Instance Dependent Testing of Samplers Using Interval Conditioning

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

[1]University of Birmingham
[2]Indian Statistical Institute Kolkata
[3]National University of Singapore
[4]Centre for Quantum Technologies, National University of Singapore

Abstract

Sampling algorithms play a pivotal role in probabilistic AI. However, verifying if a sampler program indeed samples from the claimed distribution is a notoriously hard problem. Provably correct testers like Barbarik, Teq, Flash, CubeProbe for testing of different kinds of samplers were proposed only in the last few years. All these testers focus on the worst-case efficiency, and do not support verification of samplers over infinite domains, a case occurring frequently in Astronomy, Finance, Network Security, etc.

In this work, we design the first tester of samplers with instance-dependent efficiency, allowing us to test samplers over natural numbers. Our tests are developed via a novel distance estimation algorithm between an unknown and a known probability distribution using an *interval conditioning* framework. The core technical contribution is a new connection with probability mass estimation of a continuous distribution. The practical gains are also substantial—our experiments establish up to 1000× speedup over state-of-the-art testers.

Extended version — https://arxiv.org/abs/2512.06458
Code — https://github.com/uddaloksarkar/lachesis

1 Introduction

Sampling according to a distribution is ubiquitous in probabilistic AI. Its applications in efficient and accurate information extraction and decision making are crucial for advancements of data analytics and machine learning (Yuan et al. 2004; Naveh et al. 2006; Mironov and Zhang 2006; Morawiecki and Srebrny 2013). However, practical samplers are often based on heuristic techniques and lack formal guarantees, posing risks to transparency, reproducibility, security, and safety. As a result, verifying and certifying samplers efficiently has become a key challenge in building robust and trustworthy AI systems.

In the past seven years, several efficient testing algorithms have been developed (Chakraborty and Meel 2019; Meel, Pote, and Chakraborty 2020; Pote and Meel 2021, 2022; Banerjee et al. 2023; Meel, Kumar, and Pote 2025) that are not only practically implementable but also come with formal correctness guarantees. These approaches build on algorithmic distribution testing (Goldreich and Ron 1997; Batu et al. 2001), where sampler quality is assessed by measuring the distance between the target distribution and the actual distribution it produces. The central task is to determine whether an unknown distribution P—accessed via sampling—is *close* to or *far* from a (potentially known) distribution Q. The efficiency of these tests depends on the number of samples drawn from P.

Traditional distribution testing relies on i.i.d. samples from the unknown distribution (Goldreich 2017; Canonne 2020, 2022). However, this approach often requires a number of samples polynomial in the support size, rendering it impractical for large domains. To overcome this limitation, the *conditional sampling* model was introduced (Chakraborty et al. 2016; Canonne, Ron, and Servedio 2015), where an algorithm can query an oracle with a subset S and receive samples from the conditional distribution $P_{|S}$. This model is provably more powerful: for instance, it enables super-exponential reductions in sample complexity for testing distribution closeness. Despite its theoretical appeal, it was not immediately apparent how and in which contexts this more powerful sampling model could be implemented.

Recognizing the power of conditional sampling led to the development of several of its variants tailored to different contexts, typically by restricting the structure of the condition set S; for example, *sub-cube conditions* (Bhattacharyya and Chakraborty 2018), and *pair* or *interval conditions* (Canonne, Ron, and Servedio 2015). These models enable practical implementations across diverse settings and have been used to design efficient verifiers for various samplers. For instance, standard conditional sampling has been applied to test CNF-samplers (Chakraborty and Meel 2019; Meel, Pote, and Chakraborty 2020; Banerjee et al. 2023), while sub-cube conditioning has enabled testers for self-reducible samplers (Bhattacharyya et al. 2024; Meel, Kumar, and Pote 2025). Overall, conditional sampling has played a key role in enabling practical and efficient testing, while also motivating further study of sample complexity across different models.

In this paper, we focus on the *interval conditional sampling* model, where distributions P and Q are defined over $\mathbb{Z}$, and samples can be drawn from the conditional distribution $P_{|S}$ for any interval $S = [\![a, b]\!]$ with $a, b \in \mathbb{R}$. This model has been extensively studied, and the current state-of-the-art for estimating the *total variation* distance

between an unknown and a known distribution over domain $[n]$ achieves a sample complexity of $\widetilde{\mathcal{O}}(\log^2 n)$ (Bhattacharyya et al. 2024), while the best lower bound is $\Omega(\log n/\log\log n)$ (Canonne, Ron, and Servedio 2015).

However, existing algorithms and their sample complexity bounds are largely oblivious to the finer structure of the distributions that are being tested. Most prior works focus on optimizing the *worst-case* sample complexity—i.e., over all possible distributions on $[n]$—with complexity typically measured in terms of n. This worst-case perspective has notable limitations. First, it does not extend to distributions over infinite discrete domains (e.g., $\mathbb{Z}$ or $\mathbb{N}$). Second, it misses opportunities for efficiency gains by ignoring the structural properties of the distributions P and Q.

In practice, many commonly encountered distributions—such as uniform, geometric, binomial, Poisson, or normal—exhibit smoothness properties (akin to Lipschitz continuity). This raises natural questions: *Do we need to pay the worst-case sample complexity even when testing samplers over such smooth distributions? Can we design efficient verifiers for samplers over discrete, infinite domains? And, can the sample complexity be expressed in terms of a distribution's smoothness?*

We answer all these questions in the *affirmative*. We propose a new approach for estimating the total variation distance between the output of a sampler and a target distribution using interval conditional sampling. The sample complexity of our algorithm depends on a formal notion of smoothness of the distributions, allowing significantly improved performance when the known distribution is smooth. In particular, for a wide class of smooth distributions (including uniform, Gaussian, binomial, Poisson, and geometric), our algorithm achieves sample complexity significantly better than prior worst-case bounds of $\widetilde{\mathcal{O}}(\log^2 n)$.

We introduce a novel algorithmic framework that imports techniques from the analysis of continuous distributions into the setting of discrete distribution testing under interval conditional sampling. This connection, previously unexplored, enables new algorithmic strategies for the discrete regime by systematically leveraging tools originally developed for continuous domains.

