TOWARDS PRACTICAL DISTRIBUTION TESTING

by

YASH POTE

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in the

GRADUATE DIVISION

of the

NATIONAL UNIVERSITY OF SINGAPORE

2025

Thesis Advisor:
Associate Professor Kuldeep S. Meel, Georgia Institute of Technology

Examiners:
A/P Jonathan Mark Scarlett
Professor Arnab Bhattacharyya

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Yash Pote

1 June 2025

To my parents

Acknowledgments

I am deeply grateful to my thesis committee members—Arnab Bhattacharyya, Jonathan Mark Scarlett, and Kuldeep S. Meel—for their mentoring and generosity with their time as I developed my thesis.

I would like to single out Kuldeep, my advisor, for special thanks. My research journey started with an internship with Kuldeep, where his enthusiasm infected me and convinced me to alter my life plans and start a PhD instead. This was one of the best decisions of my life. Kuldeep always went to bat for his students on every issue and ensured that the only thing we would have to worry about was the quality of our research. It is my belief that my PhD would not have worked out with anyone else. I hope the PhD is just the start of our collaborations.

I also thank all of my coauthors for contributing greatly to my development: Rajarshi Roy, David Parker, Marta Kwiatkowska, Rishiraj Bhattacharyya, Uddalok Sarkar, Sayantan Sen, Jiong Yang, Puru Sharma, Cheng-Kai Lim, Dehui Lin, Djordje Jevdjic, Saurabh Joshi, Durgesh Agrawal, and Yasamin Tabatabaee. Djordje Jevdjic posed some of the most fascinating problems which led to some rather exciting research.

In particular, I'm indebted to my coauthors Kuldeep, Sourav Chakraborty, and Gunjan Kumar, without whom this thesis would not have been possible. Sourav was of immense help when it came to building my technical chops. I must also thank Mate Soos for teaching me nearly everything I know about software development and also for adding a lot of joie de vivre to the group.

I am also very grateful to Clément Canonne and Marta Kwiatkowska for hosting me and providing me with invaluable opportunities and advice. Some of the best days of my PhD were spent at Oxford, and at the University of Sydney. At Oxford, Abheek, Daniella, Xiyue, Rajarshi, Thomas, and Emanuele quickly became

I must also thank my labmates for the time well spent together (with deep apologies to anyone I am forgetting): Alexandru Dinu, Anna Latour, Arijit Shaw, Bishwamittra Ghosh, Durgesh Agrawal, Gunjan Kumar, Jiong Yang, Joy Yang, Lawqueen Kanesh, Mate Soos, Mohimenul Kabir, Pranjal, Rahul Gupta, Shubham Sharma, Suwei Yang, Teodora Baluta, Tim van Bremen, Uddalok Sarkar, Yacine Izza, Yong Lai, and Zhanzhong Pang.

This journey would not have been the same without my close confidants in Singapore: Aashish Kolluri, Vipul Arora, Philips George John, Soundarya Ramesh, Rishav Chourasia, Yunjeong Lee, Vijeth Aradhya, Olivia, and Sasi Kumar Murakonda. Most of my time in Singapore was spent in the company of my housemates Aashish, Vipul, and Philips, and one could not have asked for better companions.

Equally important were the long-distance friendships I cherish: Utkarsha Agwan, Akshit Tyagi, Yash Jetwani, Aditya Kanthale, Harshit Singh, Veena Janaj, Vinamra Bhatia, Ritvik Rawat, Soumitra Agarwal, Mitali Ronghe, Chinmay Singhal, Shubhangi Gupta, Shreyas Kulkarni, Shubham Ugare, and Chhavi Shrivastava. Utkarsha and I started our PhDs at similar times, and she served as a source of perspective and advice, as she is usually correct about everything. I've felt at home with almost everyone mentioned, literally, as I have stayed with them frequently during the PhD. Traveling to meet and spend time with my distant friends was the greatest perk of PhD life.

One thing that was constant throughout the years was my yearning to meet my childhood friends: Aditya Divekar, Sahil Bhatnagar, Tanmay Thakkar, Dhruva Sahasrabuddhe, Shaurya Rawat, and Arnab Banerjee. I had a countdown timer, ticking down the days until I met them again, and I expect that to continue for the rest of time.

During the course of my studies, I have received generous financial support from a number of sources, including the Descartes Project at CNRS-CREATE, the Simons Foundation, and, most importantly, Singaporean taxpayers via NUS. I am immensely grateful to all these sources for putting their trust in me and allowing me to focus on research without having to worry about money. I deeply appreciate all that Singapore has done for me; my time here has been magical. I will always miss the milocinos, egg tarts, and the ability to switch off my brain because everything "just works".

Finally, I would like to thank my parents, Sumitra and Pralhad, who were my most passionate cheerleaders through the entire program. They went beyond what was needed in a million ways, but I will choose a specific small example (and defer the rest to the appendix section ...): thank you both for picking me up from the airport every single time, at all times of the day, for the last 11 years!

Contents

Acknowledgments ii

Abstract vii

List of Figures viii

List of Tables ix

1 Introduction 1

1.1 Oracles for Faster Distribution Testing 2

1.2 Problem Statements 3

1.2.1 The Decision Problem 4

1.2.2 The Estimation Problem 5

2 Preliminaries 7

2.1 Complexity Basics 7

2.2 Distance Measures for Distributions 8

2.2.1 Relevance to Applications 8

2.3 Some Useful Tools 9

2.4 Access Oracles 10

I Decision Problems 11

3 Barbarik2: an Algorithm Based on Pair-Conditioning 13

3.1 Introduction 13

3.2 Notations and Preliminaries 16

3.2.1 Chain Formulas 19

3.2.2 Kernel and the Subquery Consistency Assumption 19
3.2.3 Log-Linear Distributions and Inverse Transform Sampling . 20
3.3 An Overview of the Barbarik2 Algorithm 22
3.3.1 Theoretical Analysis 26
3.4 Proof of Correctness of Barbarik2 27
3.5 Evaluation 35

4 Scalability via Bucketing **40**
4.1 Barbarik3: an Linear Query Algorithm for the Decision Problem . . 41
4.1.1 Algorithm Outline 42
4.1.2 Lower Bound 43
4.1.3 The InBucket Subroutine 44
4.1.4 The OutBucket Subroutine 53
4.2 Evaluation 54
4.2.1 Setting A - scalable benchmarks 55
4.2.2 Setting B - real-life benchmarks 56

II Estimation Problems **59**

5 Probabilistic Circuits **61**
5.1 Preliminaries 63
5.1.1 Probability distributions 63
5.2 Teq: a Tractable Algorithm for Closeness Testing 64
5.2.1 Proving the correctness of Teq 66
5.3 Evaluation 73
5.3.1 Setting A - Synthetic benchmarks 73
5.3.2 Setting B - Real-world benchmarks 74
5.3.3 One variable perturbation 74
5.4 A characterization of the complexity of testing 76
5.4.1 Upper bounds 76
5.4.2 Hardness 78
5.4.3 A Test for Equivalence 80
5.4.4 An analysis of Peq 81

6 Polynomial Query Distance Estimation **85**
6.1 Notations and Preliminaries . . . 87
6.1.1 Distance Approximation . . . 89
6.1.2 Taming Distributions . . . 91
6.1.3 Related Work . . . 93
6.1.4 Lower Bound . . . 94
6.2 DistEstimate: a Distance Estimation Algorithm . . . 95
6.2.1 High-Level Overview . . . 95
6.2.2 Theoretical Analysis . . . 97
6.2.3 The Discrete Hypergrid Σ^n . . . 103
6.3 Experiments . . . 104

Bibliography **106**

A Extended table of results **116**
A.1 From Chapter 3 . . . 116
A.1.1 Comparing sample complexity. . . . 116
A.1.2 Comparing the runtime performance . . . 121
A.2 From Chapter 4 . . . 128
A.2.1 Comparing Barbarik2 and Barbarik3 . . . 128
A.3 From Chapter 5 . . . 131
A.3.1 Synthetic PCs . . . 131
A.3.2 Real-world PCs . . . 147

Abstract

Towards Practical Distribution Testing

by

Yash Pote

As systems that employ samplers are deployed in safety-critical software, there is a need for tests that can verify the samplers' statistical correctness. This raises the question: *For a sampler $\mathcal{P}$ and a target distribution $\mathcal{Q}$, can we practically test whether $\mathcal{P}$ samples from a distribution close to $\mathcal{Q}$?*

Samplers can be accessed in a black-box manner, where one can only observe samples drawn from the sampler. Samplers can also be accessed in a white-box manner, where the code is available and can be reasoned about. In the high-dimensional setting, where the domain is $\{0,1\}^N$ for a large N, testing with black-box access is known to be statistically intractable; and in the white-box setting, testing is known to be computationally intractable. Consequently, richer "grey-box" models, such as those allowing conditional sampling, have emerged as promising alternatives.

In this thesis, we develop grey-box algorithms that are fast in theory and practice. In the first part, we focus on the decision variant of the problem, where we develop algorithms that can distinguish between close and far distributions. To this end, we develop a technique that uses pairwise conditioning to improve the query complexity from exponential to linear in N. In our experiments we use this algorithm to design a testing tool for combinatorial samplers.

In the second part, we address the estimation variant, where we estimate the distance between distributions. We first tackle this for distributions with tractable representations, such as probabilistic circuits. We then extend our approach to the broad class of self-reducible models, for which we build the first polynomial query distance estimator. Together, these algorithms provide a toolkit for the practical statistical verification of combinatorial samplers.

List of Figures

4.1 Cactus plot: Barbarik3 vs. Barbarik2. 57

6.1 Query complexity of the baseline vs. our estimator DistEstimate 88

List of Tables

3.1 Performance of Barbarik2. 38

4.1 Runtime performance of Barbarik3 58

5.1 Runtime performance of Teq 74
5.2 Runtime performance of Teq on real-world benchmarks 76
5.3 A summary of hardness results for PC distribution tests. 78

6.1 The sample complexity and runtime performance of DistEstimate on real-world instances. 105

A.1 The Extended Table 116
A.6 Extended Table of Results 131
A.7 Extended Table of Results for Real-world PCs 147

Chapter 1

Introduction

> There is nothing like looking, if you want to find something. You certainly usually find something if you look, but it is not always quite the something you were after.
>
> J.R.R. Tolkien, *The Hobbit*

In this thesis, we study the problem of *distribution testing*, which can be framed as: *Given two distributions $\mathcal{P}$ and $\mathcal{Q}$, determine whether $\mathcal{P}$ is close to $\mathcal{Q}$.*

Distribution testing is one of the main topics of study in statistics and has been studied in several different contexts over the last two centuries [66]. Classically, the analysis of the testing problem has been asymptotic, i.e., where the number of samples drawn from the distribution is assumed to go to infinity. In modern settings, we encounter *high-dimensional* distributions, i.e., distributions over n-dimensional objects such as texts or images, where n is the size of the object and is assumed to be large. To simplify things, we can consider the distributions to be over $\{0,1\}^n$. Since these high-dimensional distributions have extremely large support, the classical asymptotic results fall short, and recent work in the area has focused on tests that provide finite-sample non-asymptotic guarantees.

The first such non-asymptotic test was devised by Goldreich and Ron [52] for the problem of *uniformity testing*, where one has to determine whether $\mathcal{P}$ is the uniform distribution or ε-far from uniform[1]. The test was *black-box* in the sense that the only way to interact with the distribution was via the drawing of samples.

[1] In total variation distance, defined later in Section 2

The sample complexity of the test was $\Omega(2^{n/2})$, and this dependence was shown to be tight by Paninski [77].

The exponential lower bound on the sample complexity gave an impetus to research into opening up the black box and exploiting the fact that, in practice, distributions frequently offer more powerful access. One of the directions taken was in the study of conditional oracle models of accessing distributions.

1.1 Oracles for Faster Distribution Testing

Conditional Oracles To sidestep the exponential lower bounds on testing, the conditional sampling model, or COND, was introduced independently by Chakraborty et al. [27] and Canonne et al. [23], as a more powerful way to access distributions. A COND oracle for distribution $\mathcal{D}$ over $\{0,1\}^n$ takes as input a set $S \subseteq \{0,1\}^n$ with $\mathcal{D}(S) > 0$, and returns a sample $i \in S$ with probability $\mathcal{D}(i)/\mathcal{D}(S)$. It has been shown that the use of the COND oracle and its variants drastically reduces the sample complexity of many tasks in distribution testing [1, 51, 23, 27, 12, 61, 13, 36, 24, 74]. An extensive treatment of the subject can be found in a survey by Canonne [21].

In this thesis, we consider two restricted variants of COND:

1. Pair-conditioning (PCOND): a special case of the COND oracle, introduced by [20] with the restriction that $|S| = 2$, i.e., the size of the conditioning set has to be two. As we will see later in the thesis, the PCOND model is suitable for testing of SAT samplers.

2. Subcube Conditioning (SUBCOND): the SUBCOND model allows conditioning only on sets that are subcubes of the domain. With a view towards plausible conditional models, Canonne et al. [23], Bhattacharyya and Chakraborty [12] introduced the SUBCOND model, which is particularly suited to the Boolean hypercube $\{0,1\}^n$.

Other Oracles With the same goal of designing tests with polynomial sample complexity, a different kind of oracle, known as the DUAL oracle, was proposed

by Canonne et al. [23]. The DUAL oracle allows one to sample from a given distribution and also query the distribution for the probability of arbitrary elements of the domain. A weaker form of DUAL is the APPROXDUAL oracle which allows approximate sampling and returns the approximate probability. In this thesis, we will consider distributions that offer DUAL, and APPROXDUAL access.

A similar oracle, not considered in this thesis, is the probability revealing PR oracle introduced by Onak and Sun [75]. Unlike the DUAL oracle, the PR oracle only returns the probability of the elements that have been sampled rather than arbitrary elements.

Oracles and Costs Our choice of oracles is motivated by practical considerations. While COND is the most flexible, and therefore useful oracle to query, it is also the most expensive to implement, requiring 2^n random bits merely to describe a single conditional set, in the worst case. Even for the restricted oracles we use, we find that in practice the oracle calls are significantly more expensive than sampling. Hence, wherever possible we try to minimise the number of queries to the expensive oracles, while accepting a trade-off of an increased number of queries to the cheap oracles.

Bhattacharya and Valiant [8] studied the trade-off problem for equivalence testing, in which one has to determine whether $P = Q$ or $d_{TV}(P, Q) \geq \varepsilon$, having sample access to both P, and Q. In their setting, the cost to draw samples from P and Q are unequal, hence they show the optimal trade-off where one can draw fewer samples from P while drawing a greater number from Q. We do not explore the trade-off formally in this thesis, and we leave it for future work.

1.2 Problem Statements

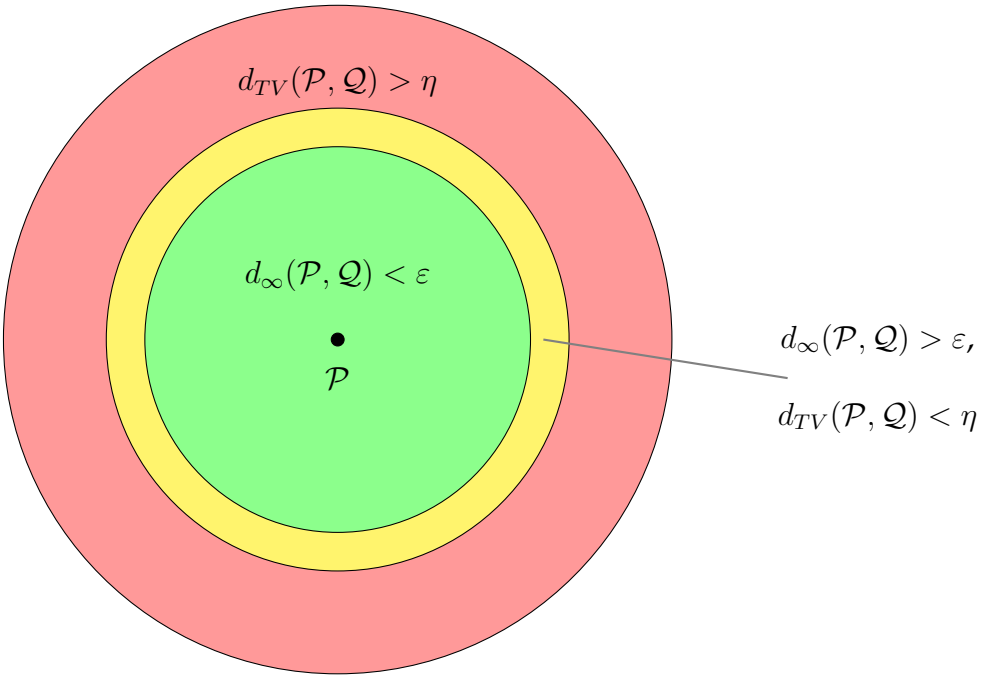
The main problem we are interested in is: *Given two distributions $\mathcal{P}$ and $\mathcal{Q}$, determine whether $\mathcal{P}$ is close to $\mathcal{Q}$*. We will make this question more formal and split the investigation into two themes: decision and estimation.

1.2.1 The Decision Problem

Given as input two distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\{0,1\}^n$, along with parameters ε, η such that $0 < \varepsilon < \eta < 1$, and $0 < \delta \leq 0.5$,

1. With probability $> 1 - \delta$ return `Accept` if $d_\infty(\mathcal{P}, \mathcal{Q}) < \varepsilon$
2. With probability $> 1 - \delta$ return `Reject` if $d_{TV}(\mathcal{P}, \mathcal{Q}) > \eta$

We will define d_{TV} and d_∞ distance functions, and discuss the motivations behind the choice later in Chapter 2.2. However, to provide an intuition, we visualize the problem statement in the following diagram.



At the center, we have the distribution $\mathcal{P}$, and the green disk represents all the distributions that are ε-close in d_∞ to $\mathcal{P}$, while the red region represents the distributions that are η-far from $\mathcal{P}$. As in the definition, we want to, with high probability, return Accept for all $\mathcal{Q}$ that are in the green region and return Reject for all $\mathcal{Q}$ in the red. Notice that there is also a band of yellow in between that represents the distributions for which we do not guarantee correctness. This band is essential for the decision problem and, as discussed later more formally, determines the query complexity of the test.

The first part of the thesis deals with the decision problem, and we present two algorithms designed to solve the problem, Barbarik2 and Barbarik3, in Chapters 3 and 4 respectively. Barbarik2 generalizes the tolerant uniformity test Barbarik [26] to arbitrary discrete distributions, and like Barbarik, uses the PCOND oracle to achieve scalability in practice.

To scale further and to show a polynomial query complexity, we present Barbarik3, another PCOND based algorithm that uses bucketing to split the problem into individually tractable parts. Barbarik3 requires $O(n)$ oracle queries to solve the decision problem in the worst case, where n is the number of dimensions of the domain.

We focus on combinatorial samplers as the objects of experimental investigation, and we conduct a detailed evaluation of these two algorithms, providing the complete data later in the appendix sections A.1 and A.2.

1.2.2 The Estimation Problem

Estimation is a quantitative generalization of the decision problem, where instead of a 0-1 decision, we provide an estimate of how far $\mathcal{P}$ and $\mathcal{Q}$ are. Formally, given as input two distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\{0,1\}^n$, along with parameter ε such that $0 < \varepsilon < 1$, and $0 < \delta \leq 0.5$, return `est` such that

$$\Pr[d_{TV}(\mathcal{P},\mathcal{Q}) - \varepsilon \leq \texttt{est} \leq d_{TV}(\mathcal{P},\mathcal{Q}) + \varepsilon] \geq 1 - \delta$$

In the second part of the thesis, we deal with the estimation question, and we present our contribution in two chapters. The first chapter (Chapter 5) deals with our research into distance estimation in probabilistic circuits, which are a class of ML models that offer tractable DUAL access. Here, we analyze the computational complexity of distance estimation and present some lower and upper bounds with respect to the underlying distribution representation. Furthermore, we design and implement the estimator for the models that allow polynomial time estimation. We present the complete set of results in appendix section A.3.

Then, in the second chapter (Chapter 6), we tackle the longstanding open problem of distance estimation in the conditional sampling model. Although there are many polynomial query algorithms known for the decision problem, to date, the best-known algorithm for distance estimation, even with full COND access,

is exponential in n. We present a part of our ongoing work in designing the first polynomial time distance estimation algorithm that uses SUBCOND, a restricted variant of COND that has the advantage of being computationally plausible to implement.

Chapter 2

Preliminaries

A probability distribution $\mathcal{D}$ over the domain Ω is a function $\mathcal{D} : \Omega \to [0, 1]$ such that $\sum_{\sigma \in \Omega} \mathcal{D}(\sigma) = 1$. In particular, we will focus on distributions over the Boolean hypercube $\{0, 1\}^n$. We use $\mathcal{D}(\sigma)$ to denote the probability of an element $\sigma \in \{0, 1\}^n$, for a distribution $\mathcal{D}$. For a set $S \subseteq \{0, 1\}^n$, we use the notation $\mathcal{D}(S)$ to denote the probability of the set S, i.e. $\mathcal{D}(S) = \sum_{\sigma \in S} \mathcal{D}(\sigma)$.

We will use $\mathcal{D}_S$ to indicate the conditional distribution generated when $\mathcal{D}$ is conditioned on S. Formally,

Definition 1 (Conditioning)**.** *For $\mathcal{D}(S) > 0$, and for some $\sigma \in \{0, 1\}^n$,*

$$\mathcal{D}_S(\sigma) = \frac{1}{\mathcal{D}(S)} \begin{cases} 0 & \text{if } \sigma \notin S \\ \mathcal{D}(\sigma) & \text{if } \sigma \in S \end{cases}$$

We denote by $[n]$ the set $\{1, \ldots, n\}$. For a random variable v, the expectation is defined as $\mathbb{E}[v]$ and the variance as $\mathbb{V}[v]$.

2.1 Complexity Basics

The polynomial hierarchy (PH) contains the classes Σ_1^{P} (NP) and Π_1^{P} (co-NP) along with generalizations of the form Σ_i^{P} and Π_i^{P} where $\Pi_{i+1}^{\mathrm{P}} = \text{co-NP}^{\Pi_i^P}$ and $\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P}$ [90]. The classes Σ_i^P and Π_i^P are said to be at level i. If it is shown that two classes on the same or consecutive levels are equal, the hierarchy collapses to that level. Such a collapse is considered unlikely, and hence is used as the basic assumption for showing many hardness results, including the ones we present in Chapter 5.

2.2 Distance Measures for Distributions

Our goal is to design a program that can test the quality of a distribution with respect to an ideal reference. The total variation distance between distributions is central in this thesis.

Definition 2. *The total variation distance between distributions $\mathcal{P}$ and $\mathcal{Q}$ is*

$$d_{TV}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{\sigma \in \{0,1\}^n} |\mathcal{Q}(\sigma) - \mathcal{P}(\sigma)|$$

We also use the notion of *pointwise* distance.

Definition 3. *The pointwise distance between distributions $\mathcal{P}$ and $\mathcal{Q}$ is*

$$d_{\infty}(\mathcal{P}, \mathcal{Q}) = \max_{\sigma \in \{0,1\}^n} \left(\frac{\mathcal{Q}(\sigma)}{\mathcal{P}(\sigma)}, \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} \right) - 1$$

We will also use the notion of *Hellinger* distance.

Definition 4. *The Hellinger distance between distributions $\mathcal{P}$ and $\mathcal{Q}$ is*

$$d_H(\mathcal{P}, \mathcal{Q}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{\sigma \in \{0,1\}^n} \left(\sqrt{\mathcal{Q}(\sigma)} - \sqrt{\mathcal{P}(\sigma)} \right)^2}$$

2.2.1 Relevance to Applications

Since the available off-the-shelf samplers that come with theoretical guarantees all provide pointwise (d_∞)-closeness guarantees, we are interested in accepting a sampler that is ε-close in d_∞ [54, 50, 28, 30].

In contrast, we would like to be more forgiving to the samplers without guarantees and would like to reject only if they are η-far in TV distance, a notion more relaxed than d_∞ closeness. This has an operational meaning in the context of testing: A randomized program that draws a single sample from $\mathcal{P}$, and after further processing outputs a distribution $\mathcal{D}$. If, $\mathcal{P}$ were to be replaced with $\mathcal{Q}$ then the new output distribution is $\mathcal{D}'$. The replacement could be for the purpose of optimisation, or due to programmer error. Using total variation we can upper bound the change in the output caused by the replacement as follows:

$$d_{TV}(\mathcal{D}, \mathcal{D}') \leq d_{TV}(\mathcal{P}, \mathcal{Q})$$

In the following definition we capture the mentioned ideas, and we will use this definition throughout Chapters 3 and 4.

Definition 5 (ε-closeness and η-farness)**.** *A distribution $\mathcal{P}$ is ε-close to an ideal $\mathcal{Q}$, if we have*

$$d_\infty(\mathcal{P}, \mathcal{Q}) < \varepsilon$$

$\mathcal{P}$ is η-far from the ideal $\mathcal{Q}$, if

$$d_{TV}(\mathcal{P}, \mathcal{Q}) > \eta$$

2.3 Some Useful Tools

Concentration Bounds

Proposition 1 (Hoeffding)**.** *For i.i.d. 0-1 random variables X_i, $X = \sum_{i=1}^{k} X_i$, and $t \geq 0$,*

$$\Pr(X - \mathbb{E}[X] > t) \leq \exp\left(-\frac{2t^2}{k}\right)$$

and

$$\Pr(\mathbb{E}[X] - X > t) \leq \exp\left(-\frac{2t^2}{k}\right)$$

When the true mean $\mathbb{E}[Y_i]$ is unknown, the following Chernoff-type bounds can be applied using a known value θ that acts as a lower bound (for case 1) or upper bound (for case 2) for $\mathbb{E}[Y_i]$:

Corollary 1. *Let $Y_1, Y_2, \ldots, Y_n$ be $i.i.d$ 0-1 random variables.*

1. *If $\mathbb{E}[Y_i] \geq \theta \geq 0$, then for any $t \leq \theta$,*

$$\Pr\left[\sum_{j\in[n]} \frac{Y_j}{n} \leq t\right] < \exp\left(-\frac{(\theta - t)^2 n}{2\theta}\right)$$

2. *If $\mathbb{E}[Y_i] \leq \theta$, then for any $t \geq \theta$,*

$$\Pr\left[\sum_{j\in[n]} \frac{Y_j}{n} \geq t\right] < \exp\left(-\frac{(t - \theta)^2 n}{2t}\right)$$

Proposition 2 (Chebyshev)**.** *Given bounded r.v. X, we have*

$$\Pr(|X - \mathbb{E}[X]| < \mathbb{E}[X]) > \frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]}$$

Sample complexity of learning a distribution. If we are given samples $\{s_1, s_2, \ldots, s_k\}$ from a distribution $\mathcal{D}$ over $[n]$, then the empirical distribution $\widehat{\mathcal{D}}$ is defined to be $\widehat{\mathcal{D}}(i) = \frac{1}{k}\sum_{j=1}^{k} \mathbb{1}_{\{s_j=i\}}$. The following proposition provides a bound on the number of samples required to learn a distribution to accuracy ε in TV with confidence $1-\delta$.

Proposition 3 (See [22] for a simple proof)**.** *Suppose $\mathcal{D}$ is a distribution over $[n]$, and $\widehat{\mathcal{D}}$ is constructed using $\max\left(\frac{n}{\eta^2}, \frac{2ln(2/\delta)}{\eta^2}\right)$ samples from $\mathcal{D}$. Then $d_{TV}(\mathcal{D}, \widehat{\mathcal{D}}) \leq \eta$ with probability at least $1-\delta$.*

2.4 Access Oracles

Our algorithms will interact with distributions only via oracles. We consider a range of oracles, and they can be thought of as variants of three main archetypes, SAMP, EVAL, and COND, which can be informally described as:

1. SAMP($\mathcal{D}$): the sampling oracle, returns a single sample σ from $\mathcal{D}$.
2. EVAL($\mathcal{D}, \sigma$): the evaluation oracle, returns the probability of element σ in the distribution $\mathcal{D}$.
3. COND($\mathcal{D}, S$): the conditional oracle, returns a sample from the $\mathcal{D}$ conditioned on set S, where $S \subseteq \{0,1\}^n$.

In the rest of the thesis, we investigate the practicality of various models by implementing them and testing their efficiency and applicability on real-world sampling benchmarks.

Part I

Decision Problems

This chapter is based on the following publications:

1. **On Testing of Samplers**
   Kuldeep S. Meel 𝔗 Yash Pote 𝔗 Sourav Chakraborty.
   In Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2020.[1]

2. **On Scalable Testing of Samplers**
   Yash Pote 𝔗 Kuldeep S. Meel.
   In Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2022.

We focus on the decision problem in this part of the thesis, and we divide it into two chapters. Chapter 3 focuses on Barbarik2, the first practical algorithm for the decision problem. In this chapter, we present a detailed recipe to implement conditional access into *samplers*, i.e. randomised programs that return samples from a given distribution.

Chapter 4 then presents Barbarik3, a linear query algorithm that improves upon Barbarik2's worst-case performance. In this chapter we focus on purely algorithmic improvements and hence to simplify matters, we only deal with distributions and not samplers.

[1] 𝔗 indicates randomly chosen author ordering.

Chapter 3

Barbarik2: an Algorithm Based on Pair-Conditioning

3.1 Introduction

Motivated by the success of statistical techniques, automated decision-making systems are increasingly employed in critical domains such as medical [40], aeronautics [72], criminal sentencing [45], and military [3]. The potential long-term impact of the ensuing decisions has led to research in the correct-by-construction design of AI-based decision systems. There has been a call for the design of randomised and quantitative formal methods [87] to verify the basic building blocks of the modern AI systems. In this chapter, we focus on one such core building block: *constrained sampling*.

Given a set of constraints φ over a set of variables X and a weight function $\mathtt{wt}$ over assignments to X, the problem of constrained sampling is to sample a satisfying assignment σ of φ with probability proportional to $\mathtt{wt}(\sigma)$. Constrained sampling is a fundamental problem that encapsulates a wide range of sampling formulations [54, 50, 28, 67, 30]. For example, $\mathtt{wt}$ can be used to capture a given prior distribution often represented implicitly through probabilistic models, and φ can be used to capture the evidence arising from the observed data, then the problem of constrained sampling models the problem of sampling from the resulting posterior distribution.

The problem of constrained sampling is computationally hard [60], even for the problems where the corresponding search problem is easy. Consequently, sampling has witnessed a sustained interest from theoreticians and practitioners,

resulting in the proposal of several approximation techniques. Of these, Monte Carlo Markov Chain (MCMC)-based methods form the backbone of modern sampling techniques [4, 15]. The runtime of these techniques depends on the length of the random walk, and the Markov chains that require polynomial walks are called rapidly mixing Markov chains. Unfortunately, for most distributions of practical interest, it is infeasible to design rapidly mixing Markov chains [58], and the practical implementations of such techniques have to resort to the usage of heuristics that violate theoretical guarantees. The developers of such techniques, often and rightly so, strive to demonstrate their effectiveness via empirical behavior in practice [16].

The need for the usage of heuristics to achieve scalability is not restricted to just MCMC methods but is widely observed for other methods such as variational methods [39], hashing-based techniques [28, 50, 29, 70], and simulated annealing techniques [63]. Consequently, a fundamental problem for the designers of sampling techniques is: *how can one efficiently test whether a given technique samples from the desired distribution?* Most of the existing approaches rely on the computations of statistical metrics such as variation distance and KL-divergence by drawing samples and perform hypothesis testing with a preset p-value. Sound computations of statistical metrics require a large number of samples that is proportional to the support of the posterior distribution [7, 92], which is prohibitively large; it is not uncommon for the distribution support to be significantly larger than 2^{70}. Consequently, the existing approaches tend to estimate the desired quantities using a fraction of the required samples, and such estimates are often without the required confidence. The usage of unsound metrics may lead to unsound conclusions, as demonstrated by a recent study where the usage of unsound metric would lead one to conclude that two samplers were indistinguishable (it is worth mentioning that the authors of the study clearly warn the reader about the unsoundness of the underlying metrics) [46].

The researchers in the sub-field of property testing within theoretical computer science have analyzed the sample complexity of testing under different models of samplers and computation. The resulting frameworks have not witnessed widespread adoption to practice due to a lack of samplers that can precisely fit the models under which results are obtained. In a recent work, Chakraborty

and Meel [26], building on the concepts developed in the condition sampling model (rf. [1]), designed the first practical algorithmic procedure, called Barbarik, that can rigorously test whether a given sampler produces the uniform distribution using a constant number of samples assuming that the given sampler is *subquery-consistent* (see Definition 14). Empirically, Barbarik was shown to be able to distinguish samplers that were indistinguishable in prior studies based on unsound metrics. While Barbarik made significant progress, it is marred by its ability to handle only the uniform distribution. Therefore, one wonders: *Can we design an algorithmic framework to test whether the distribution generated by a given sampler is close to a desired (but arbitrary) posterior distribution of interest?*

This chapter's primary contribution is the first efficient algorithmic framework, Barbarik2, to test whether the distribution generated by a sampler is ε-close or η-far from the desired distribution specified by the set of constraints φ and a weight function `wt`. In contrast to statistical techniques that require an exponential or sub-exponential number of samples for samplers whose support can be represented by n bits, the number of samples required by Barbarik2 depends on the *tilt* of the distribution, where *tilt* is defined as the maximum ratio of non-zero weights of two solutions of φ. Like Barbarik, the key technical idea of Barbarik2 sits at the intersection of *property testing* and *formal methods* and uses ideas from conditional sampling and employs chain formulas. However, the key algorithmic framework of Barbarik2 differs significantly from Barbarik, and, as demonstrated, the proof of its correctness and sample complexity requires an entirely new set of technical arguments.

Given access to an ideal sampler $\mathcal{P}$, Barbarik2 accepts every sampler that is ε-close to $\mathcal{P}$ while its ability to reject a sampler that is η-far from $\mathcal{P}$ assumes that the sampler under test is subquery-consistent. Since Barbarik2 assumes access to an ideal sampler, one might wonder if a tester such as Barbarik2 is needed when we already have access to an ideal sampler. Since sampling is computationally intractable, it is almost always the case that an ideal sampler $\mathcal{P}$ is significantly slow and one would prefer to use some other efficient sampler $\mathcal{Q}$ instead of $\mathcal{P}$, if $\mathcal{Q}$ can be certified to be close to $\mathcal{P}$.

To demonstrate the practical efficiency of Barbarik2, we developed a prototype implementation in Python and performed an experimental evaluation with several

samplers. While our framework does not put a restriction on the representation of wt, we perform empirical validation with weight distributions corresponding to log-linear models, a widely used class of distributions. Our empirical evaluation shows that Barbarik2 returns Accept for the samplers with formal guarantees but returns Reject for other samplers that are without formal guarantees. Our ability to reject samplers providence evidence in support of our assumption of subquery consistency of samplers. We believe our formalization of testing of samplers and the design of the algorithmic procedure, Barbarik2, contributes to the design of *randomised formal methods* for verified AI, a principle argued by Seshia et al [87].

3.2 Notations and Preliminaries

A Boolean variable is denoted by a lowercase letter. For a Boolean formula φ, the set of variables appearing in φ, called the *support* of φ, is denoted by $Supp(\varphi)$. An assignment $\sigma \in \{0,1\}^{|Supp(\varphi)|}$ to the variables of φ is a *satisfying assignment* or *witness* if it makes φ evaluate to 1. We denote the set of all satisfying assignments of φ as R_φ.

Definition 6 (Projection of an Assignment)**.** *Let σ be an assignment for the variables in $Supp(\varphi)$ and let $S \subseteq Supp(\varphi)$. The projection of σ onto S, denoted $\sigma_{\downarrow S}$, is an assignment over the variables in S such that for every variable $v \in S$, the value of v under $\sigma_{\downarrow S}$ is identical to its value under σ.*

We denote the set of all unique projections of the witnesses of φ onto S as $R_{\varphi\downarrow S}$, where $R_{\varphi\downarrow S} = \{\sigma_{\downarrow S} \mid \sigma \in R_\varphi\}$.

Definition 7 (Weight Function)**.** *For a set S of Boolean variables, a weight function $\mathtt{wt} : \{0,1\}^{|S|} \to (0,1)$ maps each assignment to some weight.*

Definition 8 (Sampler)**.** *A sampler $\mathcal{Q}(\varphi, S, \mathtt{wt})$ is a randomised algorithm that takes in a Boolean formula φ, a weight function wt, a set $S \subseteq Supp(\varphi)$, and outputs a sample from $R_{\varphi\downarrow S}$. For brevity of notation, we may sometimes refer to a sampler as $\mathcal{Q}(\varphi)$ or simply, $\mathcal{Q}$.*

For any $\sigma \in \{0,1\}^{|S|}$ the probability of the sampler $\mathcal{Q}$ outputting σ is denoted by $\mathcal{Q}(\varphi, \sigma)$ or $\mathcal{Q}(\sigma)$ if φ is clear from context.

We use $\mathcal{Q}(\varphi, S)$ to represent the distribution induced by $\mathcal{Q}$ on $R_{\varphi\downarrow S}$. When the set S is understood from the context, we will denote $\mathcal{Q}(\varphi, S)$ by $\mathcal{Q}(\varphi)$.

Example 3.1 (Illustration of Notations)**.** Let's consider a simple Boolean formula $\varphi \equiv (x_1 \vee x_2) \wedge x_3$.

- **Support:** The set of variables is $Supp(\varphi) = \{x_1, x_2, x_3\}$.
- **Witnesses:** The set of satisfying assignments, R_φ, consists of all assignments where $x_3 = 1$ and $(x_1 \vee x_2) = 1$.

$$R_\varphi = \{\underbrace{(1,1,1)}_{\sigma_1}, \underbrace{(1,0,1)}_{\sigma_2}, \underbrace{(0,1,1)}_{\sigma_3}\}$$

- **Projection:** Let's choose the projection set $S = \{x_1, x_3\}$. We find the set of projected witnesses, $R_{\varphi\downarrow S}$, by projecting each witness in R_φ onto the variables x_1 and x_3.

$$(\sigma_1)_{\downarrow S} = (1,1)$$
$$(\sigma_2)_{\downarrow S} = (1,1)$$
$$(\sigma_3)_{\downarrow S} = (0,1)$$

  Note that two distinct witnesses, σ_1 and σ_2, project to the same assignment $(1,1)$. The set of unique projected witnesses is therefore smaller:

$$R_{\varphi\downarrow S} = \{(1,1), (0,1)\}$$

- **Weight Function:** We can define a weight function $\mathtt{wt}$ over all possible assignments for $S = \{x_1, x_3\}$. For instance:

$$\mathtt{wt}(1,1) = 5, \qquad \mathtt{wt}(1,0) = 2,$$
$$\mathtt{wt}(0,1) = 2, \qquad \mathtt{wt}(0,0) = 1$$

- **Sampler:** A sampler $\mathcal{Q}(\varphi, S, \mathtt{wt})$ would produce samples from the set $R_{\varphi\downarrow S} = \{(1,1), (0,1)\}$. An ideal sampler would output each sample $\tau \in R_{\varphi\downarrow S}$ with probability proportional to its weight $\mathtt{wt}(\tau)$.

The normalization constant Z is the sum of weights for the elements in $R_{\varphi\downarrow S}$:

$$Z = \mathtt{wt}(1,1) + \mathtt{wt}(0,1) = 5 + 2 = 7$$

The distribution $\mathcal{Q}(\varphi, S)$ induced by the sampler is:

$$\mathcal{Q}(1,1) = \frac{\mathtt{wt}(1,1)}{Z} = \frac{5}{7}$$
$$\mathcal{Q}(0,1) = \frac{\mathtt{wt}(0,1)}{Z} = \frac{2}{7}$$

Definition 9 (Ideal Sampler)**.** *For a weight function* $\mathtt{wt}$*, a sampler* $\mathcal{P}(\varphi, S, \mathtt{wt})$ *is called an ideal sampler w.r.t. weight function* $\mathtt{wt}$ *if for all* $\sigma \in R_{\varphi\downarrow S}$*:*

$$\mathcal{P}(\varphi, S, \mathtt{wt})(\sigma) = \frac{\mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_{\varphi\downarrow S}} \mathtt{wt}(\sigma')}$$

.

In the rest of the chapter, $\mathcal{P}(\cdot,\cdot,\cdot)$ denotes the ideal sampler, and we will use $\mathcal{P}(\varphi)$ wherever the set S and function $\mathtt{wt}$ is clear from context. If

$$\forall_{\sigma \in R_\varphi}, \mathtt{wt}(\sigma) = \frac{1}{|R_\varphi|}$$

then the ideal sampler is called a uniform sampler.

Definition 10 (Tilt)**.** *For a Boolean formula* φ *and weight function* $\mathtt{wt}$*, we define*

$$tilt(\mathtt{wt}, \varphi) = \max_{\sigma_1, \sigma_2 \in R_\varphi} \frac{\mathtt{wt}(\sigma_1)}{\mathtt{wt}(\sigma_2)}$$

Our goal is to design a program that can test the quality of a sampler with respect to an ideal sampler.

Definition 11 ($(\varepsilon, \eta, \delta)$-tester for samplers)**.** *A* $(\varepsilon, \eta, \delta)$*-tester for samplers is a randomised algorithm that takes a sampler* $\mathcal{Q}$*, an ideal sampler* $\mathcal{P}$*, a tolerance parameter* ε*, an intolerance parameter* η*, a guarantee parameter* δ *and a CNF formula* φ *such that* (1) *If* $\mathcal{Q}(\varphi)$ *is* ε*-close to* $\mathcal{P}(\varphi)$*, then the tester returns* Accept *with probability at least* $1 - \delta$*, and* (2) *If* $\mathcal{Q}(\varphi)$ *is* η*-far from* $\mathcal{P}(\varphi)$*, then the tester returns* Reject *with probability at least* $1 - \delta$*.*

3.2.1 Chain Formulas

A crucial component in our algorithm is the chain formula. Chain formulas, introduced in [31], are a special class of Boolean formulas. Given a positive integer k and m, chain formulas provide an efficient construction of a Boolean formula $\psi_{k,m}$ with exactly k satisfying assignments with $log(k) \leq m$ variables. We employ chain formulas for inverse transform sampling and in the subroutine Kernel.

