SYMPAIS: SYMbolic Parallel Adaptive Importance Sampling for Probabilistic Program Analysis

YICHENG LUO, Imperial College London and University College London
ANTONIO FILIERI, Imperial College London
YUAN ZHOU, University of Oxford

Probabilistic software analysis aims at quantifying the probability of a target event occurring during the execution of a program processing uncertain incoming data or written itself using probabilistic programming constructs. Recent techniques combine classic static analysis methods with inference procedure to obtain accurate quantification of the probability of rare target events, such as failures in a mission-critical system. However, current techniques face several scalability and applicability limitations when analyzing software processing with high-dimensional multivariate distributions. In this paper, we present SYMbolic Parallel Adaptive Importance Sampling (SYMPAIS), a new algorithm that combines symbolic execution with adaptive importance sampling to analyze probabilistic programs. Our method provides a general solution that scales to systems with high-dimensional inputs and demonstrates superior performance in quantifying rare events compared to prior work. Preliminary experimental results support the potential efficacy of our solution.

1 INTRODUCTION

Uncertainty in the real-world presents many challenges in reasoning and verification about the programs. Probabilistic software analysis extends classic static analyses techniques to consider the effects of probabilistic uncertainty, whether explicitly embedded within the code – as in probabilistic program – or externalized in a probabilistic input distribution [Dwyer et al. 2016]. Similarly to their classic counterparts, these analyses aim at inferring the probability of a target event to occur during execution, e.g., reaching a program state or triggering an exception.

For the probabilistic analysis of programs written in general-purpose programming languages, probabilistic symbolic execution (PSE) [Geldenhuys et al. 2012] exploits established symbolic execution engines for the language to extract constraints on probabilistic input or program variables that lead to the occurrence of the target event. The probability of satisfying any such constraints is then computed via model counting or inferred via solution space quantification methods, depending on the variable types and the characteristic of the constraints. Variations of PSE include incomplete analyses inferring probability bounds from a finite sample of program paths executed symbolically [Filieri et al. 2014], methods for non-deterministic programs [Luckow et al. 2014] and data structures [Filieri et al. 2015]. While PSE can solve more general inference problem, the overhead of symbolic execution is typically justified when the probability of the target event is very low (rare events) or high accuracy standards are required, e.g., for the certification purposes of safety-critical systems. Applications include, for example, reliability [Filieri et al. 2013] and security analyses [Brennan et al. 2018; Malacaria et al. 2018].

A core element of PSE is computing the probability of satisfying a constraint given the distribution of the constrained variables. In this paper, we focus on numerical constraints over floating-point variables. For limited classes of constraints and input distributions, analytical solutions or numerical integration might be possible [Büeler et al. 2000; Gehr et al. 2016]. However, these methods are inapplicable for more complex classes of constraints or intractable for high-dimensional problems. Monte Carlo methods offer more general applicability. Nonetheless, while theoretically insensitive to the dimensionality of the problems, care must be taken to apply direct Monte Carlo methods in

Authors' addresses: Yicheng Luo, Imperial College London, University College London, yicheng.luo.20@ucl.ac.uk; Antonio Filieri, Imperial College London, a.filieri@imperial.ac.uk; Yuan Zhou, University of Oxford, yuan.zhou@cs.ox.ac.uk.

Listing 1 Code snippet for an example safety monitor of an autopilot navigation system.

```
// Assumptions
altitude ::= Gaussian(10000, 100);
headFlap, tailFlap ::= Gaussian([0, 0], [[0.2, 0.1], [0.1, 0.2]]);
// Program
if (altitude <= 9000) { ...
    if (Math.sin(headFlap * tailFlap) > 0.999) {
        callSupervisor();} ...
} else { callSupervisor(); }
```

quantifying the probability of rare events. To reduce the variance of direct Monte Carlo estimators and increase the overall accuracy of the estimation, recent work [Borges et al. 2014, 2015] uses interval constraint propagation and branch-and-bound techniques to partition the input domain of a program into regions that contain *only*, *no*, or *in part* solutions to a constraint. This step analytically eliminates uncertainty about the regions containing *only* or *no* solutions, requiring samples to be taken only for the remaining parts. The probability mass from the input distribution enclosed into each region serves as its weight in the stratified sampling scheme, potentially reducing the estimator's variance when the mass enclosed into *in part* regions is small compared to the input domain. Nonetheless, limitations remain in that both the partitioning phase becomes intractable in high dimensions and computing the probability mass enclosed within a region requires computing the cumulative distribution function (CDF) of the input analytically or to manage the propagation of CDF estimation errors.