Our approach leads to several significant implications. First, it enables the design of an efficient tester for samplers over discrete infinite domains—an advancement we believe to be the first of its kind. Second, it introduces an instance-dependent framework for distance estimation in the interval conditional sampling model, where the sample complexity adapts to the structural properties of the distributions involved. This marks a shift from traditional worst-case analysis toward a more refined, distribution-aware perspective. Notably, this viewpoint aligns with the growing interest in instance-optimal algorithms, where the goal is to design algorithms whose performance matches the inherent difficulty of each individual instance (Valiant and Valiant 2017; Hao and Orlitsky 2020; Feldman et al. 2024; Diakonikolas and Kane 2016; Narayanan et al. 2024; Le et al. 2024).

We provide an implementation of our algorithms and introduce a practical testing framework for a broad class of samplers known as *inverse transform samplers*. Our results demonstrate how interval conditional sampling can be effectively applied in this context. The practical gains are substantial—our method achieves up to 1000× speedup over state-of-the-art worst-case algorithms (Bhattacharyya et al. 2024) (see Figure 1).

Organization The necessary preliminaries are in Section 2, followed by our contributions in Section 3. The simulation of continuous interval conditioning is in Section 4, and our algorithms are described in Section 5. Our experimental findings are in Section 6. Due to space constraints, all proofs of theorems, lemmas, and claims are provided in the supplementary material.

2 Preliminaries

For a positive real number $a \in \mathbb{R}^+$, $\mathsf{rnd}(a)$ denotes the nearest integer of a. $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For concise expressions and readability, we use the asymptotic complexity notion of $\widetilde{\mathcal{O}}(\cdot)$, where we hide polylogarithmic dependencies of the parameters. For integers $a, b \in [n]$, we use the standard bracket notation $[a, b]$ to denote the discrete interval $\{a, a+1, \ldots, b\}$, and for real numbers $u, v \in \mathbb{R}$, we denote the closed interval by $[\![u, v]\!]$. In this paper, we will deal with discrete distributions defined on a sample space $[n]$ (with $n \geq 2$) and continuous distributions defined on the real line $(-\infty, \infty)$. The definition of discrete distributions naturally extends for $\mathbb{Z}$ such that $P(x) = 0$ for all $x \notin [n]$. For any $S \subseteq [n]$ or $S \subseteq \mathbb{R}$, we denote the probability mass of the set S under a distribution P by $P(S)$. For example, for integers $a \leq b$ write $P([a, b]) = \sum_{t=a}^{b} P(t)$. The Poisson distribution with parameter λ is denoted by Poi^{λ}. We will use the Triangular distribution in this paper: Triangular distribution Tri is defined for $x \in [-\frac{1}{2}, \frac{1}{2}]$ as

$$\mathsf{Tri}(x) = \begin{cases} 4x + 2 & -\frac{1}{2} \leq x \leq 0 \\ -4x + 2 & 0 \leq x \leq \frac{1}{2} \end{cases}$$

We define the *convolution* operation of a discrete distribution P with the continuous distribution Tri as follows:

$$P^{\mathsf{Tri}}(x) = \sum_{t \in [n]} P(t)\mathsf{Tri}(x - t)$$

Notably, P^{Tri} is a continuous distribution.

Definition 1 (ℓ_∞ and $\mathsf{d_{TV}}$-distance)**.** For distributions P and Q over $[n]$ the multiplicative distance is defined as

$$\ell_\infty(P, Q) = \max_{x \in [n]} |P(x)/Q(x) - 1|$$

The $\mathsf{d_{TV}}$-distance between P and Q is defined as

$$\mathsf{d_{TV}}(P, Q) = \frac{1}{2} \sum_{x \in [n]} |P(x) - Q(x)|$$

Definition 2 (tilt)**.** Let P be a distribution supported on $[n]$. Then for every $x \in [n]$ such that $P(x) > 0$, we define:

$$\mathsf{tilt}_P(x) = \max\left\{ \max_{y<x} \frac{P(y)}{P([y+1, x])}, \max_{y>x} \frac{P(y)}{P([x, y-1])} \right\}$$

This captures the maximum ratio between the probability mass at any point y and the total probability mass in the interval between x and y (excluding y).

Access to Distributions To sample from a distribution P is to draw an element $x \in [n]$ with probability $P(x)$. For a known distribution, the probability value $P(x)$ can be queried in constant time for any $x \in [n]$. In contrast, an unknown distribution allows only sample-based access, that is, drawing independent samples according to P. We also define sampling from a *mixture distribution* of P and Q, denoted by $P \diamond Q$ and defined as $\frac{1}{2}(P+Q)$, that is, drawing a sample from P with probability $\frac{1}{2}$ and from Q with probability $\frac{1}{2}$. In this work, we assume a stronger form of access to the unknown distribution via an oracle, called the *interval conditioning oracle*.

Definition 3 (Interval Conditioning)**.** Given a distribution P, the ICOND oracle (short for 'interval conditioning') takes an interval $[i,j] \subseteq [n]$ and returns a sample drawn from P conditioned on $x \in [i,j]$, i.e., $x \sim P_{|[i,j]}$. We extend this to the continuous setting with the $\mathsf{ICOND}^{\mathsf{Cont}}$ oracle: given a distribution μ over $\mathbb{R}$ and an interval $[\![u,v]\!] \subseteq \mathbb{R}$, it returns a sample from μ conditioned on $x \in [\![u,v]\!]$, i.e., $x \sim \mu_{|[\![u,v]\!]}$.

