $k = 5$ times, using uniform random sampling, determining five worst-case performance values and configurations, $\{\mathbf{c}_1^*, \ldots, \mathbf{c}_5^*\}$ and computing the respective posterior risk values $\hat{\varepsilon}$. Let the maximum values be $\mathbf{c}_1^*(\mathcal{S}_1) = \mathbf{c}_{99}$, $\mathbf{c}_2^*(\mathcal{S}_2) = \mathbf{c}_{73}$, $\mathbf{c}_3^*(\mathcal{S}_3) = \mathbf{c}_{80}$, $\mathbf{c}_4^*(\mathcal{S}_4) = \mathbf{c}_{83}$, and $\mathbf{c}_5^*(\mathcal{S}_5) = \mathbf{c}_{60}$. That is, the posterior risk for the respective sample sets is given by $\hat{\varepsilon}(\mathcal{S}_1) = 0.01$, $\hat{\varepsilon}(\mathcal{S}_2) = 0.27$, $\hat{\varepsilon}(\mathcal{S}_3) = 0.20$, $\hat{\varepsilon}(\mathcal{S}_4) = 0.17$, and $\hat{\varepsilon}(\mathcal{S}_5) = 0.40$. Using this information, we can compute the posterior confidence level of having a specific risk ε using Equation (5). For example, when $\varepsilon = 0.01$, then $\hat{\gamma}(0.01, \mathcal{S}_1, \ldots, \mathcal{S}_5) = {}^1\!/_5 \sum_{i=1}^{5} [0.01 \geq \hat{\varepsilon}(\mathcal{S}_i)] = {}^1\!/_5 = 0.2$.

$$\hat{\alpha}(\mathcal{S}_1, \ldots, \mathcal{S}_k) = \int_0^1 \hat{\gamma}(\varepsilon, \mathcal{S}_1, \ldots, \mathcal{S}_k)\ \mathrm{d}\varepsilon\,. \tag{6}$$

By comparing the posterior integrated confidence $\hat{\alpha}$ with the prior integrated confidence $\int_0^1 \gamma(\varepsilon, N)\ \mathrm{d}\varepsilon$, we are able to assess how a sampling strategy's probabilistic guarantees vary with sample size.

Example 3 (Integrated Confidence Level)**.** Continuing Example 2, we can compute the integrated confidence level for the sample set size $N = 6$. Using our sampled data, for every $\varepsilon \in [0.4, 1]$ the posterior confidence $\hat{\gamma}(\varepsilon, \mathcal{S}_1, \ldots, \mathcal{S}_5)$ is equal to 1. We can compute the integrated confidence level using Equation (6). Computing the integral $\int_0^1 \hat{\gamma}(\varepsilon, \mathcal{S}_1, \ldots, \mathcal{S}_5)\ \mathrm{d}\varepsilon$, we obtain $\hat{\alpha}(\mathcal{S}_1, \ldots, \mathcal{S}_5) = 0.79$. In comparison, the prior (integrated) confidence level for the sample set size $N = 6$ is $\int_0^1 \gamma\ \mathrm{d}\varepsilon = 0.86$.

Definition 2 (Integrated Confidence Discrepancy ω)**.** Given a sample set size N and k repetitions, we define the *integrated confidence discrepancy* as

$$\begin{aligned} \omega &= \hat{\alpha}(\mathcal{S}_1, \ldots, \mathcal{S}_k) - \int_0^1 \gamma(\varepsilon, N)\ \mathrm{d}\varepsilon \\ &= \int_0^1 \hat{\gamma}(\varepsilon, \mathcal{S}_1, \ldots, \mathcal{S}_k) - \gamma(\varepsilon, N)\ \mathrm{d}\varepsilon\,. \end{aligned} \tag{7}$$

A zero integrated confidence discrepancy indicates that the sample sets offer probabilistic guarantees comparable to those of a uniform random sample set; positive values denote better guarantees, while negative values indicate worse guarantees.

In summary, we introduce posterior risk and confidence levels to evaluate the probabilistic guarantees offered by different sampling strategies in estimating a configurable software system's performance bound—without restricting sampling to be uniformly random. Figure 1 illustrates the relationship between the prior and posterior confidence levels, the confidence discrepancy, and the integrated confidence discrepancy for the running example at a sample size of $N = 6$. The leftmost plot shows an example of the relationship between the prior confidence level (orange) and the posterior confidence level (green) for a sample budget of $N = 6$. The second plot displays the confidence discrepancy η, defined as the difference between the green and orange curves from the first plot, for the same sample budget. The third plot presents the integrated confidence discrepancy ω, with the green triangle marking its value for the sample budget used in the first two plots. The novel measures of confidence discrepancy and integrated confidence discrepancy

enable us to quantify the probabilistic guarantees of different sampling strategies in identifying the configuration that yields the worst-case performance for a given metric.

4 Evaluation

To demonstrate the merits of our prababilistic framework, we compare the probabilistic guarantees that five sampling strategies offer for estimating worst-case performance bounds, leveraging data from seven real-world configurable software systems. We are particularly interested in how these guarantees evolve as the sample set size increases ($|\mathcal{S}| = N$). To this end, we compare the a posteriori risk and confidence levels achieved by the sampling strategies against the a priori risk and confidence derived from scenario theory, serving as a theoretical baseline.

4.1 Research Question

Our primary objective is to demonstrate the utility of a posteriori knowledge—specifically, risk and confidence levels—and to discuss possible insights practitioners and researchers can gain from evaluating and comparing sampling strategies. Rather than identifying a single best strategy, we focus on showcasing possible applications of our measures. To this aim, we evaluate our approach guided by the following overarching research question:

Research Question: *To what extent are posterior risk and confidence able to provide actionable insights for practitioners and researchers in selecting and evaluating sampling strategies to estimate performance bounds?*

To answer this research question, we design two experiments examining how posterior risk and confidence inform the probabilistic guarantees of sampling strategies.

Experiment 1. Experiment 1 examines whether posterior risk and confidence reveal meaningful differences between sampling strategies in terms of their probabilistic guarantees for estimating worst-case performance bounds at a fixed sample set size. This inquiry arises from the observation that different sampling strategies prioritize distinct goals during configuration selection. For example, uniform random sampling aims to evenly spread samples across the configuration space, whereas distance-based sampling emphasizes maximizing configuration diversity based on a given distance metric. Although both strategies target representative sampling, their relative efficacy and stability in ensuring reliable performance guarantees remains unexplored. Thus, we investigate how their varied assumptions and objectives influence the empirical guarantees they provide, offering valuable insights into the design and evaluation of sampling methods.

Experiment 2. Experiment 2 investigates whether posterior risk and confidence help assess how the probabilistic guarantees of different sampling strategies change as the sample size increases. Note that, some strategies, such as uniform random sampling, support flexible sample set sizes, whereas others such as 2-wise sampling do not. A fundamental consideration is the trade-off between reducing the sample set size and preserving its representativeness of the entire configuration space. This raises the question of how sampling strategies maintain consistent guarantees across different sample set sizes. Identifying strategies that offer robust guarantees even with smaller sample sizes would be especially beneficial in practice. These insights can help practitioners select an appropriate sampling strategy and sample budget tailored to their specific needs.

Table 1. Subject systems used in our evaluation.