In this paper, we propose a new estimation method that addresses the challenge of inferring the probability of a rare event to occur during the execution of a program processing high-dimensional inputs. Our method combines ideas from symbolic execution with adaptive importance sampling to efficiently tackle the challenges that arise in analyzing critical systems. In particular, we iteratively adapt the proposal distribution to improve its efficiency in sampling within the solution space of a path constraint identified via symbolic execution as triggering the target event. This avoids exact CDF computations and improves accuracy and efficiency for higher-dimensional input spaces.

2 BACKGROUND

2.1 Probabilistic Symbolic Execution

Probabilistic symbolic execution (PSE) is a static analysis technique aiming at quantifying the probability of a target event occurring during execution. It uses symbolic execution to extract conditions on the values of inputs or specified random variables that lead to the occurrence of the target event. It then computes the probability of such constraints being satisfied given a probability distribution over the inputs or specified random variables. These constraints are called *path conditions* since they uniquely identify the execution path induced by an input satisfying them [King 1976].

Consider the simplified example in Listing 1, adapted from [Borges et al. 2014] using a C-like syntax and hypothetical random distributions for the input variables. The snippet represents part of the safety controller for a flying vehicle whose purpose is to detect environmental conditions that may compromise the safety of the crew and call for a supervisor's intervention. The purpose of the analysis is to estimate the probability of invoking `callSupervisor` at any point in the code. Safety-critical applications may require this probability to be very small (e.g., $< 10^{-7}$) and to be estimated with high accuracy. The symbolic execution of the snippet, where random variables are marked as symbolic, would return the following two path conditions, corresponding to all the possible invocations of `callSupervisor`: 1) PC_0: `altitude` > 9000; and 2) PC_1: `altitude` $\leq$

$9000 \wedge \sin(\texttt{headFlap} \cdot \texttt{tailFlap}) > 0.999$. The probability of satisfying a path condition is,

$$p_{PC} \stackrel{\text{def}}{=} \Pr(x \models PC) = \int_{x \models PC} \mathbb{1}_{PC}(\boldsymbol{x}) p(\boldsymbol{x})\, d\boldsymbol{x} \approx \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{PC}(\boldsymbol{x}^{(i)}), \boldsymbol{x}^{(i)} \sim p(\boldsymbol{x}) \tag{1}$$

where $\mathbb{1}_{PC}(\boldsymbol{x})$ denotes the indicator function, which returns 1 if $\boldsymbol{x} \models PC$ and 0 otherwise. Because analytical solutions to the integral are in general intractable or infeasible, Monte Carlo methods are used to approximate p_{PC}, as formalized in the last part of Equation (1).

Since the path conditions are disjoint [King 1976] (i.e., $\boldsymbol{x} \models PC_i \wedge \boldsymbol{x} \models PC_j \implies i = j$), an unbiased estimator for the probability of the target event to occur through any execution path is $\hat{p}_{PC} = \sum_i \hat{p}_{PC_i}$. Constraint slicing methods can further split individual path conditions in the conjunction of independent clauses, whose probabilities can thus be multiplied to obtain the probability of all the clauses to be satisfied [Filieri et al. 2013].

2.2 Constraint Propagation and Stratified Sampling

Constraint slicing reduces estimating the probability of satisfying a path condition to estimating the probability of satisfying its independent clauses. Such clauses predicate on a subset of program variables, thus reducing the dimensionality of each estimation problem. However, a direct Monte Carlo estimation may still suffer a slow convergence rate, and particular care should be placed when dealing with low-probability constraints.

Borges et al. [2014] proposed using interval constraint propagation [Granvilliers and Benhamou 2006] to find a union of *hyper-boxes* that reliably encloses all the solutions of a constraint. Portions of the domain outside the boxes are guaranteed to contain no solutions ($\mathbb{1}_{\cdot}(\cdot) = 0$). An hyper-box is identified as either an *inner box*, which contains only solutions, or an *outer box*, which may contain both solution and non-solution points (See Figure 3 for the visualization). Assuming it is possible to compute the CDF of the program's random variables at each vertex of the hyper-boxes, the partition of the variables domain induced by interval constraint propagation lends itself to an effective stratified sampling scheme, where each stratum is the hyper-box. The weight of a hyper-box can be computed multiplying the difference of the CDF at its vertices along each dimension (similarly to computing the volume of a hypercube), normalized by the same measure computed on the entire domain. The strata estimators would then be 1 for inner boxes, a direct Monte Carlo within each of the outer boxes, and 0 for the part of the domain not enclosed within any box.