Definition 12. [31] *Let $c_1 c_2 \cdots c_m$ be the m-bit binary representation of k, where c_m is the least significant bit. We then construct a chain formula $\varphi_{k,m}(\cdot)$ on m variables $a_1, \ldots a_m$ as follows. For every j in $\{1, \ldots m-1\}$, let C_j be the connector "$\vee$" if $c_j = 1$, and the connector "$\wedge$" if $c_j = 0$. Define*

$$\varphi_{k,m}(a_1, \cdots a_m) = a_1 \, C_1 \, (a_2 \, C_2 (\cdots (a_{m-1} \, C_{m-1} \, a_m) \cdots))$$

For example, consider $k = 11$ and $m = 4$. The binary representation of 11 using 4 bits is 1011. Therefore, $\varphi_{5,4}(a_1, a_2, a_3, a_4) = a_1 \vee (a_2 \wedge (a_3 \vee a_4))$.

Lemma 3.1 ([31])**.** *Let $m > 0$ be a natural number, $k < 2^m$, and $\varphi_{k,m}$ as defined above. Then $|\varphi_{k,m}|$ is linear in m and $\varphi_{k,m}$ has exactly k satisfying assignments. Every chain formula ψ on n variables is equivalent to a CNF formula ψ^{CNF} having at most n clauses. In addition, $|\psi^{CNF}|$ is in $O(n^2)$.*

3.2.2 Kernel and the Subquery Consistency Assumption

Kernel is a crucial subroutine that we use in our algorithm to help us draw *conditional samples* from $R_{\varphi \downarrow S}$.

Definition 13. Kernel$(\varphi, S, \sigma_1, \sigma_2)$ *is a function that takes a Boolean formula φ, a set of variables $S \subseteq Supp(\varphi)$, and two assignments $\sigma_1, \sigma_2 \in R_{\varphi \downarrow S}$, and returns $\hat{\varphi}$ such that $R_{\hat{\varphi} \downarrow S} = \{\sigma_1, \sigma_2\}$.*

The notion of *subquery consistency* plays a crucial role in our analysis. Since each subquery can be viewed as conditioning and given that conditioning is a fundamental operation, one would expect that off-the-shelf samplers would be subquery consistent. At the same time, in contrast to practical applications, the set T is arbitrarily chosen, and therefore, it is possible that certain samplers

do not satisfy the property of subquery consistency. It is, however, not known how to test whether a sampler is subquery consistent w.r.t a particular Kernel. While our empirical evaluation provides weak evidence to our claim that off the shelf samplers are subquery consistent, we believe checking whether a sampler is subquery consistent is an interesting and important problem for future work.

Definition 14. *A sampler $\mathcal{D}$ is subquery consistent w.r.t. the target distribution $\mathcal{D}^*$, and* $\mathsf{Kernel}(\varphi, S, \sigma_1, \sigma_2)$ *for φ if the following conditions hold:*

- $\forall(S \subseteq Supp(\varphi)), \sigma_1, \sigma_2 \in R_{\varphi\downarrow S}$
- *let* $\hat{\varphi} \leftarrow \mathsf{Kernel}(\varphi, S, \sigma_1, \sigma_2)$ *then* $\mathcal{D}(\hat{\varphi}, \mathtt{wt}, S) = \mathcal{D}^*(\varphi, \mathtt{wt}, S)_{|T}$*, where* $T = \{\sigma_1, \sigma_2\}$.

Example 3.2. Let there be a function $\varphi := x_3 \wedge (x_1 \vee x_2)$, and a set $S = \{x_1, x_2\}$. Then, for a given pair of assignments $\sigma_1 = \{x_1 = 0, x_2 = 1\}$, and $\sigma_2 = \{x_1 = 1, x_2 = 0\}$, a possible output of $\mathsf{Kernel}(\varphi, S, \sigma_1, \sigma_2)$ is the following function:

$$\hat{\varphi} = (x_1 \rightarrow x_4) \wedge (x_1 \oplus x_2)$$

It can be verified that $R_{\hat{\varphi}\downarrow S} = \{\{x_1 = 0, x_2 = 1\}, \{x_1 = 1, x_2 = 0\}\}$.

3.2.3 Log-Linear Distributions and Inverse Transform Sampling

Log-linear models capture wide class of distributions of interest including those arising from graphical models, conditional random fields, skip-gram models [73]. Formally, for $\sigma \in \{0,1\}^n$, we define

$$\Pr[\sigma|\theta] \propto e^{\theta \cdot \sigma}$$

Following Chavira and Darwiche [34], we describe the following equivalent representation, called literal-weighted functions, of log-linear models.

Definition 15 (Literal-Weighted Functions)**.** *For a CNF formula φ and set $S \subseteq Supp(\varphi)$, a weight function $\mathtt{wt} : \{0,1\}^{|S|} \rightarrow (0,1)$ is called a literal-weighted function if there is a map $\mathtt{W} : S \rightarrow (0,1)$ such that for any assignment $\sigma \in R_{\varphi\downarrow S}$*

$$\mathtt{wt}(\sigma) = \prod_{x \in \sigma} \begin{cases} \mathtt{W}(x) & if \quad x = 1 \\ 1 - \mathtt{W}(x) & if \quad x = 0 \end{cases}$$

In this case we call $\mathtt{wt}$ *a literal-weighted function w.r.t.* $\mathtt{W}$*. Note that we have* $\Pr[\sigma] \propto \mathtt{wt}(\sigma)$*.*

We now discuss the standard technique of inverse transform sampling for completeness. For completeness, we follow the description due to Chakraborty et al [31].

Lemma 3.2. *For any* ε*-close to uniform sampler* $\mathcal{V}$*, any CNF formula* φ *with support* S *and a literal-weighted function* $\mathtt{wt} : \{0,1\}^{|S|} \to (0,1)$*, we can construct a* $\hat{\varphi}$ *s.t.*

$$\forall_{\sigma \in R_\varphi}, \frac{(1-\varepsilon)\mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_\varphi} \mathtt{wt}(\sigma')} \leq \mathcal{V}(\hat{\varphi}, S, \sigma) \leq \frac{(1+\varepsilon)\mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_\varphi} \mathtt{wt}(\sigma')}$$

Proof. Let $S_i = \{x_{i,1}, \cdots, x_{i,m_i}\}$ be a set of m_i "fresh" variables (i.e. variables that were not used before) for each $x_i \in S$. Given any integer $m_i > 0$ and a positive odd number $k_i < 2^{m_i}$, we construct $\varphi_{k_i,m_i}(x_{i,1}, \cdots x_{i,m_i})$ using the chain formula construction in [31] such that $|R_{\varphi_{k_i,m_i}}| = k$. For notational clarity, we simply write φ_{k_i,m_i} when the arguments of the chain formula are clear from context. For each variable $x_i \in S$, such that $\mathtt{W}(x_i^1) = \frac{k_i}{2^{m_i}}$, and $\mathtt{W}(x_i^0) = 1 - \mathtt{W}(x_i)$, let $(x_i \leftrightarrow \varphi_{k_i,m_i})$ be the representative clause. Thus let $\varphi^{CNF} = \bigwedge_{i \in S}(x_i \leftrightarrow \varphi_{k_i,m_i})$. We then define the formula $\hat{\varphi}$ as follows:

$$\hat{\varphi} = \varphi \wedge \varphi^{CNF}$$

We can see that model count of the formula $|R_{\hat{\varphi}}|$ can be given by:

$$|R_{\hat{\varphi}}| = \sum_{\hat{\sigma} \in R_{\hat{\varphi}}} 1 = \sum_{\sigma \in R_\varphi} \sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma)} 1 \tag{3.1}$$

Since the representative formula of every variable uses a fresh set of variables, we have from the structure of $\hat{\varphi}$ that if σ is a witness of φ then:

$$\sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma)} 1 = \prod_{i \in \sigma^0} (2^{m_i} - k_i) \prod_{i \in \sigma^1} k_i \tag{3.2}$$

For any $\sigma \in R_\varphi$:

$$\begin{aligned}
\mathcal{U}(\hat{\varphi}, S, \sigma) &= \sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma)} \mathcal{U}(\hat{\varphi}, \hat{S}, \hat{\sigma}) \\
&= \sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma)} \frac{1}{|R_{\hat{\varphi}}|} \\
&= \frac{\sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma)} 1}{\sum_{\sigma' \in R_\varphi} \sum_{(\hat{\sigma} \in R_{\hat{\varphi}} : \hat{\sigma}_{\downarrow S} = \sigma')} 1} \quad \text{Using (3.1)} \\
&= \frac{\prod_{i \in \sigma^0} (2^{m_i} - k_i) \prod_{i \in \sigma^1} k_i}{\sum_{\sigma' \in R_\varphi} \prod_{i \in \sigma'^0} (2^{m_i} - k_i) \prod_{i \in \sigma'^1} k_i} \quad \text{Using (3.2)} \\
&= \frac{\prod_{i \in \sigma^0} (2^{m_i} - k_i) \prod_{i \in \sigma^1} k_i}{\prod_{i \in S} 2^{m_i}} \cdot \frac{\prod_{i \in S} 2^{m_i}}{\sum_{\sigma' \in R_\varphi} \prod_{i \in \sigma'^0} (2^{m_i} - k_i) \prod_{i \in \sigma'^1} k_i} \\
&= \frac{\prod_{i \in S} \mathtt{W}(\sigma_{\downarrow x_i})}{\sum_{\sigma' \in R_\varphi} \prod_{i \in S} \mathtt{W}(\sigma'_{\downarrow x_i})} \\
&= \frac{\mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_\varphi} \mathtt{wt}(\sigma')}
\end{aligned} \tag{3.3}$$

From the definition of ε-additive closeness (Def. 5) we have:

$$(1+\varepsilon)^{-1} \mathcal{U}(\varphi, S, \sigma) \leq \mathcal{V}(\varphi, S, \sigma) \leq (1+\varepsilon) \mathcal{U}(\varphi, S, \sigma)$$

Substituting into 3.3, we get:

$$\forall_{\sigma \in R_\varphi}, \frac{(1+\varepsilon)^{-1} \mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_\varphi} \mathtt{wt}(\sigma')} \leq \mathcal{V}(\hat{\varphi}, S, \sigma) \leq \frac{(1+\varepsilon) \mathtt{wt}(\sigma)}{\sum_{\sigma' \in R_\varphi} \mathtt{wt}(\sigma')}$$

□

Remark 1. Lemma 3.2 implies that if $\mathcal{V}$ is ε-close to being a uniform sampler, then it can be used as a blackbox to obtain an ε-close-to-ideal sampler $\mathcal{V}_{\mathtt{wt}}$ w.r.t any literal-weighted function wt. It should also be noted that Lemma 3.2 does not imply that if $\mathcal{V}$ is η-far from a uniform sampler, then the new sampler (obtained using the above transformation) is also far from the ideal sampler w.r.t wt. Thus, Lemma 3.2 by itself, does not allow us to reduce the problem to testing the uniformity of $\mathcal{V}$. Reduction to uniformity is not ruled out in general, but only in our transformation framework.

3.3 An Overview of the Barbarik2 Algorithm

In this section, we present the algorithmic framework of Barbarik2, the pseudocode, presented as Algorithm 1, and then the theoretical justification for the algorithm.

Barbarik2 takes as input a blackbox sampler $\mathcal{Q}$, a Boolean formula φ with the associated weight function `wt` and three parameters $(\varepsilon, \eta, \delta)$. It also has access to an ideal sampler $\mathcal{P}$. Barbarik2 is an $(\varepsilon, \eta, \delta)$-tester for samplers. Also if Barbarik2 returns Reject (that is, when $\mathcal{Q}$ is η-far from $\mathcal{P}$), it provides as witness a new formula $\hat{\varphi}$ which is similar to φ, except that $\hat{\varphi}$ has only two assignments to the variables in S (namely σ_1 and σ_2) that can be extended to satisfying assignments of $\hat{\varphi}$ and the relative probability masses of σ_1 and σ_2 in $\mathcal{Q}$ are significantly different from that in $\mathcal{P}$.

The core idea of Barbarik2 is that for verifying the quality of the sampler $\mathcal{Q}(\varphi)$, we can proceed in two stages. In the first stage, if the sampler is far from the ideal sampler $\mathcal{P}$, we hope to find a witness (in the form of two satisfying assignments) for farness with good probability. This can be guaranteed by drawing one sample each from $\mathcal{Q}(\varphi)$ and $\mathcal{P}(\varphi)$. In the second stage, we confirm whether the witness is indeed far. That is, if the witness is the (σ_1, σ_2) pair, we check that the probability of σ_1 and σ_2 in $\mathcal{Q}(\varphi)$ and $\mathcal{P}(\varphi)$ are similar or not.

Here Barbarik2 differs from Barbarik in a significant way. While Barbarik employs a bucketing strategy, Barbarik2 chooses a simpler yet equally effective method to check the similarity between σ_1 and σ_2. This is also the most difficult stage of the tester as one may have to draw a exponential number of samples to confirm this similarity. We manage by drawing samples from the conditional distribution $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\}$ instead of $\mathcal{Q}(\varphi)$. Since $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\}$ is supported on a set of size only two, estimating the distance of $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\}$ from $\mathcal{P}(\varphi) \mid \{\sigma_1, \sigma_2\}$ can be done with constant number of samples.

Now since we do not have direct access to the distribution $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\}$ we circumvent the problem by drawing samples from a new distribution $\mathcal{Q}(\hat{\varphi})$ where $\hat{\varphi}$ is obtained from φ and has similar structure as φ (with $Supp(\varphi) \subseteq Supp(\hat{\varphi})$) and there are only two assignments (namely σ_1 and σ_2 to the variables in $Supp(\varphi)$ that can be extended to satisfying assignments of $\hat{\varphi}$. The subroutine Kernel helps us to simulate the drawing of samples from $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\}$ by drawing of samples from $\mathcal{Q}(\hat{\varphi})$. The subroutine Bias helps to estimate the distance of $\mathcal{Q}(\hat{\varphi})$ from $\mathcal{P}(\hat{\varphi})$.

Finally, we repeat the whole process for a certain number of rounds, and we argue that if the sampler is indeed far, then, with high probability, in at least one round, we will find a witness of farness. On the other hand, if the sampler is close

to ideal, then there does not exist any such witness of farness.

Barbarik2 accesses two subroutines, Bias and Kernel: Bias$(\hat{\sigma}, \Gamma, S)$ takes as input an assignment $\hat{\sigma}$, a list Γ of assignments and a sampling set S. It returns the fraction of assignments of Γ whose projections on S is equal to $\hat{\sigma}$.

Kernel$(\varphi, \sigma_1, \sigma_2)$ is a subroutine (Definition 13) that aims to create a $\hat{\varphi}$ such the behaviour of the sampler on $\hat{\varphi}$ is similar to its behaviour on φ, i.e. $\mathcal{Q}(\varphi) \mid \{\sigma_1, \sigma_2\} \approx \mathcal{Q}(\hat{\varphi})$.

In Barbarik2, in the for loop (in lines 7−20), in each round, the algorithm draws one sample σ_1 according to the distribution $\mathcal{Q}(\varphi)$ and one sample σ_2 according to the ideal distribution on R_φ (line 8). In the case that $\sigma_1 = \sigma_2$, it moves to the next iteration (in line 9-10). In line 16, the subroutine Kernel uses φ and the two samples σ_1 and σ_2, to output a new formula $\hat{\varphi}$ such that $Supp(\varphi) \subseteq Supp(\hat{\varphi})$. In line 17, Barbarik2 draws a list, Γ_3, of N samples according to the distribution $\mathcal{Q}(\hat{\varphi})$. Kernel ensures that for all $\sigma \in \Gamma_3$, $\sigma_{\downarrow S}$ is either σ_1 or σ_2. In line 18 Barbarik2 uses Bias to compute the fraction of samples that are equal to σ_1 (on the variable set S), and if the fraction is greater than the threshold, then Barbarik2 returns Reject (in line 20).

Algorithm 1 Barbarik2($\mathcal{Q}, \mathcal{P}, \varepsilon, \eta, \delta, \varphi, S, \mathtt{wt}$)

$t \leftarrow \log_e(1/\delta) \log_e \left(\frac{5}{5-\eta(\eta-3\varepsilon)}\right)^{-1}$ {t iterations boost the confidence to $1-\delta$}
$n \leftarrow 8 \log_e (t/\delta)$
$lo = (1+\varepsilon)^2$
$hi = 1 + (\eta + 3\varepsilon)/2$
$\Gamma_1 \leftarrow \mathcal{Q}(\varphi, S, t)$; {$\Gamma_1$ gets t samples from $\mathcal{Q}$ }
$\Gamma_2 \leftarrow \mathcal{P}(\varphi, S, t)$; {$\Gamma_2$ gets t samples from $\mathcal{P}$ }
for $i = 1$ to t **do**
  $\sigma_1 \leftarrow \Gamma_1[i]$; $\sigma_2 \leftarrow \Gamma_2[i]$;
  if $\sigma_1 = \sigma_2$ **then**
    continue
  $\alpha \leftarrow \mathtt{wt}(\sigma_1)/\mathtt{wt}(\sigma_2)$ {α is the ratio of probabilities in $\mathcal{P}$}
  $L \leftarrow (\alpha \cdot lo) \,/\, (1 + \alpha \cdot lo)$
  $H \leftarrow (\alpha \cdot hi) \,/\, (1 + \alpha \cdot hi)$
  $T = (H + L)/2$
  $N \leftarrow n \cdot H/(H-L)^2$
  $\hat{\varphi} \leftarrow \mathsf{Kernel}(\varphi, \sigma_1, \sigma_2)$
  $\Gamma_3 \leftarrow \mathcal{Q}(\hat{\varphi}, S, N)$
  $Bias \leftarrow \mathsf{Bias}(\sigma_1, \Gamma_3, S)$ {Bias compares the ratio of probs. in $\mathcal{P}$ and $\mathcal{Q}$ }
  if $Bias > T$ **then**
    return Reject
return Accept

Algorithm 2 Kernel($\varphi, \sigma_1, \sigma_2$)

$m \leftarrow 12, k \leftarrow 2^m - 1$
$\mathbf{Lits_1} \leftarrow (\sigma_1 \setminus \sigma_2)$
$\mathbf{Lits_2} \leftarrow (\sigma_2 \setminus \sigma_1)$
$\mathbf{V} \leftarrow NewVars(\varphi, m)$;
$\hat{\varphi} \leftarrow \varphi \wedge (\sigma_1 \vee \sigma_2)$
$l \sim \mathbf{Lits_1} \cup \mathbf{Lits_2}$
$\hat{\varphi} \leftarrow \hat{\varphi} \wedge (\neg l \rightarrow \psi_{k,m}(\mathbf{V}))$
$\hat{\varphi} \leftarrow \hat{\varphi} \wedge (l \rightarrow \psi_{k,m}(\mathbf{V}))$
return $\hat{\varphi}$

Algorithm 3 Bias($\hat{\sigma}$, Γ, S)

$count = 0$
for $\sigma \in \Gamma$ **do**
  if $\sigma_{\downarrow S} = \hat{\sigma}$ **then**
    $count \leftarrow count + 1$
return $\frac{count}{|\Gamma|}$

Algorithm 2 presents the pseudocode of subroutine Kernel. As stated above, Kernel takes in a Boolean formula φ, a set $S \subseteq Supp(\varphi)$ and two partial assignments $\sigma_1, \sigma_2 \in R_{\varphi\downarrow S}$. Since the set S is implicit from σ_1 and σ_2, it may not be explicitly given as an input. Kernel assumes access to a subroutine $NewVars$ which takes in two parameters, a formula φ and a number m, and returns a set of m fresh variables that do not appear in φ. Kernel first constructs two sets of literals, denoted by $\mathbf{Lits_1}$ (resp. $\mathbf{Lits_2}$), which appear in σ_1 (resp. σ_2) but not σ_2 (resp. σ_1). The algorithm then constructs the formula $\hat{\varphi}$. First it generates $\varphi \wedge (\sigma_1 \vee \sigma_2)$ on Line 5, a formula with exactly two solutions. Next, it randomly chooses a literal l from $\mathbf{Lits_1} \cup \mathbf{Lits_2}$ and constructs a chain formula $(l \rightarrow \psi_{k,m})$ over the fresh Boolean variables $\mathbf{V}[1], \mathbf{V}[2] \cdots, \mathbf{V}[m]$ where k is the number of satisfying assignments the formula has. Conjuncting the two generated formulas, we get $\hat{\varphi} \equiv \varphi \wedge (\sigma_1 \vee \sigma_2)$. Therefore, at the end of Kernel, i.e. line 8, $\hat{\varphi}$ has $2k$ solutions. We choose the value of k such that it is odd (see [31]). The chain formula is linked to a random Boolean literal from the given set of literals for two reasons,

1. An ideal or ε-close to ideal sampler would not be affected by the randomization and would generate the same distribution over $\hat{\varphi}$ as it does over $\varphi \wedge (\sigma_1 \vee \sigma_2)$.

2. If the sampler under test $\mathcal{Q}$ is η-far from ideal, then we want to construct a formula which *cannot* be easily guessed by $\mathcal{P}$. We wish to avoid the scenario where $\mathcal{P}$, an η-far sampler on φ, somehow behaves as an almost-ideal sampler over $\hat{\varphi}$ and hence manages to fool Barbarik2.

3.3.1 Theoretical Analysis

The following theorem gives the mathematical guarantee about the correctness of Barbarik2. Note that the weight function `wt` is used to implement EVAL access to $\mathcal{Q}$.

Theorem 3.1. *Given* PCOND+SAMP *access to sampler $\mathcal{Q}$, and ideal sampler $\mathcal{P}$, $\eta > 3\varepsilon$, δ, φ, and weight function* `wt`, Barbarik2 *draws $\widetilde{O}\left(\frac{tilt(\mathtt{wt},\varphi)^2}{\eta(\eta-3\varepsilon)^3}\right)$ samples, where $\widetilde{O}$ hides a poly logarithmic factor of $1/\delta$. With probability at least $1-\delta$:*

- *If $\mathcal{Q}$ is ε-close to $\mathcal{P}$, then* Barbarik2 *returns* Accept

- *If the distribution $\mathcal{Q}(\varphi)$ is η-far from $\mathcal{P}(\varphi)$, then* Barbarik2 *returns* Reject.

Remark 2. For the Reject case we assume subquery-consistency, to account for adversarial samplers that may behave differently on the Kernel, and on other inputs. We have not observed any such adversarial samplers, and it hence it is possible that we can do without the assumption; we don't have a proof in either direction. On the other hand, if $\mathcal{Q}$ is ε-close to $\mathcal{P}$ then Barbarik2 accepts (with high probability) even if the sampler $\mathcal{Q}$ is not subquery consistent w.r.t Kernel.

It is also worth noting that Barbarik2 terminates with Reject as soon as the check in line 19 succeeds. Therefore, we expect Barbarik2 to require significantly less number of samples when it returns Reject. Furthermore, in the case of Accept, the bound on N calculated in line 15 in terms of $tilt$ is pessimistic as the probability of observing σ_1 and σ_2 such that $\alpha \approx tilt$ for a sampler close to ideal sampler is very small when the tilt is large. We defer detailed discussion of observed sample complexity to Section 3.5.

3.4 Proof of Correctness of Barbarik2

In this section, we present the theoretical analysis of Barbarik2, and the proof of Theorem 3.1. The proof clearly follows from the the following three lemmas.

Lemma 3.3. *If a sampler $\mathcal{Q}$ is ε-close* [1] *to the ideal sampler $\mathcal{P}$, then* Barbarik2 *returns* Accept *with probability at least $1-\delta$.*

Lemma 3.4. *If $\mathcal{Q}$ is subquery consistent w.r.t* Kernel *and if the distribution $\mathcal{Q}(\varphi)$ is η-far from the ideal sampler, then* Barbarik2 *returns* Reject *with probability at least $1-\delta$.*

Lemma 3.5. *Given ε, η and δ,* Barbarik2 *needs at most $\widetilde{O}\left(\frac{tilt(\mathtt{wt},\varphi)^2}{\eta(\eta-3\varepsilon)^3}\right)$ samples for any input formula φ and weight function* `wt`*, where the tilde hides a poly logarithmic factor of $1/\delta, 1/\eta$ and $1/(\eta-3\varepsilon)$.*

We will present the proofs of Lemma 3.3, Lemma 3.4 and Lemma 3.5 in Section 3.4, Section 3.4 and Section 3.4 respectively. In the rest of this section we will use the following notations:

[1] for any ε and $\eta > 3\varepsilon$

- We use $\mathbb{1}(E)$ to represent the indicator variable for the event E.
- We use R_i to denote the event that Barbarik2 returns Reject in iteration i.

We are now ready to present the proofs of Lemma 3.3, Lemma 3.4 and Lemma 3.5.

Proof of Lemma 3.3

Lemma 3.3. *If a sampler $\mathcal{Q}$ is ε-close [2] to the ideal sampler $\mathcal{P}$, then* Barbarik2 *returns* Accept *with probability at least $1-\delta$.*

For the proof of Lemma 3.3 we will firstly show (in Lemma 3.6) that in each iteration of the loop, the probability that Barbarik2 returns Reject is less than δ/t and then the proof of Lemma 3.3 follows by the application of the Chernoff Bound. Recall that R_i denotes the event that Barbarik2 returns Reject in iteration i.

Lemma 3.6. *If sampler $\mathcal{Q}$ is ε-close to an ideal sampler $\mathcal{P}$, then the probability that* Barbarik2 *returns* Reject *in any particular iteration of the loop, is no more than δ/t. Then*

$$\Pr\left[\overline{R_i} \mid \bigwedge_{j\in[i-1]} \overline{R_j}\right] \geq \left(1-\frac{\delta}{t}\right)$$

Proof. (of Lemma 3.6) Barbarik2 returns Reject in the ith iteration if the $Bias$ (in the ith iteration) is more than T, where $T = \frac{L+H}{2}$ with

$$L = \frac{(1+\varepsilon)\mathcal{P}(\varphi,S,\sigma_1)}{(1+\varepsilon)\mathcal{P}(\varphi,S,\sigma_1)+(1+\varepsilon)^{-1}\mathcal{P}(\varphi,S,\sigma_2)}$$

And since, by definition, all the elements in Γ_1, Γ_2 and Γ_3 are obtained by drawing independent samples from $\mathcal{Q}(\varphi)$, $\mathcal{P}(\varphi)$ and $\mathcal{Q}(\hat{\varphi})$ respectively so

$$\begin{aligned}\Pr\left[\overline{R_i} \mid \bigwedge_{j\in[i-1]} \overline{R_j}\right] &= \Pr\left[\, Bias \leq T \text{ in the } i\text{th iteration}\right] \\ &= 1-\Pr\left[\, Bias > T \text{ in the } i\text{th iteration}\right] \\ &= 1-\Pr\left[\sum_{j\in[N]} \frac{\mathbb{1}(\Gamma_3[j]_{\downarrow S}=\sigma_1)}{N} > T\right]\end{aligned}$$

Note that the random variables $\mathbb{1}(\Gamma_3[j]_{\downarrow S}=\sigma_1)$ are i.i.d 0-1. Since the sampler $\mathcal{Q}$ is assumed to be ε-close to the ideal sampler we have

$$(1+\varepsilon)^{-1}\mathcal{P}(\hat{\varphi},\Gamma_3[j]) \leq \mathcal{Q}(\hat{\varphi},\Gamma_3[j]) \leq (1+\varepsilon)\mathcal{P}(\hat{\varphi},\Gamma_3[j]).$$

[2] for any ε and $\eta > 3\varepsilon$

Since the random variable $\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)$ takes value 1 when we draw $\sigma_1 \sim \mathcal{Q}(\hat{\varphi}, S)$, we have,

$$\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)] = \mathcal{Q}(\hat{\varphi}, S, \sigma_1) \leq (1+\varepsilon)\mathcal{P}(\hat{\varphi}, S, \sigma_1)$$

We first note that

$$\mathcal{P}(\hat{\varphi}, S, \sigma_1) = \frac{\mathcal{P}(\varphi, S, \sigma_1)}{\mathcal{P}(\varphi, S, \sigma_1) + \mathcal{P}(\varphi, S, \sigma_2)} \tag{3.4}$$

Now we consider two cases depending on whether $\mathcal{P}(\varphi, S, \sigma_1)$ is greater or lesser than $\mathcal{P}(\varphi, S, \sigma_2)$. If $\mathcal{P}(\varphi, S, \sigma_1) \leq \mathcal{P}(\varphi, S, \sigma_2)$ then we have

$$\begin{aligned}
\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)] &= \mathcal{Q}(\hat{\varphi}, S, \sigma_1) \\
\text{(From Equation 3.4)} \qquad &= \frac{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1)}{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_2)} \\
&\leq \frac{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1)}{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)} = L
\end{aligned} \tag{3.5}$$

However if $\mathcal{P}(\varphi, S, \sigma_1) \geq \mathcal{P}(\varphi, S, \sigma_2)$ then again from Equation 3.4 we have

$$\begin{aligned}
\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_2)] &= \mathcal{Q}(\hat{\varphi}, S, \sigma_2) \\
&= \frac{(1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)}{(1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)} \\
&\geq \frac{(1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)}{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)}
\end{aligned}$$

In that case since $\mathcal{P}(\hat{\varphi}, S, \sigma_1) + \mathcal{P}(\hat{\varphi}, S, \sigma_2) = 1$ we have

$$\begin{aligned}
\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)] &= \mathcal{Q}(\hat{\varphi}, S, \sigma_1) \\
&= 1 - \mathcal{Q}(\hat{\varphi}, S, \sigma_2) \\
&\leq 1 - \left(\frac{(1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)}{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)} \right) \\
&\leq \frac{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1)}{(1+\varepsilon)\mathcal{P}(\varphi, S, \sigma_1) + (1+\varepsilon)^{-1}\mathcal{P}(\varphi, S, \sigma_2)} = L
\end{aligned} \tag{3.6}$$

Thus in either case, from Equation (3.5) and Equation (3.6) we have $\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} =$

$\sigma_1)] \leq L$. Now applying the Chernoff bound from Cor. 1 we have

$$\begin{aligned}
\Pr\left[Bias \geq T\right] &= \Pr\left[\sum_{j\in[N]} \frac{\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)}{N} > T\right] \\
&= \exp\left(-\frac{(T-L)^2 N}{2L}\right) = exp\left(-\frac{(H-L)^2 N}{8L}\right) \\
&\leq \exp\left(-\frac{(H-L)^2 N}{8H}\right) \quad \text{(Because } H \geq L\text{)} \qquad (3.7) \\
&\leq \frac{\delta}{t}, \qquad (3.8)
\end{aligned}$$

where the inequality in line (3.7) follows because $H \geq L$ when[3] $\eta \geq 3\varepsilon$ and last inequality follows because $N = n.H/(H-L)^2$ where $n = 8\log(t/\delta)$. □

Proof. (of Lemma 3.3) Let R_i denote the event that Barbarik2 returns Reject in iteration i and $\overline{R}$ denote the event that Barbarik2 returns Accept. Thus $\overline{R} = \cap_i \overline{R_i}$.

In the i^{th} iteration if the bias is less than the threshold, Barbarik2 fails to Reject. Thus from Lemma 3.6 if the sampler $\mathcal{Q}$ is ε-close to the ideal sampler $\mathcal{P}$ then

$$\Pr\left[\overline{R_i} | \bigwedge_{j\in[i-1]} \overline{R_j}\right] \geq 1 - \frac{\delta}{t}$$

If Barbarik2 has not returned Reject in any of the iteration then after the last iteration Barbarik2 returns Accept. The probability of Barbarik2 returning Accept (event $\overline{R}$) is

$$\Pr\left[\overline{R}\right] \geq \prod_{i\in[t]} \Pr\left[\overline{R_i} | \bigwedge_{j\in[i-1]} \overline{R_j}\right] \geq \left(1 - \frac{\delta}{t}\right)^t \geq 1 - \delta$$

□

Proof of Lemma 3.4

Lemma 3.4. *If $\mathcal{Q}$ is subquery consistent w.r.t* Kernel *and if the distribution $\mathcal{Q}(\varphi)$ is η-far from the ideal sampler, then* Barbarik2 *returns* Reject *with probability at least $1-\delta$.*

Proof. To prove the Lemma, we will start by splitting the set R_φ into disjoint subsets depending on the distribution $D_{\mathcal{Q}(\varphi)}$.

Definition 16. *We define the following sets for use in the soundness proof:*

[3] $H \geq L$ if $hi \geq lo$ that is $\eta \geq 2\varepsilon^2 + \varepsilon$

- $D = \{x \in R_\varphi : \ \mathcal{Q}(x) \leq \mathcal{P}(x)\}$
- $U = R_\varphi \setminus D$
- $U_0 = \{x \in R_\varphi : \ \mathcal{P}(x) < \mathcal{Q}(x) \leq \left(1 + \frac{\eta+3\varepsilon}{2}\right) \mathcal{P}(x)\}$.
- $U_1 = \{x \in R_\varphi : \ \left(1 + \frac{\eta+3\varepsilon}{2}\right) \mathcal{P}(x) < \mathcal{Q}(x)\}$

Recall, R_i is the event that Barbarik2 returns Reject in the ith iteration of the for loop. Then the following lemmas helps us to lower bound the probability of $\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D$ and the probability of R_i under the condition that $\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D$.

Lemma 3.7. *If the sampler $\mathcal{Q}$ is η-far from the ideal sampler then*

$$\Pr\left[R_i \mid (\bigwedge_{j\in[i-1]} \overline{R_j}) \wedge (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right] \geq \frac{4}{5}.$$

Lemma 3.8. *If the sampler $\mathcal{Q}$ is η-far from the ideal sampler on input φ then*

$$\Pr\left[\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D\right] \geq \frac{\eta(\eta - 3\varepsilon)}{4}.$$

Now using Lemmas 3.8 and 3.7 we can complete the proof of soundness. The probability that Barbarik2 returns Reject in the ith iteration of the for loop is

$$\begin{aligned}
&\Pr\left[R_i \mid \bigwedge_{j\in[i-1]} \overline{R_j}\right] \\
&\geq \Pr\left[R_i \mid (\bigwedge_{j\in[i-1]} \overline{R_j}) \wedge (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right] \cdot \Pr[\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D] \\
&\geq \left(\frac{4}{5}\right) \frac{\eta(\eta - 3\varepsilon)}{4} \quad \text{(From Lemma 3.8 and Lemma 3.7)} \qquad (3.9)
\end{aligned}$$

The probability of Barbarik2 returning Reject in any iteration (event R) is given by

$$\begin{aligned}
\Pr\left[\cup_i R_i\right] &= 1 - \prod_{i\in[t]} \Pr\left[\overline{R_i} \mid \bigwedge_{j\in[i-1]} \overline{R_j}\right] \\
&\geq 1 - \prod_{i\in[t]} \left(1 - \frac{\eta(\eta - 3\varepsilon)}{5}\right) \quad \text{(Using Equation (3.9))} \\
&\geq 1 - \left(1 - \frac{\eta(\eta - 3\varepsilon)}{5}\right)^t \\
&\geq 1 - \delta \quad \text{(Substituting } t\text{)}
\end{aligned}$$

□

Now to complete the proof of Lemma 3.4 we have to prove the Lemma 3.7 and Lemma 3.8. They are presented next.

Lemma 3.7. *If the sampler $\mathcal{Q}$ is η-far from the ideal sampler then*

$$\Pr\left[R_i \mid (\bigwedge_{j\in[i-1]} \overline{R_j}) \wedge (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right] \geq \frac{4}{5}.$$

Proof. Let us assume $\Gamma_1[i] \in U_1$ and $\Gamma_2[i] \in D$. That is, we have $\mathcal{Q}(\varphi, S, \Gamma_2[i]) \leq \mathcal{P}(\varphi, S, \Gamma_2[i])$ and $\mathcal{Q}(\varphi, S, \Gamma_1[i]) > \left(1 + \frac{\eta+3\varepsilon}{2}\right)\mathcal{P}(\varphi, S, \Gamma_1[i])$. It follows that

$$\frac{\mathcal{Q}(\varphi, S, \Gamma_1[i])}{\mathcal{Q}(\varphi, S, \Gamma_2[i])} \geq \left(1 + \frac{3\varepsilon+\eta}{2}\right) \cdot \frac{\mathcal{P}(\varphi, S, \Gamma_1[i])}{\mathcal{P}(\varphi, S, \Gamma_2[i])} \tag{3.10}$$

Since $\forall x > 0, a/b > x \implies a/(a+b) > x/(x+1)$, we have from Equation 3.10

$$\begin{aligned}
&\frac{\mathcal{Q}(\varphi, S, \Gamma_1[i])}{\mathcal{Q}(\varphi, S, \Gamma_2[i]) + \mathcal{Q}(\varphi, S, \Gamma_1[i])} \\
&\geq \left(1 + \frac{3\varepsilon+\eta}{2}\right) \cdot \frac{\mathcal{P}(\varphi, S, \Gamma_1[i])}{\mathcal{P}(\varphi, S, \Gamma_2[i])} \cdot \left(1 + \left(1 + \frac{3\varepsilon+\eta}{2}\right) \cdot \frac{\mathcal{P}(\varphi, S, \Gamma_1[i])}{\mathcal{P}(\varphi, S, \Gamma_2[i])}\right)^{-1}
\end{aligned}$$

Thus we have

$$\begin{aligned}
&\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)] = \mathcal{Q}(\hat{\varphi}, S, \Gamma_1[i]) \\
=&\frac{\mathcal{Q}(\varphi, S, \Gamma_1[i])}{\mathcal{Q}(\varphi, S, \Gamma_2[i]) + \mathcal{Q}(\varphi, S, \Gamma_1[i])} \quad [\text{ by the subquery consistent sampler assumption}] \\
\geq &\left(1 + \frac{3\varepsilon+\eta}{2}\right) \cdot \frac{\mathcal{P}(\varphi, S, \Gamma_1[i])}{\mathcal{P}(\varphi, S, \Gamma_2[i])} \cdot \left(1 + \left(1 + \frac{3\varepsilon+\eta}{2}\right) \cdot \frac{\mathcal{P}(\varphi, S, \Gamma_1[i])}{\mathcal{P}(\varphi, S, \Gamma_2[i])}\right)^{-1} \\
=&H \quad [\text{By definition of } H]
\end{aligned} \tag{3.11}$$

Barbarik2 returns Reject in the ith iteration if the $Bias$ (in the ith iteration) is more than T, where $T = \frac{L+H}{2}$ with

$$H = \frac{(1 + \frac{3\varepsilon+\eta}{2})\mathcal{P}(\varphi, S, \sigma_1)}{(1 + \frac{3\varepsilon+\eta}{2})\mathcal{P}(\varphi, S, \sigma_1) + \mathcal{P}(\varphi, S, \sigma_2)}$$

As defined, all the elements in Γ_1, Γ_2 and Γ_3 are obtained by drawing independent samples from $\mathcal{Q}(\varphi)$, $\mathcal{P}(\varphi)$, and $\mathcal{Q}(\hat{\varphi})$ respectively so we have

$$\begin{aligned}
&\Pr\left[R_i \mid (\bigwedge_{j\in[i-1]} \overline{R_j}) \bigwedge (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right] \\
&= \Pr\left[\, Bias > T \text{ in the } i\text{th iteration} \mid (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right] \\
&= \Pr\left[\sum_{j\in[N]} \frac{\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)}{N} \geq T \mid (\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D)\right]
\end{aligned}$$

Now since $\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)$ are i.i.d 0-1 random variables and since $\Gamma_1[i] \in U_1$ and $\Gamma_2[i] \in D$ implies $\mathbb{E}[\mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1)] \geq H$ (from Equation 3.11) by applying Chernoff bound from Cor. 1 we have:

$$\begin{aligned}
\Pr\left[\frac{1}{N}\sum_{j\in[N]} \mathbb{1}(\Gamma_3[j]_{\downarrow S} = \sigma_1) \geq T\right] &\leq exp\left(-\frac{(H-T)^2 N}{8H}\right) \\
\text{by the choice of N} \quad &\leq \frac{\delta}{t} \\
\text{since } \delta < 0.5 \text{ and } t \geq 3 \quad &\leq 1/5
\end{aligned}$$

□

Lemma 3.8. *If the sampler $\mathcal{Q}$ is η-far from the ideal sampler on input φ then*

$$\Pr\left[\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D\right] \geq \frac{\eta(\eta - 3\varepsilon)}{4}.$$

Proof. Since the sampler $\mathcal{Q}$ is ε-far from the ideal sampler on input φ so, the TV distance between $\mathcal{Q}(\varphi)$ and $\mathcal{P}(\varphi)$ is at least η. By the definition of sets U and D we have,

$$\sum_{x\in U}(\mathcal{Q}(x) - \mathcal{P}(x)) = \sum_{x\in D}(\mathcal{P}(x) - \mathcal{Q}(x)) \geq \eta \tag{3.12}$$

Now by definition of U_0, we have

$$\sum_{x\in U_0}(\mathcal{Q}(x) - \mathcal{P}(x)) < \frac{\eta + 3\varepsilon}{2} \sum_{x\in U_0} \mathcal{P}(x) < \frac{\eta + 3\varepsilon}{2} \tag{3.13}$$

As $U = U_0 \cup U_1$,

$$\sum_{x\in U_1}(\mathcal{Q}(x) - \mathcal{P}(x)) = \sum_{x\in U}(\mathcal{Q}(x) - \mathcal{P}(x)) - \sum_{x\in U_0}(\mathcal{Q}(x) - \mathcal{P}(x)) \tag{3.14}$$

Substituting Equation (3.13) and Equation (3.12) in Equation (3.14) we get:-

$$\sum_{x \in U_1} (\mathcal{Q}(x) - \mathcal{P}(x)) \geq \eta - \frac{\eta + 3\varepsilon}{2} = \frac{\eta - 3\varepsilon}{2}$$

$$\text{Therefore, } \sum_{x \in U_1} \mathcal{Q}(x) \geq \frac{\eta - 3\varepsilon}{2}$$

Thus we have,

$$\Pr\left[\Gamma_1[i] \in U_1\right] = \sum_{x \in U_1} \mathcal{Q}(x) \geq \frac{\eta - 3\varepsilon}{2} \tag{3.15}$$

From Equation (3.12) we know that,

$$\Pr\left[\Gamma_2[i] \in D\right] = \sum_{x \in D} \mathcal{P}(x) \geq \eta \tag{3.16}$$

Since $\Gamma_1[i] \in U_1$ and $\Gamma_2[i] \in D$ are independent events, putting together Equation (3.15) and Equation (3.16), we see that

$$\Pr\left[\Gamma_1[i] \in U_1 \wedge \Gamma_2[i] \in D\right] \geq \frac{\eta(\eta - 3\varepsilon)}{2}$$

□

Proof of Lemma 3.5

Lemma 3.5. *Given ε, η and δ,* Barbarik2 *needs at most $\widetilde{O}\left(\frac{tilt(\mathtt{wt},\varphi)^2}{\eta(\eta-3\varepsilon)^3}\right)$ samples for any input formula φ and weight function* `wt`*, where the tilde hides a poly logarithmic factor of $1/\delta, 1/\eta$ and $1/(\eta - 3\varepsilon)$.*

Proof. From Algorithm 1, line 1, we see that the number of trials is:

$$t = \frac{\log_e(1/\delta)}{\log_e\left(\frac{5}{5-\eta(\eta-3\varepsilon)}\right)}$$

$$(\log_e(x) \leq x - 1) \quad t \leq \log_e(1/\delta)\frac{5}{(\eta(\eta - 3\varepsilon))}$$

In every iteration we calculate a value N according to the expression:

$$\begin{aligned} N &= 8\log_e\left(\frac{t}{\delta}\right) \cdot \frac{\alpha \cdot hi}{1 + \alpha \cdot hi} \cdot \left(\frac{\alpha \cdot hi}{1 + \alpha \cdot hi} - \frac{\alpha \cdot lo}{1 + \alpha \cdot lo}\right)^{-2} \\ &= 8\log_e\left(\frac{t}{\delta}\right) \cdot \left(\frac{1}{hi - lo}\right)^2 \cdot hi \cdot \frac{1 + \alpha \cdot hi}{\alpha} \cdot (1 + \alpha \cdot lo)^2 \\ (1 < lo < hi < 2) \quad &< 8\log_e\left(\frac{t}{\delta}\right) \cdot \left(\frac{1}{hi - lo}\right)^2 \cdot 2 \cdot \frac{1 + \alpha \cdot 2}{\alpha} \cdot (1 + \alpha \cdot 2)^2 \end{aligned}$$

On Line (11) in Algorithm 1 we define:

$$\alpha = \frac{\mathtt{wt}(\sigma_1)}{\mathtt{wt}(\sigma_2)}$$

$$tilt(\mathtt{wt}, \varphi) = \max_{\sigma_1, \sigma_2 \in R_\varphi} \frac{\mathtt{wt}(\sigma_1)}{\mathtt{wt}(\sigma_2)} \quad \text{(Definition 10)}$$

Thus, $\alpha \leq tilt(\mathtt{wt}, \varphi)$. Substituting the values of α, lo and hi, we get:

$$N < 2\log_e\left(\frac{t}{\delta}\right) \cdot \left(\frac{tilt(\mathtt{wt}, \varphi)}{\eta - 3\varepsilon}\right)^2$$

The maximum number of samples drawn after t trials is:

$$\begin{aligned} & 2t + tN < 2tN \\ \text{(Substituting for t,}N\text{)} \qquad & < 2\log_e\left(\frac{1}{\delta} \cdot \frac{5 \cdot \log_e(1/\delta)}{\eta(\eta - 3\varepsilon)}\right) + \frac{5 \cdot \log_e(1/\delta)}{\eta(\eta - 3\varepsilon)} \times \frac{tilt(\mathtt{wt}, \varphi)^2}{(\eta - 3\varepsilon)^2} \\ & = \tilde{O}\left(\frac{tilt(\mathtt{wt}, \varphi)^2}{\eta(\eta - 3\varepsilon)^3}\right) \end{aligned}$$

□

3.5 Evaluation

The objective of our evaluation was to answer the following questions:

RQ1. Is Barbarik2 able to distinguish between off-the-shelf samplers by returning Accept for samplers ε-close to the ideal distribution and Reject for the η-far samplers?