Subject System	#Configuration Options	$\lvert\mathcal{V}\rvert$	$N_{2\text{-wise}}$	Application Domain
7z	44	68 640	600	File archive utility
BerkeleyDB	19	2 560	97	Embedded database
Dune	32	2 304	265	Multigrid solver
Hipacc	55	13 485	843	Image processing
JavaGC	39	193 536	468	Garbage collector
LLVM	12	1 024	56	Compiler infrastructure
Polly	40	60 000	345	Code optimizer

4.2 Subject Systems

We evaluate our framework on seven configurable software systems (see Table 1). The systems were selected according to three methodological criteria: (1) substantial variability in configuration space characteristics, including the number of options, constraint structure, and total number of valid configurations; (2) established use in empirical studies on sampling and performance prediction [8, 30, 45]; and (3) availability of complete performance measurements for all valid configurations, which is essential for computing posterior risk with respect to the ground truth. For each system, $\mathcal{V}$ denotes the set of valid configurations as reported in the literature [30]. The selected systems span multiple application domains and exhibit heterogeneous performance landscapes, ranging from comparatively smooth behavior to highly interaction-driven and irregular performance effects. This structural diversity allows us to assess how posterior risk and confidence respond to structurally different configuration spaces and performance landscapes. All performance measurements are publicly available [30], ensuring full reproducibility and enabling controlled evaluation against complete ground truth. By relying on well-established benchmark systems with documented configuration spaces and performance behavior, we strengthen the internal validity of our analysis while maintaining comparability with existing studies.

4.3 Sampling Strategies

We evaluate five different sampling strategies: uniform random sampling, distance-based sampling, plain solver sampling, 2-wise sampling, and statistical recursive search[2]. You can find the implementation of uniform random sampling and statistical recursive search in our supplement, c.f., Section 7. For distance-based and plain solver sampling, we reused the implementation used in the literature [30]. These strategies represent the categories discussed in Section 2, reflecting their diverse objectives and theoretical foundations. Without claiming completeness, we provide results for additional sampling strategies for all subject systems in the supplementary material.

4.4 Results

We conducted two experiments to evaluate our framework's ability to provide insights into the probabilistic guarantees of different sampling strategies for estimating performance bounds. In the following, we present the results of these experiments for five representative sampling strategies across seven subject systems; the full set (including eight additional sampling strategies) is provided in the supplement.

Experiment 1. Experiment 1 examines the guarantees that sampling strategies provide for performance bound estimation at a fixed sample set size. We use a sample set size equal to the commonly

[2]We slightly modified the statistical recursive search algorithm so that it continues until the predetermined sample size is reached, supplementing with random configurations if necessary. This adjustment ensures consistency in sample set size across all strategies, providing a fair basis for comparison.

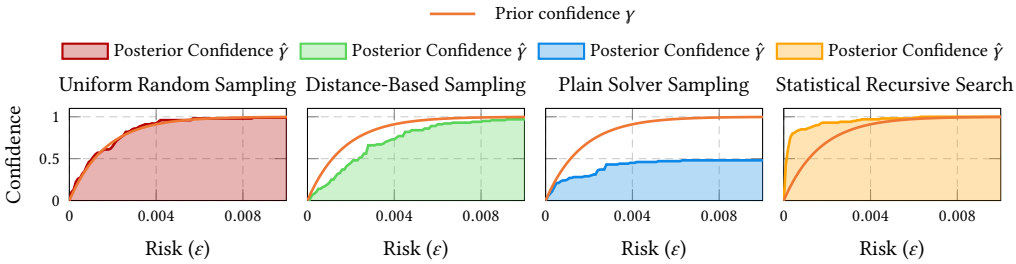


Fig. 2. Confidence level per sampling strategy for subject system 7z: prior (γ, calculated as a function of ε with $N = 600$) and posterior ($\hat{\gamma}$, calculated as a function of ε with $N = 600$ and $k = 100$). The posterior confidence level obtained with uniform random sampling is very similar to the theoretical level guaranteed by scenario theory. Statistical recursive search improves on the theoretical guarantees for lower values of ε.

used 2-wise sampling size [27, 46, 47, 52, 59] for each system, enabling a direct comparison of its risk with the other strategies. Using performance measurements for all $n = |\mathcal{V}|$ configurations of each system, we compute the posterior risk $\hat{\varepsilon}$ for each sampling strategy according to Equation (4), using $k = 100$ random seeds. This process yields an empirical probability distribution that represents the confidence level $\hat{\gamma}$ for performance bound predictions, as defined in Equation (5). We compare these posterior results to the a priori estimations derived from scenario theory, which correspond to uniform random sampling.

Figure 2 displays the prior confidence level (γ, orange line) and posterior confidence level ($\hat{\gamma}$, shaded areas) for the subject system 7z, up to a 1% risk. As expected, uniform random sampling (leftmost plot) yields posterior confidence closely aligned with the prior. In contrast, distance-based and plain solver sampling (second and third plots) exhibit reduced posterior confidence relative to the prior. Statistical recursive search (fourth plot), however, achieves higher posterior confidence at lower risk levels and converges to the prior as risk increases. These results indicate that uniform random sampling provides probabilistic guarantees that align well with the a priori expectations of scenario theory. The probabilistic guarantees of distance-based and plain solver sampling are consistently lower than the a priori knowledge, whereas statistical recursive search provides small risk values with higher confidence and very close to the a priori confidence for larger risk values. In order to judge the probabilistic guarantees of plain solver sampling, we show the confidence level for the whole risk range in Figure 3. In this case, the confidence level that plain solver sampling provides converges to the confidence level obtained using scenario theory for larger risk (at around 15%).

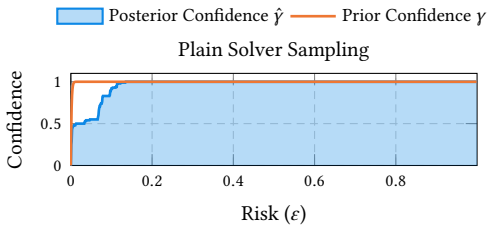


Fig. 3. Confidence level of plain solver sampling for the 7z subject system.

For a more compact visualization, we plot the *confidence discrepancy* η (i.e., the difference between posterior and prior confidence levels) for each sampling strategy and subject system in Figure 4 (with the first plot being a processed version of the data from Figure 2). This enables an efficient comparison of confidence discrepancies across sampling strategies. Notably, statistical recursive search generally outperforms the other strategies by consistently yielding posterior confidence that exceeds the prior. That is, statistical recursive search offers particularly strong probabilistic guarantees at very low accepted risk levels across all seven subject systems. Conversely, plain solver sampling is the least effective, with its posterior confidence falling below the prior level. An anomaly is observed for LLVM, where all strategies except uniform random sampling perform