This method – which we will refer to as `qCoral` in the following – proved very effective for the analysis of several software artifacts (and further optimizations have been explored in [Borges et al. 2015]). However, its effectiveness relies on 1) the interval constraint propagation pruning out a significant part of the domain as not containing solutions, 2) the ability to evaluate the CDF at the vertices of each hyper-box, and 3) an efficient sampling procedure to generate samples from within each outer box, which typically requires evaluating a truncated CDF.

The performance of interval constraint propagation degrades exponentially with the dimensionality of the problem, failing to produce significant pruning of the input domain within an acceptable time. In the worst-case scenario, the union of the hyper-boxes is the entire domain, with no area pruned out. Furthermore, the CDF of general multivariate distributions cannot be computed analytically; numerical solutions, on the other hand, require careful error propagation analysis, especially when the initial domain is partitioned in thousands to millions of hyper-boxes whose inaccuracies add up. See also Appendix A.2 for additional discussions on the qCoral method.

2.3 Importance Sampling

In this paper, we investigate an adaptive importance sampling (AIS) method [Owen 2013], which is more general and efficient compared to the stratified sampler in qCoral. Importance sampling uses a

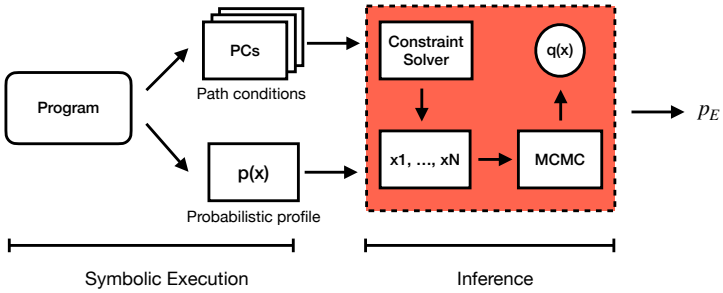


Fig. 1. Overview of SYMPAIS. We first perform probabilistic symbolic execution (PSE) to extract the input distribution and path conditions from the program. We then combine a constraint solver (*e.g.* Z3) and adaptive importace sampling (PI-MAIS) to estimate the probability of the target event $p_{\mathcal{E}}$ (orange box).

Algorithm 1 Symbolic Parallel Interacting Markov Adaptive Sampling (SYMPAIS)

Given: program P.
$PC, p(\boldsymbol{x}), \boldsymbol{x}_{\texttt{init}} \leftarrow \text{PSE}(P)$ ▷ Probabilistic symbolic execution
$\overline{p}(\boldsymbol{x}) \leftarrow p(\boldsymbol{x})\mathbb{1}_{PC}(\boldsymbol{x})$
Initialize the proposal distribution $q_{n,0}$ for $n = 1, \ldots, N$.
for $t \leftarrow 1, \ldots, T$ **do** ▷ PI-MAIS
    Update the proposal distribution $q_{n,t}$ for $n = 1, \ldots, N$ using MCMC
    Draw M samples from each proposal distribution $q_{n,t}$
    $\boldsymbol{x}_{n,t}^{(m)} \sim q_{n,t}(\boldsymbol{x})$ for $m = 1, \ldots, M$ and $n = 1, \ldots, N$.
    $w_{n,t}^{(m)} \leftarrow \frac{\overline{p}(\boldsymbol{x}_{n,t}^{(m)})}{\frac{1}{N}\sum_{j=1}^{N} q_{j,t}(\boldsymbol{x}_{n,t}^{(m)})}$ ▷ Compute importance-sampling weights
end for
$\hat{p}_{\mathcal{E}} \leftarrow \frac{1}{TMN} \sum w_{n,t}^{(m)}$

proposal distribution $q(\boldsymbol{x})$ to generate weighted samples to compute p_{PC} and avoids the need to find a stratification of the input domain, allowing us to achieve scalability to high-dimensional problems. Additionally, it achieves broader applicability than qCoral since it does not require sampling and evaluation of truncated CDFs.

In the context of estimating the probability of satisfying path conditions PC, the optimal proposal distribution $q^*(\boldsymbol{x})$ is $p(\boldsymbol{x})$ truncated and normalized in PC, *i.e.*,

$$q^*(\boldsymbol{x}) = \frac{1}{p_{PC}} p(\boldsymbol{x})\mathbb{1}_{PC}(\boldsymbol{x}). \tag{2}$$

In general, we are unable to sample from $q^*(\boldsymbol{x})$. Even when $p(\boldsymbol{x})$ is simple (*e.g.* Gaussians), the optimal proposal distribution may be intractable due to the complexity in the geometry of constraints in PC. Compared to direct Monte Carlo, we employ an adaptive scheme to automatically refine the proposal distribution that approximates $q^*(\boldsymbol{x})$ to improve sample efficiency.