RQ2. What improvements do we observe over the baseline?

RQ3. How does the required number of samples scale with the $tilt(\mathtt{wt}, \varphi)$ of the distribution?

To evaluate the runtime performance of Barbarik2 and test the quality of some state of the art samplers, we implemented a prototype of Barbarik2 in Python. Our algorithm utilizes an ideal sampler, for which we use the state of the art sampler WAPS [56]. All experiments were conducted on a high performance computing cluster with 600 E5-2690 v3 @2.60GHz CPU cores. For each benchmark, we use a single core with a timeout of 24 hours.

We focus on the log-linear distributions given their ubiquity of usage in machine learning; a formal description is provided in Section 3.2.3 for completeness. Observe that Barbarik2 does not put any restrictions on the representation of the weight distribution. We conducted our experiments on 72 publicly available benchmarks, which have been employed in the evaluation of samplers proposed in the past [29, 46]. The $tilt$ of the benchmarks spans many orders of magnitude, between 1 and 10^{11}.

Samplers Tested The past few years have witnessed a multitude of sampling techniques ranging from variational methods [95], MCMC-based techniques [59, 68], mutation-based sampling [46], importance sampling-based methods [49], knowledge-compilation techniques [56] and the like. The conceptual simplicity of uniform samplers encourages designers to tune their algorithms for uniform sampling, and the standard technique for weighted sampling employs the well-known method of the inverse transform. For the sake of completeness, we provide a detailed discussion of the transformation technique in Section 3.2.3. We perform empirical evaluation with the three state of the art samplers wUnigen, wQuicksampler, and wSTS constructed by augmenting inverse sampling with underlying samplers Unigen [29], Quicksampler [46] and STS(SearchTreeSampler) [49] respectively.

While wUnigen is known to have theoretical guarantees of $\varepsilon-$closeness, there is no theoretical analysis of the distributions generated by wQuicksampler and wSTS. Of the 72 instances, wUnigen can handle only 35 instances while wQuicksampler and wSTS can handle all the 72 instances. The variation in the number of instances that are amenable to sampling for a particular sampler highlights the trade-off between the runtime performance and theoretical guarantees. It is perhaps worth emphasizing that wQuicksampler and wSTS are significantly more efficient in runtime performance than the ideal sampler WAPS.

Test Parameters We set tolerance parameter ε, intolerance parameter η, and confidence δ for Barbarik2 to be and 0.1, 0.8 and 0.2 respectively. The chosen setting of parameters implies that for a given Boolean formula φ, if the sampler under test $\mathcal{G}(\varphi)$ is ε-close to the ideal sampler, then Barbarik2 returns Accept with probability at least 0.8, otherwise if the sampler is η-far from ideal sampler then Barbarik2

returns Reject with probability at least 0.8. Note that, the number of samples required for Accept depends only on the parameters $(\varepsilon, \eta, \delta)$ and $tilt(\mathtt{wt}, \varphi)$. We instantiate Kernel with the values $m = 12$ and $k = 2^m - 1$. Observe that Theorem 3.1 does not put restrictions on k and m.

Description of the table We present the experimental results in Table 3.1. Due to lack of space, we present results for a subset of benchmarks while the extended table is presented in the Appendix. The first column indicates the name of the benchmark, the second the $tilt$, and the following columns indicate the outcome of the experiments with wUnigen, wSTS and wQuicksampler in that order. Every cell in the table has two entries. In the second column, the first entry shows the value of $tilt$ for the corresponding benchmark, while in the other columns, it contains "A" and "R" to indicate the output of Barbarik2 for the corresponding sampler. The second entry for the cells in the column corresponding to $tilt$ indicates the theoretical upper bound on the samples required for Barbarik2 to terminate, while for rest of the columns, the second entry indicates the number of samples consumed by Barbarik2 for the corresponding instance and the sampler.

RQ1 Our experiments demonstrate that Barbarik2 returns Reject for wQuicksampler on 68 benchmarks and Accept on the remaining four benchmarks. For wSTS we found Barbarik2 returned Reject on 62 of the benchmarks and Accept on the seven while it times out on the remaining three. Since wSTS and wQuicksampler are samplers with no formal guarantees and therefore one may expect them to generation distributions away from the ideal distributions. In this context, the results in Table 3.1 provide strong evidence for the reasonableness of the *non-adversarial* assumption in practice.

In contrast, Barbarik2 returned Accept for wUnigen on all the 35 benchmarks for which wUnigen could sample. Recall, wUnigen formally guarantees ε-closeness of the samples to the required distribution, hence Barbarik2 returning Accept on all the benchmarks provides evidence in support of soundness of Barbarik2.

RQ2 We also computed the number of samples required by the baseline approach owing to [7]. Since the number of samples is so large that exhaustive

Benchmark	$tilt$ (maxSamp)	Barbarik2		
		wUnigen (samples)	wSTS (samples)	wQuicksampler (samples)
s349_3_2	28 (3e+07)	A (1e+05)	A (1e+05)	R (22854)
s820a_3_2	37 (5e+07)	A (96212)	R (87997)	A (2e+05)
UserServiceImpl.sk	140 (6e+08)	A (1e+05)	R (1e+05)	R (4393)
LoginService2.sk	232 (2e+09)	A (1e+05)	R (38044)	R (13350)
s349_7_4	603 (1e+10)	A (75555)	R (4284)	R (5150)
s344_3_2	3300 (3e+11)	A (1e+05)	R (59952)	R (5150)
s420_new_7_4	3549 (4e+11)	A (82312)	A (96659)	R (49955)
54.sk_12_97	4e+11 (6e+27)	DNS	R (14012)	R (4627)
s641_7_4	9e+07 (3e+20)	DNS	R (8747)	A (1e+06)
s838_3_2	2e+08 (1e+21)	DNS	R (9504)	R (4627)

Table 3.1: "A"(resp. "R") represents Barbarik2 returning Accept(resp. Reject). maxSamp represents the upper bound on the number of samples required by Barbarik2 to return Accept/Reject.

experimentation is infeasible, we had to resort to estimating the average time taken by a sampler for a given instance. Based on the estimated time, we can estimate the time taken by the baseline for our benchmark set. We observe that the time taken by the baseline would be over 10^6 seconds for 43, 42 and 16 benchmarks for wQuicksampler, wSTS and wUnigen respectively. In this context, it is worth highlighting that Barbarik2 terminates within 24 hours for all the instances for all the samplers. We observe that the geometric means of the speedups over the baseline approach are $10^{5.0}$, $10^{20.2}$ and 58 for wSTS, wQuicksampler and, wUnigen respectively. The lower speedup in the case of wUnigen owes to its ability to handle only small benchmarks, for which the number of models was not very large. The extended results are available in appendix A.1.

RQ3 The number of trials required (indicated by the the variable t as on Line 7 of Algorithm 1) depends only on $(\varepsilon, \eta, \delta)$, so for the values we use, $(0.1, 0.8, 0.2)$, we find that we require $t = 14$ trials. The analysis of the algorithm reveals an upper bound on the sample complexity of the tester (See Section 3.3, Theorem 3.1) which is quadratic in terms of the $tilt(\mathtt{wt}, \varphi)$. We now return to Table 3.1 and observe that the number of samples required by Barbarik2 before returning Accept were significantly lower than the theoretical bound provided in the second column. Furthermore, as noted earlier, the number of samples required before Barbarik2 returns Reject is typically significantly less than the worst case – a trend demonstrated in Table 3.1.

Chapter 4

Scalability via Bucketing

In the previous chapter Barbarik2, could test a given sampler while providing $(\varepsilon, \eta, \delta)$ guarantees, using $\tilde{O}\left(\frac{tilt(\mathcal{P})^2}{\eta(\eta-3\varepsilon)^3}\right)$ queries, where

$$tilt(\mathcal{P}) := \max_{\sigma_1,\sigma_2 \in \{0,1\}^n} \frac{\mathcal{P}(\sigma_1)}{\mathcal{P}(\sigma_2)}$$

for $\mathcal{P}(\sigma_2) > 0$. Since the $tilt(\mathcal{P})$ can take arbitrary values, we observed that the query complexity could be prohibitively large[1]. On the other hand, the best known lower bound for the problem, derived from [74], is $\tilde{\Omega}\left(\frac{\sqrt{n/\log(n)}}{\eta^2}\right)$. In this chapter, we take a step towards bridging this gap with our algorithm, Barbarik3, which has a query complexity of $\tilde{O}\left(\frac{\sqrt{n}\log n}{(\eta-11.6\varepsilon)\eta^3} + \frac{n}{\eta^2}\right)$, representing an exponential improvement over the state of the art.

To be of any real value, testing tools must be able to scale to larger instances. In the case of constrained samplers, the only existing testing tool, Barbarik2, is not scalable owing to its query complexity. The lack of scalability is illustrated by the following fact: product distributions are the simplest possible constrained distributions, and given a union of two n-dimensional product distributions, Barbarik2 requires more than 10^8 queries for $n > 30$. On the other hand, the query complexity of Barbarik3 scales linearly with n, the number of dimensions, thus making it more appropriate for practical use.

We implement Barbarik3 and compare it against Barbarik2 to determine their relative performance. In our experiments, we consider two sets of problems, (1) constrained sampling benchmarks, (2) scalable benchmarks and two constrained samplers wSTS and wUnigen. We found that to complete the test Barbarik3 required at least $450\times$ fewer samples from wSTS and $10\times$ fewer samples from wUnigen as

[1] A simple modification reveals that in terms of n, η, ε, the bound is $\tilde{O}\left(\frac{4^n}{\eta(\eta-3\varepsilon)^3}\right)$

compared to Barbarik2. Moreover, Barbarik3 terminates with a result on at least 3× more benchmarks than Barbarik2 in each experiment.

Our contributions can be summarised as follows:

1. For the problem of testing of samplers, we provide an exponential improvement in query complexity over the current state of the art test Barbarik2. Our test, Barbarik3, makes a total of $\tilde{O}\left(\frac{\sqrt{n}\log n}{(\eta-11.6\varepsilon)\eta^3}+\frac{n}{\eta^2}\right)$ queries, where $\tilde{O}$ hides polylog factors of ε, η and δ.

2. We present an extensive empirical evaluation of Barbarik3 and contrast it with Barbarik2. The results indicate that Barbarik3 requires far fewer samples and terminates on more benchmarks when compared to Barbarik2.

We then present the main contribution of the chapter, the test Barbarik3, and its proof of correctness in Section 4.1. We present our experimental findings in Section 4.2 We defer the full experimental results to the appendix section A.2.

4.1 Barbarik3: an Linear Query Algorithm for the Decision Problem

We start by providing a brief overview of our testing algorithm before providing the full analysis.

Algorithm 4 Barbarik3$(\mathcal{P}, \mathcal{Q}, \eta, \varepsilon, \delta)$

$k \leftarrow n + \lceil \log_2(100/\eta) \rceil$
for $i = 1$ to k **do**
  $S_i = \{b : 2^{-i} < \mathcal{P}(b) \leq 2^{-i+1}\}$
$S_0 = \{0,1\}^n \setminus \bigcup_{i\in[k]} S_i$
$B_{\mathcal{P}}$ is the distribution over $[k] \cup \{0\}$ where we sample $i \sim B_{\mathcal{P}}$ if we sample $j \sim \mathcal{P}$ and $j \in S_i$
$B_{\mathcal{Q}}$ is the distribution over $[k] \cup \{0\}$ where we sample $i \sim B_{\mathcal{Q}}$ if we sample $j \sim \mathcal{Q}$ and $j \in S_i$
$\theta \leftarrow \eta/20$
$\widehat{d} \leftarrow$ OutBucket$(B_{\mathcal{P}}, B_{\mathcal{Q}}, k, \theta, \delta/2)$
if $\widehat{d} > \varepsilon/2 + \theta$ **then**
  Return Reject
$\varepsilon_2 \leftarrow \widehat{d} + \theta$
Return InBucket$(\mathcal{P}, \mathcal{Q}, k, \varepsilon, \varepsilon_2, \eta, \delta/2)$

4.1.1 Algorithm Outline

The pseudocode of Barbarik3 is given in Algorithm 4. We adapt the definition of bucketing of distributions from [74] for use in our analysis.

Definition 17. *For a given $k \in \mathbb{N}_{>0}$, the bucketing of $\{0,1\}^n$ with respect to $\mathcal{P}$ is defined as follows: For $1 \leq i \leq k$, let $S_i = \{b : 2^{-i} < \mathcal{P}(b) \leq 2^{-i+1}\}$ and let $S_0 = \{0,1\}^n \setminus \bigcup_{i \in [k]} S_i$. Given any distribution $\mathcal{D}$ over $\{0,1\}^n$, we define a distribution $B_{\mathcal{D}}$ over $[k] \cup \{0\}$ as: for $0 \leq i \leq k$, $B_{\mathcal{D}}(i) = \mathcal{D}(S_i)$. We call $B_{\mathcal{D}}$ the bucket distribution of $\mathcal{D}$ and S_i the i^{th} bucket.*

Barbarik3 takes as input two distributions $\mathcal{P}$ and $\mathcal{Q}$ defined over the support $\{0,1\}^n$, along with the parameters for closeness(ε), farness(η), and confidence(δ). On Line 1, Barbarik3 computes the value of k using η and the number of dimensions n. Then, using DUAL access to $\mathcal{P}$, and SAMP access to $\mathcal{Q}$, Barbarik3 creates bucket distributions $B_{\mathcal{P}}$ and $B_{\mathcal{Q}}$ as in Defn. 17, in the following way: To sample from $B_{\mathcal{P}}$, Barbarik3 first draws a sample $j \sim \mathcal{P}$, then using the DUAL oracle, determines the value of $\mathcal{P}(j)$. Then, if j lies in the i^{th} bucket i.e., $2^{-i} < \mathcal{P}(j) \leq 2^{-i+1}$, the algorithm takes sample i as the sample from $B_{\mathcal{P}}$. Similarly, to draw a sample from $B_{\mathcal{Q}}$, Barbarik3 draws a sample $j \sim \mathcal{Q}$ and then, using the DUAL oracle to find $\mathcal{P}(j)$, finds i such that j lies in the i^{th} bucket, and then uses i as the sample.

Barbarik3 then calls two subroutines, OutBucket (Section 4.1.4) and InBucket (Section 4.1.3). The OutBucket subroutine returns a θ-additive estimate of the TV distance between $B_{\mathcal{P}}$ and $B_{\mathcal{Q}}$, the two bucket distributions of $\mathcal{P}$ and $\mathcal{Q}$, with an error probability of at most $\delta/2$. If it is found on Line 9 that the estimate $\widehat{d}$ is greater than $\varepsilon/2 + \theta$, we know that $d_{TV}(\mathcal{P}, \mathcal{Q}) > \varepsilon/2$ and also that $d_{\infty}(\mathcal{P}, \mathcal{Q}) > \varepsilon$, and hence the algorithm returns Reject. Otherwise, the algorithm calls the InBucket subroutine.

Now suppose that $d_{TV}(\mathcal{P}, \mathcal{Q}) \geq \eta$. Then, for ε_2 (Line 11), it is either the case that $d_{TV}(B_{\mathcal{P}}, B_{\mathcal{Q}}) > \varepsilon_2$ or else $d_{TV}(B_{\mathcal{P}}, B_{\mathcal{Q}}) \leq \varepsilon_2$. In the former case, the algorithm returns Reject on Line 10, and in the latter case the InBucket subroutine returns Reject. In both cases, the failure probability is at most $\delta/2$. Thus Barbarik3 returns Reject on given η-far input distributions with probability at least $1 - \delta$.

We will prove the following theorem:

Theorem 4.1. Barbarik3$(\mathcal{P}, \mathcal{Q}, \eta, \varepsilon, \delta)$ *takes in distributions* $\mathcal{P}$ *and* $\mathcal{Q}$ *defined over* $\{0,1\}^n$*, and parameters* $\eta \in (0,1]$*,* $\varepsilon \in [0, \eta/11.6)$ *and* $\delta \in (0, 1/2]$*.* Barbarik3 *has* DUAL *access to* $\mathcal{P}$*, and* PCOND+SAMP *access to* $\mathcal{Q}$*. With probability at least* $1-\delta$*,* Barbarik3 *returns*

- Accept *if* $d_\infty(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$
- Reject *if* $d_{TV}(\mathcal{P}, \mathcal{Q}) > \eta$

Barbarik3 *has query complexity* $\tilde{O}\left(\frac{\sqrt{n}\log(n)}{\eta^3(\eta-11.6\varepsilon)} + \frac{n}{\eta^2}\right)$*, where* $\tilde{O}$ *hides polylog factors of* ε, η *and* δ*.*

We will use the following proposition in our analysis.

Proposition 4. *Given distributions* $\mathcal{P}$ *and* $\mathcal{Q}$ *supported on* $\{0,1\}^n$*, and a set* $S \subseteq \{0,1\}^n$*,*

$$\sum_{i \in S} \mathcal{P}(i)\mathcal{Q}(i) > \frac{(\mathcal{P}(S) + \mathcal{Q}(S) - 2d_{TV(S)}(\mathcal{P}, \mathcal{Q}))^2}{4|S|}$$

Proof. The Hellinger distance of distributions $\mathcal{P}, \mathcal{Q}$ restricted to a set $S \subseteq \{0,1\}^n$, is defined as $d_{H(S)}(\mathcal{P}, \mathcal{Q}) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i \in S}(\sqrt{\mathcal{Q}(i)} - \sqrt{\mathcal{P}(i)})^2}$,

$$\begin{aligned}
d_{H(S)}(\mathcal{P}, \mathcal{Q}) &= \frac{1}{\sqrt{2}}\sqrt{\sum_{i \in S}(\sqrt{\mathcal{Q}(i)} - \sqrt{\mathcal{P}(i)})^2} \\
d^2_{H(S)}(\mathcal{P}, \mathcal{Q}) &= \frac{1}{2}\sum_{i \in S}(\sqrt{\mathcal{Q}(i)} - \sqrt{\mathcal{P}(i)})^2 \\
&= \frac{1}{2}\sum_{i \in S}\left(\mathcal{Q}(i) + \mathcal{P}(i) - 2\sqrt{\mathcal{P}(i)\mathcal{Q}(i)}\right) \\
&= \frac{\mathcal{P}(S) + \mathcal{Q}(S)}{2} - \sum_{i \in S}\sqrt{\mathcal{P}(i)\mathcal{Q}(i)}
\end{aligned}$$

Then using the fact that $d^2_{H(S)}(\mathcal{P}, \mathcal{Q}) \leq d_{TV(S)}(\mathcal{P}, \mathcal{Q})$ we see that, $\sum_{i \in S}\sqrt{\mathcal{P}(i)\mathcal{Q}(i)} \geq \frac{\mathcal{P}(S)+\mathcal{Q}(S)}{2} - d_{TV(S)}(\mathcal{P}, \mathcal{Q})$. Then we use the Cauchy-Schwarz inequality:

$$\sum_{i \in S} \mathcal{P}(i)\mathcal{Q}(i) \geq \frac{(\mathcal{P}(S) + \mathcal{Q}(S) - 2d_{TV(S)}(\mathcal{P}, \mathcal{Q}))^2}{4|S|}$$

□

4.1.2 Lower Bound

The lower bound comes from the paper of Narayanan [74], where it appears in Theorem 1.6. Phrased in the jargon of our paper, the lower bound states that distinguishing between $d_{TV}(\mathcal{P}, \mathcal{Q}) > \eta$ and $d_\infty(\mathcal{P}, \mathcal{Q}) = 0$ requires $\tilde{\Omega}(\sqrt{n/\log(n)}/\eta^2)$

samples. Note that the lower bound is shown on a special case ($\varepsilon = 0$) of our problem. Hence the lower bound applies to our problem as well. Furthermore, the lower bound is shown for the case where distribution $\mathcal{P}$ provides full access, i.e., the algorithm can make arbitrary queries to $\mathcal{P}$. This is a stronger access model than DUAL. Since the lower bound is for a stronger access model, again it extends to our problem.

4.1.3 The InBucket Subroutine

In this section, we present the InBucket subroutine, whose behavior is stated in the following lemma.

Lemma 1. InBucket$(\mathcal{P}, \mathcal{Q}, k, \varepsilon, \varepsilon_2, \eta, \delta)$ *takes as input two distributions* $\mathcal{P}, \mathcal{Q}$*, an integer* k *and parameters* $\varepsilon, \varepsilon_2, \eta, \delta$*. If* $d_\infty(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$*,* InBucket *returns* Accept*. If* $d_{TV}(\mathcal{P}, \mathcal{Q}) \geq \eta$ *and* $d_{TV}(B_\mathcal{P}, B_\mathcal{Q}) < \varepsilon_2$*, then* InBucket *returns* Reject*.* InBucket *errs with probability at most* δ *.*

Before we dig into the analysis, we will present a high-level overview of the InBucket subroutine. The main task of InBucket is to take in two distributions $\mathcal{P}$ and $\mathcal{Q}$ and distinguish between the case where the distributions are ε-close to each other, and the case where the distributions are η-far with the added promise that if the distributions are far, there are sufficiently many witness pairs. A witness pair is a pair of elements $\{\sigma_1, \sigma_2\}$ such that their relative probabilities under $\mathcal{P}$ and $\mathcal{Q}$ are different, i.e.

$$\frac{\mathcal{P}(\sigma_1)/\mathcal{P}(\sigma_2)}{\mathcal{Q}(\sigma_1)/\mathcal{Q}(\sigma_2)} = 1 + \mathcal{O}(\varepsilon)$$

To find such a pair, our algorithm requires the elements to be from the same bucket, incurring a sample complexity of $\mathcal{O}(\sqrt{k})$, reflected on Lines 6 and 8. Then, given such a witness pair is found, InBucket makes use of the Bias subroutine to determine the relative probabilities, and we describe the same in the following subsection.

The Bias **subroutine** The Bias subroutine takes in distribution $\mathcal{Q}$, two elements p, q and a positive integer r. Then, using the PCOND oracle, Bias draws r samples from the conditional distribution $\mathcal{Q}_{\{p,q\}}$ and returns the number of times it sees p

Algorithm 5 InBucket$(\mathcal{P}, \mathcal{Q}, k, \varepsilon, \varepsilon_2, \eta, \delta)$

1: $\varepsilon_1 \leftarrow (0.99\eta - 3.25\varepsilon_2 - 2\varepsilon/(1-\varepsilon))/1.05 + 2\varepsilon/(1-\varepsilon)$
2: $m \leftarrow \lceil \sqrt{k}/(0.99\eta - 3.25\varepsilon_2 - \varepsilon_1) \rceil$
3: $\alpha \leftarrow (\varepsilon_1 + 2\varepsilon/(1-\varepsilon))/2$
4: $t \leftarrow \left\lceil \frac{\log_e(4/\delta)}{\log_e(10/(10-\varepsilon_1+\alpha))} \right\rceil$
5: **for** t iterations **do**
6: $\quad \Gamma_{\mathcal{P}} \leftarrow m$ samples from $\mathcal{P}$
7: $\quad \forall_{i\in[k]} \Gamma^i_{\mathcal{P}} \leftarrow \Gamma_{\mathcal{P}} \cap S_i$ {S_i is defined in Defn. 17}
8: $\quad \Gamma_{\mathcal{Q}} \leftarrow m$ samples from $\mathcal{Q}$
9: $\quad \forall_{i\in[k]} \Gamma^i_{\mathcal{Q}} \leftarrow \Gamma_{\mathcal{Q}} \cap S_i$
10: $\quad$ **for all** $j \in [k]$ s.t. $|\Gamma^j_{\mathcal{P}}|, |\Gamma^j_{\mathcal{Q}}| > 0$ **do**
11: $\quad\quad p \leftarrow \Gamma^j_{\mathcal{P}}$ {p is an arbitrary element from the set $\Gamma^j_{\mathcal{P}}$}
12: $\quad\quad q \leftarrow \Gamma^j_{\mathcal{Q}}$ {q is an arbitrary element from the set $\Gamma^j_{\mathcal{Q}}$}
13: $\quad\quad h \leftarrow \frac{\mathcal{P}(p)}{\mathcal{P}(p)+\mathcal{P}(q)(1+\frac{2\varepsilon}{1-\varepsilon})}$
14: $\quad\quad \ell \leftarrow \frac{\mathcal{P}(p)}{\mathcal{P}(p)+\mathcal{P}(q)(1+\alpha)}$
15: $\quad\quad r \leftarrow \left\lceil \frac{2\log_e(4mt/\delta)}{(h-\ell)^2} \right\rceil$
16: $\quad\quad \widehat{c} \leftarrow \mathsf{Bias}(Q, p, q, r)$
17: $\quad\quad$ **if** $\widehat{c} \le (h+\ell)/2$ **then**
18: $\quad\quad\quad$ **Return** Reject
19: **Return** Accept

Algorithm 6 Bias$(\mathcal{Q}, p, q, r)$

1: **if** p and q are identical **then**
2: $\quad$ **Return** 0.5
3: $\Gamma_{\mathcal{Q}_{\{p,q\}}} \leftarrow r$ samples from $\mathcal{Q}_{\{p,q\}}$
4: **Return** the fraction of times p appears in $\Gamma_{\mathcal{Q}_{\{p,q\}}}$

in the r samples. It can be seen that the returned value is an empirical estimate of $\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)}$. Let the estimate be $\widehat{c_{pq}}$. We use the Hoeffding bound in Cor. 1, and the value of r from Line 15 of Alg. (5) to show that:

$$\Pr\left[\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} - \widehat{c_{pq}} \ge \frac{h-\ell}{2}\right] \le \frac{\delta}{4mt} \qquad \Pr\left[\widehat{c_{pq}} - \frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} \ge \frac{h-\ell}{2}\right] \le \frac{\delta}{4mt}$$

Here t represents the number of iterations of the outer loop (Line 4), and m is the number of samples drawn from $B_{\mathcal{P}}$ and $B_{\mathcal{Q}}$. Together, there are at most mt pairs of samples that are passed to the Bias oracle. Since in each invocation of Bias, the probability of error is $\delta/4mt$, using the union bound we find that the probability that all mt Bias calls return correctly is at least $1 - \delta/4$ and thus with probability at least $1 - \delta/4$, the empirical estimate $\widehat{c_{pq}}$ is closer than $(h-\ell)/2$ to

$\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)}$. Henceforth we assume:

$$\left|\widehat{c_{pq}} - \frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)}\right| \leq \frac{h-\ell}{2} \tag{4.1}$$

4.1.3.1 The Accept case

In this section we will provide an analysis of the case when $d_\infty(\mathcal{P},\mathcal{Q}) < \varepsilon$. We will now prove a proposition required for the remaining proofs.

Proposition 5. *Let $\mathcal{P},\mathcal{Q}$ be distributions and let $p \sim \mathcal{P}$ and $q \sim \mathcal{Q}$. Then,*

1. *If $d_\infty(\mathcal{P},\mathcal{Q}) < \varepsilon$ then*

$$\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} \geq \frac{\mathcal{P}(p)}{\mathcal{P}(p)+(1+\frac{2\varepsilon}{1-\varepsilon})\mathcal{P}(q)}$$

2. *If $d_{TV}(\mathcal{P},\mathcal{Q}) > \varepsilon_1$, then for $0 \leq \alpha < \varepsilon_1$, with probability at least $(d_{TV}(\mathcal{P},\mathcal{Q}) - \alpha)/2$,*

$$\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} < \frac{\mathcal{P}(p)}{\mathcal{P}(p)+(1+\alpha)\mathcal{P}(q)}$$

Proof. If $d_\infty(\mathcal{P},\mathcal{Q}) < \varepsilon$ then

$$\begin{aligned}\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} &\geq \frac{\mathcal{P}(p)(1-\varepsilon)}{\mathcal{P}(p)(1-\varepsilon)+(1+\varepsilon)\mathcal{P}(q)} \\ &= \frac{\mathcal{P}(p)}{\mathcal{P}(p)+(1+\frac{2\varepsilon}{1-\varepsilon})\mathcal{P}(q)}\end{aligned}$$

and hence we show the first part of the claim.

For the second part of the proof we introduce the some sets. Let $H_0 = \{h | 1 \leq \frac{\mathcal{Q}(h)}{\mathcal{P}(h)} < 1+\alpha\}$ and $H_1 = \{h | 1+\alpha \leq \frac{\mathcal{Q}(h)}{\mathcal{P}(h)}\}$ and $H = H_0 \cup H_1$. Similarly define, $L_0 = \{\ell | 1-\alpha < \frac{\mathcal{Q}(\ell)}{\mathcal{P}(\ell)} < 1\}$, $L_1 = \{\ell | \frac{\mathcal{Q}(\ell)}{\mathcal{P}(\ell)} \leq 1-\alpha\}$ and $L = L_0 \cup L_1$.

Now consider that we have a pair of samples, $p \sim \mathcal{P}$ and $q \sim \mathcal{Q}$. We know that either $\mathcal{P}(L) \geq 1/2$ or $\mathcal{P}(H) > 1/2$.

$\mathcal{P}(L) \geq 1/2$: We see that $\Pr[p \in L] \geq 1/2$. Then from the definition of H_0, $\mathcal{Q}(H_0) - \mathcal{P}(H_0) < \alpha$ and recall that $\mathcal{Q}(H) - \mathcal{P}(H) = d_{TV}(\mathcal{P},\mathcal{Q})$. Thus we have that

$\mathcal{Q}(H_1) - \mathcal{P}(H_1) > d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha$ and hence $\Pr[q \in H_1] > d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha$. We can now confirm that $q \in H_1 \wedge p \in L$ with probability at least $(d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha)/2$. Then,

$$\begin{aligned}\frac{\mathcal{Q}(p)}{\mathcal{Q}(p) + \mathcal{Q}(q)} &< \frac{\mathcal{P}(p)}{\mathcal{P}(p) + \mathcal{Q}(q)} \quad \text{(From } \mathcal{P}(p) > \mathcal{Q}(p)\text{)} \\ &< \frac{\mathcal{P}(p)}{\mathcal{P}(p) + (1+\alpha)\mathcal{P}(q)} \quad \text{(Since } q \in H_1\text{)}\end{aligned}$$

$\mathcal{P}(H) > 1/2$**:** We see that $\Pr[q \in H] \geq 1/2$. Then we have that $\mathcal{P}(L_0) - \mathcal{Q}(L_0) < \alpha$ and also that $\mathcal{P}(L) - \mathcal{Q}(L) = d_{TV}(\mathcal{P}, \mathcal{Q})$, we have that $\mathcal{P}(L_1) - \mathcal{Q}(L_1) < d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha$. Then, we deduce that probability at least $(d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha)/2$, $q \in H \wedge p \in L_1$. Then,

$$\begin{aligned}\frac{\mathcal{Q}(p)}{\mathcal{Q}(p) + \mathcal{Q}(q)} &< \frac{\mathcal{Q}(p)}{\mathcal{Q}(p) + \mathcal{P}(q)} \quad \text{(From } \mathcal{P}(q) < \mathcal{Q}(q)\text{)} \\ &< \frac{\mathcal{P}(p)(1-\alpha)}{\mathcal{P}(p)(1-\alpha) + \mathcal{P}(q)} \quad \text{(Since } p \in L_1\text{)} \\ &< \frac{\mathcal{P}(p)}{\mathcal{P}(p) + (1+\alpha)\mathcal{P}(q)}\end{aligned}$$

□

From our assumption (4.1), we know that for all invocations of Bias, with probability at least $1 - \delta/4$, $\left|\widehat{c_{pq}} - \frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)}\right| \leq (h - \ell)/2$. Using Prop. 5, and using the value of h given on Line 13, we can see that $\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} > h$. From this we can observe that for all invocations of Bias, $\widehat{c_{pq}} > (h + \ell)/2$ and the test does not return Reject in any iteration, hence eventually returning Accept. Thus, in the case that $d_\infty(\mathcal{P}, \mathcal{Q}) < \varepsilon$, the InBucket subroutine returns Accept with probability at least $1 - \delta/4$.

4.1.3.2 The Reject case

In this section we analyse the case when $d_{TV}(\mathcal{P}, \mathcal{Q}) \geq \eta$ and $d_{TV}(B_\mathcal{P}, B_\mathcal{Q}) \leq \varepsilon_2$ and we will show that the algorithm returns Reject with probability at least $1 - \delta$. For the purpose of the proof we will define a set of bad buckets $Bad \subseteq [k]$. Note that bucket $\{0\}$ is not in Bad.

Definition 18. $Bad = \{i \in [k] : d_{TV}(\mathcal{P}_{S_i}, \mathcal{Q}_{S_i}) > \varepsilon_1 \wedge B_\mathcal{P}(i)/B_\mathcal{Q}(i) \in [0.2, 2]\}$

Suppose we have an indicator variable $X_{r,s}$ constructed as follows: draw m samples from $\mathcal{P}$ and $\mathcal{Q}$, and if the r^{th} sample from $\mathcal{P}$ and the s^{th} sample from $\mathcal{Q}$ both belong to some bucket $b \in Bad$, then $X_{r,s} = 1$ else $X_{r,s} = 0$. Then,

$$\mathbb{E}[X_{r,s}] = \sum_{b \in Bad} B_{\mathcal{P}}(b) B_{\mathcal{Q}}(b) > \frac{(B_{\mathcal{P}}(Bad) + B_{\mathcal{Q}}(Bad) - 2d_{TV(Bad)}(B_{\mathcal{P}}, B_{\mathcal{Q}}))^2}{4k} \quad (4.2)$$

The inequality is by the application of Prop. 4.

We analyse the expression that appears in the expectation bound above in the following lemma.

Lemma 2.

$$B_{\mathcal{Q}}(Bad) + B_{\mathcal{P}}(Bad) - 2d_{TV(Bad)}(B_{\mathcal{Q}}, B_{\mathcal{P}}) > 2\left(0.99\eta - \frac{13}{4}\varepsilon_2 - \varepsilon_1\right)$$

Proof. Let $\mathcal{PQ}$ be a distribution constructed from $\mathcal{P}$ and $\mathcal{Q}$, where we first sample $j \sim B_{\mathcal{Q}}$ and then sample $i \sim \mathcal{P}_{S_j}$, thus $\mathcal{PQ}(i) = \sum_{j \in [k] \cup \{0\}} B_{\mathcal{Q}}(j) \mathcal{P}_{S_j}(i)$. We know that if $i \in S_j$, then $i \notin S_{j'}$ for $j' \neq j$. This allows us to simplify and write $\mathcal{PQ}(i) = B_{\mathcal{Q}}(j) \mathcal{P}_{S_j}(i)$. Then,

$$\begin{aligned}
d_{TV}(B_{\mathcal{P}}, B_{\mathcal{Q}}) &= \frac{1}{2} \sum_{j \in [k] \cup \{0\}} |B_{\mathcal{P}}(j) - B_{\mathcal{Q}}(j)| \\
&= \frac{1}{2} \sum_{j \in [k] \cup \{0\}} \sum_{i \in S_j} \mathcal{P}_{S_j}(i) |B_{\mathcal{P}}(j) - B_{\mathcal{Q}}(j)| \\
&= \frac{1}{2} \sum_{j \in [k] \cup \{0\}} \sum_{i \in S_j} |\mathcal{P}(i) - \mathcal{PQ}(i)| \\
&= \frac{1}{2} \sum_{i \in \{0,1\}^n} |\mathcal{P}(i) - \mathcal{PQ}(j)| = d_{TV}(\mathcal{P}, \mathcal{PQ})
\end{aligned}$$

Since $d_{TV}(B_{\mathcal{P}}, B_{\mathcal{Q}}) < \varepsilon_2$, we have $d_{TV}(\mathcal{P}, \mathcal{PQ}) < \varepsilon_2$.