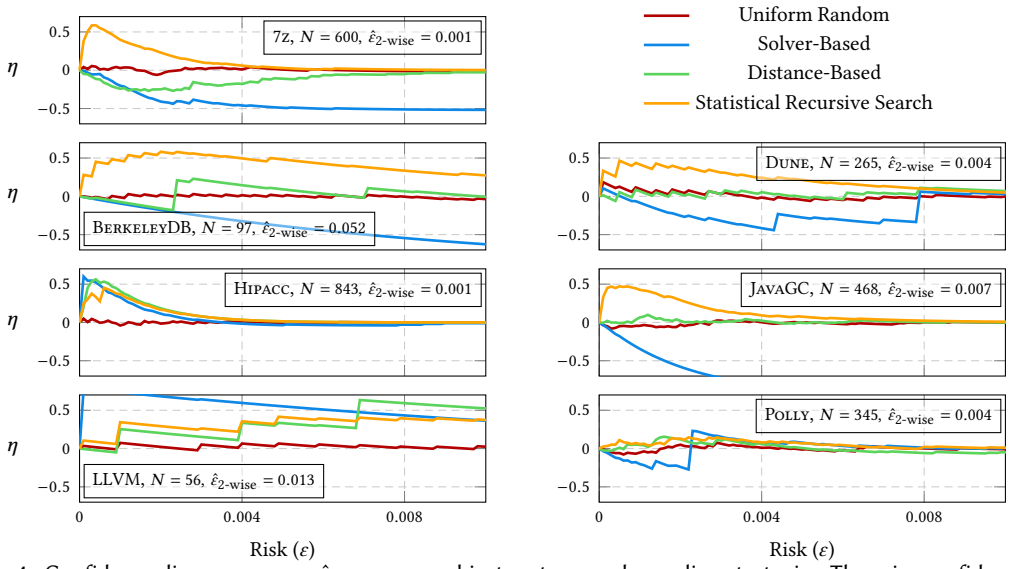


Fig. 4. Confidence discrepancy $\eta = \hat{\gamma} - \gamma$ across subject systems and sampling strategies. The prior confidence γ is calculated as a function of ε with N from 2-wise sampling (see Table 1). The posterior confidence $\hat{\gamma}$ is calculated as a function of ε with the same N and $k = 100$. The $\hat{\varepsilon}$ value for 2-wise sampling is added to each plot (no randomness).

well. This may be due to specific characteristics of LLVM, such as the number of configurations and a lack of constraints in its configuration space. Furthermore, for systems such as 7z, BerkeleyDB, and JavaGC, plain solver sampling fails to achieve small risk values with a confidence level exceeding 0.5. This implies that, for these systems, the confidence in performance bound estimates from plain solver sampling is effectively no better than chance. To further investigate the performance of plain solver sampling, we present its confidence discrepancy across all subject systems up to a risk of 65% (where the posterior risk converged to the prior risk in every system) in Figure 5. We see that plain solver sampling is inconsistent in providing probabilistic guarantees across the subject systems. For some systems, the confidence discrepancy is above zero for small risk values whereas for others it is below zero. That inconsistency means that we cannot generalize the probabilistic guarantees plain solver sampling provides for estimating performance bounds across subject systems.

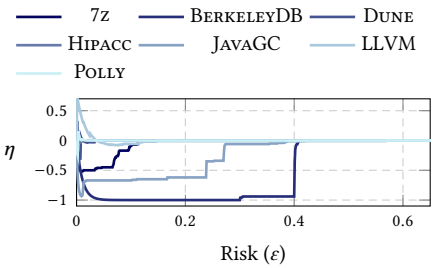


Fig. 5. Confidence discrepancy $\eta = \hat{\gamma} - \gamma$ across subject systems for plain solver sampling. The prior confidence γ is calculated as a function of ε with N from 2-wise sampling (see Table 1). The posterior confidence $\hat{\gamma}$ is calculated as a function of ε with the same N and $k = 100$.

Experiment 1: Our findings clearly indicate that the choice of sampling strategy significantly influences the posterior confidence in estimating performance bounds. Some strategies consistently provide probabilistic guarantees that meet or exceed the scenario theory baseline, while others exhibit greater sensitivity to the characteristics of the subject system.

Table 2. Proportional integrated confidence discrepancy improvement $\omega/\omega_{\max}$ for different sample set sizes across subject systems and sampling strategies. Positive values (highlighted in green) indicate that the sampling strategy provides better probabilistic guarantees than scenario theory for the given sample set size and subject system. Negative values (highlighted in red) indicate that the sampling strategy provides worse probabilistic guarantees than scenario theory for the given sample size and system.

Sampling Strategy	Subject System	Sample Set Size						
		10	50	100	150	200	250	300
Uniform Random	7z	-0.155	0.024	0.099	0.067	0.000	0.029	-0.064
	BerkeleyDB	-0.081	-0.160	-0.158	0.064	0.069	0.040	-0.033
	Dune	-0.060	-0.000	0.024	0.010	0.107	0.139	0.125
	Hipacc	-0.001	-0.168	-0.034	-0.091	-0.141	-0.218	-0.189
	JavaGC	-0.161	0.160	0.116	-0.027	0.018	0.012	0.037
	LLVM	0.090	0.153	0.226	0.327	0.375	0.407	0.404
	Polly	0.077	0.177	0.036	-0.107	-0.050	-0.077	-0.053
Distance-Based	7z	0.316	0.217	-0.228	-0.404	-0.513	-0.509	-0.594
	BerkeleyDB	-0.138	-0.935	-0.489	0.068	0.027	-0.136	-0.318
	Dune	0.886	0.610	0.434	0.331	0.266	0.191	0.092
	Hipacc	0.945	0.930	0.867	0.855	0.829	0.810	0.803
	JavaGC	-0.673	-2.460	-0.063	0.054	0.169	0.089	0.032
	LLVM	-0.560	0.688	0.859	0.844	0.800	0.758	0.717
	Polly	-0.794	-1.789	-0.636	-0.399	-0.441	-0.117	-0.015
Plain Solver	7z	-7.332	-3.031	-3.748	-5.414	-6.583	-7.469	-8.918
	BerkeleyDB	-4.116	-2.383	-1.494	-1.330	-1.548	-1.359	-1.353
	Dune	0.865	0.737	0.642	0.511	0.452	0.352	0.268
	Hipacc	0.918	0.735	0.576	0.542	0.497	0.413	0.350
	JavaGC	-3.291	-8.455	-6.517	-6.346	-6.011	-5.922	-6.081
	LLVM	-5.885	-0.436	0.206	0.404	0.577	0.642	0.624
	Polly	-4.121	-12.508	-24.199	-35.819	-46.640	-57.713	-68.484
Statistical Recursive Search	7z	-0.243	0.143	0.068	-0.218	0.028	0.323	0.395
	BerkeleyDB	-0.023	0.294	0.725	0.765	0.800	0.764	0.775
	Dune	-0.013	0.211	0.396	0.452	0.440	0.427	0.443
	Hipacc	-0.186	0.488	0.663	0.768	0.744	0.740	0.718
	JavaGC	0.064	0.265	0.444	0.332	0.310	0.445	0.406
	LLVM	-0.007	0.399	0.478	0.569	0.563	0.505	0.472
	Polly	0.008	0.063	0.213	0.194	0.243	0.250	0.197

Experiment 2. Experiment 2 explores how the probabilistic guarantees of sampling strategies change as the sample set size increases. Figure 2 illustrates the prior and posterior confidence levels for a sample set size of $N = 600$. Comparing the areas under the prior (orange) and posterior (colored) curves in Figure 2 reveals insights into the empirical guarantees of the various sampling strategies at this sample size (cf. Experiment 1).