2.1 Tootsie Pop Algorithm

The Tootsie Pop Algorithm (TPA) (Banks et al. 2018), estimates the probability mass of a target interval within the domain of a continuous distribution. TPA works by iteratively refining the interval until it reaches the target interval. We describe the Tootsie Pop Algorithm with a slight modification in Algorithm 1, adapted to operate using the $\mathsf{ICOND}^{\mathsf{Cont}}$ oracle. It takes as input a continuous distribution P^{Tri}, an element x, number of iterations r, and parameters Thresh and δ which are related to the ICOND oracle. The algorithm returns an estimate of the probability mass of the interval $[\![x-\frac{1}{2}, x+\frac{1}{2}]\!]$, or $\perp$ if it fails to estimate it within the given parameters. Within each iteration, the algorithm starts with the initial interval $[\![-n, 2n]\!]$ (noting that P^{Tri} is supported only within $[\![0,n]\!]$) and refines to the target interval $[\![x-\frac{1}{2}, x+\frac{1}{2}]\!]$ by sampling from the distribution P^{Tri} conditioned on the current interval. It counts the number of steps (λ) taken to reach the target interval, and returns the average number of steps taken across all iterations. If the number of steps exceeds a threshold Thresh at any iteration or if the sampling fails, it returns $\perp$. (Banks et al. 2018, Lemma 2) has shown that the stopping time λ follows a suitable Poisson distribution, which establishes the correctness of the algorithm TPA, that is, $\lambda \sim \mathsf{Poi}^{\mu}$, where $\mu = -\log P^{\mathsf{Tri}}([\![x-\frac{1}{2}, x+\frac{1}{2}]\!])$.

2.2 Inverse Transform Samplers

Inverse transform sampling is a powerful and widely used method for sampling from complex distributions. The core idea is to draw samples from a simpler distribution, typically uniform, and then apply a series of transformations to produce samples from the target distribution. This approach is especially effective when direct sampling from the target distribution is challenging—as in the case of Binomial (Hörmann 1993), Poisson (Hörman and Derflinger 1994), or Geometric distributions (Fishman 2001). In fact, standard libraries commonly implement samplers for these distributions using this paradigm.

Algorithm 1: $\mathsf{TPA}(P^{\mathsf{Tri}}, x, r, \mathsf{Thresh}, \delta, \theta)$

$k \leftarrow 0$
for $i = 1$ to r **do**
  $\lambda \leftarrow 0, \beta \leftarrow n$
  while $\beta > \frac{1}{2}$ **do**
    $y \leftarrow \mathsf{ICOND}^{\mathsf{Cont}}\left(P^{\mathsf{Tri}}, x-\beta, x+\beta, \frac{\delta}{r\mathsf{Thresh}}, \theta\right)$
    if $y = \perp$ or $\lambda \geq \mathsf{Thresh}$ **then**
      Return $\perp$
    $\lambda \leftarrow \lambda + 1, \beta \leftarrow |y - x|$
  $k \leftarrow k + (\lambda - 1)$
$\lambda \leftarrow \frac{k}{r}$
Return λ

Listing 1: An inverse transform sampler for the geometric distribution.

```
def Geometric(p):
    U = uniform(0,1)
    return ceil(log(1 - U) / log(1 - p))
```

When the inverse CDF of the target distribution is tractable (e.g., for the Geometric distribution; see Listing 1), direct inverse transform sampling suffices. For distributions with intractable inverse CDFs, such as Binomial or Poisson, a common workaround involves using a *hat distribution* h that upper-bounds the target distribution Q, i.e., there exists a constant $C > 0$ such that $Q(x) \leq C\mathsf{h}(x)$ for all $x \in \Omega$, and for which the CDF is easy to compute. Samples are then drawn from h using: (1) inverse transform sampling, by drawing u from uniform distribution over $[\![0,1]\!]$ and computing $x = \mathsf{H}^{-1}(u)$, followed by (2) rejection sampling, where x is accepted with probability proportional to $\frac{Q(x)}{C\mathsf{h}(x)}$.

3 Our Contributions

Our main contributions are two algorithms toltest and ERtoltest, the first set of algorithms for identity testing of distributions in the interval conditioning model with instance-dependent bounds. We also developed the first practical realization of the interval conditioning oracle to test inverse transform samplers. Our technical contributions are summarized in the following theorem.

Theorem 4. *Let P be an unknown distribution and Q a known distribution over $\mathbb{Z}$. Given access to $\mathsf{ICOND}(P)$, accuracy parameters $\varepsilon, \eta \in (0,1)$ with $\eta > \varepsilon$, and confidence parameter $\delta \in (0,1)$, the algorithms* toltest *and* ERtoltest *satisfy the following:*

- toltest *can distinguish between the cases $\mathsf{d_{TV}}(P,Q) \leq \varepsilon$ and $\mathsf{d_{TV}}(P,Q) \geq \eta$ with probability $\geq 1-\delta$.*
  The expected number of ICOND *queries is*

$$\widetilde{\mathcal{O}}\left(\frac{1}{(\eta-\varepsilon)^4} \mathop{\mathbb{E}}_{x\sim P\diamond Q}\left[\mathsf{tilt}_{P\diamond Q}(x) \cdot \log^2 \frac{1}{P\diamond Q(x)}\right]\right).$$

  where $P \diamond Q$ is the distribution defined by $P \diamond Q(x) = (P(x)+Q(x))/2$.
- ERtoltest *can distinguish between the cases $\ell_\infty(P,Q) \leq 2\varepsilon$ and $\mathsf{d_{TV}}(P,Q) \geq \eta$ with probability $\geq 1-\delta$. The*

expected number of oracle queries made by the ERtoltest *is at most*

$$\widetilde{\mathcal{O}}\left(\frac{1}{(\eta-\varepsilon)^4}\,\mathop{\mathbb{E}}_{x\sim P}\left[\mathsf{tilt}_Q(x)\cdot K(x)\right]\right)$$

Where $K(x)=\min\left(\log^2\frac{1}{P(x)},\log^2\frac{1}{Q(x)}\right)$.

Both our algorithms use the Tootsie Pop Algorithm (TPA). However, TPA crucially needs ICOND to a continuous distribution, whereas the distribution P under test is a discrete distribution. To address this, we consider the convolution of P with a continuous distribution. We needed a distribution to convolve with such that we can simulate drawing samples (or interval conditional samples) from the convolved distribution using ICOND on P. For this, we chose the triangular distribution and work with the distribution P^{Tri}. Finally, we employ rejection sampling in (Algorithm 2) to simulate $\mathsf{ICOND}^{\mathsf{Cont}}$ on P^{Tri} using ICOND on P. This simulation is presented in Section 4.