From the definition of TV, we have

$$\begin{aligned}
d_{TV}(\mathcal{Q}, \mathcal{P}\mathcal{Q}) =& \frac{1}{2} \sum_{i \in \{0,1\}^n} |\mathcal{Q}(i) - \mathcal{P}\mathcal{Q}(i)| \\
=& \frac{1}{2} \sum_{j \in [k] \cup \{0\}} \sum_{i \in S_j} |\mathcal{Q}(i) - \mathcal{P}\mathcal{Q}(i)| \\
=& \frac{1}{2} \sum_{j \in [k] \cup \{0\}} \sum_{i \in S_j} |B_{\mathcal{Q}}(j)\mathcal{Q}_{S_j}(i) - B_{\mathcal{Q}}(j)\mathcal{P}_{S_j}(i)| \\
=& \frac{1}{2} \sum_{j \in [k] \cup \{0\}} B_{\mathcal{Q}}(j) \sum_{i \in S_j} |\mathcal{Q}_{S_j}(i) - \mathcal{P}_{S_j}(i)| \\
=& \sum_{j \in ([k] \cup \{0\})} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) \\
=& \sum_{j \in ([k] \cup \{0\}) \setminus Bad} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) + \sum_{j \in Bad} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j})
\end{aligned}$$

We will need the following sets:

$$R_1 = \{j : B_{\mathcal{P}}(j) > 2B_{\mathcal{Q}}(j)\} \quad R_2 = \{j : B_{\mathcal{Q}}(j) > 5B_{\mathcal{P}}(j)\}$$

From the triangle inequality we have $d_{TV}(\mathcal{P}, \mathcal{Q}) \leq d_{TV}(\mathcal{P}, \mathcal{P}\mathcal{Q}) + d_{TV}(\mathcal{P}\mathcal{Q}, \mathcal{Q})$. We also know that $d_{TV}(\mathcal{P}, \mathcal{P}\mathcal{Q}) < \varepsilon_2$ and $d_{TV}(\mathcal{P}, \mathcal{Q}) > \eta$. Thus we have:

$$\begin{aligned}
\eta - \varepsilon_2 &< d_{TV}(Q, PQ) \\
\eta - \varepsilon_2 &< \sum_{j \in ([k] \cup \{0\}) \setminus Bad} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) + \sum_{j \in Bad} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) \\
\eta - \varepsilon_2 &< \sum_{j \in \{0\} \cup R_1 \cup R_2} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) + \sum_{j \in [k] \setminus \{R_1 \cup R_2 \cup Bad\}} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) \\
&\quad + \sum_{j \in Bad} B_{\mathcal{Q}}(j) d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j})
\end{aligned}$$

Since from the definition, we know that if $j \in [k] \setminus \{R_1 \cup R_2 \cup Bad\}$, then $d_{TV}(\mathcal{P}_{S_j}, \mathcal{Q}_{S_j}) \leq \varepsilon_1$, we have

$$\begin{aligned}
\eta - \varepsilon_2 < \sum_{j \in \{0\} \cup R_1 \cup R_2} B_{\mathcal{Q}}(j) + \sum_{j \in [k] \setminus \{R_1 \cup R_2 \cup Bad\}} B_{\mathcal{Q}}(j)\varepsilon_1 + \sum_{j \in Bad} B_{\mathcal{Q}}(j) \\
\eta - \varepsilon_2 - \varepsilon_1 < B_{\mathcal{Q}}(\{0\} \cup R_1 \cup R_2) + B_{\mathcal{Q}}(Bad) \\
\eta - \varepsilon_2 - \varepsilon_1 - B_{\mathcal{Q}}(\{0\} \cup R_1 \cup R_2) < B_{\mathcal{Q}}(Bad) \qquad (4.3)
\end{aligned}$$

If $i \in R_1$, then $B_{\mathcal{P}}(i) > 2B_{\mathcal{Q}}(i)$, and thus $B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i) > B_{\mathcal{Q}}(i)$. Thus,

$$B_{\mathcal{Q}}(R_1) < \sum_{i \in R_1} (B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i)) \qquad (4.4)$$

If $i \in R_2$, then $B_{\mathcal{Q}}(i) > 5B_{\mathcal{P}}(i)$, and thus $B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i) > 4B_{\mathcal{P}}(i)$, giving

$$\begin{aligned} B_{\mathcal{P}}(R_2) &< \frac{1}{4}\sum_{i\in R_2}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) \\ B_{\mathcal{P}}(R_2) + \sum_{i\in R_2}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) &< \frac{5}{4}\sum_{i\in R_2}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) \\ B_{\mathcal{Q}}(R_2) &< \frac{5}{4}\sum_{i\in R_2}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) \end{aligned} \tag{4.5}$$

Since $|S_0| \leq 2^n$ and all elements $i \in S_0$ satisfy $\mathcal{P}(i) \leq 2^{-k}$, we have $B_{\mathcal{P}}(0) \leq 2^{n-k}$, where we substitute $k = n + \log_2(100/\eta)$ to get

$$B_{\mathcal{P}}(0) \leq \frac{\eta}{100} \tag{4.6}$$

Then,

$$\begin{aligned} &B_{\mathcal{Q}}(\{0\} \cup R_1 \cup R_2) = B_{\mathcal{Q}}(\{0\}) + B_{\mathcal{Q}}(R_1) + B_{\mathcal{Q}}(R_2) \\ &\leq \frac{\eta}{100} + \sum_{i\in\{0\}}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) + \sum_{i\in R_1}(B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i)) + \frac{5}{4}\sum_{i\in R_2}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) \end{aligned} \tag{4.7}$$

The last inequality is from the use of inequations (4.4),(4.5) and (4.6). Here we partition the set $Bad \cup \{0\}$ into two sets Bad^+ and Bad^-, where $Bad^+ = \{i \in Bad \cup \{0\} | B_{\mathcal{P}}(i) \geq B_{\mathcal{Q}}(i)\}$ and similarly $Bad^- = \{i \in Bad \cup \{0\} | B_{\mathcal{P}}(i) < B_{\mathcal{Q}}(i)\}$.

$$\begin{aligned} &B_{\mathcal{Q}}(Bad) + B_{\mathcal{P}}(Bad) - 2d_{TV(Bad)}(B_{\mathcal{Q}}, B_{\mathcal{P}}) \\ &\qquad \geq 2(B_{\mathcal{Q}}(Bad) - 2d_{TV(Bad)}(B_{\mathcal{P}}, B_{\mathcal{Q}})) \\ \text{(From 4.3)} \quad &> 2\left(\eta - \varepsilon_2 - \varepsilon_1 - B_{\mathcal{Q}}(\{0\} \cup R_1 \cup R_2) - \sum_{i\in Bad}|B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i)|\right) \\ \text{(From 4.7)} \quad &> 2\left(.99\eta - \varepsilon_2 - \varepsilon_1 - \sum_{i\in R_1\cup Bad^+}(B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i)) - \frac{5}{4}\sum_{i\in R_2\cup Bad^-}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i))\right) \end{aligned}$$

Using the following two facts

$$d_{TV}(B_{\mathcal{P}}, B_{\mathcal{Q}}) = \sum_{i:B_{\mathcal{P}}(i)\leq B_{\mathcal{Q}}(i)}(B_{\mathcal{Q}}(i) - B_{\mathcal{P}}(i)) = \sum_{i:B_{\mathcal{P}}(i)>B_{\mathcal{Q}}(i)}(B_{\mathcal{P}}(i) - B_{\mathcal{Q}}(i)) = \varepsilon_2$$

and

$$\forall_{i\in R_1} B_{\mathcal{P}}(i) > B_{\mathcal{Q}}(i) \qquad \forall_{i\in R_2} B_{\mathcal{Q}}(i) > B_{\mathcal{P}}(i)$$

we have,

$$B_{\mathcal{Q}}(Bad) + B_{\mathcal{P}}(Bad) - 2d_{TV(Bad)}(B_{\mathcal{Q}}, B_{\mathcal{P}}) > 2\left(0.99\eta - \frac{13}{4}\varepsilon_2 - \varepsilon_1\right)$$

□

Using Equation 4.2 along with Lemma 2 we derive the fact that

$$\mathbb{E}[X_{r,s}] > \left(0.99\eta - \frac{13}{4}\varepsilon_2 - \varepsilon_1\right)^2 / k$$

Let $X = \sum_{r,s\in[m]} X_{r,s}$. Given m samples from $\mathcal{P}$ and $\mathcal{Q}$, $\Pr(X \geq 1)$ is the probability that there is at least one bucket in Bad(among k buckets) that is sampled at least once each in both sets of samples.

Lemma 3. $\mathbb{E}[X] \geq 1$

Proof. Recall that we defined the value of m on Line 2 of Alg. 5

$$m = \lceil \sqrt{k}/(0.99\eta - 3.25\varepsilon_2 - \varepsilon_1) \rceil$$

Thus we have that,

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{r,s\in[m]} X_{r,s}\right] = m^2\mathbb{E}[X_{r,s}] \geq 1$$

□

Lemma 4. $\Pr(X \geq 1) > 1/5$

Proof. Recall that for all $r, s \in [m]$, $\mathbb{E}[X_{r,s}] = \sum_{b\in Bad} B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)$. Then since $X = \sum_{r,s\in[m]} X_{r,s}$,

$$\mathbb{E}[X] = \sum_{r,s\in[m]} \mathbb{E}[X_{r,s}] = m^2\mathbb{E}[X_{r,s}]$$

Then for $i, j, k, l \in [m]$,

- if $i = k, j = l$ then $\mathbb{E}[X_{i,j}X_{k,l}] = \sum_{b\in Bad} B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b) = \mathbb{E}[X_{r,s}]$
- if $i = k, j \neq l$ then $\mathbb{E}[X_{i,j}X_{k,l}] = \sum_{b\in Bad} B_{\mathcal{P}}(b)B^2_{\mathcal{Q}}(b)$
- if $i \neq k, j = l$ then $\mathbb{E}[X_{i,j}X_{k,l}] = \sum_{b\in Bad} B^2_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)$
- if $i \neq k, j \neq l$ then $\mathbb{E}[X_{i,j}X_{k,l}] = \left(\sum_{b\in Bad} B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^2 = \mathbb{E}[X_{r,s}]^2$

$$
\begin{aligned}
\mathbb{E}[X^2] &= \mathbb{E}\left[\sum_{i,j,k,l\in[m]} X_{i,j}X_{k,l}\right] \\
&= \mathbb{E}\left[\sum_{\substack{a\neq c, b\neq d \\ i,j,k,l\in[m]}} X_{i,j}X_{k,l}\right] + \mathbb{E}\left[\sum_{\substack{a= c, b\neq d \\ i,j,k,l\in[m]}} X_{i,j}X_{k,l}\right] \\
&\quad + \mathbb{E}\left[\sum_{\substack{a\neq c, b= d \\ i,j,k,l\in[m]}} X_{i,j}X_{k,l}\right] + \mathbb{E}\left[\sum_{\substack{a= c, b= d \\ i,j,k,l\in[m]}} X_{i,j}X_{k,l}\right] \\
&= m^2(m-1)^2\mathbb{E}[X_{r,s}]^2 + m^2(m-1)\left(\sum_{b\in Bad}(B_{\mathcal{P}}(b)+B_{\mathcal{Q}}(b))B_{\mathcal{P}}(i)B_{\mathcal{Q}}(i)\right) + m^2\mathbb{E}[X_{r,s}] \\
&\leq m^4\mathbb{E}[X_{r,s}]^2 + m^3\left(\sum_{b\in Bad}(B_{\mathcal{P}}(b)+B_{\mathcal{Q}}(b))B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right) + m^2\mathbb{E}[X_{r,s}]
\end{aligned}
$$

Then,

$$
\begin{aligned}
\frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]} &> \frac{m^4\mathbb{E}[X_{r,s}]^2}{m^4\mathbb{E}[X_{r,s}]^2 + m^3\left(\sum_{b\in Bad}(B_{\mathcal{P}}(b)+B_{\mathcal{Q}}(i))B_{\mathcal{P}}(i)B_{\mathcal{Q}}(b)\right) + m^2\mathbb{E}[X_{r,s}]} \\
&= \frac{1}{1 + m^{-1}\left(\frac{\sum_{b\in Bad}(B_{\mathcal{P}}(b)+B_{\mathcal{Q}}(b))B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^2}\right) + m^{-2}\mathbb{E}[X_{r,s}]^{-1}}
\end{aligned}
$$

We will now focus on finding the maxima for the large ratio in the denominator:

$$
\begin{aligned}
\frac{\sum_{b\in Bad}(B_{\mathcal{P}}(b)+B_{\mathcal{Q}}(b))B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^2} &= \frac{\sum_{b\in Bad}\left(\sqrt{\frac{B_{\mathcal{P}}(b)}{B_{\mathcal{Q}}(b)}}+\sqrt{\frac{B_{\mathcal{Q}}(b)}{B_{\mathcal{P}}(b)}}\right)(B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b))^{3/2}}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^2} \\
(b\in Bad \implies B_{\mathcal{P}}(b)/B_{\mathcal{Q}}(b)\in[0.2,2]) \quad &\leq \frac{(\sqrt{1/5}+\sqrt{5})\sum_{b\in Bad}(B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b))^{3/2}}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^2} \\
&< 3\cdot\frac{\sum_{b\in Bad}(B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b))^{3/2}}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^{3/2}\cdot\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^{1/2}} \\
\text{(Using the monotonicity of } \ell_p \text{ norms)} \quad &\leq \frac{3}{\left(\sum_{b\in Bad}B_{\mathcal{P}}(b)B_{\mathcal{Q}}(b)\right)^{1/2}} = 3\mathbb{E}[X_{r,s}]^{-1/2}
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]} &> \frac{1}{1+3m^{-1}\mathbb{E}[X_{r,s}]^{-1/2}+m^{-2}\mathbb{E}[X_{r,s}]^{-1}} \\
&> \frac{1}{5} \qquad (\text{Since } m^2\mathbb{E}[X_{r,s}]\geq 1)
\end{aligned}
$$

The Chebyshev bound (Proposition 2) tells us that

$$\Pr[|X - \mathbb{E}[X]| < \mathbb{E}[X]] > \mathbb{E}[X]^2/\mathbb{E}[X^2] > \frac{1}{5}$$

Recall that from Lemma 3, we have $\mathbb{E}[X] \geq 1$ which lets draw the following implication

$$|X - \mathbb{E}[X]| < \mathbb{E}[X] \implies X > 0$$

which finally gives us the claim:

$$\Pr[X > 0] > \frac{1}{5}$$
$$\Pr[X \geq 1] > \frac{1}{5} \qquad \text{(Since } X \text{ takes only integer values)}$$

□

Henceforth we will condition on the the event that $X \geq 1$. In such a case, we know that for some $k \in Bad$, there is a sample $p \sim \mathcal{P}_{S_k}$ and a sample $q \sim \mathcal{Q}_{S_k}$. Then for such a pair of samples (p, q), and some α, Prop. 5 tells us that with probability at least $(d_{TV}(\mathcal{P}, \mathcal{Q}) - \alpha)/2$ we have

$$\frac{\mathcal{Q}(p)}{\mathcal{Q}(p) + \mathcal{Q}(q)} < \frac{\mathcal{P}(p)}{\mathcal{P}(p) + (1+\alpha)\mathcal{P}(q)}$$

Using the assumption made in (4.1), we immediately have that $\widehat{c_{pq}} \leq \frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} + \frac{h-\ell}{2}$. From Prop. 5 we have that $\frac{\mathcal{Q}(p)}{\mathcal{Q}(p)+\mathcal{Q}(q)} < \ell$ and hence $\widehat{c_{pq}} < (h+\ell)/2$. Since $d_{TV}(\mathcal{P}, \mathcal{Q}) \geq \varepsilon_1$, we see that if $X \geq 1$, then with probability at least $(\varepsilon_1 - \alpha)/2$, the iteration returns Reject.

Then, using Lemma 4 we see that in every iteration, with probability at least $(\varepsilon_1 - \alpha)/10$, InBucket returns Reject. There are t iterations, where t (line 4) is chosen such that the overall probability of the test returning Reject is at least $1 - \delta/2$.

4.1.4 The OutBucket Subroutine

Algorithm 7 OutBucket$(B_{\mathcal{P}}, B_{\mathcal{Q}}, k, \theta, \delta)$

1: Sample $\max\left(\frac{4(k+1)}{\theta^2}, \frac{8\log_e(4/\delta)}{\theta^2}\right)$ times from $B_{\mathcal{P}}$ and $B_{\mathcal{Q}}$ and construct empirical distributions $\widehat{B_{\mathcal{P}}}$ and $\widehat{B_{\mathcal{Q}}}$.
2: **Return** $d_{TV}(\widehat{B_{\mathcal{P}}}, \widehat{B_{\mathcal{Q}}})$

The OutBucket subroutine takes as input two distributions $\mathcal{D}_1, \mathcal{D}_2$ over $k+1$ elements and two parameters θ and δ. Then with probability at least $1-\delta$, InBucket returns a θ-additive estimate for $d_{TV}(\mathcal{D}_1, \mathcal{D}_2)$.

The OutBucket starts by drawing $\max\left(\frac{4(k+1)}{\theta^2}, \frac{8\log_e(4/\delta)}{\theta^2}\right)$ samples from the two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, and constructs the empirical distributions $\widehat{\mathcal{D}_1}$ and $\widehat{\mathcal{D}_2}$. Then from Prop. 3, we know that with probability at least $1-\delta$, both $d_{TV}(\mathcal{D}_1, \widehat{\mathcal{D}_1}) \leq \theta/2$ and $d_{TV}(\mathcal{D}_2, \widehat{\mathcal{D}_2}) \leq \theta/2$.

From the triangle inequality we have that,

$$d_{TV}(\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2}) \leq d_{TV}(\mathcal{D}_1, \widehat{\mathcal{D}_1}) + d_{TV}(\mathcal{D}_2, \widehat{\mathcal{D}_2}) + d_{TV}(\mathcal{D}_1, \mathcal{D}_2) < \theta + d_{TV}(\mathcal{D}_1, \mathcal{D}_2)$$

and also that,

$$d_{TV}(\mathcal{D}_1, \mathcal{D}_2) \leq d_{TV}(\mathcal{D}_1, \widehat{\mathcal{D}_1}) + d_{TV}(\mathcal{D}_2, \widehat{\mathcal{D}_2}) + d_{TV}(\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2}) < \theta + d_{TV}(\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2})$$

Thus with probability at least $1-\delta$, the returned estimate $d_{TV}(\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2})$ satisfies $|d_{TV}(\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2}) - d_{TV}(\mathcal{D}_1, \mathcal{D}_2)| < \theta$.

Query and runtime complexity The number of queries made by OutBucket to $\mathcal{P}$ and $\mathcal{Q}$ is given by $\tilde{O}\left(\frac{n}{\eta^2}\right)$, where $\tilde{O}$ hides polylog factors of ε, η and δ. The number of queries required by InBucket is given by mtr. Bounding the terms individually, we see that $m = \tilde{O}\left(\frac{\sqrt{n}}{\eta - 11.6\varepsilon}\right)$, $t = \tilde{O}\left(\frac{1}{\eta}\right)$ and $r = \tilde{O}\left(\frac{\log n}{\eta^2}\right)$. Thus $mtr = \tilde{O}\left(\frac{\sqrt{n}\log n}{(\eta-11.6\varepsilon)\eta^3}\right)$ and hence the total query complexity is $\tilde{O}\left(\frac{\sqrt{n}\log n}{(\eta-11.6\varepsilon)\eta^3} + \frac{n}{\eta^2}\right)$.

4.2 Evaluation

To evaluate the performance of Barbarik3 and test the quality of publicly available samplers, we implemented Barbarik3 in Python. Our evaluation took inspiration from the experiments presented in previous work [26, 71], and we used the same framework to evaluate our proposed algorithm.

In the previous chapter we provided and analysed an algorithm that could test whether an input distribution $\mathcal{Q}$ was close to the target $\mathcal{P}$, where the target distribution was expected to have EVAL and SAMP accesses implemented. In our experiments, the role of the target distribution $\mathcal{P}$ was played by WAPS[2] [56].

[2] https://github.com/meelgroup/WAPS

WAPS compiles the input Boolean formula into a representation that allows exact sampling and exact probability computation, thereby implementing the SAMP and EVAL oracles needed for our test.

For the role of sampler $\mathcal{Q}(\varphi, \mathtt{w})$, we used the state-of-the-art samplers wSTS and wUnigen. wUnigen [88] is a hashing-based sampler that provides (ε, δ) guarantees on the quality of the samples. wSTS [49] is a sampler designed for sampling over challenging domains such as energy barriers and highly asymmetric spaces. wSTS generates samples much faster than wUnigen, albeit without any guarantees on the quality of the samples. In our previous experiments we had evaluated wQuicksampler, however we chose to not evaluate wQuicksampler in this series of experiments, as wQuicksampler was rejected immediately in all cases.

To implement PCOND access, we use the Kernel construction from [26]. Kernel takes in φ and two assignments σ_1, σ_2, and returns a function $\widehat{\varphi}$ on m variables, such that: (1) $m > n$, (2) φ and $\widehat{\varphi}$ are similar in structure, and (3) for $\sigma \in \widehat{\varphi}^{-1}(1)$, it holds that $\sigma_{\downarrow supp(\varphi)} \in \{\sigma_1, \sigma_2\}$. Here $\sigma_{\downarrow supp(\varphi)}$ denotes the projection of σ on the variables of φ.

For the closeness(ε), farness(η), and confidence(δ) parameters, we choose the values $0.05, 0.9$ and 0.2. This setting implies that for a given distribution $\mathcal{P}$, and for a given sampler $\mathcal{Q}(\varphi, \mathtt{w})$, Barbarik3 returns (1) Accept if $d_\infty(\mathcal{P}, \mathcal{Q}(\varphi, \mathtt{w})) < 0.05$, and (2) Reject if $d_{TV}(\mathcal{P}, \mathcal{Q}(\varphi, \mathtt{w})) > 0.9$, with probability at least 0.8. Our empirical evaluation sought to answer the question: How does the performance of Barbarik3 compare with the state-of-the-art tester Barbarik2?

Our experiments were conducted on a high-performance compute cluster with Intel Xeon(R) E5-2690v3@2.60GHz CPU cores. We use a single core with 4GB memory with a timeout of 16 hours for each benchmark. We set a sample limit of 10^8 samples for our experiments due to our limited computational resources. The complete experimental data along with the running time of instances, is presented in the Appendix A.2.

4.2.1 Setting A - scalable benchmarks

Dataset Our dataset consists of the union of two n-dimensional product distributions, for $n \in \{4, 7, 10, \ldots, 118\}$. We have 39 problems in the dataset.

We represent the union of two product distributions as the constraint: $\varphi(\sigma) = \bigwedge_{i=1}^{2k}(\sigma_{3k+1} \vee \sigma_i) \wedge \bigwedge_{i=2k+1}^{3k}(\neg\sigma_{3k+1} \vee \sigma_i)$, and the weight function: $\mathtt{w}(\sigma) = \prod_{i=2k+1}^{3k} 3^{\sigma_i}$, where σ_i is the value of σ at position i.

Results We observe that in the case of wSTS, Barbarik2 can handle only 12 instances within the sample limit of 10^8. On the other hand, Barbarik3 can handle all 39 instances using at the most 10^6 samples. In the case of wUnigen, Barbarik2 solves 5 instances, and Barbarik3 can handle 17 instances.

Figure 4.1 shows a cactus plot comparing the sample requirement of Barbarik3 and Barbarik2. The x-axis represents the number of benchmarks and y-axis represents the number of samples, a point (x, y) implies that the relevant tester took less than y number of samples to distinguish between $d_{TV}(\mathcal{P}, \mathcal{Q}(\varphi, \mathtt{w})) > \eta$ and $d_{\infty}(\mathcal{P}, \mathcal{Q}(\varphi, \mathtt{w})) < \varepsilon$, for x many benchmarks. We display the set of benchmarks for which at least one of the two tools terminated within the sample limit of 10^8. We want to highlight that the y-axis is in log-scale, thus showing the sample efficiency of Barbarik3 compared to Barbarik2. For every benchmark, we compute the ratio of the number of samples required by Barbarik2 to test a sampler and the number of samples required by Barbarik3. The geometric mean of these ratios indicates the mean speedup. We find that the Barbarik3's speedup on wSTS is $451\times$ and on wUnigen is $10\times$.

4.2.2 Setting B - real-life benchmarks

Dataset We experiment on 87 constraints drawn from a collection of publicly available benchmarks arising from sampling and counting tasks[3]. We use distributions from the log-linear family. In a log-linear distribution, the probability of an element $\sigma \in \varphi^{-1}(1)$ is given as: $\Pr[\sigma] \propto \exp\left(\sum_{i=1}^{n} \sigma_i \theta_i\right)$, where $\theta_i \in \mathbb{R}^n_{>0}$. We found that wUnigen was not able to sample from most of the benchmarks in the dataset within the given time limit, and hence we present the results only for wSTS.

Results We find that Barbarik3 terminated with a result on all 87 instances from the set of real-life benchmarks, while Barbarik2 could only terminate on 16. We

[3] https://zenodo.org/record/3793090

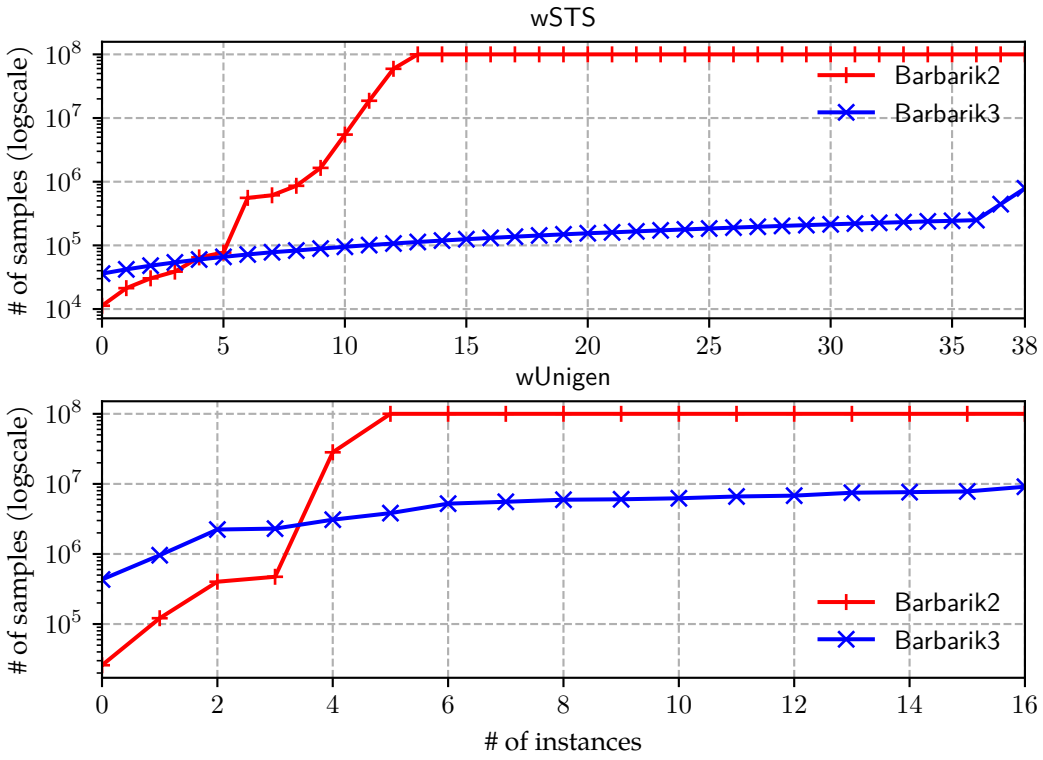


Figure 4.1: Cactus plot: Barbarik3 vs. Barbarik2. We set the sample limit to be 10^8, and our dataset consists of 39 benchmarks. The plot shows all the instances where at least one of the two tools terminated within the time limit of 16 hours and sample limit of 10^8.

present the results of our experiments in Table 4.1. The first column indicates the benchmark's name, and the second column has the number of dimensions of the space the distribution is defined on. The third and fifth columns indicate the number of samples required by Barbarik2 and Barbarik3. The fourth and sixth columns report the output of Barbarik2 and Barbarik3.

		Barbarik2		Barbarik3	
Benchmark	Dimensions	Result	# of samples	Result	# of samples
SetTest.sk_9_21	21	R	2817	R	58000
Pollard.sk_1_10	10	R	7606	R	36000
s444_3_2	24	R	848148	R	64000
s526a_3_2	24	R	848148	R	64000
s510_15_7	25	R	12708989	R	66000
s27_new_7_4	7	A	23997012	R	30000
s298_15_7	17	R	38126967	R	50000
s420_3_2	34	TO	-	R	83000
s382_3_2	24	TO	-	R	64000
s641_3_2	54	TO	-	R	123000
111.sk_2_36	36	TO	-	R	87000
7.sk_4_50	50	TO	-	R	115000
56.sk_6_38	38	TO	-	R	91000
s820a_15_7	23	TO	-	R	62000
ProjectService3.sk	55	TO	-	R	125000

Table 4.1: Runtime performance of Barbarik3. We experiment with 87 benchmarks, and out of the 87 benchmarks we display 15 in the table and we display the full data in Appendix A.2 . In the table 'A' represents Accept, 'R' represents Reject and 'TO' represents that the tester either asked for more than 10^8 samples or did not terminate in the given time limit of 16 hours.

Part II

Estimation Problems

This chapter is based on the following publications:

1. **Testing Probabilistic Circuits**
   Yash Pote 𝔒 Kuldeep S. Meel.
   In Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2021.

2. **Distance Estimation for High-Dimensional Discrete Distributions**
   Gunjan Kumar 𝔒 Kuldeep S. Meel 𝔒 Yash Pote
   In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2025.

This part will focus on the estimation problem. The first chapter of this part, Chapter 5 will focus on the estimation of distance between distributions encoded as probabilistic circuits(PCs).

Then in Chapter 6 we extend our estimation framework to a larger class of distributions beyond PCs, namely all distributions that allow conditioning. Prior to our work no polynomial query distance estimator was known, and our work paves

Chapter 5

Probabilistic Circuits

Probabilistic modeling is at the heart of modern computer science, with applications ranging from image recognition and generation [82, 84] to weather forecasting [19]. Probabilistic models have a multitude of representations, such as probabilistic circuits (PCs) [37], graphical models [64], generative networks [55], and determinantal point processes [65]. Of particular interest to us are PCs, which are known to support guaranteed inference and thus have applications in safety-critical fields such as healthcare [6, 76].

Determining the closeness of models has applications in AI planning [43], bioinformatics [85, 89, 96] and probabilistic program verification [47]. Equivalence testing is a special case of closeness testing, where one tests if $d_{TV}(P, Q) = 0$. Darwiche and Huang [43] initiated the study of equivalence testing of PCs by designing an equivalence test for d-DNNFs. An equivalence test is, however, of little use in contexts where the PCs under test encode non-identical distributions that are nonetheless close enough for practical purposes. Such situations may arise due to the use of approximate PC compilation [38] and sampling-based learning of PCs [79, 80]. As a concrete example, consider PCs that are learned via approximate methods such as stochastic gradient descent [80]. In such a case, PCs are likely to converge to close but non-identical distributions. Given two such PCs, we would like to know whether they have converged to distributions close to each other. Thus, we raise the question: *Does there exist an efficient algorithm to test the closeness of two PC distributions?*

In this chapter, we design the first closeness test for PCs with respect to TV distance, called Teq. Assuming the tested PCs allow poly-time approximate weighted model counting and sampling, Teq runs in polynomial time. Formally, given

two PC distributions P and Q, and three parameters (ε,η,δ), for closeness(ε), farness(η), and tolerance(δ), Teq returns Accept if $d_{TV}(P,Q) \leq \varepsilon$ and Reject if $d_{TV}(P,Q) \geq \eta$ with probability at least $1-\delta$. Teq makes $O((\eta-\varepsilon)^{-2}\log(\delta^{-1}))$ calls to the sampler and exactly two calls to the counter.

Teq builds on a general distance estimation technique of Canonne and Rubinfeld [20] that estimates the distance between two distributions with a small number of samples. In the context of PCs, the algorithm requires access to an exact sampler and an exact counter. Since not all PCs support exact sampling and counting, we modify existing techniques to allow for approximate samples and counts. Furthermore, we implement and test Teq on a dataset of publicly available PCs arising from applications in the testing of circuits. Our results show that closeness testing can be accurate and scalable in practice.

For some NNF fragments, such as DNNF, no sampling algorithm is known, and for fragments such as PI, sampling is known to be NP-hard [86]. Since Teq requires access to approximate weighted counters and samplers to achieve tractability, the question of determining the closeness of the PCs mentioned above remains unanswered. Thus, we investigate further and characterize the complexity of closeness testing for a broad range of PCs. Our characterization reveals that PCs from the fragments d-DNNFs and SDNNFs can be tested for closeness in poly-time via Teq, owing to the algorithms of Darwiche [41] and Arenas et al. [5]. We show that the SDNNF approximate counting algorithm of Arenas et al. [5] can be extended to log-linear SDNNFs using chain formulas [31]. Then, using previously known results, we also find that there are no poly-time equivalence tests for PCs from PI and DNNF, conditional on widely believed complexity-theoretic conjectures. Our characterization also reveals some open questions regarding the complexity of closeness and equivalence testing of PCs.

The rest of the chapter is organized as follows: we present the main contribution, the closeness test Teq, and the associated proof of correctness in Section 5.2. We present our experimental findings in Section 5.3 and then discuss the complexity landscape of closeness testing in Section 5.4.

5.1 Preliminaries

Let $\varphi : \{0,1\}^n \to \{0,1\}$ be a circuit over n Boolean variables. An assignment $\sigma \in \{0,1\}^n$ to the variables of φ is a *satisfying assignment* if $\varphi(\sigma) = 1$. The set of all satisfying assignments of φ is R_φ. If $|R_\varphi| > 0$, then φ is said to be *satisfiable* and if $|R_\varphi| = 2^n$, then φ is said to be *valid*. We use $|\varphi|$ to denote the size of circuit φ, where the size is the total number of vertices and edges in the circuit DAG.

5.1.1 Probability distributions

A weight function $\mathtt{w} : \{0,1\}^n \to \mathbb{Q}^+$ assigns a positive rational weight to each assignment σ. We extend the definition of $\mathtt{w}$ to also allow circuits as input: $\mathtt{w}(\varphi) = \sum_{\sigma \in R_\varphi} \mathtt{w}(\sigma)$. For weight function $\mathtt{w}$ and circuit φ, $\mathtt{w}(\varphi)$ is the weighted model count (WMC) of φ w.r.t. $\mathtt{w}$.

In this chapter, we focus on log-linear weight functions as they capture a wide class of distributions, including those arising from graphical models, conditional random fields, and skip-gram models [73]. Log-linear models are represented as literal-weighted functions, defined as:

Definition 19. *For a set X of n variables, a weight function $\mathtt{w}$ is called literal-weighted if there is a poly-time computable map $\mathtt{w} : X \to \mathbb{Q} \cap (0,1)$ such that for any assignment $\sigma \in \{0,1\}^n$:*

$$\mathtt{w}(\sigma) = \prod_{x \in \sigma} \begin{cases} \mathtt{w}(x) & if \quad x = 1 \\ 1 - \mathtt{w}(x) & if \quad x = 0 \end{cases}$$

For all circuits φ, and log-linear weight functions $\mathtt{w}$, $\mathtt{w}(\varphi)$ can be represented in size polynomial in the input.

PC's are a very broad class of distributional representations (see survey by Choi et al.), and in this chapter, we focus on a fragment of the Negation Normal Form (NNF).

Definition 20 (Probabilistic circuits)**.** *A probabilistic circuit is a satisfiable circuit φ along with a weight function $\mathtt{w}$. φ and $\mathtt{w}$ together define a discrete probability distribution*

on the set $\{0,1\}^n$ *that is supported over* R_φ*. We denote the p.m.f. of this distribution as:*

$$P(\varphi, \mathtt{w})(\sigma) = \begin{cases} 0 & \varphi(\sigma) = 0 \\ \mathtt{w}(\sigma)/\mathtt{w}(\varphi) & \varphi(\sigma) = 1 \end{cases}$$

A circuit φ in NNF is a rooted, directed acyclic graph (DAG), where each leaf node is labeled with true, false, v or $\neg v$; and each internal node is labeled with a $\wedge$ or $\vee$ and can have arbitrarily many children. We focus on four fragments of NNF, namely, Decomposable NNF(DNNF), deterministic-DNNF(d-DNNF), Structured DNNF(SDNNF), and Prime Implicates(PI). For further information regarding circuits in NNF, refer to the survey [44] and the paper [81].

5.2 Teq: a Tractable Algorithm for Closeness Testing

In this section, we present our main contribution: a closeness test for PCs, Teq. The pseudocode of Teq is given in Algorithm 8.

Given satisfiable circuits φ_1, φ_2 and weight functions $\mathtt{w}_1, \mathtt{w}_2$ along with parameters $(\varepsilon, \eta, \delta)$, Teq decides whether the TV distance between $P(\varphi_1, \mathtt{w}_1)$ and $P(\varphi_2, \mathtt{w}_2)$ is lesser than ε or greater than η with confidence at least $1-\delta$. Teq assumes access to an approximate weighted counter $\mathsf{Awct}(\alpha, \beta, \varphi, \mathtt{w})$, and an approximate weighted sampler $\mathsf{Samp}(\alpha, \beta, \varphi, \mathtt{w})$. We define their behavior in the following two definitions.

Definition 21. $\mathsf{Awct}(\alpha, \beta, \varphi, \mathtt{w})$ *takes a circuit* φ*, a weight function* $\mathtt{w}$*, a tolerance parameter* $\alpha > 0$ *and a confidence parameter* $\beta > 0$ *as input and returns the approximate weighted model count of* φ *w.r.t.* $\mathtt{w}$ *such that*

$$\Pr\left[\frac{\mathtt{w}(\varphi)}{1+\alpha} \leq \mathsf{Awct}(\alpha, \beta, \varphi, \mathtt{w}) \leq (1+\alpha)\mathtt{w}(\varphi)\right] \geq 1-\beta$$

Tractable approximate counting algorithms for PCs are known as Fully Polynomial Randomised Approximation Schemes (FPRAS). The running time of an FPRAS is given by $T(\alpha, \beta, \varphi) = poly(\alpha^{-1}, \log(\beta^{-1}), |\varphi|)$.

Definition 22. $\mathsf{Samp}(\alpha, \beta, \varphi, \mathtt{w})$ *takes a circuit* φ*, a weight function* $\mathtt{w}$*, a tolerance parameter* $\alpha > 0$ *and a confidence parameter* $\beta > 0$ *as input and returns either* (1) *a satisfying assignment* σ *sampled approximately w.r.t. weight function* $\mathtt{w}$ *with probability* $\geq 1-\beta$ *or*

(2) *a symbol* $\perp$ *indicating failure with probability* $< \beta$*. In other words,*

$$\frac{P(\varphi, \mathtt{w})(\sigma)}{1+\alpha} \leq \Pr[\mathsf{Samp}(\alpha, \beta, \varphi, \mathtt{w}) = \sigma | \sigma \neq \perp] \leq (1+\alpha) P(\varphi, \mathtt{w})(\sigma)$$

The running time for a call to $\mathsf{Samp}(\alpha, \beta, \varphi, \mathtt{w})$ *is given by*

$$T(\alpha, \beta, \varphi) = poly(\alpha^{-1}, \log(\beta^{-1}), |\varphi|)$$

The algorithm Our algorithm follows from a result by Bhattacharyya et al. [9][Theorem 2.3]. Specifically, the theorem states that given approximate EVAL access to two distributions, one can estimate the TV distance upto additive ε in $O(\varepsilon^{-2})$. On a high-level we will use $O(\varepsilon^{-2})$ Samp and Awct oracle accesses to simulate the a single approximate EVAL access. Thus with only a constant overhead we can implement their distance approximation algorithm.

Teq starts by computing constants γ and m. Then it queries the Awct routine with circuit φ_1 and weight function $\mathtt{w}_1$ to obtain a $\sqrt{1+\gamma/4} - 1$ approximation of $\mathtt{w}_1(\varphi_1)$ with confidence at least $1 - \delta/8$. A similar query is made for φ_2 and $\mathtt{w}_2$ to obtain an approximate value for $\mathtt{w}_2(\varphi_2)$. These values are stored in k_1 and k_2, respectively. Teq maintains a m-sized array Γ, to store the estimates for $r(\sigma_i)$. Teq now iterates m times. In each iteration, it generates one sample σ_i through the Samp call on line 7. There is a small probability of at most $\delta/4m$ that this call fails and returns $\perp$. Teq only samples from one of the two PCs.

The algorithm then proceeds to compute the weight of assignment σ_i w.r.t. the weight functions $\mathtt{w}_1$ and $\mathtt{w}_2$ and stores it in s_1 and s_2, respectively. Using the weights and approximate weighted counts stored in k_1, k_2 the algorithm computes the value $r(\sigma_i)$ on line 10, where $r(\sigma_i)$ is an approximation of the ratio of the probability of σ_i in the distribution $P(\varphi_2, \mathtt{w}_2)$ to its probability in $P(\varphi_1, \mathtt{w}_1)$. Since σ_i was sampled from $P(\varphi_1, \mathtt{w}_1)$, its probability in $P(\varphi_1, \mathtt{w}_1)$ cannot be 0, ensuring that there is no division by 0. If the ratio $r(\sigma_i)$ is less than 1, then $\Gamma[i]$ is updated with the value $1 - r(\sigma_i)$ otherwise the value of $\Gamma[i]$ remains 0. After the m iterations, Teq sums up the values in the array Γ. If the sum is found to be less than threshold $m(\varepsilon + \gamma)$, Teq returns Accept and otherwise returns Reject.

The following theorem asserts the correctness of Teq. To improve readability, we use $\mathcal{P}_1$ to refer to the distribution $P(\varphi_1, \mathtt{w}_1)$ and P_2 to refer to $P(\varphi_2, \mathtt{w}_2)$.

Algorithm 8 $\mathsf{Teq}(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \varepsilon, \eta, \delta)$

1: $\gamma \leftarrow (\eta - \varepsilon)/2$
2: $m \leftarrow \lceil 2\log(4/\delta)/\gamma^2 \rceil$
3: $\Gamma \leftarrow [0] * m$
4: $k_1 \leftarrow \mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_1, \mathtt{w}_1)$
5: $k_2 \leftarrow \mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_2, \mathtt{w}_2)$ {$\mathsf{Awct}(\cdot) > 0$ for satisfiable formula}
6: **for all** $i \in \{1, 2 \ldots, m\}$ **do**
7: $\quad \sigma_i \leftarrow \mathsf{Samp}(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathtt{w}_1)$
8: $\quad$ **if** $\sigma_i \neq \perp$ **then**
9: $\quad\quad s_1 \leftarrow \mathtt{w}_1(\sigma_i), s_2 \leftarrow \mathtt{w}_2(\sigma_i)$
10: $\quad\quad r(\sigma_i) \leftarrow \frac{s_2}{k_2} \cdot \frac{k_1}{s_1}$
11: $\quad\quad$ **if** $r(\sigma_i) < 1$ **then**
12: $\quad\quad\quad \Gamma[i] \leftarrow 1 - r(\sigma_i)$
13: **if** $\sum_{i \in [m]} \Gamma[i] \leq m(\varepsilon + \gamma)$ **then**
14: $\quad$ **Return** Accept
15: **else**
16: $\quad$ **Return** Reject

Theorem 5.1. *Given two satisfiable probabilistic circuits φ_1, φ_2 and weight functions $\mathtt{w}_1, \mathtt{w}_2$, along with parameters $\varepsilon < \eta < 1$ and $\delta < 1$,*

A. If $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) \leq \varepsilon$, then $\mathsf{Teq}(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \varepsilon, \eta, \delta)$ returns Accept with probability at least $(1 - \delta)$.

B. If $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) \geq \eta$, then $\mathsf{Teq}(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \varepsilon, \eta, \delta)$ returns Reject with probability at least $(1 - \delta)$.

5.2.1 Proving the correctness of Teq

In this subsection, we present the theoretical analysis of Teq, and the proofs of Theorem 5.1(A) and 5.1(B).