To systematically analyze this behavior, we consider sample sizes ranging from $N = 10$ to $N = 300$ for each subject system and strategy (except for 2-wise sampling, whose size is fixed by construction). The integrated confidence discrepancy aggregates deviations over the entire risk range $\varepsilon \in [0, 1]$. As the sample size increases, differences between sampling strategies increasingly concentrate at smaller risk values. Since the integral weights the interval uniformly, deviations confined to narrow low-risk regions contribute only marginally to the overall area. Consequently, exploring sample sizes substantially larger than $N = 300$ would primarily refine behavior at very low risk values without materially altering the aggregated assessment. Thus, $N = 300$ suffices to characterize comparative probabilistic behavior under the proposed measure. For each sample set

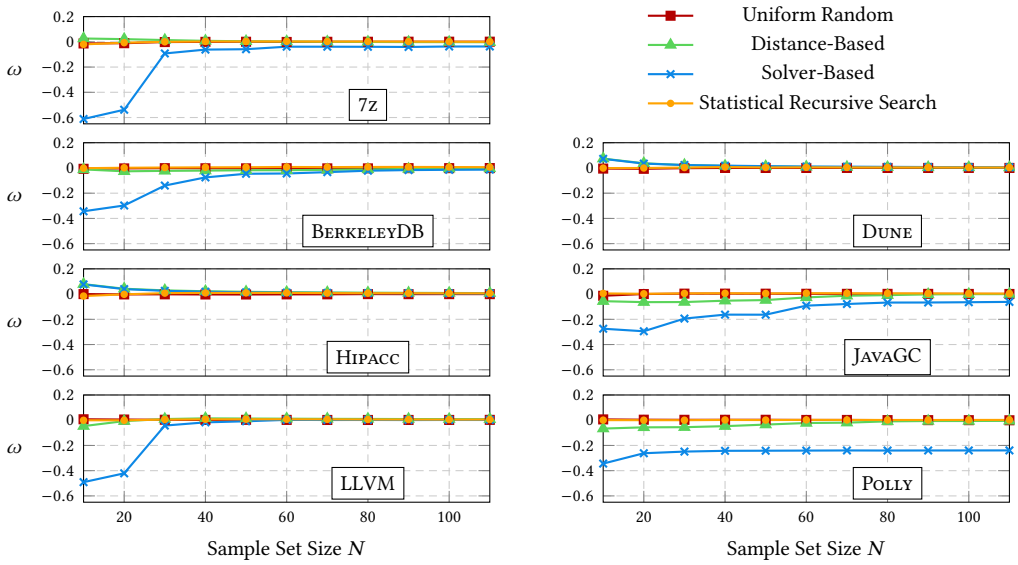


Fig. 6. Integrated confidence discrepancy ω over all possible risk values $\varepsilon \in [0, 1]$ per sampling strategy and subject system for increasing sample set sizes (the integral is approximated using 10 000 evenly spaced values of ε). The prior confidence γ is calculated with as a function of ε. The posterior confidence $\hat{\gamma}$ is calculated as a function of ε with $k = 100$.

size, we compute the ratio between the integrated confidence discrepancy ω and the maximum possible improvement in confidence $\omega_{\max}$, defined as

$$\omega_{\max}(N) = 1 - \int_0^1 \gamma(\varepsilon, N)\ \mathrm{d}\varepsilon\,. \tag{8}$$

Since the potential improvement in confidence is bounded by the prior confidence (which increases with sample size), we report the ratio of *integrated confidence discrepancy* ω to $\omega_{\max}$ in Table 2. Positive ratios (highlighted in green) indicate that the strategy outperforms the scenario theory baseline for the given sample size and subject system, reaching a maximum value of 1 if the true performance bound is always identified. Negative ratios (highlighted in red) suggest that the strategy underperforms relative to the scenario theory baseline. For a detailed analysis, Figure 6 plots the integrated confidence discrepancy ω over all risk values ($\varepsilon \in [0, 1]$) for sample sizes $N \in [10, 110]$, per strategy and subject system. Uniform random sampling consistently yields an integrated confidence discrepancy near zero across all systems and sample sizes (except for LLVM). This indicates that its probabilistic guarantees closely match the scenario theoretical predictions, as expected. Distance-based sampling exhibits a fluctuating integrated confidence discrepancy, rendering it difficult to generalize its probabilistic guarantees across different sample sizes. Even worse, plain solver sampling displays substantial variation in the integrated confidence discrepancy across sample sizes and systems. In contrast, statistical recursive search consistently achieves an integrated confidence discrepancy near zero or positive across all systems and sample sizes.

Experiment 2: Our findings indicate that the sampling strategy impacts probabilistic guarantees, particularly with small sample sizes. Plain solver sampling is least consistent across varying sample sizes and systems, whereas statistical recursive search consistently meets or surpasses the probabilistic guarantees provided by scenario theory.

5 Discussion

Our evaluation shows that our probabilistic framework around posterior risk and confidence can be used to effectively assess the probabilistic guarantees offered by different sampling strategies and enable a comparison with the theoretical baseline by scenario theory. In this section, we discuss potential threats to validity as well as implications of our results for research and practice.

5.1 Threats to Validity

Our experiments face *internal validity* threats, particularly regarding the chosen sample size and the number of repetitions for each sampling strategy. We addressed these issues through a sensitivity analysis of these factors. For most subject systems, we limited our sample size to $N = 300$, at which point most sampling strategies yielded consistent results. The exception is plain solver sampling, which required further exploration to understand its asymptotic behavior—a topic we discuss further in Section 5.2. We set a high repetition count ($k = 100$) to ensure statistical significance and robustness, noticing no additional benefit from increasing k further. The close agreement between uniform random sampling and the theoretical bounds of scenario theory for most systems and sample sizes suggests sufficient internal validity.

The *external validity* of our experiments may be limited by our choice of systems. We selected seven systems commonly referenced in previous work, which proved adequate for comparing sampling strategies and affirming the applicability of scenario theory in this context. Although our selection of sampling strategies represents a subset of all possible methods, it is nonetheless sufficient to reveal significant differences among them and thus to demonstrate that our probabilistic framework around posterior risk and confidence is able to effectively assess and compare their probabilistic guarantees. While for conciseness we concentrated on five representative sampling strategies in the paper, we provide results for additional strategies in the supplement to provide further evidence for external validity, without claiming completeness, though; these additional results are consistent with the trends reported here.

5.2 Implications for Research

Over time, researchers have developed various sampling strategies for different purposes, such as performance prediction, optimization, testing, and modeling (see Section 2). These strategies vary significantly and are often tailored to specific use cases. For example, plain solver sampling is typically applied in highly constrained configuration spaces where uniform sampling is impractical, whereas uniform sampling is common in performance testing when little is known about the configuration space. By our two experiments, we have shown two applications of our framework: (1) comparing sampling strategies for the risk and confidence of performance bounds at certain sample sizes and (2) analyzing the convergence of posterior risk and confidence for increasing sample sizes. Our experiments confirm that the choice of sampling strategy significantly influences the reliability and risk of performance estimates.

Let us illustrate how powerful our probabilistic framework is for researchers by the example of Polly and the comparison of the risk and confidence levels of plain solver sampling and statistical recursive search. Here, we increased the sample size for both plain solver sampling and statistical recursive search to $N = 2100$ to analyze if their confidence and risk levels would approach the a priori expectations. The analysis in Figure 7 shows that, even at $N = 2100$, plain solver sampling's confidence and risk remain significantly below the scenario theoretical baseline (with ω staying negative). A detailed examination across sample sizes—from as small as $N = 15$ up to $N = 2000$—reveals that plain solver sampling yields notably lower confidence levels for low risk values compared to random sampling or statistical recursive search. This comparison allows us to

