

Testing Probabilistic Circuits ^{*†}

Yash Pote  Kuldeep S. Meel
School of Computing, National University of Singapore

Abstract

Probabilistic circuits (PCs) are a powerful modeling framework for representing tractable probability distributions over combinatorial spaces. In machine learning and probabilistic programming, one is often interested in understanding whether the distributions learned using PCs are close to the desired distribution. Thus, given two probabilistic circuits, a fundamental problem of interest is to determine whether their distributions are close to each other.

The primary contribution of this paper is a closeness test for PCs with respect to the total variation distance metric. Our algorithm utilizes two common PC queries, counting and sampling. In particular, we provide a poly-time probabilistic algorithm to check the closeness of two PCs, when the PCs support tractable approximate counting and sampling. We demonstrate the practical efficiency of our algorithmic framework via a detailed experimental evaluation of a prototype implementation against a set of 475 PC benchmarks. We find that our test correctly decides the closeness of all 475 PCs within 3600 seconds.


1 Introduction

Probabilistic modeling is at the heart of modern computer science, with applications ranging from image recognition and image generation [29, 30] to weather forecasting [3]. Probabilistic models have a multitude of representations, such as probabilistic circuits (PCs) [9], graphical models [19], generative networks [16], and determinantal point processes [20]. 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 [2, 25]. In this work, we will focus on PCs that are fragments of the Negation Normal Form (NNF), specifically DNNFs, d-DNNFs, SDNNFs, and PIs [13]. We refer to the survey by Choi et al. [9] for more details regarding PCs.

Given two distributions P and Q , a fundamental problem is to determine whether they are close. Closeness between distributions is frequently quantified using the total variation (TV) distance, $d_{TV}(P, Q) = (1/2)\|P - Q\|_1$, where $\|\cdot\|$ is the ℓ_1 norm [21, 6]. Thus, stated formally, closeness testing is the problem of deciding whether $d_{TV}(P, Q) \leq \varepsilon$ or $d_{TV}(P, Q) \geq \eta$ for $0 \leq \varepsilon < \eta \leq 1$. Determining the closeness of models has applications in AI planning [13], bioinformatics [31, 33, 35] and probabilistic program verification [15, 23].

Equivalence testing is a special case of closeness testing, where one tests if $d_{TV}(P, Q) = 0$. Darwiche and Huang [13] 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

*The accompanying tool, available open source, can be found at <https://github.com/meelgroup/teq>. The Appendix is available in the accompanying supplementary material.

†The authors decided to forgo the old convention of alphabetical ordering of authors in favor of a randomized ordering, denoted by . The publicly verifiable record of the randomization is available at <https://www.aeaweb.org/journals/policies/random-author-order/search> with confirmation code: icoxrj0sNB2L. For citation of the work, authors request that the citation guidelines by AEA for random author ordering be followed.

encode non-identical distributions that are nonetheless close enough for practical purposes. Such situations may arise due to the use of approximate PC compilation [10] and sampling-based learning of PCs [26, 27]. As a concrete example, consider PCs that are learned via approximate methods such as stochastic gradient descent [27]. 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 work, 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 $(\varepsilon, \eta, \delta)$, for closeness(ε), fairness(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 at most $O((\eta - \varepsilon)^{-2} \log(\delta^{-1}))$ calls to the sampler and exactly 2 calls to the counter.

Teq builds on a general distance estimation technique of Canonne and Rubinfeld [4] 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 the technique presented in [4] to allow for approximate samples and counts. Furthermore, we implement and test Teq on a dataset of publicly available PCs arising from applications in circuit testing. 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 [32]. 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 [11] and Arenas et al. [1]. We show that the SDNNF approximate counting algorithm of Arenas et al. [1] can be extended to log-linear SDNNFs using chain formulas [8]. 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 paper is organized in the following way. We define the notation and discuss related work in Section 2. We then present the main contribution of the paper, the closeness test Teq, and the associated proof of correctness in Section 3. We present our experimental findings in Section 4, and then discuss the complexity landscape of closeness testing in Section 5. We conclude the paper and discuss some open problems in Section 6. Due to space constraints, we defer some proofs to the supplementary Section A.

2 Background

Let $\varphi : \{0, 1\}^n \rightarrow \{0, 1\}$ 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.

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 $\Pi_{i+1}^P = \text{co-NP}^{\Pi_i^P}$ and $\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P}$ [34]. 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 hardness results, including the ones we present in the paper.

2.1 Probability distributions

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

In this paper, 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 [24]. Log-linear models are represented as literal-weighted functions, defined as:

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

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

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

Probabilistic circuits: A probabilistic circuit is a satisfiable circuit φ along with a weight function \mathbf{w} . φ and \mathbf{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: $P(\varphi, \mathbf{w})(\sigma) = \begin{cases} 0 & \varphi(\sigma) = 0 \\ \mathbf{w}(\sigma)/\mathbf{w}(\varphi) & \varphi(\sigma) = 1 \end{cases}$

In this paper, we study circuits that are fragments of the Negation Normal Form (NNF). 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 [14] and the paper [28].