For the purpose of the proof, we will first define events $\mathtt{Pass}_1, \mathtt{Pass}_2$ and $\mathtt{Good}$. Events $\mathtt{Pass}_1$ and $\mathtt{Pass}_2$ are defined w.r.t. the function calls $\mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_1, \mathtt{w}_1)$ and $\mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_2, \mathtt{w}_2)$, respectively (as on lines 4, 5 of Algorithm 8). $\mathtt{Pass}_1$ and $\mathtt{Pass}_2$ represent the events that the two calls correctly return $\sqrt{1+\gamma/4}$ approximations of the weighted model counts of φ_1 and φ_2 i.e. $\frac{\mathtt{w}_1(\varphi_1)}{\sqrt{1+\gamma/4}} \leq \mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_1, \mathtt{w}_1) \leq (\sqrt{1+\gamma/4})\mathtt{w}_1(\varphi_1)$, and $\frac{\mathtt{w}_2(\varphi_2)}{\sqrt{1+\gamma/4}} \leq$

$\mathsf{Awct}(\sqrt{1+\gamma/4}-1, \delta/8, \varphi_2, \mathtt{w}_2) \leq (\sqrt{1+\gamma/4})\mathtt{w}_2(\varphi_2)$. From the definition of Awct, we have $\Pr[\mathtt{Pass}_1], \Pr[\mathtt{Pass}_2] \geq 1-\delta/8$.

Let $\mathtt{Fail}_i$ denote the event that Samp (Algorithm 8, line 7) returns the symbol $\perp$ in the ith iteration of the loop. By the definition of Samp we know that $all_{i\in[m]} \Pr[\mathtt{Fail}_i] < \delta/4m$.

The analysis of Teq requires that all m Samp calls and both Awct calls return correctly. We denote this super-event as $\mathtt{Good} = \bigcap_{i\in[m]} \overline{\mathtt{Fail}_i} \cap \mathtt{Pass}_1 \cap \mathtt{Pass}_2$. Applying the union bound we see that the probability of all calls to Awct and Samp returning without error is at least $1-\delta/2$:

$$\begin{aligned}\Pr[\mathtt{Good}] &= 1 - \Pr\left[\bigcup_{i\in[m]} \mathtt{Fail}_i \cup \overline{\mathtt{Pass}_1} \cup \overline{\mathtt{Pass}_2}\right] \\ &\geq 1 - m\cdot\delta/4m - 2\cdot\delta/8 \\ &= 1-\delta/2 \end{aligned} \tag{5.1}$$

Lemma 5. $\mathtt{Good} \rightarrow \left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| \leq \gamma/4 \cdot \frac{P_2(\sigma)}{P_1(\sigma)}$

Proof. The quantity $r(\sigma)$ (line 10 from Algorithm 8) conditioned on the event $\overline{\mathtt{Fail}_i} \subset \mathtt{Good}$:

$$r(\sigma) = \frac{\mathtt{w}_2(\sigma)}{\mathsf{Awct}(\sqrt{1+\gamma/4}-1, \delta/8, \varphi_2, \mathtt{w}_2)} \cdot \frac{\mathsf{Awct}(\sqrt{1+\gamma/4}-1, \delta/8, \varphi_1, \mathtt{w}_1)}{\mathtt{w}_1(\sigma)}$$

Conditioned on the events $\mathtt{Pass}_1, \mathtt{Pass}_2 \subset \mathtt{Good}$, we know that with probability 1:

$$\frac{\mathtt{w}_2(\sigma)\mathtt{w}_1(\varphi_1)}{\mathtt{w}_2(\varphi_2)\mathtt{w}_1(\sigma)}(\sqrt{1+\gamma/4})^{-2} < r(\sigma) < (\sqrt{1+\gamma/4})^2\frac{\mathtt{w}_2(\sigma)\mathtt{w}_1(\varphi_1)}{\mathtt{w}_2(\varphi_2)\mathtt{w}_1(\sigma)}$$

Which gives us:

$$\frac{P_2(\sigma)}{P_1(\sigma)}(1+\gamma/4)^{-1} < r(\sigma) < (1+\gamma/4)\frac{P_2(\sigma)}{P_1(\sigma)}$$

and therefore,

$$\left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| \leq \frac{P_2(\sigma)}{P_1(\sigma)} \cdot \max_{0<\gamma<1}\left(\gamma/4, 1-\frac{1}{1+\gamma/4}\right) \leq \frac{P_2(\sigma)}{P_1(\sigma)}\cdot\gamma/4$$

□

We now prove the lemma critical for our proof of correctness of Teq.

Lemma 6. *Assuming the event* Good*, let* $A = \sum_{\sigma\in\{0,1\}^n} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) P_1(\sigma)$ *, then*

1. *If* $d_{TV}(P_1, P_2) \leq \varepsilon$ *, then* $A \leq \varepsilon + \gamma/4$
2. *If* $d_{TV}(P_1, P_2) \geq \eta$ *, then* $A \geq \eta - \gamma/4$

Proof. If $\sum_x \left(P_1(x) - P_2(x)\right) = 0$, then $\frac{1}{2}\sum_x |P_1(x) - P_2(x)| = \sum_{x:P_1(x)-P_2(x)>0} (P_1(x) - P_2(x))$. Using this fact we see that,

$$\begin{aligned} d_{TV}(P_1, P_2) &= \sum_{\sigma:P_2(\sigma)<P_1(\sigma)} P_1(\sigma) - P_2(\sigma) = \sum_{\sigma:\frac{P_2(\sigma)}{P_1(\sigma)}<1} \left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) P_1(\sigma) \\ &= \sum_{\sigma\in\{0,1\}^n} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) P_1(\sigma) \\ &= A + \underbrace{\sum_{\sigma\in\{0,1\}^n} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) P_1(\sigma) - A}_{B} \end{aligned}$$

Thus we have that $d_{TV}(P_1, P_2) - A = B$. We now divide the set of assignments $\sigma \in \{0,1\}^n$ into three disjoint partition S_1, S_2 and S_3 as following: $S_1 = \{\sigma : \mathbb{1}(\frac{P_2(\sigma)}{P_1(\sigma)} < 1) = \mathbb{1}(r(\sigma) < 1)\}$; $S_2 = \{\sigma : \mathbb{1}(\frac{P_2(\sigma)}{P_1(\sigma)} < 1) > \mathbb{1}(r(\sigma) < 1)\}$; $S_3 = \{\sigma : \mathbb{1}(\frac{P_2(\sigma)}{P_1(\sigma)} < 1) < \mathbb{1}(r(\sigma) < 1)\}$. The definition implies that the indicator $\mathbb{1}(r(\sigma) < 1)$ is 0 for all assignments in the set S_2, and is 1 for all assignments in S_3. Similarly $\mathbb{1}(\frac{P_2(\sigma)}{P_1(\sigma)} < 1)$ takes value 1 and 0 for all elements in S_2 and S_3, respectively.

Now we bound the magnitude of B,

$$|B| = \left| \sum_{\sigma\in\{0,1\}^n} \left[\left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right) - \left(1 - r(\sigma)\right) \mathbb{1}\left(r(\sigma) < 1\right)\right] P_1(\sigma) \right|$$

For $b_j > 0$, we have that $|\sum_j a_j b_j| \leq \sum_j |a_j| b_j$, and thus:

$$|B| \leq \sum_{\sigma\in\{0,1\}^n} \left| \left[\left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right) - \left(1 - r(\sigma)\right) \mathbb{1}\left(r(\sigma) < 1\right)\right] \right| P_1(\sigma)$$

We can split the summation into three terms based on the sets in which the assignments lie. Some summands take the value 0 in a particular set, so we don't

include them in the term.

$$
\begin{aligned}
|B| \leq & \sum_{\sigma \in S_1} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| P_1(\sigma) \\
& + \sum_{\sigma \in S_2} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left(1 - \frac{P_2(\sigma)}{P_1(\sigma)}\right) P_1(\sigma) \\
& + \sum_{\sigma \in S_3} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) P_1(\sigma)
\end{aligned}
$$

Since we know that $\forall \sigma \in S_2, r(\sigma) > 1$ and $all \sigma \in S_3, \frac{P_2(\sigma)}{P_1(\sigma)} > 1$, we can alter the second and third terms of the inequality in the following way:

$$
\begin{aligned}
|B| \leq & \sum_{\sigma \in S_1} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| P_1(\sigma) \\
& + \sum_{\sigma \in S_2} \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)\left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| P_1(\sigma) \\
& + \sum_{\sigma \in S_3} \mathbb{1}\left(r(\sigma) < 1\right)\left|\frac{P_2(\sigma)}{P_1(\sigma)} - r(\sigma)\right| P_1(\sigma) \\
|B| \leq & \sum_{\sigma \in S_1 \cup S_2 \cup S_3} \left|r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)}\right| P_1(\sigma)
\end{aligned}
$$

Using our assumption of the event $\texttt{Good}$ and Lemma 5, $|B| \leq \sum_{\sigma \in \{0,1\}^n} \gamma/4 \cdot P_1(\sigma) \leq \gamma/4$ Since $d_{TV}(P_1, P_2) - A = B$, we get $|d_{TV}(P_1, P_2) - A| \leq \gamma/4$. We can now deduce that if $d_{TV}(P_1, P_2) \leq \varepsilon$, then $A \leq \varepsilon + \gamma/4$ and if $d_{TV}(P_1, P_2) \geq \eta$, then $A \geq \eta - \gamma/4$. □

Proof of Theorem 5.1(A)

Proof. We assume the event $\texttt{Good}$. Let σ_i be the sample returned by the sampler Samp in the ith iteration. If $r(\sigma_i) > 1$, $\Gamma[i]$ takes value 0, else $\Gamma[i] = 1 - r(\sigma_i)$. Thus $\Gamma[i]$ is a r.v. which takes on a value from $[0, 1]$. We can write $\Gamma[i] = \mathbb{1}\left(r(\sigma_i) < 1\right)\left(1 - r(\sigma_i)\right)$ The expectation of $\Gamma[i]$ is:

$$
\mathbb{E}[\Gamma[i]] = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) \cdot \Pr[\mathsf{Samp}(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathtt{w_1}) = \sigma] \tag{5.2}
$$

According to definition 22, and our assumption of $\overline{\texttt{Fail}_\texttt{i}} \subset \texttt{Good}$, we know that with probability 1, $\Pr[\mathsf{Samp}(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathtt{w_1}) = \sigma] \leq (1 + \gamma/(4\eta - 2\gamma))P_1(\sigma)$.

Thus we have,

$$\mathbb{E}[\Gamma[i]] \leq \sum_{\sigma \in \{0,1\}^n} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) \cdot (1 + \gamma/(4\eta - 2\gamma))P_1(\sigma)$$

Recall that in Lemma 6, we define $A = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) P_1(\sigma)$. Therefore, we can simplify the above expression as: $\mathbb{E}[\Gamma[i]] = (1 + \gamma/(4\eta - 2\gamma)) \cdot A$. We can then use the assumption of ε-closeness and the result of Lemma 6-1 to find a bound on the expectation,

$$\begin{aligned}
\mathbb{E}[\Gamma[i]] &\leq (1 + \gamma/(4\eta - 2\gamma))\,(\varepsilon + \gamma/4) \\
&= (1 + \gamma/(6\gamma + 4\varepsilon))(\varepsilon + \gamma/4) \\
&= \varepsilon + \gamma/4 + (\gamma/(6\gamma + 4\varepsilon))(\varepsilon + \gamma/4) \\
&= \varepsilon + \gamma/4 + \gamma/4(2/(3\gamma + 2\varepsilon))(\varepsilon + \gamma/4) \\
&= \varepsilon + \gamma/4 + \gamma/4(2/(3\eta + \varepsilon))(\eta/4 + 7\varepsilon/4) \\
&= \varepsilon + \gamma/4 + \gamma/4 \cdot \frac{1}{2} \cdot \frac{1 + 7\varepsilon/\eta}{3 + \varepsilon/\eta} \\
\text{(Since } \eta/\varepsilon < 1\text{)} \qquad &< \varepsilon + \gamma/2
\end{aligned}$$

Using the linearity of expectation we get: $\mathbb{E}\left[\sum_{i \in [m]} \Gamma[i]\right] < m(\varepsilon + \gamma/2)$. Teq returns Reject when $\sum_{i \in [m]} \Gamma[i] > m(\varepsilon + \gamma)$ on line 13. Since the $\Gamma[i]$'s are i.i.d random variables taking values in $[0, 1]$, we apply the Chernoff bound to find the probability of Accept, assuming the event `Good`:

$$\begin{aligned}
\Pr\left[\text{Teq returns Accept} \,\middle|\, \texttt{Good}\right] &= 1 - \Pr\left[\sum_{i \in [m]} \Gamma[i] > m(\varepsilon + \gamma)\right] \\
&\geq 1 - 2e^{-\gamma^2 m/2} \geq 1 - \delta/2
\end{aligned}$$

The value for m is taken from line 2 of Algorithm 8. Using (5.1), we see that the probability of Teq returning Accept is:

$$\begin{aligned}
\Pr[\text{Teq returns Accept}] &\geq \Pr[\text{Teq returns Accept} \mid \texttt{Good}]\,\Pr[\texttt{Good}] \\
&= (1 - \delta/2)(1 - \delta/2) \geq 1 - \delta
\end{aligned}$$

□

Proof of Theorem 5.1(B)

Proof. First we assume the event Good. Then according to definition 22, we know that with probability 1 (since we assume event $\overline{\texttt{Fail}_i} \subset \texttt{Good}$)

$$\Pr[\mathsf{Samp}(\gamma/(4\eta-2\gamma), \delta/4m, \varphi_1, \mathtt{w}_1) = \sigma] \geq \frac{P_1(\sigma)}{(1+\gamma/(4\eta-2\gamma))}$$

Thus substituting into (5.2), we get

$$\mathbb{E}[\Gamma[i]] \geq \sum_{\sigma\in\{0,1\}^n} \mathbb{1}\left(r(\sigma)<1\right)(1-r(\sigma))\frac{P_1(\sigma_i)}{1+\gamma/(4\eta-2\gamma)} \tag{5.3}$$

Then we use the η-farness assumption and Lemma 6-2

$$\mathbb{E}[\Gamma[i]] \geq \frac{\eta-\gamma/4}{1+\gamma/(4\eta-2\gamma)} = \eta-\gamma/2 \tag{5.4}$$

The algorithm returns Accept when $\sum_{i\in[m]}\Gamma[i] \leq m(\varepsilon+\gamma)$ (on line 13). Then using (5.4) and the linearity of expectation.

$$\mathbb{E}\left[\sum_{i\in[m]}\Gamma[i]\right] \geq m(\eta-\gamma/2)$$

Since the $\Gamma[i]$'s are i.i.d random variables taking values in $[0, 1]$, we apply the Chernoff bound to find the probability of Reject, given the assumption of the event Good:

$$\begin{aligned}
\Pr\left[\mathsf{Teq\ returns\ Reject} \mid \texttt{Good}\right] &= 1-\Pr\left[\sum_{i\in[m]}\Gamma[i] \leq m(\varepsilon+\gamma)\right] \\
&\geq 1-\Pr\left[m(\eta-\gamma/2)-\sum_{i\in[m]}\Gamma[i] \geq m(\eta-\gamma/2-\varepsilon-\gamma)\right] \\
&\geq 1-\Pr\left[\left|\sum_{i\in[m]}\Gamma[i]-m(\eta-\gamma/2)\right| \geq m\gamma/2\right] \\
&\geq 1-2e^{-\gamma^2 m/2} \geq 1-\delta/2 \quad \text{(Substituting } m \text{ as in line 2)}
\end{aligned}$$

Hence, the probability that Algorithm 8 returns Reject is

$$\begin{aligned}
\Pr[\mathsf{Teq\ returns\ Reject}] &\geq \Pr\left[\mathsf{Teq\ returns\ Reject} \mid \texttt{Good}\right]\Pr\left[\texttt{Good}\right] \\
&= (1-\delta/2)(1-\delta/2) \geq 1-\delta \quad \text{(Using (5.1))}
\end{aligned}$$

□

The following theorem states the running time of the algorithm,

Theorem 5.2. *Let $\gamma = \eta - \varepsilon$, then the time complexity of* Teq *is in*

$$O\left(T_{\mathsf{Awct}}(\gamma, \delta, \max(|\varphi_1|, |\varphi_2|)) + T_{\mathsf{Samp}}(\gamma, \delta, \max(|\varphi_1|, |\varphi_2|))\frac{\log(\delta^{-1})}{\gamma^2}\right) \tag{5.5}$$

If the underlying PCs support approximate counting and sampling in polynomial time, then the running time of Teq *is also polynomial in terms of $\gamma, \log(\delta^{-1})$ and $max(|\varphi_1|, |\varphi_2|)$.*

Proof. Teq makes two calls to Awct on line 4 and 5 of Algorithm 8. According to definition 21, the runtime of the Awct$(\sqrt{1+\gamma/4}-1, \delta/8, \varphi, \mathtt{w})$ query is $T(\sqrt{1+\gamma/4}-1, \delta/8, \varphi) = poly((\sqrt{1+\gamma/4}-1)^{-1}, \log(\delta^{-1}), |\varphi|)$.

Using the identity $1 + \frac{x}{2} - \frac{x^2}{2} \leq \sqrt{1+x}$ for $x \geq 0$ and the fact that $\gamma \in (0, 1)$

$$\frac{1}{\sqrt{1+\gamma/4}-1} \leq \frac{1}{\gamma/8 - \gamma^2/32} < \frac{11}{\gamma}$$

Hence any $poly((\sqrt{1+\gamma/4}-1)^{-1})$ algorithm also runs in $poly(\gamma^{-1})$. Thus the Awct queries run in $O(poly(\gamma^{-1}, \log(\delta^{-1}), max(|\varphi_1|, |\varphi_2|)))$

Teq makes $m = \lceil \log(2/\delta)/2\gamma^2 \rceil$ calls to Samp on lines 7 of Algorithm 8. According to definition 22, the runtime of the Samp$(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathtt{w}_1)$ query is $T(\gamma/(4\eta-2\gamma), \delta/4m, |\varphi_1|) = poly((\gamma/(4\eta-2\gamma))^{-1}, \log((\delta/4m)^{-1}), |\varphi_1|)$. First we see that $\frac{4\eta-2\gamma}{\gamma} < \frac{4}{\gamma}$, thus the algorithm remains in $poly(\gamma^{-1})$. We then see that $\log(4m/\delta) = \log(4m) + \log(\delta^{-1})$. Since $\log(m) \in poly(\log(\gamma^{-1}), \log\log(\delta^{-1}))$, we know that Samp queries run in $O(poly(\gamma^{-1}, \log(\delta^{-1}), max(|\varphi_1|, |\varphi_2|)))$.

Since each Samp call and each Awct call requires atmost polynomial time in terms of $\gamma^{-1}, \log(\delta^{-1})$ and $max(|\varphi_1|, |\varphi_2|)$ we know that the algorithm itself runs in time polynomial in $\gamma^{-1}, \log(\delta^{-1})$ and $max(|\varphi_1|, |\varphi_2|)$. □

Using Teq to test PCs in general. Exact weighted model counting (WMC) is a commonly supported query on PCs. In the language of PC queries, a WMC query is known as the marginal (MAR) query. Conditional inference (CON) is another well studied PC query. Using CON and MAR, one can sample from the distribution encoded by a given PC. It is known that if a PC has the structural properties of *smoothness* and *decomposability*, then the CON and MAR queries can be computed tractably. For definitions of the above terms and further details, please refer to the survey [37].

5.3 Evaluation

To evaluate the performance of Teq, we implemented a prototype in Python. The prototype uses WAPS[1] [56] as a weighted sampler to sample over the input d-DNNF circuits. The primary objective of our experimental evaluation was to seek an answer to the following question: Is Teq able to determine the closeness of a pair of probabilistic circuits by returning Accept if the circuits are ε-close and Reject if they are η-far? We test our tool Teq in the following two settings:

A. The pair of PCs represent small randomly generated circuits and weight functions.

B. The pair of PCs are from the set of publicly available benchmarks arising from sampling and counting tasks.

Our experiments were conducted on a high performance compute cluster with Intel Xeon(R) E5-2690 v3@2.60GHz CPU cores. For each benchmark, we use a single core with a timeout of 7200 seconds.

5.3.1 Setting A - Synthetic benchmarks

Dataset Our dataset for experiments conducted in setting **A** consisted of randomly generated 3-CNFs and with random literal weights. Our dataset consisted of 3-CNFs with $\{14, 15, 16, 17, 18\}$ variables. Since the circuits are small, we validate the results by computing the actual total variation distance using brute-force.

Results Our tests terminated with the correct result in less than 10 seconds on all the randomly generated PCs we experimented with. We present the empirical results in Table 5.1. The first column indicates the benchmark's name, the second and third indicate the parameters ε and η on which we executed Teq. The fourth column indicates the actual d_{TV} distance between the two benchmark PCs. The fifth column indicates the output of Teq, and the sixth indicates the expected result. The full detailed results are presented in the appendix Section A.3.

[1] https://github.com/meelgroup/WAPS

	d_{TV}				
Benchmark	$\leq \epsilon$	$\geq \eta$	Actual	Result	Expected Result
15_3	0.75	0.94	0.804	R	A/R
14_2	0.8	0.9	0.764	A	A
17_4	0.75	0.9	0.941	R	R
14_1	0.9	0.99	0.740	A	A
18_2	0.75	0.9	0.918	R	R

Table 5.1: Runtime performance of Teq. We experiment with 375 random PCs with known d_{TV}, and out of the 375 benchmarks we display 5 in the table and the rest in the appendix Section A.3. In the table 'A' represents Accept and 'R' represents Reject. In the last column 'A/R' represents that both Accept and Reject are acceptable outputs for Teq.

5.3.2 Setting B - Real-world benchmarks

Dataset We conducted experiments on a range of publicly available benchmarks arising from sampling and counting tasks[2]. Our dataset contained 100 d-DNNF circuits with weights. We have assigned random weights to literals wherever weights were not readily available. For the empirical evaluation of Teq, we needed pairs of weighted d-DNNFs with known d_{TV} distance. To generate such a dataset, we first chose a circuit and a weight function, and then we synthesised new weight functions using the technique of *one variable perturbation*, described in the following section.

5.3.3 One variable perturbation

Consider two weight functions $\mathtt{w}_1$ and $\mathtt{w}_2$ that differ only in the weight assigned to the literals v^0 and v^1. Then, from the definition of d_{TV}:

$$d_{TV}(P(\varphi, \mathtt{w}_1), P(\varphi, \mathtt{w}_2)) = \frac{1}{2} \sum_{\sigma \in \{0,1\}^n} \left| \frac{\mathtt{w}_1(\sigma)}{\mathtt{w}_1(\varphi)} - \frac{\mathtt{w}_2(\sigma)}{\mathtt{w}_2(\varphi)} \right|$$

[2] https://zenodo.org/record/3793090

Let $S \subseteq \{0,1\}^n$ be the set of assignments for which $\frac{\mathtt{w}_1(\sigma)}{\mathtt{w}_1(\varphi)} > \frac{\mathtt{w}_2(\sigma)}{\mathtt{w}_2(\varphi)}$. Thus,

$$d_{TV}(P(\varphi,\mathtt{w}_1),P(\varphi,\mathtt{w}_2)) = \sum_{\sigma\in S}\left(\frac{\mathtt{w}_1(\sigma)}{\mathtt{w}_1(\varphi)} - \frac{\mathtt{w}_2(\sigma)}{\mathtt{w}_2(\varphi)}\right)$$

Lets assume wlog that $\mathtt{w}_1$ assigns a larger weight to v^1 than $\mathtt{w}_2$ does. Then, S contains all and only those assignments that have literal v^1, i.e. $S \equiv \varphi \wedge v^1$. Thus,

$$d_{TV}(P(\varphi,\mathtt{w}_1),P(\varphi,\mathtt{w}_2)) = \frac{\mathtt{w}_1(\varphi\wedge v^1)}{\mathtt{w}_1(\varphi)} - \frac{\mathtt{w}_2(\varphi\wedge v^1)}{\mathtt{w}_2(\varphi)}$$

We can rewrite $\mathtt{w}_1(\varphi\wedge v^1) = \mathtt{w}'_1(\varphi)\times\mathtt{w}_1(v^1)$, where $\mathtt{w}'_1$ is $\mathtt{w}_1$ with the weight of v^1 set to 1. Using a similar transformation on $\mathtt{w}_2(\varphi\wedge v^1)$ we get

$$d_{TV}(P(\varphi,\mathtt{w}_1),P(\varphi,\mathtt{w}_2)) = \frac{\mathtt{w}'_1(\varphi)\times\mathtt{w}_1(v^1)}{\mathtt{w}_1(\varphi)} - \frac{\mathtt{w}'_2(\varphi)\times\mathtt{w}_2(v^1)}{\mathtt{w}_2(\varphi)}$$

We know that $\mathtt{w}'_1(\varphi) = \mathtt{w}'_2(\varphi)$ as $\mathtt{w}_1$ and $\mathtt{w}_2$ differed only on the one variable v^1.

$$d_{TV}(P(\varphi,\mathtt{w}_1),P(\varphi,\mathtt{w}_2)) = \mathtt{w}'_1(\varphi)\times\left(\frac{\mathtt{w}_1(v^1)}{\mathtt{w}_1(\varphi)} - \frac{\mathtt{w}_2(v^1)}{\mathtt{w}_2(\varphi)}\right)$$

All quantities in the above expression are either known constants or they are defined w.r.t the already compiled d-DNNF, thus guaranteeing that $d_{TV}(P(\varphi,\mathtt{w}_1),P(\varphi,\mathtt{w}_2))$ can be computed in poly-time.

Results We set the closeness parameter ε, farness parameter η and confidence δ for Teq to be $0.01, 0.2$ and 0.01, respectively. The chosen parameters imply that if the input pair of probabilistic circuits are ≤ 0.01 close in d_{TV}, then Teq returns Accept with probability atleast 0.99, otherwise if the circuits are ≥ 0.2 far in d_{TV}, the algorithm returns Reject with probability at least 0.99. The number of samples required for Teq (indicated by the variable m as on line 2 of Algorithm 8) depends only on $\varepsilon, \eta, \delta$ and for the values we have chosen, we find that we require $m = 294$ samples.

Our tests terminated with the correct result in less than 3600 seconds on all the PCs we experimented with. We present the empirical results in Table 5.2. The first column indicates the benchmark's name, and the second and third indicate the result and runtime of Teq when presented with a pair of ε-close PCs as input. Similarly, the fourth and fifth columns indicate the result and observed runtime of Teq when the input PCs are η-far. The full set of results is presented in the appendix A.3.

Benchmark	$d_{TV} \leq \varepsilon$		$d_{TV} \geq \eta$	
	Result	Teq(s)	Result	Teq(s)
or-70-10-8-UC-10	A	23.2	R	22.82
s641_15_7	A	33.66	R	33.51
or-50-5-4	A	414.17	R	408.59
ProjectService3	A	356.15	R	356.14
s713_15_7	A	24.86	R	24.41
or-100-10-2-UC-30	A	31.04	R	31.0
s1423a_3_2	A	153.13	R	152.81
s1423a_7_4	A	104.93	R	103.51
or-50-5-10	A	283.05	R	282.97
or-60-20-6-UC-20	A	363.32	R	362.8

Table 5.2: Runtime performance of Teq. We experiment with 100 PCs with known d_{TV}, and out of the 100 benchmarks, we display 10 in the table and the rest in the appendix A.3. In the table, 'A' represents Accept and 'R' represents Reject. The value of the closeness parameter is $\varepsilon = 0.01$, and the farness parameter is $\eta = 0.2$.

5.4 A characterization of the complexity of testing

In this section, we characterize PCs according to the complexity of closeness and equivalence testing. We present the characterization in Table 5.3. The results presented in the table can be separated into (1) hardness results, and (2) upper bounds. The hardness results, presented in Section 5.4.2, are largely derived from known complexity-theoretic results. The upper bounds, presented in Section 5.4.1, are derived from a combination of established results, our algorithm Teq and the exact equivalence test of Darwiche and Huang [43](presented at the end of this section for completeness).

5.4.1 Upper bounds

In Table 5.3 we label the pair of classes of PCs that admit a poly-time closeness and equivalence test with green symbols C and E respectively. Darwiche and Huang [43] provided an equivalence test for d-DNNF s. From Theorem 5.1, we know that PCs that supports the Awct and Samp queries in poly-time must also admit a poly-time approximate equivalence test. A weighted model counting algorithms for d-DNNFs was first provided by Darwiche [41], and a weighted

sampler was provided by Gupta et al. [56]. Arenas et al. [5] provided the first approximate counting and uniform sampling algorithm for SDNNFs. Using the following lemma, we show that with the use of chain formulas, the uniform sampling and counting algorithms extend to log-linear SDNNF distributions as well.

Lemma 7. *Given a* SDNNF *formula* φ *(with a v-tree* T*)*[3]*, and a weight function* $\mathtt{w}$*,* $\mathsf{Samp}(\varphi, \mathtt{w})$ *requires polynomial time in the size of* φ*.*

Proof. Here we will assume that the weights are in the dyadic form i.e. they can be represented as the fraction $d/2^p$ for $d, p \in \mathbb{Z}^+$. Then using the weighted to unweighted construction from [31], the problem of approximate weighted sampling over SDNNF can be reduced to approximate uniform sampling. Given a SDNNF φ, and a weight function $\mathtt{w}$, we generate a SDNNF $\varphi_w \equiv \varphi \wedge \bigwedge_{i\in[n]}(\neg x_i \vee C_i^1) \wedge \bigwedge_{i\in[n]}(x_i \vee C_i^0)$. Here, C_i^0 is chain formula having exactly $w(\neg x_i) \times 2^p = 2^p - d$ satisfying assignments, and C_i^1 is a chain formula with $\mathtt{w}(x_i) \times 2^p = d$ satisfying assignments.

The property of decomposability on the $\wedge$ nodes of φ is preserved as each C_i introduces a new set of variables disjoint from the set of variables in φ and and also from all C_j, such that $j \neq i$. The $\wedge$ nodes in the chain formula are also trivially decomposable and structured as each chain formula variable appears exactly once in the formula.

If σ is an assignment to the set of variables of S and if $S' \subseteq S$, then let $\sigma_{\downarrow S'}$ denote the projection of σ on the variables in S'. The weighted formula φ is defined over variable set $var(\varphi)$. The formula φ_w defined above has the property that if $\varphi(\sigma) = 1$, then $|\{\sigma' | \varphi_w(\sigma') = 1 \wedge \sigma'_{\downarrow var(\varphi)} = \sigma\}|/|R_{\varphi_w}| = \mathtt{w}(\sigma)$. Thus a uniform distribution on R_{φ_w}, when projected on $var(\varphi)$ induces the weighted distribution $P(\varphi, \mathtt{w})$. This property allows weighted sampling and counting on φ with the help of a uniform sampler for the generated formula φ_w.

□

[3] A variable-tree, or **v-tree**, for a set of variables V is a full, rooted binary tree whose leaves are in one-to-one correspondence with the variables in V.

	NNF	PI	DNNF	SDNNF	d-DNNF
NNF	EC				
PI	EC	UU			
DNNF	EC	EU	EU		
SDNNF	EC	EU	EU	EC	
d-DNNF	EC	UU	EU	EC	EC

Table 5.3: Summary of results. C (resp. E) indicates that a poly-time closeness (resp. equivalence) test exists. C (resp. E) indicates that a poly-time closeness (equivalence) test exists only if PH collapses. 'U' indicates that the existence of a poly-time test is not known. The table is best viewed in color.

5.4.2 Hardness

In Table 5.3, we claim that the pairs of classes of PCs labeled with symbols C and E , cannot be tested in poly-time for closeness equivalence, respectively. Our claim assumes that the polynomial hierarchy (PH) does not collapse. To prove the hardness of testing the labeled pairs, we combine previously known facts about PCs and a few new arguments. Summarizing for brevity,

- We start off by observing that PC families are in a hierarchy, with CNF $\subseteq$ NNF and DNF $\subseteq$ SDNNF $\subseteq$ DNNF [44].
- We then reduce the problem of satisfiability testing of CNFs (NP-hard) and validity testing of DNFs (co-NP-hard) into the problem of equivalence and closeness testing of PCs, in Propositions 6, 7 and 10.
- We then connect the existence of poly-time algorithms for equivalence to the collapse of PH via a complexity result due to Karp and Lipton [62].

The NP-hardness of deciding the equivalence of pairs of DNNFs and pairs of SDNNFs was first shown by Pipatsrisawat and Darwiche [81]. We recast their proofs in the language of distribution testing for the sake of completeness.

Proposition 6. *If there exists a poly-time randomised algorithm for deciding the equivalence of a pair of PCs with at least one PC in* CNF*, then NP=RP.*

Proof. For CNFs, testing satisfiability is known to be NP-hard. Consider a CNF φ defined over variables $\{x_1, \ldots, x_n\}$ and a circuit ψ s.t. $\psi \equiv \bigwedge_{i\in[n+1]} x_i$. Define

$$\hat{\varphi} = (\neg x_{n+1} \rightarrow \varphi) \wedge (x_{n+1} \rightarrow \bigwedge_{i\in[n]} x_i)$$

We see that the size of the new CNF is $|\hat{\varphi}| \in O(|\varphi| + n)$. $\hat{\varphi}$ has at least one satisfying assignment, specifically the assignment $\forall_{i\in[n+1]} x_i = 1$. We notice that $d_{TV}(P(\hat{\varphi}, \mathtt{w}), P(\psi, \mathtt{w})) = 0$ if and only if $|R_\varphi| = 0$. Thus the existence of a poly-time randomised algorithm for deciding whether $d_{TV}(P(\hat{\varphi}, \mathtt{w}), P(\psi, \mathtt{w})) = 0$ would imply NP $\subseteq$ RP and hence NP=RP. □

Proposition 7. *If there exists a poly-time randomised algorithm for deciding the closeness of a pair of PCs with at least one PC in* CNF*, then NP = RP.*

Proof. $d_{TV}(P(\hat{\varphi}, \mathtt{w}), P(\psi, \mathtt{w})) \geq 0.5$ if and only if $|R_\varphi| > 0$. Assume there exists a poly-time randomised algorithm which returns Reject if $d_{TV}(P(\hat{\varphi}, \mathtt{w}), P(\psi, \mathtt{w})) \geq 0.4$ and Accept if $d_{TV}(P(\hat{\varphi}, \mathtt{w}), P(\psi, \mathtt{w})) \leq 0.1$ with probability $> 2/3$. Such an algorithm would imply BPP $\subseteq$ NP, and hence NP=RP. □

Proposition 8. *If there exists a poly-time randomised algorithm for deciding the equivalence of a pair of PCs with at least one PC in* DNF*, then co-NP = co-RP.*

Proof. For DNFs, deciding validity is known to be co-NP-hard. Given DNF φ and a circuit $\psi = True$, the existence of a poly-time randomised algorithm for checking the equivalence of ψ and φ would imply that co-NP $\subseteq$ co-RP and hence co-NP = co-RP. □

Using Corollary 6.3 from [62], we see that PH collapses due to either of the above implications. From the set inclusions DNF$\subseteq$ SDNNF$\subseteq$ DNNF and CNF$\subseteq$ NNF, we obtain all hardness results. From the fact that d-DNNFs support weighted counting and sampling, we have the existence results.

The following lemma supports our claim in table 5.3.

Lemma 8. *Given a* SDNNF *formula φ (with a v-tree T)*[4]*, and a weight function* w*,* Samp$(\varphi, \mathtt{w})$ *requires polynomial time in the size of φ.*

[4]A variable-tree, or **v-tree**, for a set of variables V is a full, rooted binary tree whose leaves are in one-to-one correspondence with the variables in V.

5.4.3 A Test for Equivalence

For completeness, we recast the d-DNNF circuit equivalence test of Darwiche and Huang [43] into an equivalence test for log-linear probability distributions.

Algorithm 9 Peq$(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \delta)$

1: $m \leftarrow \lceil n/\delta \rceil$
2: $\theta \sim [m]^n$
3: **if** $\pi(\varphi_1, \mathtt{w}_1)(\theta) = \pi(\varphi_2, \mathtt{w}_2)(\theta)$ **then**
4: **Return** Accept
5: **else**
6: **Return** Reject

The algorithm: The pseudocode for Peq is shown in Algorithm 9. Peq takes as input two satisfiable circuits φ_1, φ_2 defined over n Boolean variables, a pair of weight functions $\mathtt{w}_1, \mathtt{w}_2$ and a tolerance parameter $\delta \in (0, 1)$. Recall that a circuit φ and a weight function $\mathtt{w}$ together define the probability distribution $P(\varphi, \mathtt{w})$. Peq returns Accept with confidence 1 if the two probability distributions $P(\varphi_1, \mathtt{w}_1)$ and $P(\varphi_2, \mathtt{w}_2)$ are equivalent, i.e. $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) = 0$. If $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) > 0$, then it returns Reject with confidence at least $1 - \delta$.

The algorithm starts by drawing a uniform random assignment θ from $[m]^n$, where $m = \lceil n/\delta \rceil$. Using the procedure given in Proposition 5.4.4 (in Section 5.4.4), Peq computes the values $\pi(\varphi_1, \mathtt{w}_1)(\theta)$ and $\pi(\varphi_2, \mathtt{w}_2)(\theta)$, where $\pi(\varphi, \mathtt{w})$ is the network polynomial [42]. $\pi(\varphi, \mathtt{w})$ defined as:

$$\pi(\varphi, \mathtt{w}) = \sum_{\sigma \in R_\varphi} \frac{\mathtt{w}(\sigma)}{\mathtt{w}(\varphi)} \left(\prod_{x_i \models \sigma} x_i \prod_{\neg x_j \models \sigma} (1 - x_j) \right)$$

The two values are then compared on line 3, and if they are equal the algorithm returns Accept and otherwise returns Reject. The central idea of the test is that whenever the two distributions $P(\varphi_1, \mathtt{w}_1)$ and $P(\varphi_2, \mathtt{w}_2)$ are equivalent, the polynomials $\pi(\varphi_1, \mathtt{w}_1)$ and $\pi(\varphi_2, \mathtt{w}_2)$ are also equivalent, however when they are not equivalent, the polynomials disagree on atleast $1 - \delta$ fraction of assignments from the set $[m]^n$.

We formally claim and prove the correctness of Peq in Lemma 9 in the following section.

5.4.4 An analysis of Peq

In this section, we present the theoretical analysis of Algorithm 9 (Peq) and the proof of the following lemma.

Lemma 9. *Given two satisfiable probabilistic circuits φ_1, φ_2 and weight functions $\mathtt{w}_1, \mathtt{w}_2$, along with confidence parameter $\delta \in (0, 1)$.*

A. If $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) = 0$, then Peq$(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \delta)$ *returns* Accept *with probability 1.*

B. If $d_{TV}(P(\varphi_1, \mathtt{w}_1), P(\varphi_2, \mathtt{w}_2)) > 0$, then Peq$(\varphi_1, \mathtt{w}_1, \varphi_2, \mathtt{w}_2, \delta)$ *returns* Reject *with probability at least $(1 - \delta)$.*

Peq returns Accept if $\pi(\varphi_1, \mathtt{w}_1)(\sigma) = \pi(\varphi_2, \mathtt{w}_2)(\sigma)$. Since $P(\varphi_1, \mathtt{w}_1) \equiv P(\varphi_2, \mathtt{w}_2) \to \pi(\varphi_1, \mathtt{w}_1) \equiv \pi(\varphi_2, \mathtt{w}_2)$, it follows that Peq always returns Accept for two equivalent probabilistic distributions.

For the proof of Lemma 9(B) we will first define some notation, and then we show (in Lemma 10) that a random assignment over $[m]^n$ is likely to be a witness for non-equivalence with probability $> 1 - \delta$. The proof immediately follows as we know that Peq returns Reject if $\pi(\varphi_1, \mathtt{w}_1)(\sigma) \neq \pi(\varphi_2, \mathtt{w}_2)(\sigma)$.

Definition 23. *$\pi|_{x_i=1}(\varphi, \mathtt{w})$ is a polynomial over $n - 1$ variables, obtained by setting the variable x_i to 1. Similarly $\pi|_{x_i=0}(\varphi, \mathtt{w})$ is obtained by setting the variable x_i to 0, thus:*

$$\pi(\varphi, \mathtt{w}) = (1 - x_i)\pi|_{x_i=0}(\varphi, \mathtt{w}) + x_i\pi|_{x_i=1}(\varphi, \mathtt{w})$$

From the definition, we can immediately infer the following proposition.

Proposition 9. *If there exists a poly-time randomised algorithm for deciding the equivalence of a pair of PCs with at least one PC in* DNF*, then co-NP = co-RP.*

If $\pi(\varphi_1, \mathtt{w}_1) \not\equiv \pi(\varphi_2, \mathtt{w}_2)$ then for all x_i, at least one of the following must be true:

- $\pi|_{x_i=1}(\varphi_1, \mathtt{w}_1) \neq \pi|_{x_i=1}(\varphi_2, \mathtt{w}_2)$
- $\pi|_{x_i=0}(\varphi_1, \mathtt{w}_1) \neq \pi|_{x_i=0}(\varphi_2, \mathtt{w}_2)$

For the proofs in this section, we will use the following notation. For a circuit φ defined over the variables $\{x_1, \ldots, x_n\}$, we define a polynomial $P(\varphi, \mathtt{w}) : \{0,1\}^n \to [0,1]$:

$$P(\varphi, \mathtt{w}) = \sum_{\sigma \in R_\varphi} \frac{\mathtt{w}(\sigma)}{\mathtt{w}(\varphi)} \left(\prod_{x_i \models \sigma} x_i \prod_{\neg x_j \models \sigma} (1 - x_j) \right)$$

We define another polynomial $\pi(\varphi, \mathtt{w})$ which is $P(\varphi, \mathtt{w})$ but defined from $[m]^n \to \mathbb{Q}$ where $[m] = \{1 \ldots, m\}$.

To show that the polynomial $\pi(\varphi, \mathtt{w})$ can be computed in time polynomial in the size of the representation, we will adapt the procedure given by [43].

Proposition 10. *If there exists a poly-time randomised algorithm for deciding the equivalence of a pair of PCs with at least one PC in* DNF*, then co-NP = co-RP.*

Let φ be a circuit over the set $X = \{x_1, \ldots, x_n\}$ of n variables , that admits poly-time WMC. Let $\mathtt{w} : X \to \mathbb{Q}^+$ be a weight function and let $\theta \in [m]^n$ be an assignment to the variables in X and $\theta(x)$ be the assignment to variable $x \in X$ in θ. For each node η in the circuit, define a function $S(\cdot)$ recursively as follows:

- $S(\eta) = \sum_i S(n_i)$, where η is an or-node with children n_i.
- $S(\eta) = \prod_i S(n_i)$, where η is an and-node with children n_i.
- $S(\eta) = \begin{cases} 0, & \text{if } \eta \text{ is a leaf node false} \\ 1, & \text{if } \eta \text{ is a leaf node true} \\ \mathtt{w}(x)\theta(x), & \text{if } \eta \text{ is a leaf node } x, x \in X \\ (1 - \mathtt{w}(x))(1 - \theta(x)), & \text{if } \eta \text{ is a leaf node } \neg x, x \in X \end{cases}$
- $\pi(\varphi, \mathtt{w}) = S(\eta)/\mathtt{w}(\varphi)$, where η is the root node

We can compute the quantity $\mathtt{w}(\varphi)$ in linear time due to our assumption of poly-time WMC, hence we can find $\pi(\varphi, \mathtt{w})(\theta)$ in time linear in the size of the d-DNNF.