3 SYMPAIS: SYMBOLIC PARALLEL ADAPTIVE IMPORTANCE SAMPLING FOR PROBABILISTIC PROGRAM ANALYSIS

We propose SYMPAIS: SYMbolic Parallel Adaptive Importance Sampling (SYMPAIS), a new technique for estimating the probability of a target event occurring during program execution. Motivated by the ideas of qCoral which makes use of PSE for integration calculation, and DCC [Zhou et al. 2019] which shows how to perform inference on probabilistic programs by dividing the paths,

SYMPAIS achieves probabilistic program analysis combing techniques from PSE and inference. An overview of SYMPAIS is provided in Figure 1 and Algorithm 1.

Given a program P, we first perform probabilistic symbolic execution (PSE) of the program source code (at line 2) to extract input distribution $p(\boldsymbol{x})$ and path conditions PC. This can be implemented by symbolic execution tools for a target programming language such as Java Symbolic PathFinder [Păsăreanu and Rungta 2010]. Given the paths information satisfying the constraints, we then estimate the probability corresponding to such paths via inference: we run AIS restricted on these paths and obtain the marginal likelihood estimates for the sub-regions restricted to the paths.

Specifically, to form the proposal distribution for the AIS in the sub-regions, we start with some initial samples satisfying the path conditions, *i.e.*, $\mathbb{1}_{PC}(\boldsymbol{x}_{\texttt{init}}) = 1$. The feasible solutions can be found either by using a constraint solver such as Z3 [De Moura and Bjørner 2008] or concolic-execution methods [Godefroid et al. 2005] where we use the former. Akin to the manner of Parallel Interacting Markov Adaptive Sampling (PI-MAIS) [Martino et al. 2017], we construct multiple independent MCMC chains to form the adaptive proposal distributions for the importance sampler. Details on PI-MAIS can be found in appendix A.3.

During inference, we set up N parallel Markov chains using the initial feasible solutions $\boldsymbol{x}_{\texttt{init}}$ obtained from line 2. For MCMC, we use $\overline{p}(\boldsymbol{x}) = p(\boldsymbol{x})\mathbb{1}_{\mathcal{E}}(\boldsymbol{x})$ (line 3) as the unnormalized target density and Random-Walk Metropolis-Hastings (RMH) [Metropolis et al. 1953] as the transition kernel. Other MCMC methods such as (constrained) Hamiltonian Monte Carlo (CHMC) [Afshar and Domke 2015; Betancourt 2010; Hoffman and Gelman 2014; Neal 2011; Nishimura et al. 2020] can also be used which can be more efficient for high-dimensional problems. Afterwards, we perform the importance sampling step in PI-MAIS in the loop from lines 5 to 10. This resembles sampling with a proposal distribution $q(\boldsymbol{x})$ that approximates the optimal proposal distribution

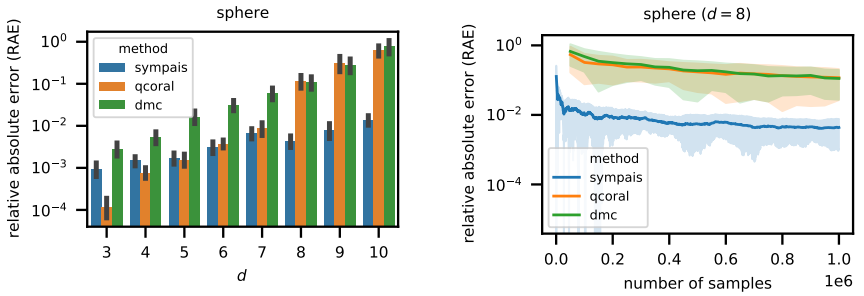
$$q^*(\boldsymbol{x}) \propto \overline{p}(\boldsymbol{x}) = p(\boldsymbol{x})\mathbb{1}_{PC}(\boldsymbol{x}).$$

4 PRELIMINARY EVALUATION

In this section, we consider the problem of estimating the probability that samples are drawn in a sphere $PC = \left\{\|\boldsymbol{x} - \boldsymbol{c}\|^2 \leq 1\right\}$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{c} \in \mathbb{R}^d$ is the center of the sphere. For this task, we use $p(\boldsymbol{x}) = \mathcal{N}(0, 1)$ and $\boldsymbol{c} = \mathbf{1}$. We compare different methods and report mean absolute error (RAE) by comparing it with a reference solution computed by running with a large number of samples.

Despite the simplicity of this problem, it illustrates the challenges faced by direct Monte Carlo methods as well as qCoral, namely the challenges of high-dimensionality and the presence of rare events. As d increases, the event becomes rarer, which makes estimation by direct Monte Carlo (DMC) more challenging. The increase in d also makes qCoral less effective due to solving for higher-dimensional constraints with the interval constraint solver.

