Testing Self-Reducible Samplers

Rishiraj Bhattacharyya[1*], **Sourav Chakraborty**[2*], **Yash Pote**[3,4*], **Uddalok Sarkar**[2*], **Sayantan Sen**[3*]

[1]University of Birmingham
[2] Indian Statistical Institute Kolkata
[3] National University of Singapore
[4] CREATE

r.bhattacharyya@bham.ac.uk, {sourav, uddalok_r}@isical.ac.in, {yashppote, sayantan789}@gmail.com

Abstract

Samplers are the backbone of the implementations of any randomised algorithm. Unfortunately, obtaining an efficient algorithm to test the correctness of samplers is very hard to find. Recently, in a series of works, testers like Barbarik, Teq, Flash for testing of some particular kinds of samplers, like CNF-samplers and Horn-samplers, were obtained. But their techniques have a significant limitation because one can not expect to use their methods to test for other samplers, such as perfect matching samplers or samplers for sampling linear extensions in posets. In this paper, we present a new testing algorithm that works for such samplers and can estimate the distance of a new sampler from a known sampler (say, uniform sampler).

Testing the identity of distributions is the heart of testing the correctness of samplers. This paper's main technical contribution is developing a new distance estimation algorithm for distributions over high-dimensional cubes using the recently proposed sub-cube conditioning sampling model. Given subcube conditioning access to an unknown distribution P, and a known distribution Q defined over $\{0,1\}^n$, our algorithm CubeProbeEst estimates the variation distance between P and Q within additive error ζ using $\mathcal{O}\left(n^2/\zeta^4\right)$ subcube conditional samples from P. Following the testing-via-learning paradigm, we also get a tester which distinguishes between the cases when P and Q are ε-close or η-far in variation distance with probability at least 0.99 using $\mathcal{O}(n^2/(\eta-\varepsilon)^4)$ subcube conditional samples.

The estimation algorithm in the sub-cube conditioning sampling model helps us to design the first tester for self-reducible samplers. The correctness of the testers is formally proved. On the other hand, we implement our algorithm to create CubeProbeEst and use it to test the quality of three samplers for sampling linear extensions in posets.

Introduction

Sampling algorithms play a pivotal role in enhancing the efficiency and accuracy of data analysis and decision-making across diverse domains (Chandra and Iyengar 1992; Yuan et al. 2004; Naveh et al. 2006; Mironov and Zhang 2006; Soos, Nohl, and Castelluccia 2009; Morawiecki and Srebrny 2013; Ashur, De Witte, and Liu 2017). With the exponential surge in data volume, these algorithms provide the means to derive meaningful insights from massive datasets without the burden of processing the complete information. Additionally, they aid in pinpointing and mitigating biases inherent in data, ensuring the attainment of more precise and equitable conclusions. From enabling statistical inferences to propelling advancements in machine learning, safeguarding privacy, and facilitating real-time decision-making, sampling algorithms stand as a cornerstone in extracting information from the vast data landscape of our modern world.

However, many advanced sampling algorithms are often prohibitively slow (hash-based techniques of (Chakraborty, Meel, and Vardi 2013; Ermon et al. 2013; Chakraborty et al. 2014; Meel et al. 2016) and MCMC-based methods of (Andrieu et al. 2003; Brooks et al. 2011; Jerrum 1998)) or lack comprehensive verification ((Ermon, Gomes, and Selman 2012), (Dutra et al. 2018), (Golia et al. 2021)). Many popular methods like "statistical tests" rely on heuristics without guarantees of their efficacy. Utilizing unverified sampling algorithms can lead to significant pitfalls, including compromised conclusion accuracy, potential privacy, and security vulnerabilities. Moreover, the absence of verification hampers transparency and reproducibility, underscoring the critical need for rigorous validation through testing, comparison, and consideration of statistical properties. Consequently, a central challenge in this field revolves around designing tools to certify sampling quality and verify correctness, which necessitates overcoming the intricate task of validating probabilistic programs and ensuring their distributions adhere to desired properties.

A notable breakthrough in addressing this verification challenge was achieved by (Chakraborty and Meel 2019), who introduced the statistical testing framework known as "Barbarik". This method proved instrumental in testing the correctness of uniform CNF (Conjunctive Normal Form) samplers by drawing samples from conditional distributions. Barbarik demonstrated three key properties: accepting an almost correct sampler with high probability, rejecting a far-from-correct sampler with high probability, and rejecting a "well-behaved" but far-from-correct sampler with high probability. There have been a series of follow-up works (Meel, Pote, and Chakraborty 2020; Pote and Meel 2021, 2022; Banerjee et al. 2023). However, in this frame-

[*]These authors contributed equally.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

work, conditioning is achieved using a gadget that does not quite generalize to applications beyond CNF sampling. For instance, for linear-extension sampling (Huber 2014), where the goal is to sample a linear ordering agreeing with a given poset, the test requires that the post-conditioning residual input be a supergraph of the original input, with the property that it has exactly two *user-specified* linear-extensions. This requirement is hard to fulfill in general. On the other hand, a generic tester that would work for any sampler implementation without any additional constraints and simultaneously be sample efficient is too good to be true (Paninski 2008). From a practical perspective, the question is: *Can we design an algorithmic framework for testers that would work for most deployed samplers and still have practical sample complexity?*

We answer the question positively. We propose algorithms that offer a generic approach to estimating the distance between a known and an unknown sampler, assuming both follow the ubiquitous self-reducible sampling strategy. Our techniques follow a constrained sampling approach, extending its applicability to wide range of samplers without mandating such specific structural conditions. A key foundational contribution of this paper includes leveraging the subcube conditional sampling techniques (Bhattacharyya and Chakraborty 2018) and devising a method to estimate the distance between samplers – a challenge often more intricate than simple correctness testing.

Organization of our paper We first present the preliminaries followed by a description of our results and their relevance. We then give a detailed description of our main algorithms CubeProbeEst and CubeProbeTester. The detailed theoretical analysis is presented in the supplementary material. We only present a high-level technical overview. Finally, we present our experimental results and conclude. The extended version of the paper is available at www.arxiv.org/abs/2312.10999.

Preliminaries

In this paper, we are dealing with discrete probability distributions whose sample space is an n-dimensional Boolean hypercube, $\{0,1\}^n$. For a distribution $\mathcal{D}$ over a universe Ω, and for any $x \in \Omega$, we denote by $\mathcal{D}(x)$ the probability mass of the point x in $\mathcal{D}$. $[n]$ denotes the set $\{1,\ldots,n\}$. For concise expressions and readability, we use the asymptotic complexity notion of $\widetilde{\mathcal{O}}$, where we hide polylogarithmic dependencies of the parameters.