Lemma 10. *For a random assignment $\sigma \sim [m]^n$,*

$$\Pr[\pi(\varphi_1, \mathtt{w}_1)(\sigma) \neq \pi(\varphi_2, \mathtt{w}_2)(\sigma) \mid P(\varphi_1, \mathtt{w}_1) \not\equiv P(\varphi_2, \mathtt{w}_2)] > 1 - \delta$$

Proof. For $n = 1$, σ is an assignment to a single variable x. The polynomial on the single variable x can be parameterised as $\pi(\varphi, \mathtt{w})(x) = \alpha x + (1 - \alpha)(1 - x)$ where parameter $\alpha = \frac{\mathtt{w(x)}}{\sum_{\theta \in R_\varphi} \mathtt{w}(\theta)}$. Let polynomials $\pi(\varphi_1, \mathtt{w_1}), \pi(\varphi_2, \mathtt{w_2})$ be parameterised with α_1, α_2, respectively. Our assumption that $P(\varphi_1, \mathtt{w_1}) \not\equiv P(\varphi_2, \mathtt{w_2})$ immediately leads to the fact that $\pi(\varphi_1, \mathtt{w_1}) \not\equiv \pi(\varphi_2, \mathtt{w_2})$ which in turn implies that $\alpha_1 \neq \alpha_2$.

The the set of inputs x for which two non-equivalent polynomials agree is given by,

$$\begin{aligned}
\pi(\varphi_1, \mathtt{w_1})(x) &= \pi(\varphi_2, \mathtt{w_2})(x) \\
\alpha_1 x + (1 - \alpha_1)(1 - x) &= \alpha_2 x + (1 - \alpha_2)(1 - x) \\
2(\alpha_1 - \alpha_2)x &= \alpha_1 - \alpha_2 \\
x &= 1/2
\end{aligned}$$

From the initial assumption we know that x can only take integer values, hence there are no inputs in the set $[m]$ for which $\pi(\varphi_1, \mathtt{w_1})(\sigma) \neq \pi(\varphi_2, \mathtt{w_2})(\sigma)$. Thus, for $n = 1$, and any σ, $\Pr[\pi(\varphi_1, \mathtt{w_1})(\sigma) \neq \pi(\varphi_2, \mathtt{w_2})(\sigma) \mid P(\varphi_1, \mathtt{w_1}) \not\equiv P(\varphi_2, \mathtt{w_2})] = 0$

We now assume that the hypothesis holds for $n - 1$ variables. Consider polynomials $\pi(\varphi_1, \mathtt{w_1}) \not\equiv \pi(\varphi_2, \mathtt{w_2})$ over n variables. From Prop 5.4.4 we know that at least one of the following holds:

- $\pi|_{x_i=1}(\varphi_1, \mathtt{w_1}) \not\equiv \pi|_{x_i=1}(\varphi_2, \mathtt{w_2})$
- $\pi|_{x_i=0}(\varphi_1, \mathtt{w_1}) \not\equiv \pi|_{x_i=0}(\varphi_2, \mathtt{w_2})$

Without any loss of generality we assume the latter. Then we know that there exists a set $\Sigma \subseteq [m]^{n-1}, |\Sigma| \geq (m-1)^{n-1}$, such that

$$\forall_{\sigma \in \Sigma}, \pi|_{x_n=0}(\varphi_1, \mathtt{w_1})(\sigma) \neq \pi|_{x_n=0}(\varphi_2, \mathtt{w_2})(\sigma)$$

The set of assignments σ for which $\pi(\varphi_1, \mathtt{w_1})(\sigma) - \pi(\varphi_2, \mathtt{w_2})(\sigma) = 0$ can we rewritten as

$$\begin{aligned}
&(1 - x_n)\pi|_{x_n=0}(\varphi_1, \mathtt{w_1})(\sigma) + x_n \pi|_{x_n=1}(\varphi_1, \mathtt{w_1})(\sigma) \\
&= (1 - x_n)\pi|_{x_n=0}(\varphi_2, \mathtt{w_2})(\sigma) + x_n \pi|_{x_n=1}(\varphi_2, \mathtt{w_2})(\sigma)
\end{aligned}$$

Factoring out x_n,we get the following:

$$\begin{aligned} x_n(\pi|_{x_n=1}(\varphi_1, \mathtt{w_1})(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathtt{w_1})(\sigma) - \pi|_{x_n=1}(\varphi_2, \mathtt{w_2})(\sigma) + \pi|_{x_n=0}(\varphi_2, \mathtt{w_2})(\sigma)) \\ = \pi|_{x_n=0}(\varphi_2, \mathtt{w_2})(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathtt{w_1})(\sigma) \end{aligned}$$

From the assumptions we know that there are at least $(m-1)^{n-1}$ assignments σ s.t. $\pi|_{x_n=0}(\varphi_2, \mathtt{w_2})(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathtt{w_1})(\sigma) \neq 0$, from which we can conclude that the RHS is non-zero. Thus for all such σ there can be at most one value of x_n for which the equality holds, which leaves $m-1$ values which x_n cannot take. Thus there are at least $(m-1) \times (m-1)^{n-1} = (m-1)^n$ assignments to n variables for which $\pi(\varphi_1, \mathtt{w_1})(\sigma) \neq \pi(\varphi_2, \mathtt{w_2})(\sigma)$.

Since the total number of assignments for n variables is m^n, out of which $(m-1)^n$ witness the non-equivalence of the two probability distributions, we know that for a randomly chosen assignment $\sigma \sim [m]^n$, we have

$$\begin{aligned} \Pr[\pi(\varphi_1, \mathtt{w_1})(\sigma) \neq \pi(\varphi_2, \mathtt{w_2})(\sigma) \mid P(\varphi_1, \mathtt{w_1}) \not\equiv P(\varphi_2, \mathtt{w_2})] &\geq \frac{(m-1)^n}{m^n} \geq \left(1 - \frac{\delta}{n}\right)^n \\ \text{(using } m \text{ from Algorithm 9)} \quad &> 1 - \delta \end{aligned}$$

□

Chapter 6

Polynomial Query Distance Estimation

In this chapter, we are interested in the computation of (ε, δ)-approximation of $d_{TV}(\mathcal{P}, \mathcal{Q})$: i.e., we would like to compute an estimate $\mathtt{est}$ such that

$$\Pr[d_{TV}(\mathcal{P}, \mathcal{Q}) - \varepsilon \leq \mathtt{est} \leq d_{TV}(\mathcal{P}, \mathcal{Q}) + \varepsilon] \geq 1 - \delta$$

TV distance is a fundamental notion in probability and finds applications in the diverse domains of computer science such as generative models [55, 57], MCMC algorithms [4, 14, 17], and probabilistic programming [2, 83].

Theoretical investigations into the problem of TV distance computation have revealed the intractability of exact computation: In particular, the problem is #P-hard even when $\mathcal{P}$ and $\mathcal{Q}$ are represented as product distributions [10]. As a consequence, the focus has been on designing approximation techniques. When $\mathcal{P}$ and $\mathcal{Q}$ are specified explicitly, randomised polynomial time approximation schemes are known for some classes of distributions, such as Bayesian networks with bounded treewidth [11]. Not every practical application allows explicit representation of probability distributions, and often, the output of some underlying process defines probability distributions. Accordingly, the field of distribution testing is concerned with the design of algorithmic techniques for different models of access to the underlying processes. Furthermore, in addition to the classical notion of time complexity, we are also concerned with the *query complexity*: how many queries do we make to a given access model?

The earliest investigations focused on the classical model of access where one is only allowed to access samples from $\mathcal{P}$ and $\mathcal{Q}$ [78, 93]; however, a lower bound of $\Omega(2^n/n)$ [91, 93] restricts the applicability of these estimators in practical scenarios. This motivates the need to focus on more powerful models. In this work, we will focus on the SUBCOND access model owing to its ability to capture

the behavior of probabilistic processes in diverse settings [60, 33, 97]. Formally, the SUBCOND oracle for a distribution $\mathcal{P}$ takes in a query string $\rho \in \{0,1,*\}^n$, constructs the conditioning set $S_\rho = \{\sigma \in \{0,1\}^n | (\rho_i = *) \vee (\rho_i = \sigma_i)\}$ and returns $\sigma \in S_\rho$ with probability $\frac{\mathcal{P}(\sigma)}{\sum_{\pi \in S_\rho} \mathcal{P}(\pi)}$. It is worth remarking that while we use the name SUBCOND to be consistent with recent literature [12], there have been algorithmic frameworks since the late 1980s that have relied on the underlying query model [60].

The starting point of our investigation is the observation that, on the one hand, practical applications of distance estimation rely on heuristic methods and hence don't provide any guarantees. On the other hand, no known algorithm, even when given access to the SUBCOND oracle, makes less than $O(2^n/n)$ queries. The primary aim of our ongoing work is to address the mentioned gap: we want to design the first algorithm that computes (ε, δ)-approximation of TV distance and makes only polynomially many queries to SUBCOND oracle.Formally,

Given two distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\{0,1\}^n$, along with parameters $\varepsilon \in (0,1)$, and $\delta \in (0,1)$, the algorithm DistEstimate$(\mathcal{P}, \mathcal{Q}, \varepsilon, \delta)$ returns estimate κ such that

$$\Pr[\kappa \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 1 - \delta$$

DistEstimate makes $\tilde{\mathcal{O}}\left(n^3 \log(1/\delta)/\varepsilon^4\right)$ queries to the SUBCOND oracle.

We now provide a high-level overview of DistEstimate: From the fact that,

$$d_{TV}(\mathcal{P}, \mathcal{Q}) = \mathop{\mathbb{E}}_{\sigma \sim \mathcal{Q}} \left[\max \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}, 0 \right) \right]$$

we can use the standard approach of sampling σ from $\mathcal{Q}$, estimating $\mathcal{P}(\sigma)$ and $\mathcal{Q}(\sigma)$ up to some multiplicative factor, and then setting the value of the random variable to be $\max(1 - \mathcal{P}(\sigma)/\mathcal{Q}(\sigma), 0)$. This approach requires a constant number of samples from $\mathcal{Q}$ to compute an approximation of $d_{TV}(\mathcal{P}, \mathcal{Q})$. The main issue is that it is not possible to approximate the value of $\mathcal{Q}(\sigma)$ for arbitrary σ with only polynomially many queries to SUBCOND since $\mathcal{Q}(\sigma)$ can be arbitrarily small and the query complexity scales inversely with $\mathcal{Q}(\sigma)$. The key technical contribution lies in showing that using polynomially many SUBCOND oracle calls, we can still compute estimates for $\mathcal{P}(\sigma)$ and $\mathcal{Q}(\sigma)$ at sufficiently many points to find a theoretically guaranteed estimate.

We are interested in designing distance estimation techniques for the SUBCOND model because it effectively captures the behavior of probabilistic processes in practice. Towards this goal, we compute the precise number of queries one would need to the test, and we find that DistEstimate offers a 10^7 factor speedup on problems of dimensionality $n = 70$, for which the baseline sample-based estimator would require $\simeq 10^{18}$ queries – a prohibitively large number. The result is presented in the Figure 6.1. Therefore, we demonstrate the application of DistEstimate in a real-world setting. Sampling from discrete domains such as $\{0,1\}^n$ under combinatorial constraints is a challenging problem; therefore, several heuristic-based samplers have been proposed over the years. We can view a sampler as a probabilistic process, and consequently, one is interested in measuring how far the distribution of a given sampler is from the ideal distribution. Our experiments focus on combinatorial samplers, and SUBCOND is particularly well suited for this problem. We use a prototype of DistEstimate to evaluate the quality of two samplers for different benchmarks. Our empirical evaluation demonstrates the promise of scalability: in particular, DistEstimate offers a 10^7 factor speedup on problems of dimensionality $n = 70$.

Organization In Section 6.1 we define the notation we use in most of the chapter, and we discuss some relevant background material. Then we present the chapter's main contribution, the estimator DistEstimate, along with its proof of correctness in Section 6.2. In Section 6.3, we present the result of the evaluation of our implementation of DistEstimate.

6.1 Notations and Preliminaries

We will focus on probability distributions over $\{0,1\}^n$. For any distribution $\mathcal{D}$ on $\{0,1\}^n$ and an element $\sigma \in \{0,1\}^n$, $\mathcal{D}(\sigma)$ is the probability of σ in distribution $\mathcal{D}$. Further, $\sigma \sim \mathcal{D}$ represents that σ is sampled from $\mathcal{D}$. The total variation (TV) distance of two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ is defined as: $d_{TV}(\mathcal{P},\mathcal{Q}) = \frac{1}{2}\sum_{\sigma\in\{0,1\}^n} |\mathcal{P}(\sigma) - \mathcal{Q}(\sigma)|$. For a random variable v, the expectation is denoted as $\mathbb{E}[v]$, and the variance as $\mathbb{V}[v]$.

For clarity of exposition, we will hide the use of the ceiling operator $\lceil x \rceil$ wher-

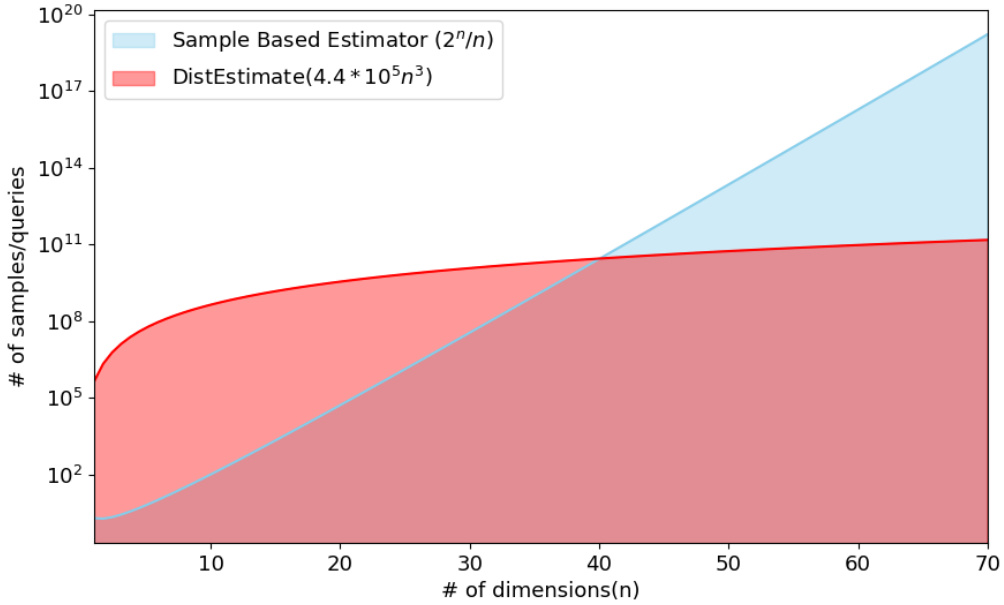


Figure 6.1: A plot comparing the sample/query complexity of the baseline non-conditional estimator vs. our estimator DistEstimate as a function of the number of dimensions n, for $\varepsilon = 0.3$. Note that the vertical axis is in the log scale.

ever integral values are required, such as the number of samples or the number of iterations of a loop. We use $[n]$ to represent the set $\{1, 2 \ldots, n\}$.

Consider a discrete r.v. that takes the value v with probability p. The count of trials required to observe k instances of v follows a *negative binomial* distribution, denoted as $\mathsf{NB}(k, p)$. The expected value $\mathbb{E}[\mathsf{NB}(k, p)]$ is k/p, and its variance $\mathbb{V}[\mathsf{NB}(k, p)]$ is $k(1-p)/p^2$. We also make use of the following tail bound for negative binomials:

Proposition 11 ([18]). *For* $\gamma > 1$, $\Pr[\mathsf{NB}(k, p) > \gamma\mathbb{E}[\mathsf{NB}(k, p)]] \leq \exp\left(-\frac{\gamma k(1-1/\gamma)^2}{2}\right)$

If σ is a string of length $n > 0$, then σ_i denotes the i^{th} element of σ, and for $1 \leq j \leq n$, $\sigma_{<i}$ denotes the substring of σ from 1 to $i-1$, $\sigma_{<i} = \sigma_1 \cdots \sigma_{i-1}$; similarly $\sigma_{\leq i} = \sigma_1 \cdots \sigma_i$, and $\sigma_{<1}$ denotes the empty (length 0) string, also denoted as $\perp$.

For any distribution $\mathcal{D}$ and string ρ, such that $0 \leq |\rho| < n$, the distribution $\mathcal{D}_\rho$ denotes the marginal distribution of $\mathcal{D}$ in the $|\rho| + 1^{th}$ dimension, conditioned on

the string ρ, i.e.,

$$\mathcal{D}_\rho(b) = \frac{\Pr_{\sigma\sim\mathcal{D}}[(\sigma_{|\rho|+1} = b) \wedge (\sigma_{\leq|\rho|} = \rho)]}{\Pr_{\sigma\sim\mathcal{D}}[\sigma_{\leq|\rho|} = \rho]}$$

Definition 24. *A sampling oracle* $\mathsf{SAMP}(\mathcal{D})$ *takes in a distribution* $\mathcal{D}$*, and returns a sample* $\sigma \in \{0,1\}^n$ *such that* $\Pr[\mathsf{SAMP}(\mathcal{D}) = \sigma] = \mathcal{D}(\sigma)$.

Definition 25. *A subcube conditioning oracle* $\mathsf{SUBCOND}(\mathcal{D}, \rho)$ *takes in a distribution* $\mathcal{D}$*, and a query string* ρ *with* $0 \leq |\rho| < n$*, and returns a sample* $\sigma \in \{0,1\}^n$ *such that* $\Pr[\mathsf{SUBCOND}(\mathcal{D}, \rho) = \sigma] = 1_{(\sigma_{\leq|\rho|}=\rho)} \prod_{i=|\rho|+1}^{n} \mathcal{D}_{\sigma_{<i}}(\sigma_i)$.

Definition 26. *A conditional marginal oracle* $\mathsf{CM}(\mathcal{D}, \rho)$ *takes in a distribution* $\mathcal{D}$*, and a query string* ρ *with* $0 \leq |\rho| < n$*, and returns a sample* $b \in \{0,1\}$ *such that* $\Pr[\mathsf{CM}(\mathcal{D}, \rho) = b] = \mathcal{D}_\rho(b)$.

Note that the chain rule implies that $\mathsf{SUBCOND}(\mathcal{D}, \perp)$ is the same as $\mathsf{SAMP}(\mathcal{D})$.

6.1.1 Distance Approximation

We adapt the distance approximation algorithm of Bhattacharyya et al. [9], that takes as input two distributions $\mathcal{P}$ and $\mathcal{Q}$, and provides an (η, δ) estimate of $d_{TV}(\mathcal{P}, \mathcal{Q})$. Recall that we had adapted the estimation algorithm in Theorem 5.1, however we state and prove it again here, as we require different constants and terminology here.

Lemma 6.1. (*Theorem 3.1 in* [9]) *For distributions* $\mathcal{P}$ *and* $\mathcal{Q}$ *over* $\{0,1\}^n$*, and* $\sigma \in \{0,1\}^n$*, let* p_σ *and* q_σ *be functions such that* $p_\sigma \in (1 \pm \eta)\mathcal{P}(\sigma)$*, and* $q_\sigma \in (1 \pm \eta)\mathcal{Q}(\sigma)$*. Given a set of samples* S *from* $\mathcal{Q}$*, and* $\eta \in (0,1)$ *along with the* p_σ *and* q_σ *for each* $\sigma \in S$*, let* $\mathtt{est} = \frac{1}{|S|} \sum_{i\in S} 1_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right)$.

$$\Pr\left[\mathtt{est} \notin \left(d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \frac{3\eta}{1-\eta}\right)\right] \leq 2\exp\left(-2|S|\left(\frac{\eta}{1-\eta}\right)^2\right)$$

Proof. Recall that $p_\sigma \in (1 \pm \eta)\mathcal{P}(\sigma)$ and $q_\sigma \in (1 \pm \eta)\mathcal{Q}(\sigma)$ then, using the definition

of $d_{TV}(\mathcal{P}, \mathcal{Q})$,

$$\begin{aligned} d_{TV}(\mathcal{P}, \mathcal{Q}) &= \sum_{\sigma \in \{0,1\}^n} \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}\right) \mathcal{Q}(\sigma) \\ &= \sum_{\sigma \in \{0,1\}^n} \mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \mathcal{Q}(\sigma) \qquad (6.1) \\ &+ \underbrace{\sum_{\sigma \in \{0,1\}^n} \left(\mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}\right) \mathcal{Q}(\sigma) - \mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \mathcal{Q}(\sigma) \right)}_{A} \end{aligned}$$

The first summand of (6.1) can be written as $\mathbb{E}_{\sigma \sim \mathcal{Q}} \left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right]$.

To bound $|A|$, we will split the domain into three sets, $B_1 = \{x : \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} = \mathbb{1}_{q_\sigma > p_\sigma}\}$, $B_2 = \{x : \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} > \mathbb{1}_{q_\sigma > p_\sigma}\}$ and $B_3 = \{x : \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} < \mathbb{1}_{q_\sigma > p_\sigma}\}$.

$$\begin{aligned} |A| &= \left| \sum_{\sigma \in \{0,1\}^n} \left(\mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}\right) \mathcal{Q}(\sigma) - \mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \mathcal{Q}(\sigma) \right) \right| \\ &\leq \sum_{\sigma \in \{0,1\}^n} \left| \left(\mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}\right) \mathcal{Q}(\sigma) - \mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \mathcal{Q}(\sigma) \right) \right| \\ &= \sum_{\sigma \in B_1} \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left| \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} - \frac{p_\sigma}{q_\sigma} \right| \mathcal{Q}(\sigma) + \sum_{\sigma \in B_2} \mathbb{1}_{\mathcal{Q}(\sigma) > \mathcal{P}(\sigma)} \left(1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)}\right) \mathcal{Q}(\sigma) \\ &+ \sum_{\sigma \in B_3} \mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \mathcal{Q}(\sigma) \end{aligned}$$

For $\sigma \in B_1$, $\left| \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} - \frac{p_\sigma}{q_\sigma} \right| \leq \frac{2\eta}{1-\eta} \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} \leq \frac{2\eta}{1-\eta}$. For $\sigma \in B_2$, $1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} \leq 1 - \frac{1-\eta}{1+\eta} = \frac{2\eta}{1+\eta}$, and for $\sigma \in B_3$, $1 - \frac{\mathcal{P}(\sigma)}{\mathcal{Q}(\sigma)} \leq 1 - \frac{1-\eta}{1+\eta} = \frac{2\eta}{1+\eta}$. Thus, $|A| \leq \sum_{\sigma \in B_1} \frac{2\eta}{1-\eta} \mathcal{Q}(\sigma) + \sum_{\sigma \in B_2} \frac{2\eta}{1+\eta} \mathcal{Q}(\sigma) + \sum_{\sigma \in B_3} \frac{2\eta}{1+\eta} \mathcal{Q}(\sigma) \leq \frac{2\eta}{1+\eta}$. Plugging the bounds on $|A|$ back into (6.1), we get

$$\left| d_{TV}(\mathcal{P}, \mathcal{Q}) - \mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right] \right| \leq \frac{2\eta}{1-\eta} \qquad (6.2)$$

Hence, $\mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right] - \frac{2\eta}{1-\eta} \leq d_{TV}(\mathcal{P}, \mathcal{Q}) \leq \mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right] + \frac{2\eta}{1-\eta}$. The distance estimation algorithm draws $|S|$ samples to estimate $\mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right]$. We will use `est` to denote the empirical estimate of $\mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right]$. Since each sample σ is drawn independently, and $\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right)$ is bounded in $[0, 1]$, we can use the Hoeffding bound as follows,

$$\Pr\left[\left| \texttt{est} - \mathbb{E}\left[\mathbb{1}_{q_\sigma > p_\sigma} \left(1 - \frac{p_\sigma}{q_\sigma}\right) \right] \right| \geq \frac{\eta}{1-\eta} \right] \leq 1 - 2\exp\left(-2|S| \left(\frac{\eta}{1-\eta} \right)^2 \right) \qquad (6.3)$$

Plugging (6.2) into (6.3), we complete the proof:

$$\Pr\left[|\texttt{est} - d_{TV}(\mathcal{P},\mathcal{Q})| \geq \frac{3\eta}{1-\eta}\right] = \Pr\left[d_{TV}(\mathcal{P},\mathcal{Q}) \notin \left(\texttt{est} \pm \frac{3\eta}{1-\eta}\right)\right]$$
$$\leq 2\exp\left(-2|S|\left(\frac{\eta}{1-\eta}\right)^2\right)$$

□

6.1.2 Taming Distributions

Given a distribution $\mathcal{D}$, we will define and construct a new distribution $\mathcal{D}'$ that has desirable properties critical for DistEstimate.

Definition 27. *A distribution $\mathcal{D}'$ is θ-tamed, if*

$$\forall \sigma \in \{0,1\}^n, \forall, \ell \in [n] \quad \mathcal{D}'_{\sigma_{<\ell}}(\sigma_\ell) \in [\theta, 1-\theta]$$

Definition 28. *For a given distribution $\mathcal{D}$, and parameter $\theta \in [0, 1/n)$, distribution $\mathcal{D}'$ is the θ-tamed sibling of $\mathcal{D}$, if $\mathcal{D}'$ is θ-tamed and $d_{TV}(\mathcal{D},\mathcal{D}') \leq \theta n$.*

Henceforth, we will use $\mathcal{D}'$ as shorthand to refer to the θ-tamed sibling of $\mathcal{D}$ and omit mentioning θ whenever θ is evident from the context. We will now show in the following lemma that given SUBCOND query access to distribution $\mathcal{D}$, CM, and SAMP access to $\mathcal{D}'$ can be simulated efficiently.

Lemma 6.2. *Given a distribution $\mathcal{D}$ and parameter $\theta \in [0, 1/n)$, every* CM *query to $\mathcal{D}'$ can be simulated by making one* SUBCOND *query to $\mathcal{D}$, and every* SAMP*query to $\mathcal{D}'$ can be simulated by making n* SUBCOND *queries to $\mathcal{D}$.*

Proof. Our proof adapts the θ-balancing trick, devised for product distributions in Canonne et al. [25, Thm. 6]. To simulate the $\mathsf{CM}(\mathcal{D}', \sigma_{<\ell})$ query using SUBCOND access to $\mathcal{D}$, we use the following process: $all i \geq \ell$, given the substring $\sigma_{<i}$, set $\sigma_i = 0$ with probability $(1-2\theta)\mathcal{D}_{\sigma_{<i}}(0)+\theta$ and $\sigma_i = 1$ with probability $(1-2\theta)\mathcal{D}_{\sigma_{<i}}(1)+\theta$. To implement the above, with probability $1-2\theta$, draw $\rho \sim \mathsf{SUBCOND}(\mathcal{D}, \sigma_{<i})$ and return ρ_i, else with probability 2θ draw a sample uniformly from $\{0,1\}$.

Observe that $all \ell \in [n]$, $c \in \{0,1\}$, and $\rho \in \{0,1\}^{\ell-1}$, we have $\mathcal{D}'_\rho(c) = (1-2\theta)\mathcal{D}_\rho(c)+\theta$. Since $\theta \leq \mathcal{D}'_\rho(c) \leq 1-\theta$, we see that $\mathcal{D}'$ is indeed θ-tamed. To simulate $\mathsf{SAMP}(\mathcal{D}')$, we use the chain rule.

Now we will show that $\mathcal{D}'$ is close to $\mathcal{D}$.

Claim 1. *For distribution $\mathcal{D}$ and its θ-tamed sibling $\mathcal{D}'$, we have $d_{TV}(\mathcal{D}, \mathcal{D}') \leq \theta n$*

Proof. Recall the definition of subcube $S_\rho = \{w \in \{0,1\}^n : w_{\leq|\rho|} = \rho\}$. For any set $S \subseteq \{0,1\}^n$, $\mathcal{D}(S)$ is the total probability of S in $\mathcal{D}$. For any distribution $\mathcal{D}$, string ρ (with $1 \leq |\rho| \leq n$) and $\omega \in \{0,1\}^{n-|\rho|}$, the distribution $\mathcal{D}^\rho$ denotes the marginal distribution of $\mathsf{SUBCOND}(\mathcal{D}, \rho)$ in the remaining dimensions, i.e. for any $\omega \in \{0,1\}^{n-|\rho|}$, $\mathcal{D}^\rho(\omega) = \Pr_{w\sim\mathsf{SUBCOND}(\mathcal{D},\rho)}[w = \rho\omega]$.

Consider the induction hypothesis that $d_{TV}(\mathcal{D}, \mathcal{D}') \leq \theta i$ if $\mathcal{D}$ is supported on $\{0,1\}^i$. To verify the hypothesis for $i = 1$, wlog assume that $\mathcal{D}(0) \leq \mathcal{D}(1)$, then $d_{TV}(\mathcal{D}, \mathcal{D}') = \mathcal{D}(1) - \mathcal{D}'(1) = 2\theta\mathcal{D}(1) - \theta \leq \theta$. Assume the hypothesis holds for all $i \in [n-1]$. Now, we show the hypothesis is true for $i = n$.

Consider a distribution $\mathcal{D}$ over $\{0,1\}^n$ and its θ-tamed sibling $\mathcal{D}'$, then:

$$
\begin{aligned}
d_{TV}(\mathcal{D}, \mathcal{D}') &= \frac{1}{2}\sum_{\sigma\in\{0,1\}^n} |\mathcal{D}(\sigma) - \mathcal{D}'(\sigma)| = \frac{1}{2}\sum_{\rho\in\{0,1\}}\sum_{\omega\in\{0,1\}^{n-1}} |\mathcal{D}(\rho\omega) - \mathcal{D}'(\rho\omega)| \\
&= \frac{1}{2}\sum_{\rho\in\{0,1\}}\sum_{\omega\in\{0,1\}^{n-1}} |\mathcal{D}(S_\rho)\mathcal{D}^\rho(\omega) - \mathcal{D}'(S_\rho)\mathcal{D}'^\rho(\omega)| \\
&= \frac{1}{2}\sum_{\rho\in\{0,1\}}\sum_{\omega\in\{0,1\}^{n-1}} |\mathcal{D}(S_\rho)\mathcal{D}^\rho(\omega) - \mathcal{D}(S_\rho)\mathcal{D}'^\rho(\omega) + \mathcal{D}(S_\rho)\mathcal{D}'^\rho(\omega) - \mathcal{D}'(S_\rho)\mathcal{D}'^\rho(\omega)| \\
&\leq \frac{1}{2}\sum_{\rho\in\{0,1\}}\sum_{\omega\in\{0,1\}^{n-1}} |\mathcal{D}(S_\rho)\mathcal{D}^\rho(\omega) - \mathcal{D}(S_\rho)\mathcal{D}'^\rho(\omega)| + |\mathcal{D}(S_\rho)\mathcal{D}'^\rho(\omega) - \mathcal{D}'(S_\rho)\mathcal{D}'^\rho(\omega)| \\
&= \frac{1}{2}\sum_{\rho\in\{0,1\}}\sum_{\omega\in\{0,1\}^{n-1}} \mathcal{D}(S_\rho)|\mathcal{D}^\rho(\omega) - \mathcal{D}'^\rho(\omega)| + \mathcal{D}'_\rho(\omega)|\mathcal{D}'(S_\rho) - \mathcal{D}(S_\rho)| \\
&= \frac{1}{2}\sum_{\rho\in\{0,1\}} (\mathcal{D}(S_\rho)2d_{TV}(\mathcal{D}^\rho, \mathcal{D}'^\rho)) + \frac{1}{2}\sum_{\rho\in\{0,1\}} |\mathcal{D}'(S_\rho) - \mathcal{D}(S_\rho)| \\
&\leq \sum_{\rho\in\{0,1\}} (\mathcal{D}(S_\rho)\theta(n-1)) + \theta = \theta n
\end{aligned}
$$

We use $|a+b| \leq |a| + |b|$ in the first inequality. In the second, we use the induction hypothesis to bound the first summand, and for the second, we observe that for $c \in \{0,1\}$, $|\mathcal{D}'(c) - \mathcal{D}(c)| \leq \theta$. □

□

6.1.3 Related Work

Distance estimation is one of the many problems in the broader area of distribution testing. Apart from estimation, there is extensive literature on the problems of identity and equivalence testing. The problem of identity testing involves returning Accept if $d_{TV}(\mathcal{P}, \mathcal{P}^*) = 0$ and returning Reject if $d_{TV}(\mathcal{P}, \mathcal{P}^*) > \varepsilon$, where $\mathcal{P}$ is an unknown distribution and $\mathcal{P}^*$ is known, i.e. you have a full description of $\mathcal{P}^*$. Equivalence testing is the generalization of identity testing. It is the problem of deciding between $d_{TV}(\mathcal{P}, \mathcal{Q}) = 0$ and $d_{TV}(\mathcal{P}, \mathcal{Q}) > \varepsilon$ where both $\mathcal{P}$ and $\mathcal{Q}$ are unknown. It is worth emphasizing that for both identity and equivalence testing problems, any answer from the tester (Accept or Reject) is considered valid if $0 < d_{TV}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon$. Provided only sample access, the sample complexity of identity testing is $\Theta\left(2^{n/2}/\varepsilon^2\right)$ [78, 94] and of equivalence testing is $\max(2^{2n/3}\varepsilon^{-4/3}, 2^{n/2}\varepsilon^{-2})$ Chan et al. [32], Valiant and Valiant [94]. While testing is of theoretical interest, its practical application faces significant limitations primarily because testers must accept only when two given distributions are identical. In real-world scenarios, distributions are rarely identical but often exhibit close similarity. Consequently, a simplistic tester that consistently returns Reject can meet the specifications. A more rigorous definition of a tester is required to address this limitation, including estimating the distance between the two distributions. Unfortunately, this introduces a considerable challenge. [93] demonstrate that in the classical sampling model, the necessary number of queries increases to $2^n/n$, a significant jump from the previous $2^{2n/3}$.

To sidestep the exponential lower bounds on testing, the conditional sampling model, or COND, was introduced independently by Chakraborty et al. [27] and Canonne et al. [23], and has been successfully applied to various problems, including identity and equivalence testing. In this model, the sample complexity of identity testing is $\Theta(\varepsilon^{-2})$ (independent of n), while for equivalence testing the best-known upper and lower bounds are $O((\log n)/\varepsilon^5)$ [51], and $\Omega(\sqrt{\log n})$ [1] respectively. A survey by Canonne [21] provides a detailed view of testing and related problems in various sampling models.

Our work investigates the distance estimation problem using the SUBCOND model, a restriction of COND. Unlike COND, which allows conditioning on arbitrary

sets, the SUBCOND model allows conditioning only on sets that are subcubes of the domain. While COND significantly improves the sample complexity, it is not easily implementable in practice, as arbitrary subsets are not efficiently represented and sampled from. With a view towards plausible conditional models, Canonne et al. [23], Bhattacharyya and Chakraborty [12] came up with the SUBCOND model, which is particularly suited to the Boolean hypercube $\{0,1\}^n$. Canonne et al. [24] used the SUBCOND model to construct a nearly-optimal $\Theta(\sqrt{n})$ uniformity testing algorithm for $\{0,1\}^n$, demonstrating its natural applicability for high-dimensional distributions. Then Chen et al. [36] used SUBCOND to study the problems of learning and testing junta distributions supported on $\{0,1\}^n$. Bhattacharyya and Chakraborty [12] developed a test for equivalence in the SUBCOND model, with query complexity of $O(n^2/\varepsilon^2)$. However, before this work, there was no distance estimation algorithm in the SUBCOND oracle model, and indeed even in the general COND model.

6.1.4 Lower Bound

Canonne et al. [25] show an $\Omega(n/\log(n))$ lower bound for the problem.

Theorem 6.1 (Theorem 11 in [25])**.** *An absolute constant $\varepsilon_0 < 1$ exists, such that the following holds. Any algorithm that, given a parameter $\varepsilon \in (0, \varepsilon_0]$, and sample access to product distributions $\mathcal{P}, \mathcal{Q}$ over $\{0,1\}^n$, distinguishes between $d_{TV}(\mathcal{P}, \mathcal{Q}) < \varepsilon$ and $d_{TV}(\mathcal{P}, \mathcal{Q}) > 2\varepsilon$, with probability at least 2/3, requires $\Omega(n/\log(n))$ samples. Moreover, the lower bound still holds in the case where $\mathcal{Q}$ is known, and provided as an explicit parameter.*

The lower bound is shown for the case where the tester has access to samples from a product distribution $\mathcal{P}$ and $\mathcal{Q}$ (over $\{0,1\}^n$). As observed by Bhattacharyya and Chakraborty [12], SUBCOND access is no stronger than SAMP when it comes to product distributions. Thus we get the relevant lower bound:

Corollary 2. *Let $\mathcal{S}(\varepsilon_1, \varepsilon_2, \mathcal{P}, \mathcal{Q})$ be any algorithm that has* SUBCOND *access to distribution $\mathcal{P}$, and explicit knowledge of $\mathcal{Q}$ (defined over $\{0,1\}^n$), and distinguishes between $d_{TV}(\mathcal{P}, \mathcal{Q}) \leq \varepsilon_1$ and $d_{TV}(\mathcal{P}, \mathcal{Q}) > \varepsilon_2$ with probability $> 2/3$. Then, $\mathcal{S}$ makes $\Omega(n/\log(n))$* SUBCOND *queries.*

6.2 DistEstimate: a Distance Estimation Algorithm

We now present the pseudocode of our algorithm DistEstimate, and the SubToEval and SubVsSub subroutines. The following subsection will provide a high-level overview of all our algorithms and formal analysis.

Algorithm 10 DistEstimate$(\mathcal{P}, \mathcal{Q}, \varepsilon, \delta)$

for $j = 1$ **to** $4.5\log(2/\delta)$ **do**
  $r_j \leftarrow \mathsf{SubVsSub}(\mathcal{P}, \mathcal{Q}, \varepsilon)$
$\kappa \leftarrow \mathtt{Median}_j(r_j)$
return κ

Algorithm 11 SubVsSub$(\mathcal{P}, \mathcal{Q}, \varepsilon)$

$\eta \leftarrow \varepsilon/(\varepsilon + 4)$
$m_{out} \leftarrow \frac{\log(24)}{2}\left(\frac{1-\eta}{\eta}\right)^2$
$m_{in} \leftarrow 32\log(48m_{out})$
$\mathtt{est} \leftarrow 0$
for all $i = 1$ **to** m_{out} **do**
  $\sigma \leftarrow \mathsf{SAMP}(\mathcal{Q}')$
  for all $j = 1$ **to** m_{in} **do**
    $p_j \leftarrow \mathsf{SubToEval}(\mathcal{P}', \sigma, \eta)$
    $q_j \leftarrow \mathsf{SubToEval}(\mathcal{Q}', \sigma, \eta)$
  $\hat{p} \leftarrow \mathtt{Median}_j(p_j)$
  $\hat{q} \leftarrow \mathtt{Median}_j(q_j)$
  if $\hat{q} > \hat{p}$ **then**
    $\mathtt{est} \leftarrow \mathtt{est} + 1 - \hat{p}/\hat{q}$
return $\mathtt{est}/m_{out}$

6.2.1 High-Level Overview

In Section 6.2.1.1, we introduce the main ideas of our algorithms, DistEstimate and SubVsSub. Then, in Section 6.2.1.2, we explain the key concepts of the SubToEval subroutine.

6.2.1.1 Outline of the DistEstimate and SubVsSub routines

The pseudocode of DistEstimate and SubVsSub is given in Alg. 10 and 11 respectively. DistEstimate takes as input two distributions $\mathcal{P}$ and $\mathcal{Q}$ defined over the support $\{0,1\}^n$, along with the parameter ε for tolerance and the parameter δ for

Algorithm 12 SubToEval$(\mathcal{D}', \sigma, \eta)$

$t \leftarrow 0$
$k \leftarrow 4n\eta^{-2}(1+\eta^2)$
for all $i = 1$ **to** n **do**
  $x_i \leftarrow 0$
  $f \leftarrow 0$
  while $f < k$ **do**
    $\alpha \leftarrow \mathsf{CM}(\mathcal{D}', \sigma_{<i})$
    $x_i \leftarrow x_i + 1$
    $t \leftarrow t + 1$
    if $t = 64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$ **then**
      return 0
    $f \leftarrow f + \mathbb{1}(\alpha = \sigma_i)$
$d \leftarrow \prod_{i=1}^{n} k/x_i$
return d

confidence, and returns an ε-additive estimate of $d_{TV}(\mathcal{P}, \mathcal{Q})$ with probability at least $1 - \delta$.

The SubVsSub subroutine call returns an estimate r_j of $d_{TV}(\mathcal{P}, \mathcal{Q})$ such that $\Pr[r_j \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 2/3$, and DistEstimate makes $48\log(1/\delta)$ calls to boost the overall probability to $1 - \delta$, using the Chernoff bound on the median of the estimates.

SubVsSub takes as input the distributions $\mathcal{P}$ and $\mathcal{Q}$, and creates their $\varepsilon/8n$-tamed siblings $\mathcal{P}'$ and $\mathcal{Q}'$ that are $\varepsilon/8$ close to $\mathcal{P}$ and $\mathcal{Q}$ in TV distance, and have the property that all of their marginal probabilities are lower bounded by $\Omega(\varepsilon/8n)$. The bounded marginal property of $\mathcal{P}'$ and $\mathcal{Q}'$ is crucial for the polynomial query complexity of SubVsSub. The construction of $\mathcal{P}'$ and $\mathcal{Q}'$, and the claimed guarantees, are discussed in Section 6.1.2. SubVsSub then computes the constants η, m_{out}, and m_{in} (the counts of iterations of the outer and inner loop).