The TV distance of two probability distributions $P(\varphi_1, \mathbf{w}_1)$ and $P(\varphi_2, \mathbf{w}_2)$ over $\{0, 1\}^n$ is defined as: $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) = \frac{1}{2} \sum_{\sigma \in \{0, 1\}^n} |P(\varphi_1, \mathbf{w}_1)(\sigma) - P(\varphi_2, \mathbf{w}_2)(\sigma)|$.

$P(\varphi_1, \mathbf{w}_1)$ and $P(\varphi_2, \mathbf{w}_2)$ are said to be (1) equivalent if $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) = 0$, (2) ε -close if $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) \leq \varepsilon$, and (3) η -far if $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) \geq \eta$.

Our closeness testing algorithm `Teq`, assumes access to an approximate weighted counter $\text{Awct}(\alpha, \beta, \varphi, \mathbf{w})$, and an approximate weighted sampler $\text{Samp}(\alpha, \beta, \varphi, \mathbf{w})$. We define their behavior as follows:

Definition 2. $\text{Awct}(\alpha, \beta, \varphi, \mathbf{w})$ takes a circuit φ , a weight function \mathbf{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. \mathbf{w} such that

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

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

Definition 3. $\text{Samp}(\alpha, \beta, \varphi, \mathbf{w})$ takes a circuit φ , a weight function \mathbf{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 \mathbf{w} with probability $\geq 1 - \beta$ or (2) a symbol \perp indicating failure with probability $< \beta$. In other words, whenever Samp samples σ :

$$\frac{P(\varphi, \mathbf{w})(\sigma)}{1 + \alpha} \leq \Pr[\text{Samp}(\alpha, \beta, \varphi, \mathbf{w}) = \sigma] \leq (1 + \alpha)P(\varphi, \mathbf{w})(\sigma)$$

Tractable approximate sampling algorithms for PCs are known as Fully Polynomial Almost Uniform Samplers (FPAUS). The running time of an FPAUS for a single sample is given by $T(\alpha, \beta, \varphi) = \text{poly}(\alpha^{-1}, \log(\beta^{-1}), |\varphi|)$.

In the rest of the paper $[m]$ denotes the set $\{1, 2, \dots, m\}$, $\mathbb{1}(e)$ represents the indicator variable for event e , and $\mathbb{E}(v)$ represents the expectation of random variable v .

2.2 Related work

Closeness testing: Viewing circuit equivalence testing through the lens of distribution testing, we see that the d-DNNF equivalence test of Darwiche and Huang [13] can be interpreted as an equivalence test for uniform distribution on the satisfying assignments of d-DNNFs. This relationship between circuit equivalence testing and closeness testing lets us rule out the existence of distributional equivalence tests for all those circuits for which circuit equivalence is already known to be hard under complexity-theoretic assumptions. We will explore this further in Section 5.2.

Distribution testing: Discrete probability distributions are typically defined over an exponentially large number of points; hence a lot of recent algorithms research has focused on devising tests that require access to only a sublinear or even constant number of points in the distribution [5]. In this work, we work with distributions over $\{0, 1\}^n$, and thus we aim to devise algorithms with running time at most polynomial in n . Previous work in testing distributions over Boolean functions has focused on the setting where the distributions offer pair-conditional sampling access [7, 22]. Using pair-conditional sampling access, Meel \oplus et al. [22] were able to test distributions for closeness using $\tilde{O}(\text{tilt}(\varphi)^2/(\eta - 6\varepsilon)^3\eta)$ queries, where tilt is the ratio of the probabilities of the most and least probable element in the support.

3 Teq: a tractable algorithm for closeness testing

In this section, we present the main contribution of the paper: a closeness test for PCs, Teq. The pseudocode of Teq is given in Algorithm 1.

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

The algorithm Teq starts by computing constants γ and m . Then it queries the Awct routine with circuit φ_1 and weight function \mathbf{w}_1 to obtain a $\sqrt{1 + \gamma/4} - 1$ approximation of $\mathbf{w}_1(\varphi_1)$ with confidence at least $1 - \delta/8$. A similar query is made for φ_2 and \mathbf{w}_2 to obtain an approximate value for $\mathbf{w}_2(\varphi_2)$. 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 \mathbf{w}_1 and \mathbf{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 $P(\varphi_2, \mathbf{w}_2)$ to its probability in $P(\varphi_1, \mathbf{w}_1)$. Since σ_i was sampled from $P(\varphi_1, \mathbf{w}_1)$, its probability in $P(\varphi_1, \mathbf{w}_1)$ 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.

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

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

The following theorem states the running time of the algorithm,

Theorem 2. *Let $\gamma = \eta - \varepsilon$, then the time complexity of Teq is in $O\left(T_{\text{Awct}}(\gamma, \delta, \max(|\varphi_1|, |\varphi_2|)) + T_{\text{Samp}}(\gamma, \delta, \max(|\varphi_1|, |\varphi_2|)) \frac{\log(\delta^{-1})}{\gamma^2}\right)$. If the underlying*

Algorithm 1 $\text{Teq}(\varphi_1, \mathbf{w}_1, \varphi_2, \mathbf{w}_2, \varepsilon, \eta, \delta)$

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

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 $\max(|\varphi_1|, |\varphi_2|)$.