The crucial subroutine that we need for both our algorithms toltest and ERtoltest is Est, which is used to estimate the value of $P(x)$ for any $x\in\mathbb{Z}$. This subroutine uses the TPA using the $\mathsf{ICOND}^{\mathsf{Cont}}$ oracles to P^{Tri}. The expected runtime of Est depends on the smoothness (tilt) of P, and this is where we observe that our algorithms are instance-dependent, that is, the runtime varies with the quality of the unknown distribution P. Finally, using Est, we describe our algorithms. The algorithms Est, toltest and ERtoltest are described in Section 5.

Experimental Results We extended our algorithms to build the first practical tester for the broader class of samplers—namely, inverse transform samplers. The key technical challenge in this direction is implementing the interval conditioning oracle for such samplers. In this work, we designed the first concrete implementation of such an ICOND oracle, specifically tailored to operate with inverse transform samplers. This implementation is non-trivial and involves formal guarantees to ensure correctness. The design and methodology of this oracle are described in detail in Section 6. We integrate this oracle to construct our practical tester Lachesis. Empirical results show that Lachesis significantly outperforms existing methods for testing samplers, offering substantial improvements in efficiency.

Technical Novelty FROM CONTINUOUS TO DISCRETE TESTING: The core of our technique is in efficient probability mass estimation exploiting the *structure* of the distributions. The main technical novelty is in adapting the Tootsie Pop Algorithm (TPA), which is used to approximate the integral of nonnegative functions over high-dimensional continuous spaces (Huber and Schott 2010). We crucially observe that TPA could be employed to estimate the probability mass of a subset of the domain if one could establish interval conditioning over the reals using the oracle that allows interval conditioning samples over $\mathbb{Z}$. To achieve this, we innovate a technique for achieving interval conditioning over reals via extended distribution and rejection sampling. The efficiency of this transformation depends on the "smoothness" of the distributions.

Notable Features of our algorithms There are a couple of notable and novel features of our algorithms that we would like to highlight:

TESTING DISTRIBUTIONS OVER INFINITE SUPPORT: The sample complexities of our algorithms are independent of the domain size. This allows our algorithms to test distributions over infinite support, e.g., $\mathbb{Z}$, often with very few queries, for a large class of distributions, for example, *log-concave distributions* (Bagnoli and Bergstrom 2005; Lovász and Vempala 2007). For example, consider the case when Q is a Poisson distribution with parameter $\lambda>0$. In such a scenario, our algorithm ERtoltest can efficiently distinguish whether an unknown distribution P is close to or far from Q, with an expected number of queries $\widetilde{\mathcal{O}}\left(\log^2\lambda\right)$.

TESTING DISTRIBUTIONS OVER SMALL SUPPORT: Many times, the sampler's output is concentrated over a small set, following the target distribution perfectly over that set. Consider, for example, the case when P is *uniform* over a subset $S\subset[n]$, such that, $|S|\ll n$. The best known algorithm, CubeProbe (Bhattacharyya et al. 2024), requires $\widetilde{\mathcal{O}}\left(\frac{\log^2 n}{(\eta-\varepsilon)^4}\right)$ queries to the interval conditioning oracle to distinguish between $\ell_\infty(P,U)\le 2\varepsilon$ and $\mathsf{d_{TV}}(P,U)\ge\eta$, where U denotes the uniform distribution over $[n]$. The following corollary of Theorem 4 provides a better result.

Corollary 5. *Let P be an unknown distribution, promised to be supported on a subset $S\subseteq[n]$, and let U denote the uniform distribution over $[n]$. Our algorithm* ERtoltest *can distinguish between $\ell_\infty(P,U)\le 2\varepsilon$ and $\mathsf{d_{TV}}(P,U)\ge\eta$, with at most $\widetilde{\mathcal{O}}\left(\frac{\log^2|S|}{(\eta-\varepsilon)^4}\right)$ queries to the interval conditioning oracle in expectation.*

4 Simulating $\mathsf{ICOND}^{\mathsf{Cont}}$ from ICOND

We will use the Tootsie Pop algorithm (TPA, Algorithm 1), which works in the context of continuous distributions. However, we only have access to the ICOND oracle for an unknown discrete distribution P. In this section, we will devise a method of simulating conditional samples from a continuous distribution with ICOND query access to P, in particular, we will introduce an algorithm that simulates the $\mathsf{ICOND}^{\mathsf{Cont}}$ oracle for a continuous distribution P^{Tri}.

Throughout this section, we assume access to a sampling procedure for the triangular distribution Tri, an assumption justified by well-established sampling methods for triangular distributions (Kotz and Van Dorp 2004). In Algorithm 2, we describe the construction of the continuous interval conditioning oracle $\mathsf{ICOND}^{\mathsf{Cont}}$, using the basic ICOND oracle and sampling access to Tri, implemented via a rejection sampling–based approach.

Given ICOND access to the unknown distribution P, interval $[\![u,v]\!]$, the confidence parameter δ and a stopping parameter θ, $\mathsf{ICOND}^{\mathsf{Cont}}$ first set the maximum number of attempts to $T=(2\theta+1)\log\frac{1}{\delta}$. Then it obtains a sample x from ICOND oracle conditioned on the interval $[\mathsf{rnd}(u),\mathsf{rnd}(v)]$. Next, it also obtains another sample r independently from the triangular distribution Tri. If $(x+r)$

Algorithm 2: $\mathsf{ICOND}^{\mathsf{Cont}}(P^{\mathsf{Tri}}, u, v, \delta, \theta)$

$T \leftarrow (2\theta+1)\log\frac{1}{\delta}$
for $i \in \{0 \dots T\}$ **do**
  $x \leftarrow \mathsf{ICOND}(P, [\mathsf{rnd}(u), \mathsf{rnd}(v)])$
  $r \leftarrow \mathsf{Tri}$
  if $(x+r) \in [\![u, v]\!]$ **then**
    return $x+r$
return $\perp$

belongs to the interval $[\![u, v]\!]$, $\mathsf{ICOND}^{\mathsf{Cont}}$ outputs $(x+r)$ as a sample from P^{Tri}. Otherwise, it restarts the same process. If no valid sample is obtained in T iterations, $\mathsf{ICOND}^{\mathsf{Cont}}$ outputs $\perp$ and terminates the algorithm. The following theorem states the correctness of the Algorithm 2.