Samplers, Estimators, and Testers A sampler $\mathcal{I}$: $\mathsf{Domain} \to \mathsf{Range}$ is a randomized algorithm which, given an input $x \in \mathsf{Domain}$, outputs an element in Range. For a sampler $\mathcal{I}$, $\mathcal{D}^{\mathcal{I},\psi}$ denotes the probability distribution of the output of $\mathcal{I}$ when the input is $\psi \in \mathsf{Domain}$. In other words,

$$\forall x \in \mathsf{Range},\ \mathcal{D}^{\mathcal{I},\psi}(x) = \Pr[\mathcal{I}(\psi) = x],$$

where the probability is over the internal random coins of $\mathcal{I}$. We define a sampler $\mathcal{I}_{\mathcal{W}}$ to be a *known sampler* if, for any input $\psi \in \mathsf{Domain}$, we know its probability distribution $\mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}$ explicitly. We note that the input ψ depends on the application. For example, in the perfect-matching and linear-extension samplers, ψ is a graph, whereas, for the CNF sampler, ψ is a CNF formula.

Definition 1 (Total variation distance). Let $\mathcal{I}_{\mathcal{W}}$ and $\mathcal{I}_{\mathcal{G}}$ be two samplers. For an input $\psi \in \mathsf{Domain}$, the variation distance between $\mathcal{I}_{\mathcal{G}}$ and $\mathcal{I}_{\mathcal{W}}$ is defined as:

$$\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) = \max_{A \subseteq \mathsf{Range}} \{\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(A) - \mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}(A)\}.$$

Definition 2 ((ζ,δ)-approx d_{TV} estimator). A (ζ,δ)-approx d_{TV} estimator is a randomized approximation algorithm that given two sampler $\mathcal{I}_{\mathcal{G}}$ and $\mathcal{I}_{\mathcal{W}}$, an input ψ, tolerance parameter $\zeta \in (0,1/3]$ and a confidence parameter $\delta \in (0,1)$, with probability $(1-\delta)$ returns an estimation $\widehat{\mathsf{dist}^{\psi}}$ of $\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}})$ such that:

$$\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) - \zeta \le \widehat{\mathsf{dist}^{\psi}} \le \mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) + \zeta$$

Definition 3 (ε-closeness and η-farness). Consider any sampler $\mathcal{I}_{\mathcal{G}}$. $\mathcal{I}_{\mathcal{G}}$ is said to be ε-close to another sampler $\mathcal{I}_{\mathcal{W}}$ on input ψ, if $\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) \le \varepsilon$ holds. On the other hand, $\mathcal{I}_{\mathcal{G}}$ is said to be η-far from $\mathcal{I}_{\mathcal{W}}$ with respect to some input ψ if $\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) \ge \eta$ holds.

Definition 4 ((ε,η,δ)-identity tester)). An $(\varepsilon,\eta,\delta)$-identity tester takes as input an unknown sampler $\mathcal{I}_{\mathcal{G}}$, a known sampler $\mathcal{I}_{\mathcal{W}}$, an input ψ to the samplers, a tolerance parameter $\varepsilon \in (0,1/3)$, an intolerance parameter $\eta \in (0,1]$ with $\eta > \varepsilon$, a confidence parameter $\delta \in (0,1)$, and with probability at least $(1-\delta)$: (1) outputs ACCEPT if $\mathcal{I}_{\mathcal{G}}$ is ε-close to $\mathcal{I}_{\mathcal{W}}$ on input ψ, (2) outputs REJECT if $\mathcal{I}_{\mathcal{G}}$ is η-far from $\mathcal{I}_{\mathcal{W}}$ on input ψ.

For practical purposes, δ can be 0.99 or any close-to-one constant. From now onwards, we shall consider the input domain and output range of a sampler to be a Boolean hypercube, that is, $\mathsf{Domain} = \{0,1\}^m$ and $\mathsf{Range} = \{0,1\}^n$ for some integers m and n. Therefore the universe of probability distributions of samplers is n-dimensional binary strings.

Self-reducible sampler. A self-reducible sampler $\mathcal{I}$: $\{0,1\}^m \to \{0,1\}^n$ generates a sample x by first sampling a bit and then sampling the rest of the substring. Formally, we can define a self-reducible sampler as follows:

Definition 5 (Self-reducible sampler). A sampler $\mathcal{I}$: $\{0,1\}^m \to \{0,1\}^n$ is said to be a self-reducible sampler if, for any input $\psi \in \{0,1\}^m$, there exists $\widehat{\psi} \in \{0,1\}^m$ for which the following is true:

$$\mathcal{D}^{\mathcal{I},\psi}(x_1x_2...x_n)|_{x_1=b_1,\ldots,x_i=b_i} = \mathcal{D}^{\mathcal{I},\widehat{\psi}}(b_1...b_ix_{i+1}...x_n)$$

where $b_i \in \{0,1\}$ for all i.

The concept of self-reducibility has been influential in the field of sampling since the work of (Jerrum, Valiant, and Vazirani 1986), which showed the computational complexity equivalence of approximate sampling and counting for problems in #P. Intuitively, self-reducibility is the idea that one can construct the solution to a given problem from the solutions of subproblems of the same problem. Self-reducibility is a critical requirement for simulating subcube conditioning. Also, it does not hamper the model's generality too much. As observed in (Khuller and Vazirani 1991; Große, Rothe, and Wechsung 2006; Talvitie, Vuoksenmaa, and Koivisto 2020), all except a few known problems are self-reducible.

Subcube Conditioning over Boolean hypercubes Let P be a probability distribution over $\{0,1\}^n$. Sampling using subcube conditioning accepts $A_1, A_2, \ldots, A_n \subseteq \{0,1\}$, constructs $S = A_1 \times A_2 \times \ldots \times A_n$ as the condition set, and returns a vector $x = (x_1, x_2, \ldots, x_n)$, such that $x_i \in A_i$, with probability $P(x)/(\sum_{w \in S} P(w))$. If $P(S) = 0$, we assume the sampling process would return an element from S uniformly at random. A sampler that follows this technique is called a subcube conditioning sampler.

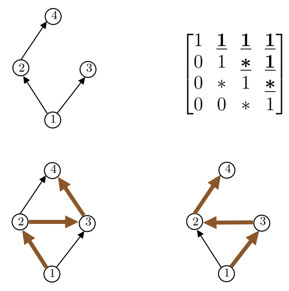
Linear-Extension of a Poset We applied our prototype implementation on verifying linear-extension samplers of a poset. Let us first start with the definition of a poset.

Definition 6 (Partially ordered set (Poset)). Let S be a set on k elements. A relation $\preceq$ (subset of $S \times S$) is said to be a partial order if $\preceq$ is (i) reflexive ($a \preceq a$ for every $a \in S$) (ii) anti-symmetric ($a \preceq b$ and $b \preceq a$ implies $a = b$ for every $a, b \in S$) and (iii) transitive ($a \preceq b$ and $b \preceq c$ implies $a \preceq c$ for every $a, b, c \in S$). We say $(S, \preceq)$ is a *partially ordered set* or *poset* in short. If all pairs of S are comparable, that is, for any $a, b \in S$, either $a \preceq b$ or $b \preceq a$ then $(S, \preceq)$ is called a *linear ordered set*.