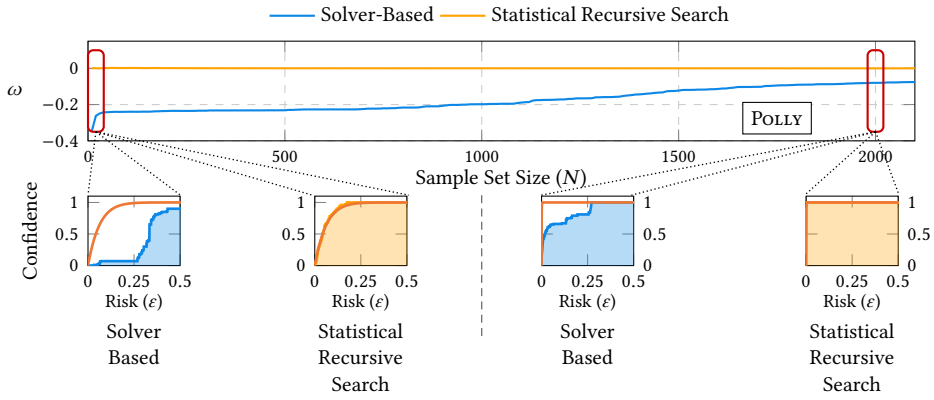


Fig. 7. Integrated confidence discrepancy ω of plain solver sampling and statistical recursive search for Polly. With two lenses (dark red rounded boxes), we show the confidence and risk for small sample sizes and their convergence for large sample sizes.

make clear statements and showcase the ability of our framework to provide insights for research: First, plain solver sampling *cannot reliably* provide probabilistic guarantees on performance bounds of this system even at very large sampling sizes, and possibly for other systems with similar characteristics. Second, statistical recursive search consistently achieves confidence and risk levels that match or exceed the a priori predictions even with very small sample sizes. In general, our probabilistic framework offers a rigorous method for developing and testing new sampling strategies for performance bound assessment. It can serve as a guiding measure and optimization criterion for researchers to develop new sampling strategies that provide reliable performance bounds with high confidence.

5.3 Implications for Practice

Practitioners often face the challenge of selecting a sampling strategy based on criteria such as the reliability and accuracy of performance estimates, the risk and confidence associated with estimated performance bounds, and the need to cover interacting configuration options for functional testing. Specialized strategies such as statistical recursive search may outperform random sampling by providing stronger probabilistic guarantees; however, they might be less effective when the goal is to learn a performance model that reliably estimates the performance of any configuration. When non-functional and functional testing are intertwined within a continuous integration process, diversified sampling approaches—such as distance-based sampling—may prove more suitable. Thus, practitioners require a framework that enables them to carefully analyze the characteristics and assumptions of each strategy to select the most appropriate one, especially when managing low risk values that significantly influence confidence levels in practice. If practitioners are interested in quantifying the risk and confidence associated with a performance bound, they can use our framework to calculate the required sample budget for a desired risk and confidence level. For instance, when using random sampling, one can apply Equation (2) (with minimal adjustments) to calculate

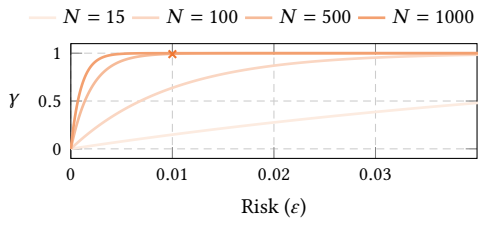


Fig. 8. Prior confidence level and risk for different sample budgets. A predefined risk of 0.01 and confidence level of 0.99 is marked with an orange cross.

the number of samples needed to achieve a target risk and confidence—independently of the system under test. Concretely, suppose we aim for a risk of 0.01 and a confidence level of $\gamma = 0.99$ that system performance will not exceed a certain threshold. Using Equation (2), we can calculate that, at least, $N \geq 458.21$ samples are required. Figure 8 shows the prior confidence and risk for various sample budgets; the orange cross marks our target, providing a visual guide for determining the necessary sample budget.

Moreover, our framework is able to guide practitioners to combine multiple strategies while maintaining probabilistic guarantees. For example, if the available sample budget exceeds the requirement for random sampling, one may use any strategy that meets or exceeds this baseline to achieve the guarantees and then apply a second strategy (e.g., coverage-based to systematically probe for interactions). In summary, these insights support practitioners in an informed selection of sampling strategies for performance estimation and testing of configurable systems.

5.4 Applicability to Stochastic and Large-Scale Systems

Our experiments thus far assume deterministic performance values $\Pi(\mathbf{c})$ and a fully known configuration space $\mathcal{V}$, which allowed us to compute posterior risk with respect to the complete configuration space. In practice, however, it is often difficult to obtain equally clean and repeatable measurements. Variability in the execution environment can introduce stochastic effects in performance measurements, for instance through caching behavior, scheduling decisions, or hardware-level noise. Even when measurements are stable, large-scale systems rarely allow for exhaustive evaluation of the complete configuration space. In the following, we outline how our framework can be adapted to account for stochastic performance measurements and incomplete configuration spaces, and briefly comment on applicability to multi-objective performance metrics.

Stochastic Performance Measurements. To account for measurement variability, we model the performance of a fixed configuration $\mathbf{c}$ as a random variable $\Pi(\mathbf{c}, \xi)$, where ξ is a parameter vector that includes all the factors that influence the run-to-run variability (note that we do not need to enumerate or even know ξ). Given repeated measurements $\{ \Pi(\mathbf{c}, \xi_1), \ldots, \Pi(\mathbf{c}, \xi_M) \}$ (illustrated in Figure 9), scenario theory can be applied at the level of an individual configuration to derive a probabilistic worst-case bound $\Pi(\mathbf{c}, \varepsilon, \gamma, \langle \xi_i \rangle_{i \in \{1..M\}})$, with risk ε and confidence γ. For brevity, we denote this scenario-theoretic bound by $\Pi_M(\mathbf{c})$. Posterior risk and confidence are then computed using $\Pi_M(\mathbf{c})$ instead of deterministic performance values. Formally, posterior risk becomes

$$\hat{\varepsilon}\,(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \left[\Pi_M\,(\mathbf{c}_i) > \Pi_M\,(\mathbf{c}^*(\mathcal{S})) \right],$$

with posterior confidence defined analogously as in Equation (5). Importantly, the definition of posterior risk remains unchanged[3]; only the performance function is replaced by its scenario-theoretic bound. Conceptually, scenario theory is applied per configuration to obtain probabilistic performance bounds under stochastic measurements (see Figure 9), and posterior analysis is subsequently performed over these bounds.

To illustrate the effect of stochasticity, we injected 1% Gaussian noise into the measurement data of our subject systems and recomputed the confidence discrepancy for all sampling strategies (see Figure 10; full results are provided in the supplementary material). As expected, we find that moderate noise does not substantially alter the relative behavior of the sampling strategies: the

[3]More precisely, although the formula remains identical, the performance values entering the comparison are now scenario-theoretic bounds rather than point estimates, so posterior risk now inherits the probabilistic guarantees of scenario theory—specifically, the bound $\Pi_M(\mathbf{c})$ is violated with probability at most ε (with confidence γ).

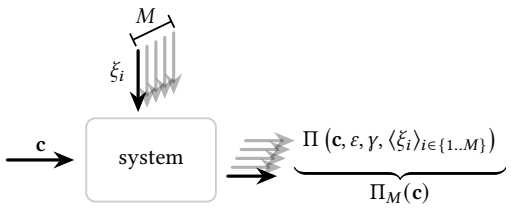


Fig. 9. Illustration of the application of scenario theory to derive probabilistic performance bounds for stochastic measurements.