Theorem 6 (Correctness of $\mathsf{ICOND}^{\mathsf{Cont}}$)**.** *Let P be a discrete distribution defined over $[n]$ and u, v are any real numbers from $[\![0,\ n]\!]$ with $v \geq u+1$. Then the algorithm $\mathsf{ICOND}^{\mathsf{Cont}}$ simulates the $\mathsf{ICOND}^{\mathsf{Cont}}$ oracle for the conditional continuous distribution $P^{\mathsf{Tri}}_{|[\![u,v]\!]}$. Moreover, if the algorithm terminates with a valid real number in $[\![u, v]\!]$, it makes at most $\mathcal{O}(L)$ calls to the ICOND oracle in expectation, where $L = \max\left\{\frac{P(u)}{P([u+1,v])}, \frac{P(v)}{P([u,v-1])}\right\}$.*

We now outline the two key components underlying the proof of Theorem 6, deferred to the supplementary material. First, we argue that when the algorithm ICOND does output a real value (i.e., the output is not $\perp$), then the value returned is a real number from $[\![u, v]\!]$ drawn according to the distribution $P^{\mathsf{Tri}}_{|[\![u,v]\!]}$. The main observation here is that the number $(x+r)$ is a real number between $[\![\mathsf{rnd}(u) - \frac{1}{2}, \mathsf{rnd}(v) + \frac{1}{2}]\!]$ drawn according to the distribution $P^{\mathsf{Tri}}_{|[\![\mathsf{rnd}(u)-\frac{1}{2}, \mathsf{rnd}(v)+\frac{1}{2}]\!]}$. If $(x+r)$ is not within the range $[\![u, v]\!]$, then the process is repeated for T times or until a number between $[\![u, v]\!]$ is picked, whichever occurs first.

Second, we show that if θ is large enough, the probability that the algorithm ICOND outputs $\perp$ is small. Informally speaking, while calling the ICOND oracle on the interval $[\mathsf{rnd}(u), \mathsf{rnd}(v)]$, as long as $\theta \geq L$, where $L = \max\left\{\frac{P(u)}{P([u+1,v])}, \frac{P(v)}{P([u,v-1])}\right\}$, the chance of $(x+r)$ not falling in the range $[\![u, v]\!]$ in any of the T rounds is very small. And if the algorithm does not output $\perp$, it takes at most $\mathcal{O}(L)$ calls to the ICOND oracle in expectation.

5 Our Algorithms

We now present our algorithms, as stated in Theorem 4, assuming access to ICOND queries on the unknown distribution P. We begin by describing a crucial subroutine, Est, that estimates the value of $P(x)$ using the TPA algorithm. Then we will describe our algorithms $\mathsf{toltest}$ and $\mathsf{ERtoltest}$, both of which use Est as a subroutine.

5.1 Probability Mass Estimator: Est

Given access to the ICOND oracle for the unknown distribution P, along with parameters ζ, δ, B, and a target element $x \in [n]$, Est estimates $P(x)$ in two phases using the TPA algorithm. In the first phase, it calls TPA to obtain a coarse estimate λ. If this call fails (returns $\perp$), Est also returns $\perp$. Otherwise, it computes a refined iteration bound r_2 based on λ, and makes a second TPA call. If the second call fails, Est again returns $\perp$; otherwise, it returns $e^{-\lambda}$ as the estimate of $P(x)$. To limit the number of iterations in TPA, Est uses thresholds Thresh_1 and Thresh_2, determined by B, in each phase. Formally, the correctness of Est is stated in the following lemma.

Algorithm 3: $\mathsf{Est}(P, x, \zeta, \delta, B, \theta)$

▶ **First phase of** TPA **for early estimate**
$r_1 \leftarrow 2\log\frac{8}{\delta}$
$\mathsf{Thresh}_1 \leftarrow B + \log\frac{2r_1}{\delta} + \sqrt{\log^2\frac{2r_1}{\delta} + 2B\log\frac{2r_1}{\delta}}$
$\lambda \leftarrow \mathsf{TPA}(P^{\mathsf{Tri}}, x, r_1, \mathsf{Thresh}_1, \frac{\delta}{4}, \theta)$
if $\lambda = \perp$ **then**
  return $\perp$
▶ **Second phase of** TPA **to refine estimate**
New iterations: $r_2 \leftarrow \frac{2(\lambda+\sqrt{\lambda}+2+\log(1+\zeta))}{\log^2(1+\zeta)} \cdot \log\frac{16}{\delta}$
Compute refined threshold: $\mathsf{Thresh}_2 \leftarrow B + \log\frac{2r_2}{\delta} + \sqrt{\log^2\frac{2r_2}{\delta} + 2B\log\frac{2r_2}{\delta}}$
$\lambda \leftarrow \mathsf{TPA}(P^{\mathsf{Tri}}, x, r_2, \mathsf{Thresh}_2, \frac{\delta}{4}, \theta)$
if $\lambda = \perp$ **then**
  return $\perp$
return Final estimate: $e^{-\lambda}$

Lemma 7 (Correctness of Est)**.** *Given ICOND access to P, an element $x \in [n]$, and parameters $\zeta, \delta \in (0,1)$, the algorithm Est returns a value $\widehat{P}_x$. If $P(x) \geq \frac{1}{\theta}$, then with probability at least $1-\delta$, $\widehat{P}_x \in (1 \pm \zeta) \cdot P(x)$ holds. The expected number of ICOND queries performed by Est is $\widetilde{\mathcal{O}}\left(\frac{1}{(\eta-\varepsilon)^2}\,\mathbb{E}_{x\sim P}\left[\min(\mathsf{tilt}_P(x), \theta) \cdot \log^2\frac{1}{P(x)}\right]\right)$.*

5.2 Our Distance Estimator: toltest

We now describe our algorithm, $\mathsf{toltest}$. Given ICOND query access to an unknown distribution P, a known distribution Q, and parameters ε, η with $\eta > \varepsilon$, outputs Accept if $\mathsf{d_{TV}}(P, Q) \leq \varepsilon$ and outputs Reject if $\mathsf{d_{TV}}(P, Q) \geq \eta$. We will first describe the algorithm and then prove its correctness. The algorithm $\mathsf{toltest}$ (Algorithm 4) uses Est (Algorithm 3) as subroutine. The algorithm works by sampling from a mixture distribution $P \diamond Q := \frac{P+Q}{2}$.