Definition 7 (Linear-extension of poset). A relation $\preceq_l \supseteq \preceq$ is called a linear-extension of $\preceq$, if $(S, \preceq_l)$ is linearly ordered. Given a poset $\mathcal{P} = (S, \preceq)$, we denote the set of all possible linear-extensions by $\mathcal{L}(\mathcal{P})$.

Definition 8 (Linear-extension sampler). Given a poset $\mathcal{P} = (S, \preceq)$, a linear-extension sampler $\mathcal{I}_{Lext}$ samples a possible linear-extension $\preceq_l$ of $\mathcal{P}$ from the set of all possible linear-extensions $\mathcal{L}(\mathcal{P})$.

Linear-extension to Boolean Hypercube Let us define a *base* linear ordering on S as $\preceq_l'$. We order the elements of S as $S_1 \preceq_l' S_2 \preceq_l' \ldots \preceq_l' S_k$ based on $\preceq_l'$, where $k = |S|$. For a poset $\mathcal{P} = (S, \preceq)$, we construct a $k \times k$ matrix $\mathcal{M}_{\mathcal{P}}$ such that for all i, $\mathcal{M}_{\mathcal{P}}(i,i) := 1$ and for all $i \neq j$, if $S_i \preceq S_j$ then $\mathcal{M}_{\mathcal{P}}(i,j) := 1$ and $\mathcal{M}_{\mathcal{P}}(i,j) := 0$ when $S_j \preceq S_i$, if $(S_i, S_j) \notin \preceq$, that is if S_i, S_j are not comparable in $\preceq$, then $\mathcal{M}_{\mathcal{P}}(i,j) := *$. The matrix $\mathcal{M}_{\mathcal{P}}$ is a unique representation of the poset $\mathcal{P} = (S, \preceq)$. $\mathcal{M}_{\mathcal{P}}$ is anti-symmetric, i.e., the upper triangle of $\mathcal{M}_{\mathcal{P}}$ is exactly the opposite of the lower triangle (apart from the $*$ and the diagonal entries). So only the upper triangle of $\mathcal{M}_{\mathcal{P}}$ without the diagonal entries can represent $\mathcal{P}$. Now unrolling of the upper triangle of $\mathcal{M}_{\mathcal{P}}$ (without the diagonal) creates a $\{0,1,*\}^{^kC_2}$ string $x_{\mathcal{M}_{\mathcal{P}}}$. Suppose for a $\mathcal{P}$ there are n $*$'s in the unrolling. Then we can say sampling a linear-extension of $\mathcal{P}$ is equivalent to sampling from a $\{0,1\}^n$ subcube of the Boolean hypercube $\{0,1\}^{^kC_2}$, where $\mathcal{P}$ induces subcube conditioning by fixing the bits of non-$*$ dimensions. Adding one more new pair, say $(S_{i'}, S_{j'})$, to $\mathcal{P}$ results in fixing one more bit of $x_{\mathcal{M}_{\mathcal{P}}}$ and vice versa. We introduce a mapping $\mathsf{SubCond}$ that can incorporate a new pair into poset $\mathcal{P}$ and subsequently fixes



$$\begin{bmatrix} 1 & \underline{\mathbf{1}} & \underline{\mathbf{1}} & \underline{\mathbf{1}} \\ 0 & 1 & \underline{\mathbf{*}} & \underline{\mathbf{1}} \\ 0 & * & 1 & \underline{\mathbf{*}} \\ 0 & 0 & * & 1 \end{bmatrix}$$

Figure 1: Top-left graph represents cover graph of a poset $\mathcal{P}$ over $S = \{1,2,3,4\}$, and poset relation $\preceq = \{(1,2),(1,3),(2,4),(1,4)\}$. The bottom row shows two possible linear-extensions $1 \preceq 2 \preceq 3 \preceq 4$ and $1 \preceq 3 \preceq 2 \preceq 4$, with corresponding cover graphs (red arrows). The matrix on the top-right corresponds to $\mathcal{M}_{\mathcal{P}}$. Unrolling of the upper triangle (bold-underline) of $\mathcal{M}_{\mathcal{P}}$ gives $x_{\mathcal{M}_{\mathcal{P}}} = 111*1*$. Fixing the 4th bit of $x_{\mathcal{M}_{\mathcal{P}}}$ to 0 is equivalent to including the relation $(3,2)$ into $\preceq$. Here $\mathsf{SubCond}(\preceq, 4) = \preceq \cup \{(3,2)\}$.

the corresponding bit in bit string $x_{\mathcal{M}_{\mathcal{P}}}$. Thus $\mathsf{SubCond}$ provides a method to achieve subcube conditioning on a poset.

Basic Probability Facts We will use the following probability notations in our algorithm. A random variable X is said to follow the exponential distribution with parameter λ if $\Pr(X = x) = \lambda e^{-\lambda x}$ if $x \geq 0$ and 0 otherwise. This is represented as $X \sim \mathsf{Exp}(\lambda)$. A random variable X is said to be *sub-Gaussian* (SubG in short) with parameter α^2 if and only if its tails are dominated by a Gaussian of parameter α^2. We include formal definitions and related concentration bounds in the supplementary material.

Our Results

The main technical contribution of this work is the algorithm $\mathsf{CubeProbeEst}$ that can estimate the variation distance between a known and an unknown self-reducible sampler. The following informal theorem captures the details.

Theorem 9. *For an error parameter $\zeta \in (0,1)$, and a constant $\delta < 1/3$, $\mathsf{CubeProbeEst}$ is (ζ, δ)-approx d_{TV} estimator between a known and unknown self-reducible samplers $\mathcal{I}_{\mathcal{W}}$ and $\mathcal{I}_G$ respectively with sample complexity of $\widetilde{\mathcal{O}}\left(n^2/\zeta^4\right)$.*

Our framework seamlessly extends to yield an $(\varepsilon, \eta, \delta)$-tester $\mathsf{CubeProbeTester}$ through the "testing-via-learning" paradigm (Diakonikolas et al. 2007; Gopalan et al. 2009; Servedio 2010). To test whether the sampler's output distribution is ε-close or η-far from the target output distribution, the resultant tester requires $\widetilde{\mathcal{O}}\left(n^2/\left(\eta - \varepsilon\right)^4\right)$ samples.

To demonstrate the usefulness of $\mathsf{CubeProbeEst}$, we developed a prototype implementation with experimental evaluations in gauging the correctness of linear-extension samplers while emulating uniform samplers. Counting the size of the set of linear extensions and sampling from them has been widely studied in a series of works by (Huber 2014; Talvitie et al. 2018a,b). The problem found extensive applications in artificial intelligence, particularly in learning graphical models (Wallace, Korb, and Dai 1996), in sorting (Peczarski 2004), sequence analysis (Mannila and Meek 2000), convex rank tests (Morton et al. 2009), preference reasoning (Lukasiewicz, Martinez, and Simari 2014), partial order plans (Muise, Beck, and McIlraith 2016) etc. Our implementation extends to a closeness tester that accepts "close to uniform" samplers and rejects "far from uniform" samplers. Moreover, while rejecting, our implementation can produce a certificate of non-uniformity. CubeProbeEst and CubeProbeTester are the first estimator and tester for general self-reducible samplers.