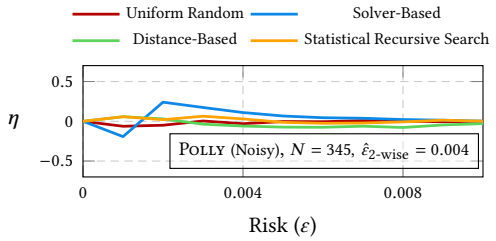


Fig. 10. Confidence difference η for Polly with noisy measurements across different sampling strategies (see Figure 4, for the corresponding confidence discrepancy for Polly without measurement noise).

confidence discrepancy remains largely consistent to the deterministic case across strategies as long as measurement variability does not dominate performance differences. This indicates that our framework extends naturally to stochastic settings such as in Just-In-Time or deep learning compilers, where performance distributions arise from hardware or runtime variability.

Incomplete Configuration Spaces. When the full configuration space $\mathcal{V}$ is unknown or infeasible to evaluate, posterior analysis can be performed with respect to a fixed baseline $\mathcal{V}' \subseteq \mathcal{V}$ with $|\mathcal{V}'| = n'$, representing the subset of configurations for which measurements are available. Posterior risk is then defined as

$$\hat{\varepsilon}\left(\mathcal{S}\right) = \frac{1}{n'} \sum_{i=1}^{n'} \left[\Pi\left(\mathbf{c}_i\right) > \Pi\left(\mathbf{c}^*(\mathcal{S})\right) \right],$$

with posterior confidence computed analogously. In this case, guarantees are interpreted relative to $\mathcal{V}'$ rather than the unknown full space $\mathcal{V}$.

We evaluated this approximation by restricting the baseline to 50%, 20%, 10%, and 5% of the original configuration space (see Figure 11; full results in the supplement). For uniform random sampling, distance-based sampling, and statistical recursive search, we find that the confidence discrepancy remains largely consistent to the results from the true ground truth across baseline sizes. In contrast, solver-based sampling exhibits sensitivity to the chosen baseline. This behavior can be attributed to the altered constraint structure induced by restricting $\mathcal{V}'$, which changes the solver's internal representation and consequently the generated sample sets. For example, for $|\mathcal{V}'| = 30\,000$, the confidence discrepancy remains negative until convergence, whereas for $|\mathcal{V}'| = 12\,000$ it changes sign, indicating that the posterior confidence temporarily exceeds the *a priori* guarantee for ε values between 0.2% and 1%. Overall, these results indicate that posterior risk and confidence remain informative under restricted baselines, while also revealing strategy-specific sensitivities that would remain hidden without an explicit probabilistic analysis.

Multi-objective Performance Metrics. The framework also extends to settings with multiple performance objectives. A straightforward case arises when objectives can be aggregated into a single scalar performance function, for instance via a weighted sum or utility function. In this case, the analysis applies unchanged, as posterior risk and confidence are defined with respect to the resulting scalar performance value. A more general and practically relevant setting optimizes one objective subject to constraints on others, for example, minimizing response time under a limit on memory consumption. Formally, if $\Pi_t(\mathbf{c})$ denotes the objective to be optimized (e.g., response time) and $\Pi_{\mathrm{M}}(\mathbf{c}) \leq \tau_{\mathrm{M}}$, denotes an additional constraint (e.g., limit on memory consumption (τ_{M})), the admissible configuration space becomes $\mathcal{V}_t = \{\, \mathbf{c} \in \mathcal{V} \mid \Pi_{\mathrm{M}}(\mathbf{c}) \leq \tau_{\mathrm{M}} \,\}$. Posterior risk and

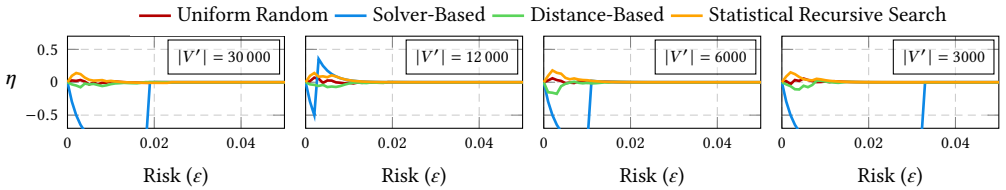


Fig. 11. Confidence difference η for Polly under four restricted measurement sets (50%, 20%, 10%, 5%).

confidence are then computed over $\mathcal{V}_t$ with respect to $\Pi_t(\mathbf{c})$. In both settings, the probabilistic framework remains unchanged; only the performance function or the constraint set is adapted.

Answer to Research Question: Our findings demonstrate that the measure of posterior risk and confidence are able to provide actionable insights for evaluating and selecting sampling strategies. They enable rigorous comparison of sampling strategies beyond observed performance values, supporting informed decision-making in performance estimation, even for stochastic measurements, incomplete configuration spaces, and multi-objective settings.

6 Conclusion

This paper explores sampling used in performance analysis of configurable systems, introducing the measures of posterior risk and posterior confidence to quantify the guarantees of sampling strategies—particularly for identifying the worst-case (or best-case) configurations. We compare five advanced sampling strategies across seven real-world configurable systems, revealing significant differences in their effectiveness. Our evaluation demonstrates that our probabilistic framework based on the measures of posterior risk and confidence is able to provide novel insights into existing sampling techniques. For example, we found that statistical recursive search generally outperforms other strategies—including uniform random sampling, distance-based sampling, and t-wise sampling. Conversely, plain solver sampling proves less reliable in providing probabilistic guarantees with sufficient confidence, especially for small samples. Additional experiments on stochastic measurements and restricted configuration spaces confirm that posterior risk and confidence can be meaningfully approximated under measurement noise and incomplete ground truth.

Overall, our findings underscore the merits of our probabilistic framework in evaluating the guarantees on worst-case (or best-case) performance approximations of sampling strategies, enabling both researchers and practitioners to assess the probabilistic guarantees of sampling strategies. We further discuss the implications of our results, emphasizing that a careful selection of sampling strategies is crucial in performance prediction and optimization of configurable systems. Our research provides a solid foundation for future work aimed at developing and refining sampling strategies that deliver improved guarantees across diverse application domains.

7 Data Availability

The code and data (including results for additional sampling strategies) to reproduce our experiments are publicly available [63].

Acknowledgements

The authors are supported by the DFG through the Collaborative Research Center TRR 248, project ID 389792660 (https://perspicuous-computing.science) and the Grant SI 2171/3-2 (699378) as well as by the KAW through the project Wallenberg AI, Autonomous Systems and Software Program (WASP) (https://wasp-sweden.org/).

References

[1] Uzma Afzal, Tariq Mahmood, Ayaz H. Khan, Sadeeq Jan, Raihan Ur Rasool, Ali Mustafa Qamar, and Rehan Ullah Khan. 2020. Feature Selection Optimization in Software Product Lines. *IEEE Access* 8 (2020), 160231–160250. doi:10.1109/ACCESS.2020.3020795

[2] Joshua Ammermann, Tim Bittner, Domenik Eichhorn, Ina Schaefer, and Christoph Seidl. 2023. Can Quantum Computing Improve Uniform Random Sampling of Large Configuration Spaces?. In *Proceedings of the 4th International Workshop on Quantum Software Engineering (Q-SE '23)*. 34–41. doi:10.1109/Q-SE59154.2023.00012

[3] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2016. *Feature-Oriented Software Product Lines*. Springer.