The results are illustrated in fig. 2 and table 1. In general, qCoral is the most efficient for low-dimensional problems, although its performance degrades quickly for larger d. SYMPAIS is consistently better than direct Monte Carlo. While it is less efficient than qCoral for low-dimensional problems, its performance closely matches qCoral as d increases and eventually becomes orders of magnitude more sample efficient for $d \geq 8$. For $d < 8$, qCoral finds constrained interval boxes $B \subset \mathcal{X}$ which helps to improve the accuracy of the estimates compared to DMC. However, this improvement in efficiency can be accomplished similarly with the adaptive importance sampling scheme in SYMPAIS for $5 \leq d < 8$. However, for $d \geq 8$, the interval constraint solver has failed to approximate $\mathcal{X}_{PC}$ completely, as a result, $B = \mathcal{X}$ and qCoral degrades to DMC. In summary, SYMPAIS achieves similar performance to qCoral for problems beyond three dimensions and can efficiently tackle problems beyond the regime that can be handled efficiently by qCoral.



(a) Comparison of RAE for sphere (b) Convergence of estimates for sphere ($d = 8$)

Fig. 2. Comparison of different methods for estimating the probability of satisfying the sphere constraints. `dmc` refers estimation direct Monte Carlo, `qcoral` is qCoral and `sympais` is our proposed method. We fix the number of samples used by all the algorithms to 10^6. For `sympais`, we use random-walk Metropolis-Hastings as the MCMC kernel with a step size of 0.1. We use $N = 100$ parallel Markov chains and draw $M = 5$ samples from each chain at every iteration. Results are averaged over 20 independent runs, and the mean RAE and standard deviations are reported.

5 RELATED WORK

Our work takes a divide-and-conquer approach that exploits the symbolic constraints of probabilistic programs for more efficient inference and analysis.

Similar to qCoral [Borges et al. 2014], we decompose the original program into the analysis of smaller path conditions which can be more analyzed independently. Borges et al. [2015] extends qCoral to handle non-uniform input distributions via discretization. We do not discretize the input and are more sample-efficient for high-dimensional, rare-event problems when interval constraint propagation is not feasible or when computing the CDF of the input distribution accurately, as required by the stratified sampling scheme of qCoral, is not possible or inefficient. Borges et al. [2015] proposes an iterative refinement scheme to update the estimates. This is orthogonal to our work and can also be incorporated in SYMPAIS.

Similar ideas have also been considered in the PPL community where the motivation is to address challenges in efficiently generating intractable posterior samples. Chaganty et al. [2013] considers breaking a probabilistic program with branching and loops into small programs and use pre-image analysis to perform efficient importance sampling. We differ from Chaganty et al. [2013] in that we use PI-MAIS, which also uses MCMC to further adapt the proposal distributions. Nori et al. [2014] similarly uses the idea of pre-image analysis to design a proposal distribution that generates samples that are less likely to be rejected for MCMC. These analysis complements our approach and can potentially be incorporated to improve our MCMC scheme. Recent work by [Zhou et al. 2019] motivates the decomposition into subproblems by considering universal probabilistic programs with stochastic support. *i.e.*, depending on values of the samples, the program may take on different control flow, and the number of random variables vary as a result. This makes it difficult for designing a proposal distribution for efficient MCMC. [Zhou et al. 2019] approaches this issue by decomposing the problem into small straight-line programs (SLPs) for which the support is fixed and posterior inference is easier. Unlike in [Zhou et al. 2019], our sub-problems are decomposed according to control-flow statements instead of stochastic support. Furthermore, we find this decomposition via symbolic execution instead of relying on a global MCMC sampler to discover the subproblems. Our approach is thus more effective in the estimation of rare events and tail integrals.

REFERENCES

Hadi Mohasel Afshar and Justin Domke. 2015. Reflection, refraction, and Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, Vol. 2015-Janua. 3007–3015.

Michael Betancourt. 2010. Nested sampling with constrained Hamiltonian Monte Carlo. In *AIP Conference Proceedings*, Vol. 1305. 165–172. https://doi.org/10.1063/1.3573613 arXiv:1005.0157

Mateus Borges, Antonio Filieri, Marcelo D Amorim, Corina S P, and Willem Visser. 2014. Compositional Solution Space Quantification for Probabilistic Software Analysis. (2014). https://doi.org/10.1145/2594291.2594329

Mateus Borges, Antonio Filieri, Marcelo d'Amorim, and Corina S Păsăreanu. 2015. Iterative distribution-aware sampling for probabilistic symbolic execution. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 866–877.