Novelty in Our Contributions In relation to the previous works, we emphasize our two crucial novel contributions.

- Our algorithm is grounded in a notably refined form of "grey-box" sampling methodology, setting it apart from prior research endeavors (Chakraborty and Meel 2019; Meel, Pote, and Chakraborty 2020; Banerjee et al. 2023). While prior approaches required arbitrary conditioning, our algorithm builds on the significantly weaker subcube conditional sampling paradigm (Bhattacharyya and Chakraborty 2018). Subcube conditioning is a natural fit for ubiquitous self-reducible sampling, and thus our algorithm accommodates a considerably broader spectrum of sampling scenarios.
- All previous works produced testers crafted to produce a "yes" or "no" answer to ascertain correctness of samplers. In essence, these testers strive to endorse samplers that exhibit "good" behavior while identifying and rejecting those that deviate significantly from this standard. However, inherent technical ambiguity exists in setting the thresholds of the distances (η and ε) that would label a sampler as good or bad. In contrast, the CubeProbeEst framework produces the estimated statistical distance that allows a practitioner to make informed and precise choices while selecting a sampler implementation. In this context CubeProbeEst is the first of its kind.

Our Contribution in the Context of Distribution Testing with Subcube Conditional Samples. The crucial component in designing our self-reducible-sampler-tester CubeProbeEst is a novel algorithm for estimating the variation distance in the subcube conditioning model in distribution testing. Given sampling access to an unknown distribution P and a known distribution Q over $\{0,1\}^n$, the distance estimation problem asks to estimate the variation distance between P and Q. The corresponding testing problem is the *tolerant identity testing* of P and Q. Distance estimation and tolerant testing with subcube conditional samples have been open since the introduction of the framework five years ago. The following theorem formalizes our result in the context of distance estimation/tolerant testing using subcube conditional samples.

Theorem 10. *Let P be an unknown distribution and Q be a known distribution defined over $\{0,1\}^n$. Given subcube conditioning access to P, an approximation parameter $\gamma \in (0,1)$ and a confidence parameter $\delta \in (0,1)$, there exists an algorithm that takes $\widetilde{\mathcal{O}}(n^2)$ subcube-conditional samples from P on expectation and outputs an estimate of $\mathsf{d}_{\mathsf{TV}}(P,Q)$ with an additive error ζ with probability at least $1-\delta$.*

This is the first algorithm that solves the variation distance estimation problem in $\widetilde{\mathcal{O}}(n^2)$ subcube conditioning samples.

Related Works

The state-of-the-art approach for efficiently testing CNF samplers was initiated by Meel and Chakraborty (Chakraborty and Meel 2019). They employed the concept of hypothesis testing with conditional samples (Chakraborty et al. 2016; Canonne, Ron, and Servedio 2015) and showed that such samples could be "simulated" in the case of CNF samplers. The approach produced mathematical guarantees on the correctness of their tester. Their idea was extended to design a series of testers for various types of CNF samplers (Barbarik (Chakraborty and Meel 2019) for uniform CNF samplers, Barbarik2 (Meel, Pote, and Chakraborty 2020) for weighted CNF samplers, Teq (Pote and Meel 2021) for testing probabilistic circuits, Flash (Banerjee et al. 2023) for Horn samplers, Barbarik3 (Pote and Meel 2022) for constrained samplers).

The theoretical foundation of our work follows the *subcube conditioning* model of property testing of probability distributions. This model was introduced by (Bhattacharyya and Chakraborty 2018) as a special case of the conditional sampling model (Chakraborty et al. 2016; Canonne, Ron, and Servedio 2015) targeted towards high-dimensional distributions. Almost all the known results in the subcube conditioning framework deal with problems in the non-tolerant regime: testing uniformity, identity, and equivalence of distributions. (Canonne et al. 2021) presented optimal algorithm for (non-tolerant) uniformity testing in this model. (Chen et al. 2021) studied the problem of learning and testing junta distributions. Recently (Mahajan et al. 2023) studied the problem of learning Hidden Markov models. (Blanca et al. 2023) studied identity testing in related coordinate conditional sampling model. (Fotakis, Kalavasis, and Tzamos 2020) studied parameter estimation problem for truncated Boolean product distributions. Recently (Chen and Marcussen 2023) studied the problem of uniformity testing in hypergrids. Very recently, in a concurrent work, the authors in (Kumar, Meel, and Pote 2023) studied the problem of tolerant equivalence testing where both the samplers are unknown and designed an algorithm that takes $\widetilde{\mathcal{O}}(n^3)$ samples.

Estimator of Self-reducible Samplers

Our estimator utilizes the subcube conditional sampling technique. The main program CubeProbeEst works with two subroutines: Est and GBAS. The algorithm GBAS is adopted from the *Gamma Bernoulli Approximation Scheme* (Huber 2014). Since its intricacies are crucial for our algorithm, we include the algorithm here for completeness.

CubeProbeEst: In this algorithm, given a known self-reducible sampler $\mathcal{I}_{\mathcal{W}}$, subcube conditioning access to an unknown self-reducible sampler $\mathcal{I}_{\mathcal{G}}$, along with an input ψ, an approximation parameter ζ and a confidence parameter δ, it estimates the variation distance between $\mathcal{I}_{\mathcal{G}}$ and $\mathcal{I}_{\mathcal{W}}$ with

Algorithm 1: CubeProbeEst $(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}, \psi, \zeta, \delta)$

```
1 $\alpha = \frac{2}{\zeta^2} \log \frac{4}{\delta}$
2 $\gamma = \frac{\zeta}{1.11(2+\zeta)}$
3 $\delta' = \delta/2\alpha$
4 $S = \emptyset$
5 $S \leftarrow \alpha$ iid samples from $\mathcal{I}_{\mathcal{G}}(\psi)$
6 $val = 0$
7 for $x \in S$ do
8     $val = val + \max\left(0, 1 - \frac{\mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}(x)}{\mathsf{Est}(\mathcal{I}_{\mathcal{G}},\psi,n,x,\gamma,\delta')}\right)$
9 return $\frac{val}{\alpha}$
```

Algorithm 2: Est $(\mathcal{I}_{\mathcal{G}}, \psi, n, x, \gamma, \delta')$

```
1 $k \leftarrow \lceil \frac{3n}{\gamma^2} \cdot \log\left(\frac{2n}{\delta'}\right) \rceil$
2 for $i = 1$ to $n$ do
3     $\widehat{\psi} \leftarrow$ SubCond $(\psi, x_1 \ldots x_{i-1})$
4     $\widehat{P}_i \leftarrow$ GBAS$(\mathcal{I}_{\mathcal{G}}, \widehat{\psi}, i, k, x_i)$
5 $\widehat{\mathcal{D}_x^{\mathcal{I}_{\mathcal{G}},\psi}} = \Pi_{i=1}^{n} \widehat{P}_i$
6 return $\widehat{\mathcal{D}_x^{\mathcal{I}_{\mathcal{G}},\psi}}$
```

additive error ζ. CubeProbeEst uses the algorithm Est as a subroutine. It starts by setting several parameters α, γ, δ' in Line 1-Line 3. In Line 4, it initializes an empty multi-set S, and then takes α samples from $\mathcal{I}_{\mathcal{G}}(\psi)$ in S in Line 5. Now it defines a counter val in Line 6, initialized to 0. Now in the for loop starting from Line 7, for every sample $x \in S$ obtained before, CubeProbeEst calls the subroutine Est in Line 8 to estimate the probability mass of $\mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}$ at x. Finally, in Line 9, we output val/α as the estimated variation distance and terminate the algorithm.