[4] Eduard Baranov and Axel Legay. 2025. Baital: Sampling Configurable Systems with High T-Wise Coverage. *Science of Computer Programming* 240 (2025), 103209. doi:10.1016/j.scico.2024.103209

[5] João Marcello Bessal, Millena Cavalcanti, Mathieu Acher, Markus Endler, and Juliana Alves Pereira. 2025. Unveiling the Impact of Sampling on Feature Selection for Performance Prediction in Configurable Systems. In *Proceedings of the 22nd International Conference on Software and Systems Reuse (ICSR '25)*. 55–66. doi:10.1109/ICSR66718.2025.00012

[6] Giuseppe C. Calafiore and Marco C. Campi. 2006. The Scenario Approach to Robust Control Design. *IEEE Trans. Automat. Control* 51, 5 (2006), 742–753. doi:10.1109/TAC.2006.875041

[7] Johannes Dorn, Sven Apel, and Norbert Siegmund. 2023. Mastering Uncertainty in Performance Estimations of Configurable Software Systems. *Empirical Software Engineering* 28, 33 (2023). doi:10.1007/s10664-022-10250-2

[8] Clemens Dubslaff, Kai Ding, Andrey Morozov, Christel Baier, and Klaus Janschek. 2019. Breaking the Limits of Redundancy Systems Analysis. In *Proceedings of the 29th European Safety and Reliability Conference (ESREL '19)*. 2317–2325.

[9] Rafael Dutra, Kevin Laeufer, Jonathan Bachrach, and Koushik Sen. 2018. Efficient Sampling of SAT Solutions for Testing. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, 549–559.

[10] Omid Gheibi, Christian Kästner, and Pooyan Jamshidi. 2025. Hardness, Structural Knowledge, and Opportunity: An Analytical Framework for Modular Performance Modeling. arXiv:2509.11000.

[11] Jingzhi Gong and Tao Chen. 2023. Predicting Software Performance with Divide-and-Learn. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23)*. ACM, 858–870.

[12] Jingzhi Gong, Tao Chen, and Rami Bahsoon. 2025. Dividable Configuration Performance Learning. *IEEE Transactions on Software Engineering* 51, 1 (2025), 106–134. doi:10.1109/TSE.2024.3491945

[13] Alexander Grebhahn, Carmen Rodrigo, Norbert Siegmund, Francisco J. Gaspar, and Sven Apel. 2017. Performance-Influence Models of Multigrid Methods: A Case Study on Triangular Meshes. *Concurrency and Computation: Practice and Experience* 29, 17 (2017), 4057.

[14] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wąsowski. 2013. Variability-Aware Performance Prediction: A Statistical Learning Approach. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE '13)*. IEEE, 301–311.

[15] Jianmei Guo, Dingyu Yang, Norbert Siegmund, Sven Apel, Atrisha Sarkar, Pavel Valov, Krzysztof Czarnecki, Andrzej Wąsowski, and Huiqun Yu. 2018. Data-Efficient Performance Learning for Configurable Systems. *Empirical Software Engineering* 23 (2018), 1826–1867. doi:10.1007/s10664-017-9573-6

[16] Jianmei Guo, Edward Zulkoski, Rafael Olaechea, Derek Rayside, Krzysztof Czarnecki, Sven Apel, and Joanne M. Atlee. 2014. Scaling Exact Multi-Objective Combinatorial Optimization by Parallelization. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE '14)*. ACM, 409–420. doi:10.1145/2642937.2642971

[17] Huong Ha, Zongwen Fan, and Hongyu Zhang. 2022. Uncertainty-Aware Performance Prediction for Highly Configurable Software Systems via Bayesian Neural Networks. arXiv:2212.13359.

[18] Huong Ha and Hongyu Zhang. 2019. DeepPerf: Performance Prediction for Configurable Software with Deep Sparse Neural Network. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE '19)*. IEEE, 1095–1106. doi:10.1109/ICSE.2019.00113

[19] Huong Ha and Hongyu Zhang. 2019. Performance-Influence Model for Highly Configurable Software with Fourier Learning and Lasso Regression. In *Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME '19)*. IEEE, 470–480.

[20] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 2015. Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In *Proceedings of the 37th IEEE International Conference on Software Engineering (ICSE '15)*. 517–528. doi:10.1109/ICSE.2015.69

[21] Ruben Heradio, David Fernandez-Amoros, José A. Galindo, and David Benavides. 2020. Uniform and Scalable SAT-Sampling for Configurable Systems. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line (SPLC '20)*. ACM. doi:10.1145/3382025.3414951

[22] Ruben Heradio, David Fernandez-Amoros, José A. Galindo, David Benavides, and Don Batory. 2022. Uniform and Scalable Sampling of Highly Configurable Systems. *Empirical Software Engineering* 27, 2 (2022), 44.

[23] Ruben Heradio, David Fernandez-Amoros, Christoph Mayr-Dorn, and Alexander Egyed. 2019. Supporting the Statistical Analysis of Variability Models. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE '19)*. 843–853. doi:10.1109/ICSE.2019.00091

[24] Tobias Heß, Tim Jannik Schmidt, Lukas Ostheimer, Sebastian Krieter, and Thomas Thüm. 2024. UnWise: High T-Wise Coverage from Uniform Sampling. In *Proceedings of the 18th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '24)*. ACM, 37–45. doi:10.1145/3634713.3634716

[25] Jose-Miguel Horcas, Daniel Strüber, Alexandru Burdusel, Jabier Martinez, and Steffen Zschaler. 2023. We're Not Gonna Break It! Consistency-Preserving Operators for Efficient Product Line Configuration. *IEEE Transactions on Software Engineering* 49 (2023), 1102–1117. doi:10.1109/TSE.2022.3171404

[26] Jingyi Huang, Paul Goulart, and Kostas Margellos. 2025. Data-Driven Performance Guarantees for Parametric Optimization Problems. In *Proceedings of the 64th Conference on Decision and Control (CDC '25)*. IEEE, 1967–1973. doi:10.1109/CDC57313.2025.11312114

[27] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. 2012. An Algorithm for Generating T-Wise Covering Arrays from Large Feature Models. In *Proceedings of the 16th International Software Product Line Conference (SPLC '12)*. ACM, 46–55. doi:10.1145/2362536.2362547

[28] Nikolai Käfer, Sven Apel, Christel Baier, Clemens Dubslaff, and Holger Hermanns. 2025. When to Sample from Feature Diagrams?. In *Proceedings of the 19th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '25)*. ACM, 11–20. doi:10.1145/3715340.3715442

[29] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 2020. The Interplay of Sampling and Machine Learning for Software Performance Prediction. *IEEE Software* 37, 4 (2020), 58–66. doi:10.1109/MS.2020.2987024

[30] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. 2019. Distance-Based Sampling of Software Configuration Spaces. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE '19)*. IEEE, 1084–1094.

[31] Christian Kaltenecker, Stefan Mühlbauer, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 2023. Performance Evolution of Configurable Software Systems: An Empirical Study. *Empirical Software Engineering* 28, 152 (2023). doi:10.1007/s10664-023-10338-3

[32] A.I. Khuri and J.A. Cornell. 1996. *Response Surfaces: Designs and Analyses* (2nd ed.). CRC Press. doi:10.1201/9780203740774

[33] Samuel Kounev, Klaus-Dieter Lange, and Jóakim von Kistowski. 2020. *Systems Benchmarking—For Scientists and Engineers*. Springer. doi:10.1007/978-3-030-41705-5

[34] Dominik M. Krupke, Ahmad Moradi, Michael Perk, Phillip Keldenich, Gabriel Gehrke, Sebastian Krieter, Thomas Thüm, and Sándor P. Fekete. 2025. How Low Can We Go? Minimizing Interaction Samples for Configurable Systems. *ACM Transactions on Software Engineering and Methodology* 34, 6 (2025). doi:10.1145/3712193

[35] Sean Luke. 2013. *Essentials of Metaheuristics* (second ed.). Lulu.