Tegan Brennan, Seemanta Saha, Tevfik Bultan, and Corina S. Păsăreanu. 2018. Symbolic Path Cost Analysis for Side-channel Detection. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. ACM, 27–37. https://doi.org/10.1145/3213846.3213867

Benno Büeler, Andreas Enge, and Komei Fukuda. 2000. *Exact Volume Computation for Polytopes: A Practical Study*. Birkhäuser Basel, Basel, 131–154. https://doi.org/10.1007/978-3-0348-8438-9_6

Arun T Chaganty, Aditya V Nori, and Sriram K Rajamani. 2013. Efficiently sampling probabilistic programs via program analysis. In *Journal of Machine Learning Research*, Vol. 31. 153–160.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT Solver. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 4963 LNCS. 337–340. https://doi.org/10.1007/978-3-540-78800-3_24

Matthew B Dwyer, Antonio Filieri, Jaco Geldenhuys, Mitchell Gerrard, Corina S Pasareanu, and Willem Visser. 2016. Probabilistic Program Analysis. (2016). https://doi.org/10.1007/978-3-319-60074-1

Antonio Filieri, MarceloF. Frias, Corina S. Păsăreanu, and Willem Visser. 2015. Model Counting for Complex Data Structures. In *Model Checking Software (LNCS)*, Bernd Fischer and Jaco Geldenhuys (Eds.), Vol. 9232. Springer, 222–241. https://doi.org/10.1007/978-3-319-23404-5_15

Antonio Filieri, Corina S Păsăreanu, and Willem Visser. 2013. Reliability analysis in symbolic pathfinder. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 622–631.

Antonio Filieri, Corina S. Pasareanu, Willem Visser, and Jaco Geldenhuys. 2014. Statistical symbolic execution with informed sampling. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Vol. 16-21-Nove. 437–448. https://doi.org/10.1145/2635868.2635899

Timon Gehr, Sasa Misailovic, and Martin Vechev. 2016. PSI: Exact symbolic inference for probabilistic programs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9779. 62–83. https://doi.org/10.1007/978-3-319-41528-4_4

Jaco Geldenhuys, Matthew B Dwyer, and Willem Visser. 2012. Probabilistic symbolic execution. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*. 166–176.

Patrice Godefroid, Nils Klarlund, and Koushik Sen. 2005. DART: Directed automated random testing. *ACM SIGPLAN Notices* 40, 6 (2005), 213–223. https://doi.org/10.1145/1064978.1065036

Laurent Granvilliers and Frédéric Benhamou. 2006. Algorithm 852: RealPaver: An interval solver using constraint satisfaction techniques. *ACM Trans. Math. Software* 32, 1 (2006), 138–156. https://doi.org/10.1145/1132973.1132980

Matthew D Hoffman and Andrew Gelman. 2014. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* 15 (2014), 1593–1623. arXiv:1111.4246

James C. King. 1976. Symbolic Execution and Program Testing. *Commun. ACM* 19, 7 (July 1976), 385–394. https://doi.org/10.1145/360248.360252

Kasper Luckow, Corina S. Păsăreanu, Matthew B. Dwyer, Antonio Filieri, and Willem Visser. 2014. Exact and Approximate Probabilistic Symbolic Execution for Nondeterministic Programs. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE '14)*. ACM, New York, NY, USA, 575–586. https://doi.org/10.1145/2642937.2643011

P. Malacaria, M. Khouzani, C. S. Pasareanu, Q. Phan, and K. Luckow. 2018. Symbolic Side-Channel Analysis for Probabilistic Programs. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF) (CSF)*. 313–327. https://doi.org/10.1109/CSF.2018.00030

L Martino, V. Elvira, D. Luengo, and J. Corander. 2017. Layered adaptive importance sampling. *Statistics and Computing* 27, 3 (2017), 599–623. https://doi.org/10.1007/s11222-016-9642-5 arXiv:1505.04732

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21, 6 (jun 1953), 1087–1092. https://doi.org/10.1063/1.1699114

Radford M Neal. 2011. MCMC using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. 113–162. https://doi.org/10.1201/b10905-6 arXiv:1206.1901

Akihiko Nishimura, David B Dunson, and Jianfeng Lu. 2020. Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods. *Biometrika* (2020). https://doi.org/10.1093/biomet/asz083 arXiv:1705.08510

Aditya V Nori, Chung Kil Hur, Sriram K Rajamani, and Selva Samuel. 2014. R2: An efficient MCMC sampler for probabilistic programs. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 4. 2476–2482. www.aaai.org

Art B. Owen. 2013. *Monte Carlo theory, methods and examples.*

Corina S Păsăreanu and Neha Rungta. 2010. Symbolic PathFinder: symbolic execution of Java bytecode. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. 179–180.