$\mathsf{toltest}$ first obtains a multiset $\mathcal{X}$ of t' samples from $P \diamond Q$. For each sample $x_i \in \mathcal{X}$, it calls the subroutine Est to estimate the probability mass of $P(x_i)$ with a maximum threshold B and confidence parameter $\frac{\delta}{4t'}$. If Est returns $\perp$ for any $x_i \in \mathcal{X}$, $\mathsf{toltest}$ discards the sample from $\mathcal{X}$. If the number of samples in $\mathcal{X}$ is less than t, it outputs Reject and terminates the algorithm. Otherwise, it computes $\widehat{d}$ depending on the estimates obtained. Finally, if the estimate $\widehat{d}$ is more than $\frac{\eta+\varepsilon}{4}$, it outputs Reject and terminates the algorithm. Otherwise it outputs Accept, and declares that P and Q are ε-close in $\mathsf{d_{TV}}$ distance. The threshold B (computed based on the min-entropy of Q) dictates the sample complexity of Est.

Algorithm 4: toltest$(P, Q, \varepsilon, \eta, \delta)$

$\varepsilon' \leftarrow \frac{\varepsilon}{2}, \eta' \leftarrow \frac{\eta}{2}$
$\zeta \leftarrow \frac{\eta'-\varepsilon'}{\eta'-\varepsilon'+2}, t \leftarrow \frac{8}{(\eta'-\varepsilon')^2}\log\frac{4}{\delta}$
$k \leftarrow 1 + \frac{1}{t}\log\frac{4}{\delta} + \sqrt{\frac{1}{t}\log^2\frac{4}{\delta} + \frac{2}{t}\log\frac{4}{\delta}}$
$t' \leftarrow 3kt$
$P \diamond Q = \frac{P+Q}{2}$
Draw a multiset $\mathcal{X} = \{x_1, \ldots, x_{t'}\}$ from $P \diamond Q$
$B \leftarrow \log\theta + \log(1+\varepsilon'),\ \theta \leftarrow \frac{1}{Q_{\min}}$
for $i = 1$ to t' **do**
    $\widehat{P}_{x_i} \leftarrow \mathsf{Est}(P \diamond Q, x_i, \varepsilon', \zeta, \frac{\delta}{4t'}, B, \theta)$
    if $\widehat{P}_{x_i} = \perp$ **then**
        $\mathcal{X}.\mathsf{remove}(x_i)$
if $|\mathcal{X}| < t$ **then**
    return Reject
$\widehat{d} \leftarrow \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \max\left(0, 1 - \frac{Q(x)}{\widehat{P}_x}\right)$
if $\widehat{d} > \frac{\eta+\varepsilon}{4}$ **then**
    return Reject
return Accept

Algorithm 5: ERtoltest$(P, Q, \varepsilon, \eta, \delta)$

$\zeta \leftarrow \frac{\eta-\varepsilon}{\eta-\varepsilon+2}, t \leftarrow \frac{8}{(\eta-\varepsilon)^2}\log\frac{4}{\delta}$
Draw a multiset $\mathcal{X} = \{x_1, \ldots, x_t\}$ from P
for $i = 1$ to t **do**
    if $Q(x_i) = 0$ **then**
        return Reject
    $B \leftarrow \log\frac{1+2\varepsilon}{Q(x_i)},\ \theta \leftarrow \frac{1+\varepsilon}{1-\varepsilon}\mathsf{tilt}_Q(x_i)$
    $\widehat{P}_x \leftarrow \mathsf{Est}(P, x_i, \zeta, \frac{\delta}{4t}, B, \theta)$
    if $\widehat{P}_x = \perp$ **then**
        return Reject
$\widehat{d} \leftarrow \frac{1}{t}\sum_{i=1}^{t} \max\left(0, 1 - \frac{Q(x_i)}{\widehat{P}_{x_i}}\right)$
if $\widehat{d} > \frac{\eta+\varepsilon}{2}$ **then**
    return Reject
return Accept

5.3 The Early Reject Tester: ERtoltest

Now we modify the identity tester toltest to obtain a simpler early reject identity tester ERtoltest (Algorithm 5) that outputs Accept if $\ell_\infty(P, Q) \leq \varepsilon$ and outputs Reject if $\mathsf{d}_{\mathsf{TV}}(P, Q) \geq \eta$. The primary difference between ERtoltest and toltest is that ERtoltest uses the full power of the Est subroutine that also uses a threshold parameter B to control the running time of the TPA subroutine. While toltest uses a fixed threshold B that is independent of the sample x_i, ERtoltest uses a sample dependent threshold B that is computed based on the probability of x_i in the known distribution Q. This allows ERtoltest to run faster than toltest, as it can adaptively adjust the threshold based on the sample being processed. Also, if Est returns $\perp$ for any sample x_i, ERtoltest immediately outputs Reject and terminates.

Listing 2: ICOND implementation for a standard geometric sampler. Variables introduced specifically to support interval conditioning are highlighted.

```
def Geometric(p, x_low, x_high):
    u_low= 1 - (1 - p) ** x_high
    u_high= 1 - (1 - p) ** x_low
    U = uniform(u_low, u_high)
    return ceil(log(1 - U) / log(1 - p))
```

6 Testing Inverse Transform Samplers

In this section, we extend ERtoltest and toltest to design the tester Lachesis for inverse transform samplers. The tester Lachesis has two modes: (1) ERtoltest mode, and (2) toltest mode. We use Lachesis to verify the correctness of three common samplers implemented following NumPy's design standards: (1) Geometric distribution sampler (Fishman 2001), (2) a Binomial sampler (Hörmann 1993), (3) a Poisson sampler (Hörman and Derflinger 1994). We assume full access to the program code, allowing us to inspect and modify the implementation. We note that despite having complete visibility, it is #P-Hard to explicitly determine the exact distribution of a randomized program's output (Holtzen, Van den Broeck, and Millstein 2020). Thus, a tester has to employ a sampling based technique, possibly using conditional samples, which approximates the probability mass function of the program's output distribution. We first describe how ICOND can be implemented for programs that realize the inverse transform samplers.