SubVsSub then draws m_{out} samples $\sigma \sim \mathcal{Q}'$, and for each sample σ, calls SubToEval m_{in} times to find the $(1 \pm \eta)$ estimates of $\mathcal{Q}'(\sigma)$ and $\mathcal{P}'(\sigma)$. The SubToEval subroutine puts an upper limit on the number of CM oracle calls, and the limit is set high enough to ensure that the estimates, $\hat{p}$ and $\hat{q}$, are correct with the required confidence. SubVsSub then computes the distance using these estimates as given in Lemma 6.1.

6.2.1.2 Outline of the SubToEval subroutine

The SubToEval subroutine takes as input an element $\sigma \in \{0,1\}^n$, a distribution $\mathcal{D}$ over $\{0,1\}^n$, and a parameter η. SubToEval outputs an η-multiplicative estimate of $\mathcal{D}(\sigma)$. The probability $\mathcal{D}(\sigma)$ can be expressed as a product of marginals, $\mathcal{D}(\sigma) = \prod_{i=1}^{n} \mathcal{D}_{\sigma_{<i}}(\sigma_i)$, by applying the chain rule. Essentially, the subroutine approximates each marginal $\mathcal{D}_{\sigma_{<i}}(\sigma_i)$ by k/x_i for each $i \in [n]$, using the CM oracle. The product $\prod_{i=1}^{n} k/x_i$ is then employed as the final estimate for $\mathcal{D}(\sigma)$.

In this context, the variable x_i represents the total count of $\mathsf{CM}(\mathcal{D}, \sigma_{<i})$ queries executed until k occurrences of σ_i are observed. Given that $\mathcal{D}_\rho(b) = \Pr_{w \sim \mathsf{CM}(\mathcal{D},\rho)}[w = b]$ for any ρ (as discussed in Section 6.1), the ratio k/x_i is an intuitive choice as an estimator for $\mathcal{D}_{\sigma_{<i}}(\sigma_i)$. Moreover, to ensure the subroutine terminates, a total number of calls to the CM oracle are monitored, and if they ever exceed the threshold $64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$, the subroutine terminates and returns 0.

We now discuss our technical contribution - showing the correctness of SubToEval when the threshold is set to $O(n^3)$ (for this discussion, we will set aside the dependency on η). To estimate $\mathcal{D}(\sigma)$, it is essential to estimate each of the n marginals, $\mathcal{D}_{\sigma_{<i}}(\sigma_i)$, to within an error margin of approximately $1 + 1/n$. This would require at least $n^2/\mathcal{D}_{\sigma_{<i}}(\sigma_i)$ queries for each marginal. Consequently, the total query complexity would sum up to $\sum_{i=1}^{n} n^2/\mathcal{D}_{\sigma_{<i}}(\sigma_i)$. This quantity is at least $\Omega(n^2)$, but it could potentially be unbounded as $\mathcal{D}_{\sigma_{<i}}(\sigma_i)$ can take arbitrarily small values. In the forthcoming section, we reduce this complexity to $O(n^3)$ through a more nuanced analysis.

6.2.2 Theoretical Analysis

In this section, we will prove our main Theorem 6. The proof of Theorem 6 relies on Lemma 6.3, which claims the correctness of the SubToEval subroutine and upper bound its query complexity. We will prove the lemma later.

Lemma 6.3. $\mathsf{SubToEval}(\mathcal{D}', \sigma, \eta)$ *takes as input distribution* $\mathcal{D}'$, $\sigma \in \{0,1\}^n$, $\eta \in (0, 1/5)$ *and returns* d, *then*

$$\Pr[d \in (1 \pm \eta)\mathcal{D}'(\sigma)] \geq 5/8$$

SubToEval *makes* $O(n^3/\eta^2)$ CM *queries to* $\mathcal{D}'$.

Given two distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\{0,1\}^n$, along with parameters $\varepsilon \in (0,1)$, and $\delta \in (0,1)$, the algorithm DistEstimate$(\mathcal{P}, \mathcal{Q}, \varepsilon, \delta)$ returns estimate κ such that

$$\Pr[\kappa \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 1 - \delta$$

DistEstimate makes $\tilde{\mathcal{O}}\left(n^3 \log(1/\delta)/\varepsilon^4\right)$ queries to the SUBCOND oracle.

Proof. We will first show that the algorithm SubVsSub$(\mathcal{P}, \mathcal{Q}, \varepsilon)$ returns $\texttt{est}$ such that

$$\Pr[\texttt{est} \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 5/6$$

Since DistEstimate returns the median of the independent estimates provided by SubVsSub, then applying the Chernoff bound, we have $\Pr[\kappa \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 1 - \delta$.

We will now consider the events that could lead to an incorrect estimate. Recalling that $\mathcal{P}'$ and $\mathcal{Q}'$ are $\varepsilon/8n$-tamed siblings of $\mathcal{P}$ and $\mathcal{Q}$ we define $\texttt{Bad}_i^{\hat{p}}$ and $\texttt{Bad}_i^{\hat{q}}$ to be the events that in the i^{th} iteration of SubVsSub, $\hat{p} \notin (1 \pm \eta)\mathcal{P}'(\sigma)$, and $\hat{q} \notin (1 \pm \eta)\mathcal{Q}'(\sigma)$, respectively. We bound the probability of $\texttt{Bad}_i^{\hat{p}}$ and $\texttt{Bad}_i^{\hat{q}}$ in the following claim.

Claim 2.

$$\Pr[\texttt{Bad}_i^{\hat{p}}] \leq 1/24m_{out} \qquad \textit{and} \qquad \Pr[\texttt{Bad}_i^{\hat{q}}] \leq 1/24m_{out}$$

Proof. For a fixed iteration j, applying Lemma 6.3 we have $\Pr[p_j \in (1 \pm \eta)\,\mathcal{P}'(\sigma)] \geq 5/8$. Since $\hat{p}$ is the median of independent observations $p_j \in [0,1]$, over $j \in [m_{in}]$, we can use the Chernoff bound to derive the claimed bound, $\Pr[\texttt{Bad}_i^{\hat{p}}] \leq 1/24m_{out}$. The proof for the claim $\Pr[\texttt{Bad}_i^{\hat{q}}] \leq 1/24m_{out}$ proceedes identically. □

Now we define $\texttt{Bad} = \bigcup_{i \in [m_{out}]}(\texttt{Bad}_i^{\hat{p}} \cup \texttt{Bad}_i^{\hat{q}})$, i.e., Bad captures the event that at least one of the estimates is incorrect. Then from Claim 2 and the union bound,

$$\begin{aligned}\Pr[\texttt{Bad}] &= \Pr\left[\bigcup_{i \in [m_{out}]} (\texttt{Bad}_i^{\hat{p}} \cup \texttt{Bad}_i^{\hat{q}})\right] \\ &\leq \sum_{i \in [m_{out}]} (\Pr[\texttt{Bad}_i^{\hat{p}}] + \Pr[\texttt{Bad}_i^{\hat{q}}]) \leq m_{out}\left(\frac{1}{24m_{out}} + \frac{1}{24m_{out}}\right) \leq \frac{1}{12}\end{aligned}$$

Now, let's assume the event $\overline{\texttt{Bad}}$. We have a set of m_{out} samples from $\mathcal{Q}'$, and for each sample σ we have $\hat{p}$ and $\hat{q}$ such that $\hat{p} \in (1 \pm \eta)\mathcal{P}'(\sigma)$ and $\hat{q} \in (1 \pm \eta)\mathcal{Q}'(\sigma)$.

This fulfills the condition of Lemma 6.1, and hence substituting $|S| = m_{out}$ (Line 2 of Alg.11) we have,

$$\Pr\left[\texttt{est} \notin \left(d_{TV}(\mathcal{P}', \mathcal{Q}') \pm \frac{3\eta}{1-\eta}\right) \cap \overline{\texttt{Bad}}\right]$$
$$\leq 2\exp\left(-2m_{out}\left(\frac{\eta}{1-\eta}\right)^2\right) \leq 2\exp\left(-\log(24)\right) = \frac{1}{12}$$

Substituting $\eta = \frac{\varepsilon}{\varepsilon+4}$ from Alg.11, we have,

$$\Pr\left[\texttt{est} \notin \left(d_{TV}(\mathcal{P}', \mathcal{Q}') \pm \frac{3\varepsilon}{4}\right) \cap \overline{\texttt{Bad}}\right] \leq \frac{1}{12}$$

Then,

$$\Pr\left[\texttt{est} \notin \left(d_{TV}(\mathcal{P}', \mathcal{Q}') \pm \frac{3\varepsilon}{4}\right)\right] \leq \Pr\left[\texttt{est} \notin \left(d_{TV}(\mathcal{P}', \mathcal{Q}') \pm \frac{3\varepsilon}{4}\right) \cap \overline{\texttt{Bad}}\right] + \Pr[\texttt{Bad}]$$
$$\leq 1/12 + 1/12 = 1/6$$

Since $\mathcal{P}'$ and $\mathcal{Q}'$ are $\varepsilon/8n$-tamed siblings of $\mathcal{P}$ and $\mathcal{Q}$, from Lemma 6.2 we know that $d_{TV}(\mathcal{P}', \mathcal{P}) \leq \varepsilon/8$ and $d_{TV}(\mathcal{Q}', \mathcal{Q}) \leq \varepsilon/8$. Then, from the triangle inequality, we have the bounds on $d_{TV}(\mathcal{P}, \mathcal{Q})$:

$$d_{TV}(\mathcal{P}', \mathcal{Q}') \in d_{TV}(\mathcal{P}, \mathcal{Q}) \pm (d_{TV}(\mathcal{P}', \mathcal{P}) + d_{TV}(\mathcal{Q}', \mathcal{Q}))$$
$$\in d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon/4$$

Combining the two, we get that $\Pr[\texttt{est} \notin (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \leq 1/6$, and hence we have our claim.

Now, we will complete the proof by showing an upper bound on the query complexity. The total number of CM queries made by $\mathsf{SubToEval}(\mathcal{D}', \sigma, \eta)$ in a single invocation is $64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1} = O(n^3\varepsilon^{-3})$. Then $\mathsf{SubVsSub}(\mathcal{P}, \mathcal{Q}, \varepsilon)$ makes $m_{in}m_{out} = O(\varepsilon^{-2}\log(\varepsilon^{-1}))$ many calls to SubToEval. Finally, DistEstimate calls SubVsSub $48\log(1/\delta)$ many times. Thus the total number of queries to the CM oracle made by DistEstimate is $O\left(n^3\log(1/\delta)\log(\varepsilon^{-1})/\varepsilon^5\right)$. □

Proof of Lemma 6.3. Consider the subroutine $\mathsf{SubToEval}_1(\mathcal{D}', \sigma, \eta)$ (Alg. 13), that is the same as $\mathsf{SubToEval}(\mathcal{D}', \sigma, \eta)$ (Alg. 12) except in one critical aspect: the termination condition on Line 10 of SubToEval has been removed. This implies that while $\mathsf{SubToEval}(\mathcal{D}', \sigma, \eta)$ terminates if the number of calls to the CMoracle exceeds

the threshold $64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$, $\mathsf{SubToEval}_1(\mathcal{D}', \sigma, \eta)$ does not enforce this restriction, thereby allowing an unlimited number of calls to the CMoracle. Note that we use variable names d_1 and t_1 in SubToEval′ to distinguish them from d of t of SubToEval. This modification is critical for our analysis as it leads to the variable x_i in $\mathsf{SubToEval}_1(\mathcal{D}', \sigma, \eta)$ following the negative binomial distribution.

Algorithm 13 $\mathsf{SubToEval}_1(\mathcal{D}', \sigma, \eta)$

1: $t_1 \leftarrow 0$
2: $k \leftarrow 4n\eta^{-2}(1+\eta^2)$
3: **for all** $i = 1$ **to** n **do**
4: $\quad x_i \leftarrow 0$
5: $\quad f \leftarrow 0$
6: $\quad$ **while** $f < k$ **do**
7: $\quad\quad \alpha \leftarrow \mathsf{CM}(\mathcal{D}', \sigma_{<i})$
8: $\quad\quad x_i \leftarrow x_i + 1$
9: $\quad\quad t_1 \leftarrow t_1 + 1$
10: $\quad\quad f \leftarrow f + \mathbb{1}(\alpha = \sigma_i)$
11: $d_1 \leftarrow \prod_{i=1}^{n} k/x_i$
12: **return** d_1

Remark 3. Henceforth we will use t_1 and x_i to denote the final values of t_1 and x_i, as on Line 11.

We will now show that the $\mathsf{SubToEval}_1(\mathcal{D}', \sigma, \eta)$ correctly estimates $\mathcal{D}'(\sigma)$ with high probability (Lemma 6.4) and then we show that it makes fewer than $64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$ calls to CM oracle with high probability (Lemma 6.5). These results will help us establish analogous results for the subroutine $\mathsf{SubToEval}(\mathcal{D}, \sigma, \eta)$ and in validating our Lemma 6.3.

Observation 1. *Comparing* SubToEval *and* $\mathsf{SubToEval}_1$, *we observe that* SubToEval *returns an incorrect estimate* d *in two cases. Either* SubToEval *returns incorrect* d_1, *or else* $\mathsf{SubToEval}_1$ *makes more than* $64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$ *queries. Stated formally,*

$$\Pr[d \notin (1 \pm \eta)\mathcal{D}'(\sigma)] \leq \Pr\left[d_1 \notin (1 \pm \eta)\mathcal{D}'(\sigma)\right] + \Pr\left[t_1 \geq 64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}\right]$$

Our proof will use the following propositions and lemmas

Proposition 12. *For* $i \in [n]$, *the value of* x_i *(in Alg. 13) is distributed as* $\mathsf{NB}(k, \mathcal{D}_{\sigma_{<i}}(\sigma_i))$

Proof. Fix any $i \in [n]$. In Alg. 13, the r.v α takes the value σ_i with probability $\mathcal{D}_{\sigma_{<i}}(\sigma_i)$. Note that while the value of x_i increments by one in every iteration of the

loop (lines 6-11), while the value of f increases by one only when $\alpha = \sigma_i$. Since the loop runs until the value of f is k, the distribution of x_i is $\mathsf{NB}(k, \mathcal{D}_{\sigma_{<i}}(\sigma_i))$. □

Lemma 6.4. $\Pr[d_1 \in (1 \pm \eta)\mathcal{D}'(\sigma)] \geq 2/3$.

Proof. We use a variance reduction technique introduced by Dyer and Frieze [48]. x_i on Line 11 is distributed according to $\mathsf{NB}(k, \mathcal{D'}_{\sigma_{<i}}(\sigma_i))$, so we have $\mathbb{E}[x_i] = k/\mathcal{D'}_{\sigma_{<i}}(\sigma_i)$, and hence, $k/\mathbb{E}[x_i] = \mathcal{D'}_{\sigma_{<i}}(\sigma_i)$. Now since $d_1 = \prod_{j=1}^{n} k/x_i$, we have $\mathbb{E}[1/d_1] = \mathbb{E}[\prod_{i=1}^{n} x_i/k] = \prod_{i=1}^{n} 1/\mathcal{D'}_{\sigma_{<i}}(\sigma_i)$.

$$\begin{aligned}
\frac{\mathbb{V}[1/d_1]}{\mathbb{E}[1/d_1]^2} &= \frac{\mathbb{E}[1/d_1^2]}{\mathbb{E}[1/d_1]^2} - 1 \\
&= \prod_{i=1}^{n} \frac{\mathbb{E}[(x_i/k)^2]}{\mathbb{E}[x_i/k]^2} - 1 \\
&= \prod_{j=1}^{n} \left(1 + \frac{\mathbb{V}[x_i/k]}{\mathbb{E}[x_i/k]^2}\right) - 1
\end{aligned}$$

Using the fact that x_i is negative binomial, we substitute $\mathbb{V}[x_i/k]$ and $\mathbb{E}[x_i/k]^2$,

$$\begin{aligned}
\frac{\mathbb{V}[1/d_1]}{\mathbb{E}[1/d_1]^2} &= \prod_{j=1}^{n} \left(1 + \frac{(1 - \mathcal{D'}_{\sigma_{<i}})/k\mathcal{D'}^2_{\sigma_{<i}}}{(1/\mathcal{D'}_{\sigma_{<i}})^2}\right) - 1 \\
&= \prod_{j=1}^{n} \left(1 + \frac{1 - \mathcal{D'}_{\sigma_{<i}}(\sigma_i)}{k}\right) - 1 \\
&\leq \prod_{j=1}^{n} \left(1 + \frac{1}{k}\right) - 1
\end{aligned}$$

Substituting the value of k from the algorithm, we have

$$\begin{aligned}
\frac{\mathbb{V}[1/d_1]}{\mathbb{E}[1/d_1]^2} &\leq \left(1 + \frac{\eta^2}{4n(1+\eta)^2}\right)^n - 1 \\
&\leq \exp\left(\frac{\eta^2}{4}\right) - 1 \\
&\leq \frac{\eta^2}{3(1+\eta)^2} \qquad (6.4)
\end{aligned}$$

The last inequality comes from the fact that for $r \in (0,1), s > 1$, $\exp\left(\frac{r}{s+1}\right) \leq 1 + \frac{r}{s}$. Recall that from the chain rule we have $\mathcal{D}'(\sigma) = \prod_{j=1}^{n} \mathcal{D'}_{\sigma_{<i}}(\sigma_i)$, then $\mathbb{E}[1/d_1] =$

$1/\mathcal{D}'(\sigma)$.

$$\begin{aligned}
\Pr[d_1 \in (1 \pm \eta)\mathcal{D}'(\sigma)] &= \Pr\left[\frac{1}{d_1} \in \left[\frac{1}{1+\eta}, \frac{1}{1-\eta}\right] \frac{1}{\mathcal{D}'(\sigma)}\right] \\
&= \Pr\left[\frac{1}{d_1} - \mathbb{E}\left[\frac{1}{d_1}\right] \in \left[-\frac{\eta}{1+\eta}, \frac{\eta}{1-\eta}\right] \mathbb{E}\left[\frac{1}{d_1}\right]\right] \\
&\geq \Pr\left[\left|\mathbb{E}\left[\frac{1}{d_1}\right] - \frac{1}{d_1}\right| \leq \frac{\eta}{1+\eta}\mathbb{E}\left[\frac{1}{d_1}\right]\right] \\
&\geq 1 - \frac{(1+\eta)^2}{\eta^2} \frac{\mathbb{V}\left[\frac{1}{d_1}\right]}{\mathbb{E}\left[\frac{1}{d_1}\right]^2} \\
&\geq 1 - \frac{1}{3} = \frac{2}{3} \qquad (6.5)
\end{aligned}$$

We use the Chebyshev bound to get the second to last inequality and then substitute (6.4). □

Note that in every iteration, t_1 gets incremented by the value of x_i. In the following lemma, we claim that t_1, the number of queries made by $\mathsf{SubToEval}_1$, exceeds the threshold on Line 10 of SubToEval with low probability.

Lemma 6.5.

$$\Pr[t_1 \geq 64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}] \leq 1/24$$

Proof. The number of CMcalls made by $\mathsf{SubToEval}_1$ in the i^{th} iteration is captured by x_i. Recall from Prop 12 that x_i is drawn from $\mathsf{NB}(k, \mathcal{D}'_{\sigma_{<i}}(\sigma_i))$, and therefore we have,

$$\mathbb{E}[x_i] = k/\mathcal{D}'_{\sigma_{<i}}(\sigma_i) = 4n\eta^{-2}(1+\eta)^2/\mathcal{D}'_{\sigma_{<i}}(\sigma_i) \qquad \text{(Using } k \text{ from Line 2 of } \mathsf{SubToEval}_1\text{)}$$

From the fact that the distribution is $\varepsilon/8n$-tamed, we know that $\mathcal{D}'_{\sigma_{<i}}(\sigma_i) \geq \varepsilon/8n$. Hence we have $\mathbb{E}[x_i] \leq 32n^2\eta^{-2}(1+\eta)^2\varepsilon^{-1}$. Since $t_1 = \sum_{i\in[n]} x_i$, we have that

$\mathbb{E}[t_1] = \mathbb{E}[\sum_{i\in[n]} x_i] = n\mathbb{E}[x_i] \leq 32n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}$. Thus,

$$\begin{aligned}
\Pr[t_1 \geq 64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}] &= \Pr\left[t_1 \geq 2\mathbb{E}[t_1]\right] \\
&\leq \Pr\left[\sum_{i\in[n]} x_i \geq 2\mathbb{E}\left[\sum_{i\in[n]} x_i\right]\right] \\
&\leq \sum_{i\in[n]} \Pr\left[x_i \geq 2\mathbb{E}[x_i]\right] \\
\text{(Prop. 11)} \qquad &\leq \sum_{i\in[n]} \exp(-2k(1-1/2)^2/2) \\
&= n\exp(-k/4) \\
\text{(Substituting } k \text{ and } \eta \leq 1/5, \varepsilon < 1) \qquad &\leq n\exp(-n\eta^{-2}(1+\eta)^2\varepsilon^{-1}) \\
&\leq n\exp(-9n) \leq 1/24
\end{aligned}$$

In the last inequality we used the fact that for $s > 0$, $xe^{-sx} \leq 1/es$. □

Putting together lemmas 6.4 and 6.5 along with the observation 1 , we complete the proof:

$$\begin{aligned}
\Pr[d \notin (1\pm\eta)\mathcal{D}'(\sigma)] &\leq \Pr\left[d_1 \notin (1\pm\eta)\mathcal{D}'(\sigma)\right] + \Pr\left[t_1 \geq 64n^3\eta^{-2}(1+\eta)^2\varepsilon^{-1}\right] \\
&\leq \frac{1}{3} + \frac{1}{24} = \frac{3}{8}
\end{aligned}$$

□

6.2.3 The Discrete Hypergrid Σ^n

This section extends our results beyond the hypercube $\{0,1\}^n$ to the hypergrid Σ^n, where Σ is any discrete set. This line of investigation is motivated by the fact that in modern ML, distributions models are frequently described over hypergrids. For instance, language models are defined to be distributions over Σ^n where Σ is the set of tokens, and n the length of the generated string. Furthermore, the SUBCOND oracle is particularly suitable for use in ML applications as it models autoregressive generation.

The SUBCOND oracle for $\mathcal{D}$ supported on Σ^n, takes a query string $\rho \in \{\Sigma \cup *\}^n$ and draws samples from the set of strings that match all the non-$*$ characters of ρ. As noted in [35], algorithms for $\{0,1\}^n$ do not immediately translate into

algorithms for Σ^n, because the SUBCOND oracle does not work with the natural reduction of replacing elements $c \in \Sigma$ with their binary encoding. Nevertheless, SubVsSub can be extended to distributions over Σ^n, incurring a linear dependence on $|\Sigma|$.

We will now restate our result adapted to the new setting: Given two distributions $\mathcal{P}$ and $\mathcal{Q}$ over Σ^n, along with parameters $\varepsilon \in (0, 1)$, $\delta \in (0, 1/2)$, the algorithm DistEstimate$(\mathcal{P}, \mathcal{Q}, \varepsilon, \delta)$, and with probability at least $1 - \delta$ returns κ, s.t.

$$\Pr[\kappa \in (d_{TV}(\mathcal{P}, \mathcal{Q}) \pm \varepsilon)] \geq 1 - \delta$$

DistEstimate$(\mathcal{P}, \mathcal{Q}, \varepsilon, \delta)$ makes $\tilde{O}\left(n^3|\Sigma| \log(1/\delta)/\varepsilon^5\right)$ SUBCOND queries.

The only change required in DistEstimate to make it work for distributions over the Σ^n is in the construction of the tamed siblings $\mathcal{P}'$, and $\mathcal{Q}'$. We update the taming parameter from $\varepsilon/8n$ to $\varepsilon/8n|\Sigma|$. Since the query complexity is proportional to $1/\theta$, we observe a linear dependence on $|\Sigma|$.

6.3 Experiments

We implemented DistEstimate in Python. We focus on distributions generated by state-of-the-art combinatorial samplers STS[49] and CMSGen[53]. Our assessment included two datasets: (1) `scalable` comprising random Boolean functions over n variables, with n ranging from 30 to 70, and (2) `real-world`, containing instances from the ISCAS89 dataset, a standard in combinatorial testing and sampling evaluations [69]. To determine the ground truth TV distance for the above instances, we implement a learning-based distance estimator [22].

For our experiments, we set the tolerance $\varepsilon = 0.3$ and confidence $\delta = 0.4$ as the default throughout the evaluation. These parameters indicate that the estimate returned by DistEstimateis expected to be within ± 0.3 of the ground truth, with a probability of at least 0.6.

The experiments were conducted on a cluster with AMD EPYC 7713 CPU cores. We use 32 cores with 4GB of memory for each benchmark and a 24-hour timeout per instance.

Our aim was to answer the question: To what extent does DistEstimate scale, i.e., how many dimensions can the estimator handle while providing guarantees?

Table 6.1: The sample complexity and runtime performance of DistEstimate on real-world instances.

Benchmark	Dimensions	STS		CMSGen	
		# of samples	time (in s)	# of samples	time (in s)
s1196a_3_2	33	1.8e+09	4.1e+05	1.9e+09	5.3e+05
53.sk_4_32	33	1.7e+09	2.5e+05	1.9e+09	1.6e+06
27.sk_3_32	33	1.7e+09	1.9e+05	1.9e+09	1.0e+06
s1196a_7_4	33	1.8e+09	4.6e+05	1.9e+09	5.5e+05
s420_15_7	35	2.1e+09	4.2e+05	2.3e+09	4.0e+05
111.sk_2_36	37	2.2e+09	3.5e+05	8.3e+08	6.6e+05

We found that DistEstimate scales to $n = 70$ dimensional problems, a regime where the baseline sample-based estimators would require $10^7\times$ more samples. The estimates are empirically confirmed to be of high quality when compared against the ground truth, falling within the allowed tolerance bound in all cases where we could determine the ground truth.

Table 6.1 details the performance of DistEstimate on 6 `real-world` benchmarks. The algorithm successfully finished on all benchmarks with dimensionality up to $n = 37$. The table specifies the benchmark name, dimensionality, sample count, and processing time for both STS and CMSGen.

The sample complexity of DistEstimate relative to a baseline sample-based estimator is illustrated in Figure 6.1. For this, we use `scalable` benchmarks. Remarkably, for the largest instance handled ($n = 70$ dimensions), DistEstimate outperformed the baseline by a factor greater than 10^7.

Bibliography

[1] Jayadev Acharya, Clément L. Canonne, and Gautam Kamath. A chasm between identity and equivalence testing with conditional queries. *Electron. Colloquium Comput. Complex.*, 2014.

[2] Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. A pre-expectation calculus for probabilistic sensitivity. *Programming Languages*, (POPL), 2021.

[3] Jürgen Altmann and Frank Sauer. Autonomous weapon systems and strategic stability. *Survival*, 2017.

[4] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 2003.

[5] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. When is approximate counting for conjunctive queries tractable? *STOC 2021*, 2021.

[6] Dominik Aronsky and Peter J Haug. Diagnosing community-acquired pneumonia with a bayesian network. In *Proceedings of the AMIA Symposium*, 1998.

[7] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 2013.

[8] Bhaswar Bhattacharya and Gregory Valiant. Testing closeness with unequal sized samples. *Advances in Neural Information Processing Systems*, 28, 2015.

[9] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, and N. V. Vinodchandran. Efficient distance approximation for structured high-dimensional distributions via learning. In *Proceedings of the 34th International Conference on*

Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[10] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. On approximating total variation distance. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 2023.

[11] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. Total variation distance meets probabilistic inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[12] Rishiraj Bhattacharyya and Sourav Chakraborty. Property testing of joint distributions using conditional samples. *ACM Transactions on Computation Theory (TOCT)*, 2018.

[13] Eric Blais, Clément L Canonne, and Tom Gur. Distribution testing lower bounds via reductions from communication complexity. *ACM Transactions on Computation Theory (TOCT)*, 2019.

[14] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 2004.

[15] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011.

[16] Stephen P Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 1998.

[17] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[18] Daniel G Brown. How i wasted too long finding a concentration inequality for sums of geometric variables. 2011.

[19] Rafael Cano, Carmen Sordo, and José M Gutiérrez. Applications of bayesian networks in meteorology. In *Advances in Bayesian networks*. 2004.

[20] Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *Automata, Languages, and Programming*, 2014.

[21] Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? *Theory of Computing*, 2020.

[22] Clément L Canonne. A short note on learning discrete distributions. *arXiv preprint arXiv:2002.11457*, 2020.

[23] Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 2015.

[24] Clément L Canonne, Xi Chen, Gautam Kamath, Amit Levi, and Erik Waingarten. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021.

[25] Clément L. Canonne, Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Testing bayesian networks. *IEEE Transactions on Information Theory*, 2020.

[26] Sourav Chakraborty and Kuldeep S. Meel. On testing of uniform samplers. In *AAAI*, 2019.

[27] Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*, 2016.

[28] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A Scalable and Nearly Uniform Generator of SAT Witnesses. In *Proc. of CAV*, 2013.

[29] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. Distribution-aware sampling and weighted model counting for SAT. In *Proc. of AAAI*, 2014.

[30] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. On parallel scalable uniform sat witness generation. In *Proc. of TACAS*, 2015. doi: 10.1007/978-3-662-46681-0_25.

[31] Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *Proc. of IJCAI*, 2015.

[32] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014.

[33] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. On random sampling over joins. *ACM SIGMOD Record*, 1999.

[34] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 2008.

[35] Xi Chen and Cassandra Marcussen. *Uniformity Testing over Hypergrids with Subcube Conditioning*.

[36] Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Learning and testing junta distributions with sub cube conditioning. In *Proceedings of Thirty Fourth Conference on Learning Theory*, Proceedings of Machine Learning Research. PMLR, 2021.

[37] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. 2020.

[38] Karine Chubarian and György Turán. Interpretability of bayesian network classifiers: Obdd approximation and polynomial threshold functions. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2020.

[39] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *ACM SIGGRAPH Papers*. 2004.

[40] Elliott Crigger and Christopher Khoury. Making policy on augmented intelligence in health care. *AMA journal of ethics*, 2019.

[41] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 2001.

[42] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 2003.

[43] Adnan Darwiche and Jinbo Huang. Testing equivalence probabilistically. In *Technical Report*, 2002.

[44] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 2002.

[45] Michael E Donohue. A replacement for justitia's scales?: Machine learning's role in sentencing. *Harvard Journal of Law & Technology*, 2019.

[46] Rafael Dutra, Kevin Laeufer, Jonathan Bachrach, and Koushik Sen. Efficient sampling of sat solutions for testing. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, 2018.

[47] Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic. Testing probabilistic programming systems. In *Proc. of Joint Meeting on ESE and FSE*, 2018.

[48] Martin Dyer and Alan Frieze. Computing the volume of convex bodies: a case where randomness provably helps. *Probabilistic combinatorics and its applications*, 1991.

[49] Stefano Ermon, Carla P. Gomes, and Bart Selman. Uniform solution sampling using a constraint solver as an oracle. In *UAI*, 2012.

[50] Stefano Ermon, Carla P Gomes, Ashish Sabharwal, and Bart Selman. Embed and project: Discrete sampling with universal hashing. In *Proc. of NIPS*, 2013.

[51] Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Faster algorithms for testing under conditional sampling. In *Conference on Learning Theory*. PMLR, 2015.

[52] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Studies in Complexity and Cryptography*, 2011.

[53] Priyanka Golia, Mate Soos, Sourav Chakraborty, and Kuldeep S. Meel. Designing samplers is easy: The boon of testers. In *Proceedings of Formal Methods in Computer-Aided Design (FMCAD)*, 8 2021.

[54] Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Near-uniform sampling of combinatorial spaces using XOR constraints. In *Proc. of NIPS*, 2007.

[55] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014.

[56] Rahul Gupta, Shubham Sharma, Subhajit Roy, and Kuldeep S Meel. Waps: Weighted and projected sampling. In *TACAS*, 2019.

[57] Zhipeng Hu Rongsheng Zhang Minlie Huang Haozhe Ji, Pei Ke. Tailoring language generation models under total variation distance. In *The Eleventh International Conference on Learning Representations*, 2023.

[58] Mark Jerrum. Mathematical foundations of the markov chain monte carlo method. In *Probabilistic methods for algorithmic discrete mathematics*. 1998.

[59] Mark R. Jerrum and Alistair Sinclair. The Markov Chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, 1996.

[60] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical computer science*, 1986.

[61] Gautam Kamath and Christos Tzamos. Anaconda: A non-adaptive conditional sampling algorithm for distribution testing. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019.

[62] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. of STOC*, 1980.

[63] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 1983.

[64] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[65] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012. ISBN 1601986289.

[66] Pierre-Simon Laplace. Mémoire sur les probabilités. *Mémoires de l'Académie Royale des sciences de Paris*, 1781.

[67] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014.

[68] Neal Madras. Lectures on Monte Carlo Methods, Fields Institute Monographs 16. *American Mathematical Society*, 2002.

[69] Kuldeep S. Meel. Model counting and uniform sampling instances, 2020. URL https://zenodo.org/record/3793090.

[70] Kuldeep S. Meel, Moshe Y. Vardi, Supratik Chakraborty, Daniel J Fremont, Sanjit A Seshia, Dror Fried, Alexander Ivrii, and Sharad Malik. Constrained sampling and counting: Universal hashing meets sat solving. In *AAAI Workshop: Beyond NP 2016*, 2016.

[71] Kuldeep S. Meel𝔗, Yash Pote 𝔗, and Sourav Chakraborty. On testing of samplers. In *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, 12 2020.

[72] Kathleen L Mosier and Linda J Skitka. Human decision makers and automated decision aids: Made for each other? In *Automation and human performance*. 2018.

[73] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[74] Shyam Narayanan. On tolerant distribution testing in the conditional sampling model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021.

[75] Krzysztof Onak and Xiaorui Sun. Probability–revealing samples. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018.

[76] Agnieszka Oniśko, Marek J Druzdzel, and Hanna Wasyluk. Extension of the hepar ii model to multiple-disorder diagnosis. In *Intelligent Information Systems*. 2000.

[77] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 2008.

[78] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

[79] Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *ICML*, 2020.

[80] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *UAI*, 2020.

[81] Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008.

[82] Arthur R Pope and David G Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 2000.

[83] Yash Pote and Kuldeep S Meel. On scalable testing of samplers. In *Advances in Neural Information Processing Systems*, 2022.

[84] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[85] Yasir Rahmatallah, Frank Emmert-Streib, and Galina Glazko. Gene sets net correlations analysis (gsnca): a multivariate differential coexpression test for gene sets. *Bioinformatics*, 2014.

[86] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 1996.

[87] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Toward verified artificial intelligence. *Commun. ACM*, June 2022.

[88] Mate Soos, Stephan Gocht, and Kuldeep S Meel. Tinted, detached, and lazy cnf-xor solving and its applications to counting and sampling. In *International Conference on Computer Aided Verification*, 2020.

[89] Nicolas Städler and Sach Mukherjee. Multivariate gene-set testing based on graphical models. *Biostatistics*, 2015.

[90] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 1976.

[91] Gregory Valiant and Paul Valiant. A clt and tight lower bounds for estimating entropy. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2010.

[92] Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011.

[93] Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011.

[94] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 2017.

[95] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 2008.

[96] Weiwei Yin, Swetha Garimalla, Alberto Moreno, Mary R Galinski, and Mark P Styczynski. A tree-like bayesian structure learning algorithm for small-sample datasets from complex biological model systems. *BMC systems biology*, 2015.

[97] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random sampling over joins revisited. In *Proceedings of the 2018 International Conference on Management of Data*, 2018.

Appendix A

Extended table of results

A.1 From Chapter 3

A.1.1 Comparing sample complexity.

"A"("R") represent Barbarik2 returning Accept(Reject). "DNS" is used against those instances on which the indicated sampler Did Not Sample. "-" indicates that Barbarik2 timed out on that particular instance on the indicated sampler. Note that "DNS" is different from "-" as "DNS" indicates the failure of the underlying sampler to sample the initial set of samples, while "-" indicates the failure of Barbarik2 to finish within the timeout period. The timeout was set to 50,000 seconds for wSTS and wQuicksampler, while for wUnigen it was 24 hours.

Table A.1: The Extended Table

		Barbarik2		
Benchmark	$tilt$ (maxSamp)	wUnigen (samples)	wSTS (samples)	wQuicksampler (samples)
107.sk_3_90	1 (2e+05)	DNS	R (5146)	R (6009)
tableBasedAddition.sk	1 (2e+05)	DNS	R (6009)	R (24534)
55.sk_3_46	1 (2e+05)	DNS	R (8911)	R (4354)
111.sk_2_36	1 (2e+05)	DNS	R (23543)	R (5150)

continued ...

Benchmark	$tilt$ (maxSamp)	Barbarik2 wUnigen (samples)	Barbarik2 wSTS (samples)	Barbarik2 wQuicksampler (samples)
17.sk_3_45	1 (2e+05)	DNS	R (1e+05)	R (4677)
80.sk_2_48	1 (2e+05)	DNS	R (4284)	R (4627)
27.sk_3_32	1 (2e+05)	A (1e+05)	R (25329)	R (6009)
70.sk_3_40	1 (2e+05)	DNS	R (10402)	R (17704)
32.sk_4_38	1 (2e+05)	A (1e+05)	R (18081)	R (14682)
84.sk_4_77	1 (2e+05)	DNS	R (5146)	R (4354)
53.sk_4_32	1 (2e+05)	A (1e+05)	R (35618)	R (6009)
s35932_3_2	3 (6e+05)	DNS	TO	R (11756)
s35932_7_4	3 (6e+05)	DNS	TO	R (11756)
s832a_3_2	3 (6e+05)	A (1e+05)	R (8708)	R (54138)
109.sk_4_36	8 (3e+06)	DNS	R (26218)	R (6009)
77.sk_3_44	11 (5e+06)	DNS	R (47582)	R (47907)
s35932_15_7	12 (6e+06)	DNS	TO	R (4354)
s832a_7_4	15 (8e+06)	A (1e+05)	R (4393)	R (13350)
51.sk_4_38	18 (1e+07)	A (78661)	R (4284)	R (4627)
29.sk_3_45	26 (2e+07)	DNS	R (4284)	R (55989)
81.sk_5_51	27 (3e+07)	DNS	R (28409)	A (2e+05)

continued ...

Benchmark	$tilt$ (maxSamp)	Barbarik2		
		wUnigen (samples)	wSTS (samples)	wQuicksampler (samples)
s349_3_2	28 (3e+07)	A (1e+05)	A (1e+05)	R (22854)
s298_3_2	32 (3e+07)	A (1e+05)	R (80883)	R (26491)
s820a_3_2	37 (5e+07)	A (96212)	R (87997)	A (2e+05)
s298_15_7	44 (6e+07)	A (1e+05)	R (42520)	R (53107)
63.sk_3_64	58 (1e+08)	DNS	R (4393)	R (4677)
s820a_15_7	79 (2e+08)	A (84310)	R (2e+05)	R (16714)
s1488_15_7	110 (4e+08)	A (86152)	R (17168)	R (7341)
s1488_3_2	132 (6e+08)	A (89686)	A (89236)	R (7341)
s382_15_7	138 (6e+08)	A (92159)	R (2e+05)	R (6009)
UserServiceImpl.sk_8_32	140 (6e+08)	A (1e+05)	R (1e+05)	R (4393)
20.sk_1_51	144 (7e+08)	DNS	R (30895)	R (5146)
s820a_7_4	167 (9e+08)	A (95566)	A (1e+05)	R (6009)
s832a_15_7	194 (1e+09)	A (96984)	R (9434)	R (13350)
s1488_7_4	206 (1e+09)	A (1e+05)	R (4677)	R (4627)
s344_15_7	218 (2e+09)	A (90183)	R (94481)	R (4354)
LoginService2.sk_23_36	232 (2e+09)	A (1e+05)	R (38044)	R (13350)
s420_new1_15_7	265 (2e+09)	DNS	R (19224)	A (3e+05)

continued ...

Benchmark	$tilt$ (maxSamp)	Barbarik2 wUnigen (samples)	Barbarik2 wSTS (samples)	Barbarik2 wQuicksampler (samples)
s349_15_7	412 (5e+09)	A (99215)	R (28400)	R (14682)
s444_15_7	501 (8e+09)	A (1e+05)	A (1e+05)	R (26627)
s349_7_4	603 (1e+10)	A (75555)	R (4284)	R (5150)
s444_7_4	644 (1e+10)	DNS	R (4393)	R (4354)
s420_new1_7_4	982 (3e+10)	A (1e+05)	R (4354)	R (18473)
s298_7_4	986 (3e+10)	A (83681)	R (8638)	R (6009)
s420_new1_3_2	1226 (5e+10)	DNS	A (1e+05)	R (5150)
s382_7_4	1283 (5e+10)	A (92307)	R (26491)	R (7341)
s420_3_2	1552 (8e+10)	A (1e+05)	R (14756)	R (48983)
s1238a_7_4	1856 (1e+11)	A (95095)	R (5150)	R (7341)
s1238a_3_2	1965 (1e+11)	A (1e+05)	R (28848)	R (4627)
s444_3_2	2028 (1e+11)	A (1e+05)	R (2e+05)	R (9500)
s1238a_15_7	2317 (2e+11)	DNS	R (9020)	R (88233)
s420_new_15_7	2317 (2e+11)	A (99198)	R (1e+05)	R (4393)
30.sk_5_76	2453 (2e+11)	DNS	R (5216)	R (4677)
s344_7_4	2607 (2e+11)	A (1e+05)	R (14170)	R (16818)
s344_3_2	3300 (3e+11)	A (1e+05)	R (59952)	R (5150)

continued ...