Yuan Zhou, Hongseok Yang, Yee Whye Teh, and Tom Rainforth. 2019. Divide, Conquer, and Combine: a New Inference Strategy for Probabilistic Programs with Stochastic Support. (2019). arXiv:1910.13324 http://arxiv.org/abs/1910.13324

A APPENDICES

A.1 Results for the sphere benchmark

		$\hat{p}_{\mathcal{E}}$		RAE	
d	method	mean	std.	mean	std.
3	dmc	0.058333	0.000200	0.002827	0.002424
	qcoral	0.058246	0.000011	0.000119	0.000153
	sympais	0.058279	0.000062	0.000954	0.000762
4	dmc	0.016571	0.000107	0.005424	0.004089
	qcoral	0.016600	0.000012	0.000772	0.000528
	sympais	0.016616	0.000028	0.001496	0.000798
5	dmc	0.004303	0.000090	0.016431	0.013140
	qcoral	0.004289	0.000008	0.001549	0.001132
	sympais	0.004287	0.000009	0.001711	0.001246
6	dmc	0.001017	0.000038	0.030994	0.020330
	qcoral	0.001020	0.000004	0.003740	0.001977
	sympais	0.001020	0.000003	0.003123	0.002261
7	dmc	0.000228	0.000017	0.061292	0.042406
	qcoral	0.000226	0.000002	0.008837	0.007029
	sympais	0.000226	0.000001	0.006708	0.004375
8	dmc	0.000045	0.000006	0.111748	0.084479
	qcoral	0.000045	0.000007	0.117763	0.091145
	sympais	0.000047	0.000000	0.004345	0.003324
9	dmc	0.000009	0.000003	0.283070	0.241214
	qcoral	0.000011	0.000004	0.313718	0.302383
	sympais	0.000009	0.000000	0.007991	0.007369
10	dmc	0.000003	0.000002	0.790323	0.638325
	qcoral	0.000002	0.000001	0.626393	0.373326
	sympais	0.000002	0.000000	0.013702	0.008431

Table 1. Benchmark results for Sphere

A.2 qCoral

qCoral [Borges et al. 2014] is a compositional way of estimating event probability in probabilistic program analysis. qCoral utilizes the idea decomposing the problem with probabilistic program analysis with using symbolic execution. For each path constraint, it uses stratified sampling based on outputs from an interval constraint solver. The intervals produced by the constraint solver may be integrated analytically or analyzed based on Monte Carlo sampling.

For each path constraint PC_i consisting of k inequality constraints $\{c_1(\boldsymbol{x}) \leq 0, \ldots, c_k(\boldsymbol{x}) \leq 0\}$ over the input variables $\boldsymbol{x} \in \mathbb{R}^n$, qCoral estimates the probability with stratified sampling to reduce variance. qCoral uses RealPaver [Granvilliers and Benhamou 2006], an interval constraint solver useful for modeling and solving nonlinear systems. In our case, we use RealPaver to quantify the solution space. The purpose of using RealPaver is to generate a set of n-dimensional boxes that cover the solution set. An interval over variable $\boldsymbol{x}_n$ is a closed real interval $[l_n, h_n]$ in the domain of $x_n \in \mathbb{R}$. An *interval box* is a Cartesian product of intervals over the variables $\boldsymbol{x}$. For example, an

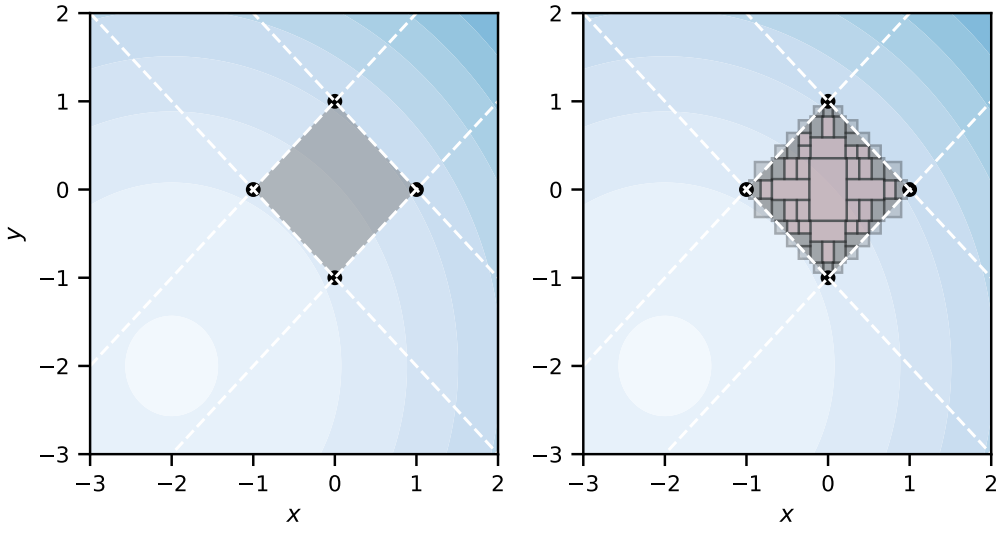


Fig. 3. Graphical illustration of the problem in estimating the path probabilities. The gray boxes correspond to outer boxes $\mathcal{O}$ and pink boxes correspond to inner boxes $\mathcal{I}$.