6.1 ICOND Implementation

In inverse transform samplers, the inverse sampling component is responsible for the primary sample generation. Therefore, the ICOND oracle for such samplers can be implemented by conditioning the output of this inverse step to fall within a specified interval. This, in turn, reduces to sampling uniformly from the corresponding pre-image of that interval under the transformation. Listing 2 illustrates the implementation of ICOND for the geometric distribution sampler, a simple example of the inverse transform sampler.

Claim 8 (ICOND Implementation)**.** *Let* sampler *be an inverse transform sampler that uses a hat distribution* h *with cumulative distribution function* H*. Then, the interval conditioning query* ICOND(sampler, $[a, b]$) *is equivalent to* ICOND(Unif, $[\![\mathsf{H}(a), \mathsf{H}(b)]\!]$)*, where* Unif *denotes the uniform distribution over* $[\![0, 1]\!]$.

6.2 Evaluation

To evaluate the practical effectiveness of Lachesis, we implemented a prototype in Python3. The objective of our empirical evaluation was to answer the following questions: **RQ1:** Is Lachesis accurate and scalable for testing inverse transform samplers? **RQ2:** Is Lachesis better than the baseline algorithm?

Baseline and Parameters We implemented the algorithm CubeProbe proposed in (Bhattacharyya et al. 2024) and used it as a baseline for evaluating our tool. CubeProbe

Parameters	1		2		3		4		5		6	
(n, p) or μ	Dec.	# Calls	Dec.	# Calls	Dec.	# Calls	Dec.	# Calls	Dec.	# Calls	Dec.	# Calls
31306, 0.16	A	46K	R	38K	R	9830K	R	9874K	R	9874K	R	9874K
83836, 0.42	A	49K	R	41K	R	7124K	R	6966K	R	6966K	R	6966K
49489, 0.25	A	49K	R	40K	R	8893K	R	8924K	R	8925K	R	8925K
39387, 0.2	A	47K	R	40K	R	9430K	R	9467K	R	9467K	R	9468K
77775, 0.39	A	50K	R	42K	R	7474K	R	7318K	R	7318K	R	7318K
89897, 0.45	A	51K	R	41K	R	6771K	R	6611K	R	6612K	R	6612K
29285	A	982K	R	1022K	R	1023K	R	2266K	A	980K	A	976K
69693	A	1024K	R	1067K	R	1067K	R	5393K	A	1024K	A	1016K
100000	A	1041K	R	1081K	R	1083K	R	7739K	A	1042K	A	1032K
83836	A	1032K	R	1074K	R	1074K	R	6487K	A	1036K	A	1037K
53530	A	1008K	R	1049K	R	1046K	R	4142K	A	1012K	A	1007K
23224	A	968K	R	999K	R	1006K	R	1797K	A	963K	A	966K

Table 1: Run of Lachesis on the original sampler (1) and its buggy variants (2–6). The Dec. column represents the outcome of Lachesis: 'A' denoting an Accept and 'R' denoting a Reject; the '# Calls' column represents the number ICOND queries. The top half reports results for Binomial samplers, and the bottom half for Poisson samplers.

uses a conditioning approach called, *subcube conditioning*, which is a generalization of the interval conditioning approach (Canonne, Ron, and Servedio 2015). To enable compatibility with inverse transform samplers, we adapted CubeProbe with non-algorithmic modifications. Following the setup of (Bhattacharyya et al. 2024), we set the parameters $\varepsilon = 0.01$ and $\eta = 0.5$ for Lachesis and CubeProbe. We set the confidence parameter $\delta = 0.1$ for both algorithms. We conducted all our experiments on a high performance computer cluster, with each node consisting of Intel Xeon Gold 6148 CPUs. We allocated one CPU core and a 5GB memory limit to each tester instance pair.

Performance Experiments We conducted performance experiments to evaluate the runtime of Lachesis (in toltest mode) and CubeProbe. Since CubeProbe is limited to distributions over finite domains, we restricted the performance comparison experiments to the Binomial samplers. The benchmarks used for the performance experiments are different n and p parameters for the Binomial sampler, where n ranges from 1,000 to 600,000 and p ranges from 0.01 to 0.5. We maintained an equal split between the benchmarks that are close to the Binomial distribution and those that are far from it to ensure a balanced evaluation. The results of the performance experiments, shown in Figure 1, demonstrate that Lachesis achieves an average speedup of over $1000\times$ compared to CubeProbe across nearly all benchmarks.

Case Study We performed a case study to evaluate the effectiveness of Lachesis in detecting incorrect implementations of Binomial and Poisson samplers. We used the ERtoltest mode in this setting. To simulate buggy implementations, we manually introduced errors by perturbing the constants used in their inverse sampling routines. For each distribution, we constructed six implementations: implementation 1 is correct, while implementations 2–6 contain injected bugs. For the implementation 5 and 6 of the Poisson sampler we apply benign changes by modifying the rejection sampling parameters that preserves correctness and should be accepted by our tester. Table 1 presents an abridged version of the results. Lachesis correctly identifies all samplers for almost all parameter setting that deviate from the intended distribution, while correctly accepting the implementations where the modifications preserve distributional correctness. The full experimental setup and detailed results are available in the supplementary material.

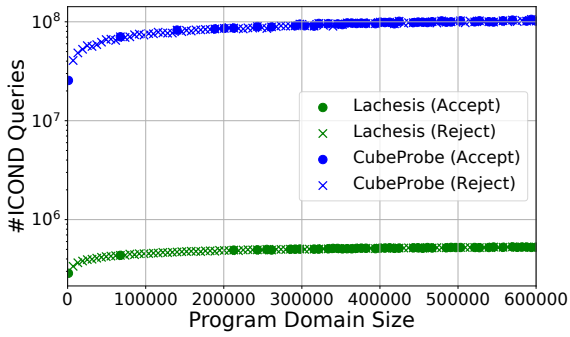

Figure 1: Performance comparison of Lachesis and CubeProbe on Binomial samplers. The y-axis represents the number of samples drawn from the sampler, while the x-axis shows the Domain size of the distribution.