Benchmark	$tilt$ (maxSamp)	Barbarik2		
		wUnigen (samples)	wSTS (samples)	wQuicksampler (samples)
s420_new_7_4	3549 (4e+11)	A (82312)	A (96659)	R (49955)
s953a_7_4	8984 (3e+12)	DNS	A (2e+05)	R (4627)
s953a_15_7	10596 (4e+12)	DNS	R (11734)	R (59735)
10.sk_1_46	15268 (7e+12)	DNS	R (35179)	R (1e+05)
s420_new_3_2	17449 (1e+13)	A (1e+05)	R (44937)	R (5150)
19.sk_3_48	18253 (1e+13)	DNS	R (59014)	R (4627)
s953a_3_2	20860 (1e+13)	DNS	R (51161)	R (1e+05)
s641_3_2	1e+06 (5e+16)	DNS	R (14454)	R (4627)
ProjectService3.sk_12_55	5e+06 (7e+17)	DNS	R (9020)	R (4393)
71.sk_3_65	1e+07 (3e+18)	DNS	R (1e+05)	R (4284)
s838_7_4	1e+07 (5e+18)	DNS	R (4393)	R (4284)
s838_15_7	3e+07 (3e+19)	DNS	R (5150)	R (4393)
s713_3_2	6e+07 (1e+20)	DNS	R (56386)	R (5827)
s713_7_4	6e+07 (1e+20)	DNS	R (5827)	R (37419)
s641_7_4	9e+07 (3e+20)	DNS	R (8747)	A (1e+06)
s838_3_2	2e+08 (1e+21)	DNS	R (9504)	R (4627)
54.sk_12_97	4e+11 (6e+27)	DNS	R (14012)	R (4627)

A.1.2 Comparing the runtime performance

In each of the following tables we compare the runtime of Barbarik2 against the runtime of the baseline approach. The runtime of Barbarik2 on Reject instances depends on which iteration the tester terminated on. The runtime of the baseline is extrapolated from the expected number of samples and the average sampling rate of the sampler. To do this we use the ℓ_1-testing algorithm given in [7]. In the current context, the algorithm assumes black box sample access to a uniform sampler over the models of a Boolean formula φ, and the sampler under test, and requires $O(\#\varphi^{2/3}(\eta - \varepsilon)^{-8/3}\log(\#\varphi/\delta))$ samples, where $\#\varphi$ is the model count, (ε, η) are the closeness and farness parameters, and δ is the confidence parameter.

A.1.2.1 For wSTS

Benchmark	Baseline	Barbarik2(s)	Speedup
s349_7_4	16457	5	3428.58
s420_new1_7_4	5.4E+6	6	8.6E+5
s298_7_4	705	8	94.02
s444_7_4	1.1E+7	8	1.3E+6
s832a_7_4	3725	10	372.53
s1488_7_4	184	12	15.16
s344_7_4	24751	15	1683.77
s420_3_2	2.2E+6	17	1.3E+5
s1238a_7_4	1.4E+6	20	66538.64
s832a_3_2	2149	22	98.60
s832a_15_7	15121	24	622.29
s838_15_7	2.9E+13	27	1.1E+12
s349_15_7	16457	28	587.76
s838_7_4	3.7E+13	29	1.3E+12
s382_7_4	14915	32	469.03
s298_15_7	384	32	12.09
s420_new1_15_7	4.1E+6	33	1.3E+5
27.sk_3_32	79531	34	2346.06

Benchmark	Baseline	Barbarik2(s)	Speedup
s1238a_15_7	1.8E+6	37	49906.05
111.sk_2_36	2.9E+8	42	6.8E+6
51.sk_4_38	2.0E+6	44	45904.52
80.sk_2_48	6.0E+7	46	1.3E+6
s1488_15_7	128	48	2.67
s953a_15_7	1.1E+9	49	2.2E+7
s344_3_2	15750	51	309.45
s298_3_2	229	52	4.42
s838_3_2	2.7E+13	57	4.8E+11
s420_new_3_2	2.9E+6	65	44288.35
84.sk_4_77	3.4E+13	68	5.0E+11
s641_3_2	4.1E+10	70	5.9E+8
55.sk_3_46	2.0E+7	70	2.9E+5
s349_3_2	30563	73	416.96
107.sk_3_90	1.7E+15	86	1.9E+13
s1238a_3_2	2.2E+6	87	25824.41
s344_15_7	24751	91	271.10
32.sk_4_38	5.8E+5	94	6228.23
10.sk_1_46	6.5E+7	112	5.8E+5
29.sk_3_45	2.2E+8	150	1.5E+6
s420_new_7_4	4.1E+6	152	27272.30
s1488_3_2	52	163	0.32
s953a_3_2	6.4E+8	165	3.9E+6
s420_new_15_7	4.5E+6	186	24014.34
70.sk_3_40	2.9E+6	201	14544.89
s444_15_7	13470	202	66.82
s420_new1_3_2	2.6E+6	211	12084.36
s820a_3_2	2189	221	9.91
s444_3_2	11186	247	45.22
s713_3_2	8.8E+10	255	3.5E+8

Benchmark	Baseline	Barbarik2(s)	Speedup
109.sk_4_36	6.6E+5	269	2459.36
s820a_7_4	4240	277	15.33
63.sk_3_64	5.8E+11	282	2.1E+9
s641_7_4	8.2E+10	311	2.6E+8
53.sk_4_32	55060	313	176.08
s382_15_7	33182	343	96.86
s820a_15_7	4154	370	11.23
ProjectService3.sk_12_55	1.3E+10	458	2.9E+7
s35932_3_2	3.6E+2	TO	-
s35932_7_4	3.6E+2	TO	-
s35932_15_7	3.6E+2	TO	-
s953a_7_4	5.7E+8	689	8.3E+5
UserServiceImpl.sk_8_32	479	720	0.67
30.sk_5_76	7.0E+14	1116	6.2E+11
77.sk_3_44	5.3E+6	1687	3156.66
tableBasedAddition.sk_240	3.8E+14	1832	2.1E+11
81.sk_5_51	5.0E+9	2099	2.4E+6
LoginService2.sk_23_36	12951	2368	5.47
20.sk_1_51	1.1E+10	2568	4.1E+6
19.sk_3_48	3.1E+8	2760	1.1E+5
17.sk_3_45	4.5E+7	3016	14948.13
71.sk_3_65	4.7E+12	4365	1.1E+9
54.sk_12_97	2.7E+18	4688	5.8E+14

A.1.2.2 For wQuicksampler

Benchmark	Baseline	Barbarik2(s)	Speedup
s344_3_2	24751.45	3	8534.98
s344_15_7	24751.45	4	7071.84
s349_7_4	28212.36	4	7624.96
s298_7_4	512.82	4	119.26
s420_new1_3_2	5.1E+6	4	1.2E+6
s420_new_3_2	2.2E+6	4	5.1E+5
s420_new_15_7	3.5E+6	4	7.8E+5
s382_7_4	12429.39	5	2589.46
s444_7_4	51980.83	5	10192.32
s820a_7_4	2283.19	5	430.79
s1488_7_4	128.07	6	20.99
s444_3_2	8700.57	6	1359.46
s838_7_4	1.3E+13	7	1.8E+12
27.sk_3_32	48942.42	7	6797.56
s1238a_3_2	1.6E+6	7	2.2E+5
s953a_7_4	6.6E+8	8	8.8E+7
s1488_3_2	65.65	8	8.31
s838_3_2	1.9E+13	8	2.4E+12
s1488_15_7	60.56	9	6.80
s349_15_7	35265.44	9	3833.20
s344_7_4	22501.32	9	2393.76
s349_3_2	14106.18	10	1424.87
55.sk_3_46	4.5E+7	10	4.3E+6
s1238a_7_4	1.1E+6	11	97431.59
s298_3_2	534.49	11	46.89
s832a_7_4	4139.28	12	344.94
111.sk_2_36	5.2E+5	12	41613.34
s838_15_7	2.6E+13	12	2.1E+12
s420_new1_7_4	2.2E+6	13	1.7E+5

Benchmark	Baseline	Barbarik2(s)	Speedup
s832a_15_7	13861.52	14	1011.79
UserServiceImpl.sk_8_32	326.81	14	23.68
s382_15_7	27149.56	15	1859.56
53.sk_4_32	91767.04	16	5595.55
s820a_15_7	5665.59	17	335.24
84.sk_4_77	2.1E+13	18	1.2E+12
51.sk_4_38	1.8E+6	19	91363.08
s444_15_7	14817.06	19	763.77
109.sk_4_36	6.6E+5	20	33425.00
107.sk_3_90	1.6E+15	21	7.4E+13
71.sk_3_65	1.3E+12	27	5.0E+10
s641_3_2	2.8E+10	28	1.0E+9
s298_15_7	1153.85	30	38.98
32.sk_4_38	1.2E+6	34	36689.91
s420_3_2	4.5E+6	34	1.3E+5
s420_new_7_4	3.5E+6	36	96896.05
80.sk_2_48	2.1E+8	37	5.7E+6
s832a_3_2	2149.58	45	47.66
19.sk_3_48	4.5E+8	50	9.0E+6
63.sk_3_64	2.1E+11	51	4.0E+9
17.sk_3_45	8.3E+7	55	1.5E+6
s713_3_2	9.4E+10	56	1.7E+9
s953a_15_7	6.7E+8	79	8.5E+6
20.sk_1_51	4.0E+9	82	4.8E+7
70.sk_3_40	4.3E+6	101	42475.10
s1238a_15_7	1.0E+6	107	9614.31
10.sk_1_46	7.1E+7	128	5.5E+5
s953a_3_2	3.4E+8	132	2.6E+6
s820a_3_2	1167.90	137	8.54
30.sk_5_76	3.0E+14	210	1.4E+12

Benchmark	Baseline	Barbarik2(s)	Speedup
ProjectService3.sk_12_55	6.4E+9	219	2.9E+7
LoginService2.sk_23_36	12692.30	229	55.52
s420_new1_15_7	3.2E+6	232	13726.91
77.sk_3_44	1.2E+7	409	30125.88
29.sk_3_45	1.3E+8	658	2.0E+5
54.sk_12_97	4.0E+17	690	5.8E+14
s641_7_4	6.8E+10	1117	6.1E+7
s35932_15_7	1.4E+356	1182	1.2E+353
tableBasedAddition.sk_240	3.0E+13	1430	2.1E+10
s35932_7_4	1.2E+356	2227	5.5E+352
s35932_3_2	1.1E+356	2346	4.5E+352
81.sk_5_51	2.0E+9	2461	8.3E+5

A.1.2.3 For wUnigen

Benchmark	Baseline	Barbarik2(s)	Speedup
s1488_3_2	229.78	6648	0.03
s298_7_4	7564.11	10758	0.70
s1488_15_7	643.45	11493	0.06
s298_15_7	2948.72	12325	0.24
s349_7_4	1.8E+06	12858	136.40
s820a_15_7	48724.11	14070	3.46
s344_15_7	3.8E+05	14074	27.18
s1488_7_4	853.78	15049	0.06
s820a_7_4	42728.33	16124	2.65
s349_15_7	3.9E+05	17690	21.80
s382_7_4	9.7E+05	21785	44.45
s349_3_2	3.0E+05	22395	13.54
s832a_15_7	5.6E+05	23036	24.45
s420_new_7_4	4.0E+09	24092	1.7E+5
s344_7_4	1.7E+06	26423	64.55
51.sk_4_38	2.7E+09	26612	1.0E+5
s820a_3_2	2.3E+05	27408	8.47
s298_3_2	2061.62	30262	0.07
s344_3_2	5.0E+05	32378	15.29
s1238a_7_4	1.5E+09	33689	45408.69
s832a_7_4	76990.55	34315	2.24
s382_15_7	1.0E+07	39024	263.98
s1238a_3_2	7.1E+08	40406	17575.38
s420_new_15_7	4.9E+09	40725	1.2E+5
27.sk_3_32	7.4E+06	41997	176.26
s832a_3_2	74844.43	42696	1.75
UserServiceImpl.sk_8_32	21547.88	45090	0.48
32.sk_4_38	4.9E+08	45126	10872.88
s420_new1_7_4	2.8E+08	48911	5639.38

Benchmark	Baseline	Barbarik2(s)	Speedup
s444_3_2	1.9E+06	55017	34.61
LoginService2.sk_23_36	1.3E+06	56229	22.38
s420_3_2	2.3E+09	68048	33247.50
53.sk_4_32	2.2E+07	70590	312.87
s420_new_3_2	1.2E+10	75284	1.6E+5

A.2 From Chapter 4

A.2.1 Comparing Barbarik2 and Barbarik3

Performance of Barbarik3. We experiment with 87 benchmarks, and out of the 87 benchmarks. In the table 'TO' represents that either the tester timed out or asked for more than 10^8 samples. The value of the parameter for closeness is $\varepsilon = 0.05$, for farness is $\eta = 0.9$ and for confidence is $\delta = 0.2$. Here R denotes Reject, and A denotes Accept.

		Barbarik2			Barbarik3		
Benchmark	Dimensions	Result	# of samples	Time(in s)	Result	# of samples	Time(in s)
SetTest	21	R	2817	170	R	58000	2290
s27_15_7	7	R	4789	0.99	R	30000	6.14
s27_7_4	7	R	4789	1.06	R	30000	6.43
polynomial.sk	25	R	4789	8.41	R	66000	95.0
Pollard.sk_1_10	10	R	7606	168	R	36000	525
s298_3_2	17	R	57431	50.75	R	50000	61.53
s27_3_2	7	R	62220	10.75	R	30000	6.49
s27_new_15_7	7	R	128264	19.04	R	30000	11.79
s526a_3_2	24	R	848148	1373	R	64000	191
s444_3_2	24	R	848148	1161	R	64000	142

s27_new_3_2	7	R	905579	138	R	30000	7.16
s510_15_7	25	R	12708989	18844	R	66000	206
s1488_15_7	14	R	12708989	38070	R	44000	198
s298_7_4	17	R	12708989	10186	R	50000	63.58
s27_new_7_4	7	A	23997012	3558	R	30000	7.24
s298_15_7	17	R	38126967	36140	R	50000	72.77
s349_7_4	24	TO	-	0.28	R	64000	130
110.sk_3_88	88	TO	-	2.98	R	190000	5082
s344_3_2	24	TO	-	0.33	R	64000	127
s526_3_2	24	TO	-	0.38	R	64000	169
53.sk_4_32	32	TO	-	0.32	R	80000	224
s420_7_4	34	TO	-	0.31	R	83000	297
10.sk_1_46	46	TO	-	0.4	R	107000	521
17.sk_3_45	45	TO	-	0.9	R	105000	801
s349_15_7	24	TO	-	0.51	R	64000	161
s820a_7_4	23	TO	-	0.28	R	62000	221
s832a_15_7	23	TO	-	0.43	R	62000	281
80.sk_2_48	48	TO	-	0.58	R	111000	656
s344_15_7	24	TO	-	0.87	R	64000	145
81.sk_5_51	51	TO	-	6.71	R	117000	13505
s420_3_2	34	TO	-	0.25	R	83000	275
s420_new1_15_7	34	TO	-	0.39	R	83000	335
UserService	32	TO	-	0.7	R	80000	706
111.sk_2_36	36	TO	-	0.33	R	87000	257
s349_3_2	24	TO	-	0.3	R	64000	120
s953a_7_4	45	TO	-	0.71	R	105000	783
s444_7_4	24	TO	-	1.9	R	64000	153
77.sk_3_44	44	TO	-	0.82	R	103000	862
51.sk_4_38	38	TO	-	1.04	R	91000	1689
109.sk_4_36	36	TO	-	0.69	R	87000	843
s832a_7_4	23	TO	-	0.3	R	62000	237
s526a_15_7	24	TO	-	86.82	R	64000	291

s420_new_7_4	34	TO	-	0.27	R	83000	276
s382_3_2	24	TO	-	0.24	R	64000	142
s641_3_2	54	TO	-	19.96	R	123000	951
s420_new_3_2	34	TO	-	0.26	R	83000	286
LoginService2	36	TO	-	18.86	R	87000	6639
s832a_3_2	23	TO	-	0.34	R	62000	233
s420_new_15_7	34	TO	-	0.29	R	83000	324
s420_new1_3_2	34	TO	-	0.26	R	83000	277
s838_3_2	66	TO	-	0.7	R	147000	1640
70.sk_3_40	40	TO	-	0.4	R	95000	450
s820a_15_7	23	TO	-	0.39	R	62000	216
29.sk_3_45	45	TO	-	3.29	R	105000	6514
19.sk_3_48	48	TO	-	3.51	R	111000	2259
57.sk_4_64	64	TO	-	0.98	R	143000	1501
s444_15_7	24	TO	-	0.48	R	64000	169
s1238a_3_2	32	TO	-	0.92	R	80000	493
s526_7_4	24	TO	-	46.45	R	64000	216
s382_7_4	24	TO	-	0.31	R	64000	138
s1238a_7_4	32	TO	-	1.1	R	80000	537
7.sk_4_50	50	TO	-	0.68	R	115000	834
55.sk_3_46	46	TO	-	0.44	R	107000	512
s713_7_4	54	TO	-	198	R	123000	1288
s420_new1_7_4	34	TO	-	0.35	R	83000	290
s641_7_4	54	TO	-	240	R	123000	1300
s1196a_15_7	32	TO	-	2.29	R	80000	613
ProjectService3	55	TO	-	184	R	125000	5557
s1196a_3_2	32	TO	-	0.91	R	80000	490
s1238a_15_7	32	TO	-	2.39	R	80000	664
s526_15_7	24	TO	-	122	R	64000	318
s820a_3_2	23	TO	-	0.32	R	62000	188
27.sk_3_32	32	TO	-	0.24	R	80000	196
s510_3_2	25	TO	-	0.28	R	66000	176

s1196a_7_4	32	TO	-	1.15	R	80000	546
s344_7_4	24	TO	-	0.32	R	64000	130
s713_3_2	54	TO	-	44.5	R	123000	1027
s953a_3_2	45	TO	-	0.65	R	105000	731
s526a_7_4	24	TO	-	44.16	R	64000	224
s420_15_7	34	TO	-	0.39	R	83000	346
s953a_15_7	45	TO	-	0.95	R	105000	912
s838_7_4	66	TO	-	0.81	R	147000	1552
56.sk_6_38	38	TO	-	0.52	R	91000	526
32.sk_4_38	38	TO	-	0.35	R	91000	358
s382_15_7	24	TO	-	8.83	R	64000	190
s1488_3_2	14	TO	-	0.42	R	44000	154
63.sk_3_64	64	TO	-	3.33	R	143000	9191

A.3 From Chapter 5

The timeout for all our experiments was set to 7200 seconds.

A.3.1 Synthetic PCs

In the following table, the first column indicates the benchmark, the second and the third indicate the closeness parameter ε and η used in the test. The fourth column indicates actual d_{TV} distance between the two benchmark PCs . The fifth column indicates the test outcome and the sixth represents the expected outcome. 'A' represents Accept and 'R' represents Reject and 'A/R' represents that both 'A' and 'R' are acceptable outputs.

Table A.6: Extended Table of Results

Benchmark	ε	η	Actual d_{TV}	Result	Expected Result
14_4	0.9	0.99	0.773	A	A
17_2	0.75	0.99	0.998	R	R

14_2	0.9	0.99	0.764	A	A
18_3	0.75	0.96	0.930	R	A/R
17_4	0.75	0.99	0.941	R	A/R
18_3	0.8	0.99	0.930	R	A/R
17_1	0.8	0.96	0.874	R	A/R
17_0	0.85	0.94	0.968	A	A/R
14_2	0.85	0.94	0.764	A	A
14_4	0.85	0.94	0.773	A	A
15_4	0.9	0.94	0.941	R	R
16_2	0.9	0.99	0.987	A	A/R
14_0	0.75	0.96	0.771	A	A/R
16_3	0.8	0.9	0.879	A	A/R
17_2	0.75	0.96	0.998	R	R
15_0	0.8	0.9	0.984	R	R
18_0	0.75	0.94	0.994	R	R
17_4	0.75	0.96	0.941	R	A/R
18_4	0.75	0.99	0.907	R	A/R
18_2	0.75	0.99	0.918	R	A/R
14_1	0.75	0.99	0.740	A	A
16_1	0.85	0.9	0.918	R	R
17_0	0.85	0.96	0.968	A	A/R
15_4	0.85	0.94	0.941	R	R
17_0	0.8	0.94	0.968	A	A/R
17_0	0.85	0.99	0.968	A	A/R
14_0	0.9	0.94	0.771	A	A

16_4	0.75	0.96	0.833	A	A/R
15_1	0.85	0.96	0.927	R	A/R
14_4	0.9	0.96	0.773	A	A
14_3	0.8	0.96	0.852	A	A/R
14_2	0.9	0.96	0.764	A	A
16_4	0.75	0.9	0.833	A	A/R
17_3	0.9	0.94	0.914	A	A/R
15_1	0.85	0.9	0.927	R	R
16_4	0.9	0.96	0.833	A	A
14_3	0.8	0.94	0.852	A	A/R
16_1	0.8	0.99	0.918	R	A/R
16_2	0.85	0.96	0.987	A	A/R
15_3	0.75	0.99	0.804	R	A/R
14_3	0.75	0.94	0.852	A	A/R
16_4	0.85	0.96	0.833	A	A
18_0	0.8	0.9	0.994	R	R
18_4	0.8	0.94	0.907	R	A/R
18_4	0.85	0.99	0.907	A	A/R
18_0	0.9	0.99	0.994	R	R
15_1	0.9	0.99	0.927	R	A/R
14_3	0.85	0.96	0.852	A	A/R
16_2	0.75	0.94	0.987	R	R
15_0	0.9	0.96	0.984	R	R
18_4	0.8	0.96	0.907	R	A/R
17_0	0.75	0.9	0.968	R	R

18_1	0.9	0.96	0.993	R	R
18_0	0.9	0.96	0.994	R	R
17_3	0.8	0.99	0.914	A	A/R
18_3	0.85	0.9	0.930	R	R
17_2	0.85	0.94	0.998	R	R
15_1	0.75	0.94	0.927	R	A/R
14_3	0.75	0.9	0.852	R	A/R
15_3	0.8	0.99	0.804	R	A/R
17_3	0.85	0.94	0.914	A	A/R
14_3	0.8	0.9	0.852	R	A/R
17_3	0.75	0.99	0.914	R	A/R
14_3	0.9	0.99	0.852	A	A
17_0	0.75	0.94	0.968	R	R
18_2	0.8	0.99	0.918	R	A/R
17_0	0.8	0.96	0.968	A	A/R
17_1	0.85	0.94	0.874	A	A/R
16_3	0.8	0.94	0.879	A	A/R
14_1	0.85	0.94	0.740	A	A
16_3	0.85	0.99	0.879	A	A/R
18_0	0.85	0.96	0.994	R	R
15_3	0.9	0.94	0.804	A	A
16_4	0.8	0.9	0.833	A	A/R
14_1	0.75	0.96	0.740	A	A
16_2	0.8	0.9	0.987	R	R
17_1	0.75	0.96	0.874	R	A/R

15_1	0.8	0.9	0.927	R	R
15_4	0.8	0.96	0.941	R	A/R
18_2	0.9	0.94	0.918	R	A/R
18_4	0.85	0.94	0.907	R	A/R
18_4	0.85	0.96	0.907	R	A/R
16_3	0.9	0.99	0.879	A	A
14_0	0.75	0.99	0.771	A	A/R
16_0	0.85	0.9	0.954	R	R
14_4	0.85	0.99	0.773	A	A
16_1	0.8	0.9	0.918	R	R
17_1	0.8	0.94	0.874	R	A/R
17_1	0.85	0.99	0.874	A	A/R
16_4	0.9	0.99	0.833	A	A
14_1	0.9	0.94	0.740	A	A
17_0	0.8	0.9	0.968	R	R
14_2	0.75	0.96	0.764	A	A/R
15_0	0.85	0.96	0.984	R	R
14_0	0.8	0.96	0.771	A	A
14_4	0.75	0.96	0.773	A	A/R
16_3	0.75	0.9	0.879	R	A/R
17_2	0.9	0.94	0.998	R	R
15_2	0.85	0.9	0.905	A	A/R
14_4	0.8	0.94	0.773	A	A
14_2	0.8	0.94	0.764	A	A
16_0	0.8	0.99	0.954	R	A/R

17_2	0.85	0.9	0.998	R	R
16_3	0.85	0.96	0.879	A	A/R
14_2	0.75	0.94	0.764	A	A/R
18_1	0.8	0.94	0.993	R	R
18_1	0.85	0.99	0.993	R	R
18_1	0.9	0.99	0.993	R	R
16_4	0.75	0.99	0.833	A	A/R
15_0	0.9	0.99	0.984	R	A/R
15_1	0.9	0.96	0.927	R	A/R
16_0	0.9	0.94	0.954	A	A/R
17_1	0.75	0.9	0.874	R	A/R
15_0	0.8	0.94	0.984	R	R
17_4	0.8	0.99	0.941	R	A/R
18_2	0.85	0.9	0.918	R	R
14_2	0.75	0.9	0.764	A	A/R
15_0	0.8	0.99	0.984	R	A/R
14_2	0.8	0.9	0.764	A	A
14_4	0.8	0.9	0.773	A	A
14_1	0.85	0.9	0.740	A	A
17_0	0.75	0.99	0.968	A	A/R
14_0	0.85	0.9	0.771	A	A
17_1	0.75	0.94	0.874	R	A/R
18_1	0.85	0.94	0.993	R	R
18_1	0.8	0.99	0.993	R	R
18_1	0.75	0.9	0.993	R	R

17_3	0.8	0.96	0.914	R	A/R
18_3	0.9	0.96	0.930	R	A/R
16_2	0.8	0.94	0.987	A	A/R
14_0	0.85	0.94	0.771	A	A
16_2	0.85	0.99	0.987	A	A/R
16_4	0.8	0.94	0.833	A	A/R
18_1	0.85	0.96	0.993	R	R
16_4	0.85	0.99	0.833	A	A
15_2	0.9	0.94	0.905	A	A/R
15_0	0.75	0.9	0.984	R	R
16_0	0.9	0.99	0.954	A	A/R
15_4	0.8	0.9	0.941	R	R
17_0	0.75	0.96	0.968	R	R
15_3	0.8	0.96	0.804	R	A/R
18_3	0.9	0.94	0.930	R	A/R
18_3	0.85	0.94	0.930	R	A/R
16_0	0.8	0.96	0.954	A	A/R
17_4	0.85	0.96	0.941	R	A/R
14_3	0.75	0.99	0.852	A	A/R
17_2	0.85	0.96	0.998	R	R
17_4	0.8	0.94	0.941	R	R
16_2	0.8	0.96	0.987	A	A/R
17_4	0.85	0.99	0.941	R	A/R
16_3	0.8	0.96	0.879	A	A/R
17_1	0.8	0.9	0.874	R	A/R

16_2	0.75	0.96	0.987	R	R
15_3	0.85	0.96	0.804	A	A
17_2	0.8	0.94	0.998	R	R
14_1	0.8	0.96	0.740	A	A
14_0	0.85	0.99	0.771	A	A
16_2	0.75	0.9	0.987	R	R
15_2	0.85	0.94	0.905	A	A/R
14_3	0.85	0.99	0.852	A	A/R
15_3	0.85	0.9	0.804	R	A/R
14_2	0.85	0.99	0.764	A	A
16_3	0.8	0.99	0.879	A	A/R
17_3	0.85	0.9	0.914	A	A/R
16_0	0.85	0.96	0.954	A	A/R
14_1	0.75	0.94	0.740	A	A
18_4	0.8	0.9	0.907	R	R
18_0	0.8	0.94	0.994	R	R
14_3	0.8	0.99	0.852	A	A/R
18_0	0.85	0.99	0.994	R	R
18_2	0.9	0.99	0.918	R	A/R
16_3	0.75	0.99	0.879	A	A/R
15_3	0.9	0.99	0.804	A	A
16_4	0.75	0.94	0.833	A	A/R
15_2	0.9	0.96	0.905	A	A/R
16_1	0.9	0.94	0.918	R	A/R
18_2	0.8	0.96	0.918	R	A/R

17_2	0.9	0.96	0.998	R	R
15_1	0.8	0.94	0.927	R	A/R
17_3	0.9	0.99	0.914	A	A/R
15_1	0.75	0.96	0.927	R	A/R
15_1	0.8	0.99	0.927	R	A/R
14_1	0.8	0.9	0.740	A	A
17_4	0.75	0.94	0.941	R	R
18_0	0.75	0.96	0.994	R	R
17_1	0.75	0.99	0.874	R	A/R
17_2	0.75	0.94	0.998	R	R
18_0	0.8	0.99	0.994	R	R
18_0	0.75	0.9	0.994	R	R
17_2	0.8	0.96	0.998	R	R
18_2	0.9	0.96	0.918	R	A/R
16_1	0.8	0.94	0.918	R	A/R
16_1	0.85	0.99	0.918	R	A/R
18_2	0.85	0.96	0.918	R	A/R
15_1	0.9	0.94	0.927	R	A/R
15_1	0.75	0.9	0.927	R	R
16_1	0.9	0.99	0.918	R	A/R
14_3	0.75	0.96	0.852	A	A/R
18_3	0.75	0.94	0.930	R	A/R
15_2	0.8	0.96	0.905	A	A/R
18_0	0.9	0.94	0.994	R	R
18_1	0.75	0.99	0.993	R	R

18_2	0.85	0.94	0.918	R	A/R
17_3	0.85	0.96	0.914	A	A/R
14_2	0.75	0.99	0.764	A	A/R
15_3	0.85	0.94	0.804	A	A
17_2	0.8	0.9	0.998	R	R
16_3	0.75	0.96	0.879	A	A/R
14_2	0.85	0.9	0.764	A	A
15_2	0.85	0.96	0.905	A	A/R
14_1	0.9	0.96	0.740	A	A
16_1	0.75	0.9	0.918	R	R
17_4	0.9	0.94	0.941	R	R
15_4	0.85	0.9	0.941	R	R
16_4	0.8	0.99	0.833	A	A/R
15_0	0.75	0.99	0.984	R	A/R
15_0	0.85	0.9	0.984	R	R
16_2	0.8	0.99	0.987	A	A/R
17_0	0.85	0.9	0.968	A	A/R
16_1	0.85	0.96	0.918	R	A/R
14_0	0.75	0.94	0.771	A	A/R
16_2	0.9	0.96	0.987	A	A/R
18_3	0.8	0.9	0.930	R	R
18_3	0.8	0.94	0.930	R	A/R
14_2	0.8	0.99	0.764	A	A
16_1	0.9	0.96	0.918	R	A/R
18_3	0.85	0.99	0.930	R	A/R

14_4	0.8	0.99	0.773	A	A
16_0	0.9	0.96	0.954	A	A/R
18_3	0.9	0.99	0.930	R	A/R
16_2	0.75	0.99	0.987	A	A/R
14_0	0.85	0.96	0.771	A	A
15_2	0.9	0.99	0.905	A	A/R
16_1	0.75	0.94	0.918	R	A/R
16_4	0.9	0.94	0.833	A	A
15_3	0.9	0.96	0.804	A	A
16_2	0.9	0.94	0.987	A	A/R
18_3	0.8	0.96	0.930	R	A/R
17_3	0.9	0.96	0.914	A	A/R
15_2	0.8	0.94	0.905	A	A/R
17_0	0.8	0.99	0.968	A	A/R
15_2	0.75	0.94	0.905	R	A/R
18_4	0.85	0.9	0.907	R	R
15_4	0.8	0.99	0.941	R	A/R
15_4	0.75	0.94	0.941	R	R
14_4	0.75	0.9	0.773	A	A/R
14_0	0.8	0.9	0.771	A	A
14_0	0.9	0.99	0.771	A	A
18_1	0.75	0.96	0.993	R	R
17_3	0.75	0.94	0.914	R	A/R
18_3	0.75	0.9	0.930	R	R
17_4	0.85	0.94	0.941	R	R

16_0	0.8	0.94	0.954	A	A/R
15_0	0.85	0.99	0.984	R	A/R
16_0	0.85	0.99	0.954	A	A/R
15_4	0.75	0.9	0.941	R	R
15_1	0.85	0.99	0.927	R	A/R
18_3	0.85	0.96	0.930	R	A/R
15_0	0.9	0.94	0.984	R	R
15_2	0.75	0.9	0.905	R	R
15_2	0.85	0.99	0.905	A	A/R
15_2	0.8	0.9	0.905	A	A/R
15_3	0.85	0.99	0.804	A	A
18_2	0.75	0.94	0.918	R	A/R
18_4	0.75	0.94	0.907	R	A/R
15_1	0.8	0.96	0.927	R	A/R
18_1	0.9	0.94	0.993	R	R
18_0	0.75	0.99	0.994	R	R
14_3	0.85	0.9	0.852	A	A/R
16_3	0.85	0.9	0.879	A	A/R
16_1	0.8	0.96	0.918	R	A/R
14_1	0.85	0.99	0.740	A	A
15_0	0.85	0.94	0.984	R	R
17_2	0.85	0.99	0.998	R	R
14_2	0.9	0.94	0.764	A	A
14_4	0.9	0.94	0.773	A	A
17_3	0.8	0.9	0.914	R	R

16_0	0.75	0.96	0.954	R	A/R
14_0	0.9	0.96	0.771	A	A
17_1	0.9	0.94	0.874	A	A
16_0	0.75	0.9	0.954	R	R
14_1	0.8	0.94	0.740	A	A
15_1	0.75	0.99	0.927	R	A/R
17_1	0.85	0.9	0.874	R	A/R
18_2	0.8	0.9	0.918	R	R
18_2	0.8	0.94	0.918	R	A/R
14_1	0.8	0.99	0.740	A	A
18_2	0.85	0.99	0.918	R	A/R
18_4	0.9	0.99	0.907	A	A/R
16_1	0.75	0.99	0.918	R	A/R
14_1	0.85	0.96	0.740	A	A
16_0	0.75	0.94	0.954	R	R
15_4	0.9	0.96	0.941	R	A/R
17_2	0.75	0.9	0.998	R	R
16_3	0.9	0.94	0.879	A	A
18_0	0.8	0.96	0.994	R	R
17_4	0.75	0.9	0.941	R	R
15_3	0.8	0.94	0.804	R	A/R
17_1	0.8	0.99	0.874	R	A/R
18_1	0.85	0.9	0.993	R	R
15_3	0.75	0.94	0.804	R	A/R
14_1	0.75	0.9	0.740	A	A

17_1	0.9	0.99	0.874	A	A
15_3	0.75	0.96	0.804	R	A/R
18_4	0.75	0.96	0.907	R	A/R
14_1	0.9	0.99	0.740	A	A
18_2	0.75	0.96	0.918	R	A/R
18_4	0.8	0.99	0.907	R	A/R
18_4	0.75	0.9	0.907	R	R
18_2	0.75	0.9	0.918	R	R
17_4	0.8	0.96	0.941	R	A/R
14_3	0.85	0.94	0.852	A	A/R
18_4	0.9	0.96	0.907	A	A/R
17_3	0.75	0.9	0.914	R	R
17_4	0.9	0.96	0.941	R	A/R
15_3	0.75	0.9	0.804	R	A/R
16_0	0.8	0.9	0.954	R	R
17_3	0.75	0.96	0.914	R	A/R
15_3	0.8	0.9	0.804	R	A/R
18_1	0.75	0.94	0.993	R	R
16_1	0.85	0.94	0.918	R	A/R
16_3	0.85	0.94	0.879	A	A/R
18_4	0.9	0.94	0.907	A	A/R
15_0	0.8	0.96	0.984	R	R
16_0	0.85	0.94	0.954	A	A/R
14_4	0.85	0.9	0.773	A	A
18_3	0.75	0.99	0.930	R	A/R

17_0	0.9	0.96	0.968	A	A/R
18_0	0.85	0.94	0.994	R	R
17_1	0.9	0.96	0.874	A	A
16_4	0.8	0.96	0.833	A	A/R
16_2	0.85	0.9	0.987	A	A/R
17_1	0.85	0.96	0.874	A	A/R
14_4	0.75	0.99	0.773	A	A/R
16_4	0.85	0.9	0.833	A	A
17_3	0.8	0.94	0.914	R	A/R
15_1	0.85	0.94	0.927	R	A/R
17_3	0.85	0.99	0.914	A	A/R
14_3	0.9	0.94	0.852	A	A
17_4	0.8	0.9	0.941	R	R
16_1	0.75	0.96	0.918	R	A/R
15_4	0.85	0.96	0.941	R	A/R
14_2	0.8	0.96	0.764	A	A
14_3	0.9	0.96	0.852	A	A
17_0	0.9	0.94	0.968	A	A/R
14_4	0.8	0.96	0.773	A	A
16_3	0.9	0.96	0.879	A	A
15_4	0.75	0.99	0.941	R	A/R
14_0	0.8	0.94	0.771	A	A
17_4	0.85	0.9	0.941	R	R
15_2	0.75	0.99	0.905	A	A/R
14_4	0.75	0.94	0.773	A	A/R

17_4	0.9	0.99	0.941	A	A/R
18_1	0.8	0.9	0.993	R	R
15_4	0.85	0.99	0.941	R	A/R
14_0	0.8	0.99	0.771	A	A
17_2	0.9	0.99	0.998	R	R
14_0	0.75	0.9	0.771	A	A/R
14_4	0.85	0.96	0.773	A	A
16_0	0.75	0.99	0.954	R	A/R
14_2	0.85	0.96	0.764	A	A
15_4	0.9	0.99	0.941	A	A/R
16_3	0.75	0.94	0.879	R	A/R
16_4	0.85	0.94	0.833	A	A
18_1	0.8	0.96	0.993	R	R
15_0	0.75	0.96	0.984	R	R
16_2	0.85	0.94	0.987	A	A/R
15_4	0.8	0.94	0.941	R	R
17_2	0.8	0.99	0.998	R	R
18_0	0.85	0.9	0.994	R	R
15_0	0.75	0.94	0.984	R	R
15_4	0.75	0.96	0.941	R	A/R
17_0	0.9	0.99	0.968	A	A/R
15_2	0.8	0.99	0.905	A	A/R
15_2	0.75	0.96	0.905	R	A/R

A.3.2 Real-world PCs

In the following table, the first column indicates the benchmark, the second indicates the time required for the test, and the third column indicates the test outcome. 'A' represents Accept and 'R' represents Reject.

Table A.7: Extended Table of Results for Real-world PCs

Benchmark	Teq(s)	Result
or-70-10-8-UC-10_0	23.2	A
or-70-10-8-UC-10_1	22.72	R
or-70-10-8-UC-10_2	22.92	R
or-70-10-8-UC-10_3	22.87	R
or-70-10-8-UC-10_4	22.78	R
or-70-10-8-UC-10_5	23.06	R
or-70-10-8-UC-10_6	22.99	R
or-70-10-8-UC-10_7	22.93	R
or-70-10-8-UC-10_8	22.82	R
or-70-10-8-UC-10_9	22.82	R
s641_15_7_0	33.66	A
s641_15_7_1	33.4	R
s641_15_7_2	33.45	R
s641_15_7_3	33.32	R
s641_15_7_4	33.51	R
s641_15_7_5	33.21	R
s641_15_7_6	33.46	R
s641_15_7_7	33.23	R
s641_15_7_8	33.61	R
s641_15_7_9	33.51	R
or-50-5-4_0	414.17	A
or-50-5-4_1	414.84	R
or-50-5-4_2	410.16	R
or-50-5-4_3	414.15	R
or-50-5-4_4	410.07	R

or-50-5-4_5	412.27	R
or-50-5-4_6	414.77	R
or-50-5-4_7	415.19	R
or-50-5-4_8	416.84	R
or-50-5-4_9	408.59	R
ProjectService3.sk_12_55_0	356.58	A
ProjectService3.sk_12_55_1	353.77	R
ProjectService3.sk_12_55_2	355.93	R
ProjectService3.sk_12_55_3	356.11	R
ProjectService3.sk_12_55_4	356.15	A
ProjectService3.sk_12_55_5	355.64	R
ProjectService3.sk_12_55_6	357.89	R
ProjectService3.sk_12_55_7	356.69	R
ProjectService3.sk_12_55_8	353.36	R
ProjectService3.sk_12_55_9	356.14	R
s713_15_7_0	24.56	R
s713_15_7_1	24.68	R
s713_15_7_2	24.28	R
s713_15_7_3	24.47	R
s713_15_7_4	24.65	R
s713_15_7_5	24.32	R
s713_15_7_6	24.4	R
s713_15_7_7	24.39	R
s713_15_7_8	24.86	A
s713_15_7_9	24.41	R
or-100-10-2-UC-30_0	31.11	R
or-100-10-2-UC-30_1	31.16	R
or-100-10-2-UC-30_2	31.04	R
or-100-10-2-UC-30_3	31.13	R
or-100-10-2-UC-30_4	31.14	R
or-100-10-2-UC-30_5	31.04	A
or-100-10-2-UC-30_6	31.03	R

or-100-10-2-UC-30_7	31.13	R
or-100-10-2-UC-30_8	31.17	R
or-100-10-2-UC-30_9	31.0	R
s1423a_3_2_0	153.8	R
s1423a_3_2_1	152.37	R
s1423a_3_2_2	152.01	R
s1423a_3_2_3	150.96	R
s1423a_3_2_4	152.64	R
s1423a_3_2_5	153.13	A
s1423a_3_2_6	151.52	R
s1423a_3_2_7	152.53	R
s1423a_3_2_8	152.4	R
s1423a_3_2_9	152.81	R
s1423a_7_4_0	104.28	R
s1423a_7_4_1	103.4	R
s1423a_7_4_2	103.82	R
s1423a_7_4_3	104.18	R
s1423a_7_4_4	103.95	R
s1423a_7_4_5	103.59	R
s1423a_7_4_6	104.31	R
s1423a_7_4_7	104.93	R
s1423a_7_4_8	104.93	A
s1423a_7_4_9	103.51	R
or-50-5-10_0	282.09	R
or-50-5-10_1	282.49	R
or-50-5-10_2	279.63	R
or-50-5-10_3	281.8	R
or-50-5-10_4	280.69	R
or-50-5-10_5	279.91	R
or-50-5-10_6	283.05	A
or-50-5-10_7	282.69	R
or-50-5-10_8	279.65	R

or-50-5-10_9	282.97	R
or-60-20-6-UC-20_0	359.89	R
or-60-20-6-UC-20_1	362.3	R
or-60-20-6-UC-20_2	363.1	R
or-60-20-6-UC-20_3	363.11	R
or-60-20-6-UC-20_4	362.76	R
or-60-20-6-UC-20_5	358.76	R
or-60-20-6-UC-20_6	363.32	A
or-60-20-6-UC-20_7	358.41	R
or-60-20-6-UC-20_8	358.8	R
or-60-20-6-UC-20_9	362.8	R