Est: Given subcube conditioning access to the unknown self-reducible sampler $\mathcal{I}_{\mathcal{G}}$, an input ψ, the dimension n, an n-bit string x, parameters γ and δ' and an integer t, the subroutine Est returns an estimate of the probability of $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}$ at x by employing the subroutine GBAS . In the for loop starting from Line 2, it first calls SubCond with ψ and $x_1, \ldots, x_{i-1}$ which outputs $\widehat{\psi}$. Now in Line 4 it calls GBAS with $\mathcal{I}_{\mathcal{W}}, \widehat{\psi}, i, k$ along with the i-th bit of x, i.e , x_i with the integer k (to be fixed such that $\delta'/n = 2\exp(-k\gamma^2/3)$) to estimate $\widehat{P}_i$, the empirical weight of $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\widehat{\psi}}$. Now in Line 5, Est computes the empirical weight of $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$ by taking a product of all marginal distributions $\widehat{P}_1, \ldots, \widehat{P}_n$ obtained from the above for loop. Finally in Line 6, Est returns $\widehat{\mathcal{D}_x^{\mathcal{I}_{\mathcal{G}},\psi}}$, the estimated weight of the distribution $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}$ on x.

GBAS: In this algorithm, given access to an unknown self-reducible sampler $\mathcal{I}_{\mathcal{G}}$, input $\widehat{\psi}$, integers i and k, and a bit HEAD, GBAS outputs an estimate $\widehat{p}$ of p. GBAS starts by declaring two variables s and r, initialized to 0 in Line 1. Then in the for loop starting in Line 2, as long as $s < k$, it first takes a sample w from the sampler $\mathcal{I}_{\mathcal{G}}$ on input $\widehat{\psi}$ in

Algorithm 3: GBAS $(\mathcal{I}_{\mathcal{G}}, \widehat{\psi}, i, k, \texttt{HEAD})$

```
1 $s \leftarrow 0$, $r \leftarrow 0$;
2 while $s < k$ do
3     $w \sim \mathcal{I}_{\mathcal{G}}(\widehat{\psi})$;
4     if HEAD $= w_i$ then
5         $s \leftarrow s + 1$ ;
6     $a \sim \mathsf{Exp}(1)$, $r \leftarrow r + a$ ;
7 $\widehat{p} \leftarrow (k-1)/r$ ;
8 return $\widehat{p}$;
```

Line 3. Then in Line 4, it checks if the value of HEAD is w_i where w_i is the i-th bit of the n-bit sample w. If the value of HEAD equals w_i, then in Line 5, it increments the value of s by 1. Then in Line 6, GBAS samples a following $\mathsf{Exp}(1)$, the exponential distribution with parameter 1 and assigns $r + a$ to r. At the end of the for loop in Line 7, it assigns the estimated probability $\widehat{p}$ as $(k-1)/r$. Finally, in Line 8, GBAS returns the estimated probability $\widehat{p}$.

Theoretical Analysis of Our Estimator

The formal result of our estimator is presented below.

Theorem 9. *For an error parameter $\zeta \in (0,1)$, and a constant $\delta < 1/3$,* CubeProbeEst *is (ζ, δ)-approx d_{TV} estimator between a known and unknown self-reducible samplers $\mathcal{I}_{\mathcal{W}}$ and $\mathcal{I}_{\mathcal{G}}$ respectively with sample complexity of $\widetilde{\mathcal{O}}\left(n^2/\zeta^4\right)$.*

The formal proof is presented in the supplementary material.

High-level Technical Overview

The main idea of CubeProbeEst stems from an equivalent characterization of the variation distance which states that $\mathsf{d}_{\mathsf{TV}}^{\psi}(\mathcal{I}_{\mathcal{G}}, \mathcal{I}_{\mathcal{W}}) = \mathbb{E}_{x\sim\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}}(1 - \mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}(x)/\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x))$. Our goal is to estimate the ratio $\mathcal{D}^{\mathcal{I}_{\mathcal{W}},\psi}(x)/\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$ for some samples x-s drawn from $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}$. As $\mathcal{I}_{\mathcal{W}}$ is known, it is sufficient to estimate $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$. It is generally difficult to estimate $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$. However, using self-reducibility of $\mathcal{I}_{\mathcal{G}}$ to mount subcube-conditioning access to $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}$, we estimate $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$ by conditioning over the n conditional marginal distributions of $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}$. Using the chain formula, we obtain the value of $\mathcal{D}^{\mathcal{I}_{\mathcal{G}},\psi}(x)$ by multiplying a number of these conditional probabilities. This is achieved by the subroutine Est . The probability mass estimation of each conditional marginal distribution is achieved by the subroutine GBAS , which is called from Est . The idea of GBAS follows from (Huber 2017), which roughly states that to estimate the probability of head (say p) of a biased coin, within (multiplicative) error γ_i and success probability at least $1-\delta$, it is sufficient to make T coin tosses on average, where $T = k/p$ with $k \geq 3\log(2/\delta)/\gamma_i^2$. The crucial parameter is the error margin γ_i that is used in Est . It should be set so that after taking the errors in all the marginals into account, the total error remains bounded by the target error margin γ. Our pivotal observation is that the error distribution in the subroutine GBAS , when estimating the mass of the conditional marginal distributions, is a SubGaussian distribution (that is, a Gaussian distribution dominates its tails). Following the tail bound on the sum of SubGaussian random variables, we could afford to estimate the mass of each of the marginal with error $\gamma_i = \gamma/\sqrt{n}$ and still get an estimation of $\mathcal{D}^{\mathcal{I}_\mathcal{G},\psi}(x)$ with a correctness error of at most γ. That way the total sample complexity of Est reduces to $\widetilde{\mathcal{O}}(n/(\gamma/\sqrt{n})^2) = \widetilde{\mathcal{O}}(n^2/\gamma^2)$. As $\alpha/\gamma^2 = \mathcal{O}\left(1/\zeta^4\right)$, we get the claimed sample complexity of CubeProbeEst.