[36] Chuan Luo, Jianping Song, Qiyuan Zhao, Binqi Sun, Junjie Chen, Hongyu Zhang, Jinkun Lin, and Chunming Hu. 2025. Solving the T-Wise Coverage Maximum Problem via Effective and Efficient Local Search-Based Sampling. *ACM Transactions on Software Engineering and Methodology* 34, 1 (2025). doi:10.1145/3688836

[37] Claudio Mandrioli and Martina Maggio. 2020. Testing Self-Adaptive Software with Probabilistic Guarantees on Performance Metrics. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20)*. ACM, 1002–1014. doi:10.1145/3368089.3409685

[38] Dusica Marijan, Arnaud Gotlieb, Sagar Sen, and Aymeric Hervieu. 2013. Practical Pairwise Testing for Software Product Lines. In *Proceedings of the 17th International Software Product Line Conference (SPLC '13)*. ACM, 227–235. doi:10.1145/2491627.2491646

[39] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. 2016. A Comparison of 10 Sampling Algorithms for Configurable Systems. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, 643–654. doi:10.1145/2884781.2884793

[40] Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes, and Don Batory. 2019. Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features. In *Proceedings of the 23rd International Systems and Software Product Line Conference (SPLC '19)*. ACM, 289–301.

[41] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2017. Using Bad Learners to Find Good Configurations. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE '17)*. 257–267.

[42] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2018. Faster Discovery of Faster System Configurations with Spectral Learning. *Automated Software Engineering* 25, 2 (2018), 247–277. doi:10.1007/s10515-017-0225-2

[43] Vivek Nair, Zhe Yu, Tim Menzies, Norbert Siegmund, and Sven Apel. 2020. Finding Faster Configurations Using FLASH. *IEEE Transactions on Software Engineering* 46, 7 (2020), 794–811. doi:10.1109/TSE.2018.2870895

[44] Jeho Oh, Don Batory, and Rubén Heradio. 2023. Finding Near-Optimal Configurations in Colossal Spaces with Statistical Guarantees. *ACM Transactions on Software Engineering and Methodology* 33, 1 (2023).

[45] Jeho Oh, Don Batory, Margaret Myers, and Norbert Siegmund. 2017. Finding Near-Optimal Configurations in Product Lines by Random Sampling. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE '17)*. ACM, 61–71. doi:10.1145/3106237.3106273

[46] Jeho Oh, Paul Gazzillo, and Don Batory. 2019. t-wise Coverage by Uniform Sampling. In *Proceedings of the 23rd International Systems and Software Product Line Conference (SPLC '19)*. ACM, 84–87. doi:10.1145/3336294.3342359

[47] Sebastian Oster, Florian Markert, and Philipp Ritter. 2010. Automated Incremental Pairwise Testing of Software Product Lines. In *Software Product Lines: Going Beyond (SPLC '10)*. Springer, 196–210.

[48] Gustavo G. Pascual, Roberto E. Lopez-Herrejon, Mónica Pinto, Lidia Fuentes, and Alexander Egyed. 2015. Applying Multiobjective Evolutionary Algorithms to Dynamic Software Product Lines for Reconfiguring Mobile Applications. *Journal of Systems and Software* 103 (2015), 392–411. doi:10.1016/j.jss.2014.12.041

[49] Kewen Peng, Christian Kaltenecker, Norbert Siegmund, Sven Apel, and Tim Menzies. 2023. VEER: Enhancing the Interpretability of Model-Based Optimizations. *Empirical Software Engineering* 28, 3 (2023). doi:10.1007/s10664-023-10296-w

[50] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. 2020. Sampling Effect on Performance Prediction of Configurable Systems: A Case Study. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20)*. ACM, 277–288. doi:10.1145/3358960.3379137

[51] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2021. Learning Software Configuration Spaces: A Systematic Literature Review. *Journal of Systems and Software* 182 (2021), 111044. doi:10.1016/j.jss.2021.111044

[52] Gilles Perrouin, Sebastian Oster, Sagar Sen, Jacques Klein, Benoit Baudry, and Yves Traon. 2012. Pairwise Testing for Software Product Lines: Comparison of Two Approaches. *Software Quality Journal* 20, 3–4 (2012), 605–643. doi:10.1007/s11219-011-9160-9

[53] Robin L Plackett and J Peter Burman. 1946. The Design of Optimum Multifactorial Experiments. *Biometrika* 33, 4 (1946), 305–325. doi:10.2307/2332195

[54] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. 2019. Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?. In *Proceedings of the 12th IEEE Conference on Software Testing, Validation and Verification (ICST '19)*. 240–251. doi:10.1109/ICST.2019.00032

[55] Andrea Saltelli. 2008. *Global Sensitivity Analysis: The Primer*. John Wiley.

[56] Atrisha Sarkar, Jianmei Guo, Norbert Siegmund, Sven Apel, and Krzysztof Czarnecki. 2015. Cost-Efficient Sampling for Performance Prediction of Configurable Systems. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE '15)*. IEEE, 342–352.

[57] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. 2013. Scalable Product Line Configuration: A Straw to Break the Camel's Back. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE '13)*. 465–474. doi:10.1109/ASE.2013.6693104

[58] Yangyang Shu, Yulei Sui, Hongyu Zhang, and Guandong Xu. 2020. Perf-AL: Performance Prediction for Configurable Software through Adversarial Learning. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '20)*. ACM, 1–11.

[59] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 2015. Performance-Influence Models for Highly Configurable Systems. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE '15)*. ACM, 284–294.

[60] Urjoshi Sinha, Mikaela Cashman, and Myra B. Cohen. 2020. Using a Genetic Algorithm to Optimize Configurations in a Data-Driven Application. In *Search-Based Software Engineering: 12th International Symposium (SSBSE '20)*. Springer, 137–152. doi:10.1007/978-3-030-59762-7_10

[61] Mate Soos, Stephan Gocht, and Kuldeep S. Meel. 2020. Tinted, Detached, and Lazy CNF-XOR Solving and its Applications to Counting and Sampling. In *Proceedings of the International Conference on Computer-Aided Verification (CAV '20)*.

[62] Helge Spieker, Théo Matricon, Nassim Belmecheri, Jørn Eirik Betten, Gauthier Le Bartz Lyan, Heraldo Borges, Quentin Mazouni, Dennis Gross, Arnaud Gotlieb, and Mathieu Acher. 2025. Prompting for Performance: Exploring LLMs for Configuring Software. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI) (ICTAI '25)*. doi:10.48550/arXiv.2507.09790

[63] Kallistos Weis, Martina Maggio, Norbert Siegmund, and Sven Apel. 2026. *Risk and Confidence in Performance Bounds of Configuration Sampling Strategies — Sources and Data*. doi:10.5281/zenodo.19682296

[64] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. 2008. The Worst-Case Execution-Time Problem: Overview of Methods and Survey of Tools. *ACM Transactions on Embedded Computing Systems* 7, 3 (2008). doi:10.1145/1347375.1347389

[65] Yi Xiang, Han Huang, Yuren Zhou, Sizhe Li, Chuan Luo, Qingwei Lin, Miqing Li, and Xiaowei Yang. 2022. Search-Based Diverse Sampling from Real-World Software Product Lines. In *Proceedings of the 44th International Conference on*

Software Engineering (ICSE '22). ACM, 1945–1957.

[66] Yi Xiang, Xiaowei Yang, Han Huang, Zhengxin Huang, and Miqing Li. 2022. Sampling Configurations from Software Product Lines via Probability-Aware Diversification and SAT Solving. *Automated Software Engineering* 29, 54 (2022).