7 Conclusion

In this work, we designed the first instance-dependent tolerant testing algorithms in the interval conditioning model. Moreover, our algorithms work very well in practice, as demonstrated by our experimental results.

Limitations of our work The query complexities of our algorithms depend on the tilt of the distributions to be tested, whereas the known lower bound of these problems is $\Omega(\log n/\log\log n)$ in the worst case (Canonne, Ron, and Servedio 2015). Finding a lower bound in terms of tilt of the distributions remains an intriguing open problem.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which improved the presentation of the paper. Rishiraj Bhattacharyya acknowledges the support of UKRI by EPSRC grant number EP/Y001680/1. Sourav Chakraborty's research is partially supported by the SERB project SQUID-1982-SC-4837. Uddalok Sarkar is supported by the Google PhD Fellowship. Sayantan Sen's research is supported by the NRF Investigatorship award (NRF-NRFI10-2024-0006) and CQT Young Researcher Career Development Grant (25-YRCDG-SS). Computations were performed on the Niagara supercomputer at the SciNet HPC Consortium. SciNet is funded by Innovation, Science and Economic Development Canada; the Digital Research Alliance of Canada; the Ontario Research Fund: Research Excellence; and the University of Toronto.

References

Bagnoli, M.; and Bergstrom, T. 2005. Log-concave probability and its applications. *Economic theory*, 26(2): 445–469.

Banerjee, A.; Chakraborty, S.; Chakraborty, S.; Meel, K. S.; Sarkar, U.; and Sen, S. 2023. Testing of Horn Samplers. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, 1301–1330. PMLR.

Banks, J.; Garrabrant, S. M.; Huber, M. L.; and Perizzolo, A. 2018. Using TPA to count linear extensions. *Journal of Discrete Algorithms*, 51: 1–11.

Batu, T.; Fortnow, L.; Fischer, E.; Kumar, R.; Rubinfeld, R.; and White, P. 2001. Testing Random Variables for Independence and Identity. In *FOCS*, 442–451. IEEE Computer Society.

Bhattacharyya, R.; and Chakraborty, S. 2018. Property Testing of Joint Distributions using Conditional Samples. *ACM Transactions on Computation Theory (TOCT)*, 10(4): 16:1–16:20.

Bhattacharyya, R.; Chakraborty, S.; Pote, Y.; Sarkar, U.; and Sen, S. 2024. Testing Self-Reducible Samplers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 7952–7960.

Canonne, C. L. 2020. A survey on distribution testing: Your data is big. But is it blue? *Theory of Computing*, 1–100.

Canonne, C. L. 2022. Topics and Techniques in Distribution Testing: A Biased but Representative Sample. *Foundations and Trends® in Communications and Information Theory*, 19(6): 1032–1198.

Canonne, C. L.; Ron, D.; and Servedio, R. A. 2015. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3): 540–616.

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.; and Meel, K. S. 2019. On Testing of Uniform Samplers. In *AAAI*, 7777–7784. AAAI Press.

Diakonikolas, I.; and Kane, D. M. 2016. A new approach for testing properties of discrete distributions. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 685–694. IEEE.

Feldman, V.; McMillan, A.; Sivakumar, S.; and Talwar, K. 2024. Instance-Optimal Private Density Estimation in the Wasserstein Distance. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Fishman, G. S. 2001. Sampling from Probability Distributions. In *Discrete-Event Simulation: Modeling, Programming, and Analysis*, 326–415. Springer.

Goldreich, O. 2017. *Introduction to property testing*. Cambridge University Press.

Goldreich, O.; and Ron, D. 1997. Property Testing in Bounded Degree Graphs. In Leighton, F. T.; and Shor, P. W., eds., *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, 406–415. ACM.

Hao, Y.; and Orlitsky, A. 2020. Data amplification: Instance-optimal property estimation. In *International Conference on Machine Learning*, 4049–4059. PMLR.

Holtzen, S.; Van den Broeck, G.; and Millstein, T. 2020. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA): 1–31.

Hörman, W.; and Derflinger, G. 1994. The transformed rejection method for generating random variables, an alternative to the ratio of uniforms method. *Communications in Statistics-Simulation and Computation*, 23(3): 847–860.

Hörmann, W. 1993. The generation of binomial random variates. *Journal of statistical computation and simulation*, 46(1-2): 101–110.

Huber, M.; and Schott, S. 2010. Using TPA for Bayesian inference. *Bayesian Statistics*, 9: 257–282.

Kotz, S.; and Van Dorp, J. R. 2004. *Beyond beta: other continuous families of distributions with bounded support and applications*. World Scientific.

Le, H.; Solomon, S.; Than, C.; Tóth, C. D.; and Zhang, T. 2024. Towards Instance-Optimal Euclidean Spanners. In *FOCS*, 1579–1609. IEEE.

Lovász, L.; and Vempala, S. 2007. The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3): 307–358.

Meel, K. S.; Kumar, G.; and Pote, Y. 2025. Distance Estimation for High-Dimensional Discrete Distributions. In *AISTATS*, volume 258 of *Proceedings of Machine Learning Research*, 955–963. PMLR.

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

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*.

Narayanan, S.; Rozhon, V.; Tetek, J.; and Thorup, M. 2024. Instance-Optimality in I/O-Efficient Sampling and Sequential Estimation. In *FOCS*, 658–688. IEEE.

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.

Pote, Y.; and Meel, K. S. 2021. Testing Probabilistic Circuits. In *NeurIPS*, 22336–22347.

Pote, Y.; and Meel, K. S. 2022. On scalable testing of samplers. *Advances in Neural Information Processing Systems*, 35: 28068–28079.

Valiant, G.; and Valiant, P. 2017. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1): 429–455.

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.*