From Estimator to Tester

We extend our design to a tester named CubeProbeTester that tests if two samplers are close or far in variation distance. As before, the inputs to CubeProbeTester are two self-reducible samplers $\mathcal{I}_\mathcal{G}, \mathcal{I}_\mathcal{W}$, an input ψ, parameters ε, η, and the confidence parameter δ. CubeProbeTester first computes the estimation margin-of-error ζ as $(\eta - \varepsilon)/2$, and sets an intermediate confidence parameter δ_t as 2δ. The algorithm estimates the distance between $\mathcal{I}_\mathcal{G}$ and $\mathcal{I}_\mathcal{W}$ on input ψ, by invoking CubeProbeEst on $\mathcal{I}_\mathcal{G}, \mathcal{I}_\mathcal{W}, \psi$ along with the estimation-margin ζ and δ_t. If the computed distance $\widehat{\mathsf{dist}}$ is more than the threshold $K = (\eta + \varepsilon)/2$, the tester rejects. Otherwise, the tester accepts.

Algorithm 4: CubeProbeTester $(\mathcal{I}_\mathcal{G}, \mathcal{I}_\mathcal{W}, \psi, \varepsilon, \eta, \delta)$

```
1 $\zeta = (\eta - \varepsilon)/2$
2 $\delta_t = 2\delta$
3 $K = (\eta + \varepsilon)/2$
4 $\widehat{\mathsf{dist}} = \mathsf{CubeProbeEst}(\mathcal{I}_\mathcal{G}, \mathcal{I}_\mathcal{W}, \psi, \zeta, \delta_t)$
5 if $\widehat{\mathsf{dist}} > K$ then
6     return REJECT
7 return ACCEPT
```

The details of CubeProbeTester are summarised below.

Theorem 11. *Consider an unknown self-reducible sampler $\mathcal{I}_\mathcal{G}$, a known self-reducible sampler $\mathcal{I}_\mathcal{W}$, an input ψ, closeness parameter $\varepsilon \in (0,1)$, farness parameter $\eta \in (0,1)$ with $\eta > \varepsilon$ and a confidence parameter $\delta \in (0,1)$. There exists a $(\varepsilon, \eta, \delta)$-Self-reducible-sampler-tester* CubeProbeTester *that takes $\widetilde{\mathcal{O}}\left(n^2/\left(\eta-\varepsilon\right)^4\right)$ samples.*

We note that our tester is general enough that when $\mathcal{I}_\mathcal{G}$ is ε-close to $\mathcal{I}_\mathcal{W}$ in ℓ_∞-distance [1], then CubeProbeTester outputs ACCEPT . Moreover, If CubeProbeTester outputs reject on input ψ, then one can extract a configuration (witness of rejection) ψ_e such that $\mathcal{I}_\mathcal{G}$ and $\mathcal{I}_\mathcal{W}$ are η-far.

Evaluation Results

To evaluate the practical effectiveness of our proposed algorithms, we implemented prototype of CubeProbeEst and CubeProbeTester in Python3 [2]. We use CubeProbeEst to estimate the variation distance ($\mathsf{d_{TV}}$) of three linear extension samplers from a perfect uniform sampler. SAT solvers power the backends of these linear extension samplers. The objective of our empirical evaluation was to answer the following:

RQ1 Can CubeProbeEst estimate the distance of linear extension samplers from a known (e.g., uniform) sampler?

RQ2 How many samples CubeProbeEst requires to estimate the distance?

RQ3 How do the linear extension samplers behave with an increasing number of dimensions?

Boolean encoding of Poset Given a poset $\mathcal{P} = (S, \preceq_P)$, we encode it using a Boolean formula $\varphi_\mathcal{P}$ in conjunctive normal form (CNF), as described in (Talvitie et al. 2018b):

1 for all elements $a, b \in S$, the formula $\varphi_\mathcal{P}$ contains the variables of the form v_{ab} such that $v_{ab} = 1$ represents $a \preceq b$ and $v_{ab} = 0$ represents $b \preceq a$.

2 The CNF formula $\varphi_\mathcal{P}$ contains the following clauses. Type-1: v_{ab} for all $a, b \in S$ such that $a \preceq_\mathcal{P} b$. This enforces the poset relation $\preceq_\mathcal{P}$. Type-2: $\neg v_{ab} \vee \neg v_{bc} \vee v_{ac}$ for all $a, b, c \in S$ to guarantee the transitivity.

This reduction requires ${}^{|S|}C_2$ many variables and ${}^{|S|}P_3$ many clauses of type-2. The number of clauses of type-1 depends on the number of edges in the cover graph of $\mathcal{P}$.

Experimental Setup

Samplers Used: To assess the performance of CubeProbeEst and CubeProbeTester, we utilized three different linear extension samplers- `LxtQuicksampler`, `LxtSTS`, `LxtCMSGen`, to estimate their $\mathsf{d_{TV}}$ distances from a uniform sampler. The backend of these samplers are powered by three state-of-the-art CNF samplers: `QuickSampler` (Dutra et al. 2018), `STS` (Ermon, Gomes, and Selman 2012), `CMSGen` (Golia et al. 2021). A poset-to-CNF encoder precedes these CNF samplers, and a Boolean string-to-poset extractor succeeds the CNF samplers to build the linear extension samplers. We also required access to a known uniform sampler which is equivalent to having access to a linear extension counter[3]. We utilized an exact model counter for CNF formulas to meet this need: `SharpSAT-TD` (Korhonen and Järvisalo 2021).

Poset Instances: We adopted a subset of the poset instances from the experimental setup of (Talvitie et al. 2018a) and (Talvitie et al. 2018b) to evaluate CubeProbeEst and CubeProbeTester. The instances include three different kinds of posets. (a) posets of type $\mathsf{avgdeg_k}$ are generated from DAGs with average indegree of $k = 3, 5$; (b) posets of type $\mathsf{bipartite_p}$ have been generated by from bipartite set $S = A \cup B$ by adding the order constraint $a \prec b$ (resp. $b \prec a$) with probability p (resp. $1 - p$) for all $(a, b) \in A \times B$; (c) posets of type bayesiannetwork is obtained from a transitive closure a randomly sampled subgraph of bayesian networks, obtained from (Elidan 1998).

[1]$\mathcal{I}_\mathcal{G}$ is ε-close to $\mathcal{I}_\mathcal{W}$ on input ψ in ℓ_∞-distance if for every $x \in \{0,1\}^n$, $(1-\varepsilon)\mathcal{D}^{\mathcal{I}_\mathcal{W},\psi}(x) \leq \mathcal{D}^{\mathcal{I}_\mathcal{G},\psi}(x) \leq (1+\varepsilon)\mathcal{D}^{\mathcal{I}_\mathcal{W},\psi}(x)$.

[2]Codes and experimental results are available at www.github.com/uddaloksarkar/cubeprobe.

[3]For a set $\mathcal{S}$ if we know the size of the set $|\mathcal{S}|$, we know the mass of each element to be $1/|\mathcal{S}|$ in a uniform sampler.