interval box produced by RealPaver for two-dimensional inputs $\boldsymbol{x} \in \mathbb{R}^2 = [x_1, x_2]$ with constraint $x_1^2 + x_2^2 \leq 1$ may be

```
x_1 in [0, 0.5]
x_2 in [0, 0.5]
```

Given a set of constraints, RealPaver constructs two sets of interval boxes $\mathcal{I}$ and $\mathcal{O}$. The inner box set $\mathcal{I} = \{I_1, \ldots, I_p\}$ consists of boxes that strictly satisfy the constraints. *i.e.*, $\boldsymbol{x} \in I_i \implies \mathbb{1}_{PC}(\boldsymbol{x})$. The outer box set $\mathcal{O} = \{O_1, \ldots, O_q\}$ consists of intervals that includes only only part of the solution set. Figure 3 gives a graphical illustration of the box generation process.

Given the two set of boxes, we can perform stratified sampling and estimate the probability of constraint as

$$\hat{p}_{\mathcal{E}}^{qCoral} = \underbrace{\sum_{i=1}^{|\mathcal{O}|} p(O_i) \sum_{j=1}^{N_j} \mathbb{1}_{PC}(\boldsymbol{x}_i^{(j)})}_{\text{stratified sampling on outer boxes}} + \underbrace{\sum_{i=1}^{|\mathcal{I}|} p(I_i)}_{\text{exact integration on inner boxes}} \tag{3}$$

where we use $p(O_i)$ to denote the probability that samples drawn in the interval box O_i and similarly $p(I_i)$ for the outer boxes. $\boldsymbol{x}_i^{(j)}$ are samples draw from the conditional distribution $p(\boldsymbol{x} \mid I_i)$.

qCoral computes the probability analytically for inner boxes I_i that strictly satisfy the constraints analytically. For outer boxes O_i that only partially intersect with the path constraint. qCoral estimates the probability using a Hit-or-Miss Monte Carlo on the domain of the interval box.

A.3 PI-MAIS

The choice of the proposal distribution q^* is critical to the success of applying importance sampling. The optimal proposal distribution q^*, in general, is not realizable. In practice, we want to find proposal distribution that is close to q^* to ensure importance sampling offers significant variance reduction. The natural question is how to construct an approximately optimal proposal distribution. In this section, we discuss one adaptive method of constructing proposal distribution.

The Parallel Interacting Markov Adaptive Sampling (PI-MAIS) [Martino et al. 2017] is an adaptive importance sampling algorithm which uses a hierarchical model as the proposal distribution. The proposal distribution in PI-MAIS is parametrized by several *sub-proposals* $q_1, \ldots, q_N$. To sample from the proposal distribution, we first choose a sub proposal q_i and then draw samples from $\boldsymbol{x}$ from q_i. Together the sub-proposals form a mixture distribution. PI-MAIS adapts the sub-proposals to the target distribution by running parallel MCMC so that it can form efficient proposal distributions for target distributions that are multi-modal and non-linear.

One particular instantiation of the algorithm is to consider a proposal distribution that is parametrized by several Gaussian distributions $q_i(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \Sigma)$. The pdf for the proposal distribution is then given by a mixture of Gaussians

$$q(\cdot) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\cdot, \boldsymbol{\mu}_i, \Sigma).$$

The mean vectors are adapted by running N parallel MCMC sampler. At each step t, the sampler produces a set of states $\boldsymbol{x}_{n,t}$. The proposal distribution at step t is given by

$$q_t(\cdot) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\cdot | \boldsymbol{x}_{n,t}, \Sigma).$$

The marginal likelihood estimator $\hat{Z}$ is

$$\hat{Z} = \frac{1}{N} \frac{1}{T} \frac{1}{M} \sum_{t=1}^{T} \sum_{n=1}^{N} \sum_{m=1}^{M} w_{n,t,m}, \; w_{n,t,m} = \frac{p(\boldsymbol{x}_m)}{q_{n,t}(\boldsymbol{x}_m)}$$