		`LxtQuickSampler`			`LxtSTS`			`LxtCMSGen`		
Instances	dim	Estd d_{TV}	#samples	A/R	Estd d_{TV}	#samples	A/R	Estd d_{TV}	#samples	A/R
avgdeg_3_008_2	19	0.1854	9986426	A	0.0205	11013078	A	0.1772	9914721	A
avgdeg_3_010_2	30	0.1551	24537279	A	0.0155	24758147	A	0.1267	24126731	A
avgdeg_5_010_3	16	0.0976	7593533	A	0.0338	7338508	A	0.1135	7261255	A
avgdeg_5_010_4	11	0.0503	3486025	A	0.0387	3475635	A	0.1147	3412151	A
bn_andes_010_1	35	0.2742	33557190	A	0.0396	33536595	A	0.1601	33235104	A
bn_diabetes_010_3	26	0.1955	19211200	A	0.0009	18847561	A	0.1478	18539480	A
bn_link_010_4	28	0.2024	21482230	A	0.0346	22377750	A	0.1635	21161624	A
bn_munin_010_1	33	0.2414	30348931	A	0.0448	30693619	A	0.1230	30218998	A
bn_pigs_010_1	36	0.3106	36917129	R	0.0569	36311963	A	0.1353	35978964	A
bipartite_0.2_008_4	25	0.3204	17761820	R	0.0073	17840945	A	0.1153	17546682	A
bipartite_0.2_010_1	41	0.3299	46244946	R	0.1528	48135745	A	0.1461	47003971	A
bipartite_0.5_008_4	22	0.2977	13144132	A	0.0528	13424946	A	0.1059	13317859	A
bipartite_0.5_010_1	36	0.3082	35875122	R	0.0037	36728064	A	0.1472	35823878	A

Table 1: For each sampler the three columns represent the estimated d_{TV}, number of samples consumed by CubeProbeEst and the output of CubeProbeTester. "A" and "R" represent ACCEPT and REJECT respectively.

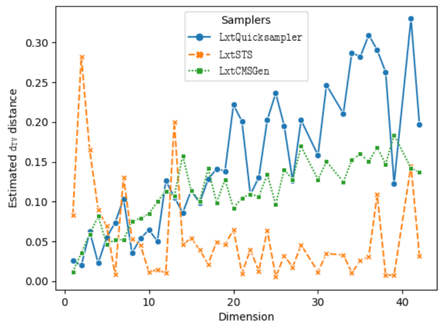


Figure 2: TV distances of samplers from uniformity are estimated across dimensions. For each dimension, we take the median d_{TV} over all the instances of that dimension.

Parameters Initialization: For our experiments with CubeProbeEst, the approximation parameter ζ and confidence parameter δ are set to be 0.3 and 0.2. Our tester CubeProbeTester takes a closeness parameter ε, farness parameter η, and confidence parameter δ. For our experiments these are set to be $\varepsilon : 0.01$, $\eta : 0.61$, and $\delta : 0.1$, respectively.

Environment All experiments are carried out on a high-performance computer cluster, where each node consists of AMD EPYC 7713 CPUs with 2x64 cores and 512 GB memory. All tests were run in multi-threaded mode with 8 threads per instance per sampler with a timeout of 12 hrs.

Experimental Results & Discussion

RQ1 Table 1 shows a subset of our experimental results. Due to space constraints, we have postponed presenting our comprehensive experimental results to the supplementary material. We found that among 90 instances:

- `LxtQuickSampler` has the maximum d_{TV} from uniformity in 48 instances, `LxtSTS` in 14 instances, and `LxtCMSGen` in 28 instances;
- `LxtQuickSampler` has the minimum d_{TV} from uniformity in 10 instances, `LxtSTS` in 69 instances, and `LxtCMSGen` in 11 instances;

These observations indicate that `LxtSTS` serves as a linear extension sampler that closely resembles uniform distribution characteristics. At the same time, `LxtQuickSampler` deviates significantly from the traits of a uniform-like linear extension sampler. `LxtCMSGen` falls in an intermediate position between these two.

RQ2 Table 1 reflects that the number of samples drawn by CubeProbeEst depends on the dimension of an instance. Again, when the dimension is kept constant, the number of samples drawn remains similar across all runs.

RQ3 In Figure 2, at lower dimensions, both `LxtQuickSampler` and `LxtCMSGen` behave relatively close to uniform sampling. However, as the dimension increases, d_{TV} between these two samplers from uniformity increases. In contrast, `LxtSTS` shows a different behavior. In lower dimensions, the estimated d_{TV} distance can be notably high for certain instances, but it tends to stabilize with increasing dimension. It is worth highlighting that, in higher dimensions, `LxtSTS` demonstrates a more uniform-like sampling behavior compared to the other two samplers.

Conclusion

In this paper, we have designed the first self-reducible sampler tester, and used it to test linear extension samplers. We have also designed a novel variation distance estimator in the subcube-conditioning model along the way.

Limitations of our work Our algorithm takes $\widetilde{\mathcal{O}}(n^2)$ samples while the known lower bound for tolerant testing with subcube conditioning is of $\Omega(n/\log n)$ for this task (Canonne et al. 2020). Moreover, our algorithm works when the samplers are self-reducible, which is required for our analysis. So our algorithm can not handle non-self-reducible samplers, such as in (Große, Rothe, and Wechsung 2006; Talvitie, Vuoksenmaa, and Koivisto 2020).

Acknowledgements

Rishiraj Bhattacharyya acknowledges the support of UKRI by EPSRC grant number EP/Y001680/1. Uddalok Sarkar is supported by the Google PhD Fellowship. Sayantan Sen's research is supported by the National Research Foundation Singapore under its NRF Fellowship Programme (NRF-NRFFAI1-2019-0002). This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. The computational works of this article were performed on the resources of the National Supercomputing Centre, Singapore www.nscc.sg.

References

Andrieu, C.; De Freitas, N.; Doucet, A.; and Jordan, M. I. 2003. An introduction to MCMC for machine learning. *Machine Learning*.

Ashur, T.; De Witte, G.; and Liu, Y. 2017. An automated tool for rotational-xor cryptanalysis of arx-based primitives. In *SITB*.

Banerjee, A.; Chakraborty, S.; Chakraborty, S.; Meel, K. S.; Sarkar, U.; and Sen, S. 2023. Testing of Horn Samplers. In *AISTATS*.

Bhattacharyya, R.; and Chakraborty, S. 2018. Property testing of joint distributions using conditional samples. *ACM Transactions on Computation Theory (TOCT)*.

Blanca, A.; Chen, Z.; Štefankovič, D.; and Vigoda, E. 2023. Complexity of High-Dimensional Identity Testing with Coordinate Conditional Sampling. In *COLT*.

Brooks, S.; Gelman, A.; Jones, G.; and Meng, X.-L. 2011. *Handbook of markov chain monte carlo*. Cambridge University Press.

Canonne, C. L.; Chen, X.; Kamath, G.; Levi, A.; and Waingarten, E. 2021. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In *SODA*.

Canonne, C. L.; Diakonikolas, I.; Kane, D. M.; and Stewart, A. 2020. Testing bayesian networks. *IEEE Transactions on Information Theory*.

Canonne, C. L.; Ron, D.; and Servedio, R. A. 2015. Testing probability distributions using conditional samples. *SIAM Journal on Computing*.

Chakraborty, S.; Fischer, E.; Goldhirsh, Y.; and Matsliah, A. 2016. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*.

Chakraborty, S.; Fremont, D.; Meel, K.; Seshia, S.; and Vardi, M. 2014. Distribution-aware sampling and weighted model counting for SAT. In *AAAI*.

Chakraborty, S.; and Meel, K. S. 2019. On testing of uniform samplers. In *AAAI*.

Chakraborty, S.; Meel, K. S.; and Vardi, M. Y. 2013. A scalable and nearly uniform generator of SAT witnesses. In *ICCAD*.

Chandra, A. K.; and Iyengar, V. S. 1992. Constraint solving for test case generation: a technique for high-level design verification. In *ICCD*.

Chen, X.; Jayaram, R.; Levi, A.; and Waingarten, E. 2021. Learning and testing junta distributions with sub cube conditioning. In *COLT*.

Chen, X.; and Marcussen, C. 2023. Uniformity Testing over Hypergrids with Subcube Conditioning.

Diakonikolas, I.; Lee, H. K.; Matulef, K.; Onak, K.; Rubinfeld, R.; Servedio, R. A.; and Wan, A. 2007. Testing for concise representations. In *FOCS*.

Dutra, R.; Laeufer, K.; Bachrach, J.; and Sen, K. 2018. Efficient sampling of SAT solutions for testing. In *ICSE*.

Elidan, G. 1998. Bayesian-Network-Repository. cs.huji.ac.il/w-galel/Repository/.

Ermon, S.; Gomes, C. P.; Sabharwal, A.; and Selman, B. 2013. Embed and project: Discrete sampling with universal hashing. *NeurIPS*.

Ermon, S.; Gomes, C. P.; and Selman, B. 2012. Uniform Solution Sampling Using a Constraint Solver As an Oracle. In *UAI*.

Fotakis, D.; Kalavasis, A.; and Tzamos, C. 2020. Efficient parameter estimation of truncated boolean product distributions. In *COLT*.

Golia, P.; Soos, M.; Chakraborty, S.; and Meel, K. S. 2021. Designing samplers is easy: The boon of testers. In *FMCAD*.

Gopalan, P.; O'Donnell, R.; Servedio, R. A.; Shpilka, A.; and Wimmer, K. 2009. Testing Fourier Dimensionality and Sparsity. In *ICALP*.

Große, A.; Rothe, J.; and Wechsung, G. 2006. On computing the smallest four-coloring of planar graphs and non-self-reducible sets in P. *Information Processing Letters*.

Huber, M. 2014. Near-linear time simulation of linear extensions of a height-2 poset with bounded interaction. *Chicago Journal of Theoretical Computer Science*.

Huber, M. 2017. A Bernoulli mean estimate with known relative error distribution. *Random Struct. Algorithms*.

Jerrum, M. 1998. Mathematical foundations of the Markov chain Monte Carlo method. In *Probabilistic methods for algorithmic discrete mathematics*.

Jerrum, M. R.; Valiant, L. G.; and Vazirani, V. V. 1986. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*.

Khuller, S.; and Vazirani, V. V. 1991. Planar graph coloring is not self-reducible, assuming P$\neq$ NP. *Theoretical Computer Science*.

Korhonen, T.; and Järvisalo, M. 2021. SharpSAT-TD Participating in Model Counting Competition 2021.

Kumar, G.; Meel, K. S.; and Pote, Y. 2023. Tolerant Testing of High-Dimensional Samplers with Subcube Conditioning. arXiv:2308.04264.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2014. Probabilistic preference logic networks. In *ECAI*.

Mahajan, G.; Kakade, S.; Krishnamurthy, A.; and Zhang, C. 2023. Learning Hidden Markov Models Using Conditional Samples. In *COLT*.

Mannila, H.; and Meek, C. 2000. Global partial orders from sequential data. In *SIGKDD*.

Meel, K. S.; Pote, Y. P.; and Chakraborty, S. 2020. On testing of samplers. *NeurIPS*.

Meel, K. S.; Vardi, M. Y.; Chakraborty, S.; Fremont, D. J.; Seshia, S. A.; Fried, D.; Ivrii, A.; and Malik, S. 2016. Constrained Sampling and Counting: Universal Hashing Meets SAT Solving. In *Beyond NP, Papers from the 2016 AAAI Workshop*, AAAI.

Mironov, I.; and Zhang, L. 2006. Applications of SAT solvers to cryptanalysis of hash functions. In *SAT*.

Morawiecki, P.; and Srebrny, M. 2013. A SAT-based preimage analysis of reduced Keccak hash functions. *Information Processing Letters*.

Morton, J.; Pachter, L.; Shiu, A.; Sturmfels, B.; and Wienand, O. 2009. Convex rank tests and semigraphoids. *SIAM Journal on Discrete Mathematics*.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research*.

Naveh, Y.; Rimon, M.; Jaeger, I.; Katz, Y.; Vinov, M.; s Marcu, E.; and Shurek, G. 2006. Constraint-based random stimuli generation for hardware verification.

Paninski, L. 2008. A Coincidence-Based Test for Uniformity Given Very Sparsely Sampled Discrete Data. *IEEE Transactions on Information Theory*.

Peczarski, M. 2004. New results in minimum-comparison sorting. *Algorithmica*.

Pote, Y.; and Meel, K. S. 2022. On Scalable Testing of Samplers. *NeurIPS*.

Pote, Y. P.; and Meel, K. S. 2021. Testing probabilistic circuits. *NeurIPS*.

Servedio, R. A. 2010. Testing by implicit learning: a brief survey. *Property Testing*.

Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending SAT solvers to cryptographic problems. In *SAT*.

Talvitie, T.; Kangas, J.-K.; Niinimäki, T.; and Koivisto, M. 2018a. A scalable scheme for counting linear extensions. In *IJCAI*.

Talvitie, T.; Kangas, K.; Niinimäki, T.; and Koivisto, M. 2018b. Counting linear extensions in practice: MCMC versus exponential Monte Carlo. In *AAAI*.

Talvitie, T.; Vuoksenmaa, A.; and Koivisto, M. 2020. Exact sampling of directed acyclic graphs from modular distributions. In *UAI*.

Wallace, C.; Korb, K. B.; and Dai, H. 1996. Causal discovery via MML. In *ICML*.

Yuan, J.; Aziz, A.; Pixley, C.; and Albin, K. 2004. Simplifying boolean constraint solving for random simulation-vector generation. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*