To improve readability, we use P_1 to refer to the distribution $P(\varphi_1, \mathbf{w}_1)$ and P_2 to refer to $P(\varphi_2, \mathbf{w}_2)$.

3.1 Proving the correctness of Teq

In this subsection, we present the theoretical analysis of Teq , and the proof of Theorem 1(A). We will defer the proofs of Theorem 1(B) and Theorem 2 to the supplementary Section A.4.2 and Section A.4.3, respectively.

For the purpose of the proof, we will first define events $\text{Pass}_1, \text{Pass}_2$ and Good . Events $\text{Pass}_1, \text{Pass}_2$ are defined w.r.t. the function calls $\text{Awct}(\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi_1, \mathbf{w}_1)$ and $\text{Awct}(\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi_2, \mathbf{w}_2)$, respectively (as on lines 4, 5 of Algorithm 1). Pass_1 and 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. $\frac{\mathbf{w}_1(\varphi_1)}{\sqrt{1 + \gamma/4}} \leq \text{Awct}(\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi_1, \mathbf{w}_1) \leq (\sqrt{1 + \gamma/4})\mathbf{w}_1(\varphi_1)$, and $\frac{\mathbf{w}_2(\varphi_2)}{\sqrt{1 + \gamma/4}} \leq \text{Awct}(\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi_2, \mathbf{w}_2) \leq (\sqrt{1 + \gamma/4})\mathbf{w}_2(\varphi_2)$. From the definition of Awct , we have $\Pr[\text{Pass}_1], \Pr[\text{Pass}_2] \geq 1 - \delta/8$.

Let Fail_i denote the event that Samp (Algorithm 1, line 7) returns the symbol \perp in the i th iteration of the loop. By the definition of Samp we know that $\forall_{i \in [m]} \Pr[\text{Fail}_i] < \delta/4m$.

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

$$\Pr[\text{Good}] = 1 - \Pr\left[\bigcup_{i \in [m]} \text{Fail}_i \cup \overline{\text{Pass}_1} \cup \overline{\text{Pass}_2}\right] \geq 1 - m \cdot \delta/4m - 2 \cdot \delta/8 = 1 - \delta/2 \quad (1)$$

We will now state a lemma, which we will prove in the supplementary Section A.4.

Lemma 1. $\text{Good} \rightarrow \left| r(\sigma) - \frac{P_2(\sigma)}{P_1(\sigma)} \right| \leq \gamma/4 \cdot \frac{P_2(\sigma)}{P_1(\sigma)}$

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

Lemma 2. Assuming the event Good , let $A = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}(r(\sigma) < 1) (1 - r(\sigma)) P_1(\sigma)$, 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 $\sum_x (P_1(x) - P_2(x)) = 0$, 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,

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

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: $S_1 = \{\sigma : \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right) = \mathbb{1}(r(\sigma) < 1)\}$; $S_2 = \{\sigma : \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right) > \mathbb{1}(r(\sigma) < 1)\}$; $S_3 = \{\sigma : \mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right) < \mathbb{1}(r(\sigma) < 1)\}$. 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 $\mathbb{1}\left(\frac{P_2(\sigma)}{P_1(\sigma)} < 1\right)$ takes value 1 and 0 for all elements in S_2 and S_3 , respectively.

Now we bound the magnitude of B ,

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

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

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

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.

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

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

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

Using our assumption of the event Good and Lemma 1, $|B| \leq \sum_{\sigma \in \{0,1\}^n} \gamma/4 \cdot P_1(\sigma) \leq \gamma/4$. 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 $A \geq \eta - \gamma/4$. \square

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 the definitions of the above terms and further details, please refer to the survey [9].

4 Evaluation

To evaluate the performance of Teq, we implemented a prototype in Python. The prototype uses WAPS³ [17] 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.

4.1 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.

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

Table 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 supplementary Section B. 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.

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 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 B.

4.2 Setting B - Real-world benchmarks

Dataset We conducted experiments on a range of publicly available benchmarks arising from sampling and counting tasks⁴. 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 synthesized new weight functions using the technique of *one variable perturbation*, described in the appendix Section B.1.

³<https://github.com/meelgroup/WAPS>

⁴<https://zenodo.org/record/3793090>

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 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 B. In the table ‘A’ represents Accept and ‘R’ represents Reject. The value of the closeness parameter is $\varepsilon = 0.01$ and the fairness parameter is $\eta = 0.2$.

Results We set the closeness parameter ε , fairness 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 1) 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 2. The first column indicates the benchmark’s name, 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 are presented in the supplementary Section B.

5 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 3. The results presented in the table can be separated into (1) hardness results, and (2) upper bounds. The hardness results, presented in Section 5.2, are largely derived from known complexity-theoretic results. The upper bounds, presented in Section 5.1, are derived from a combination of established results, our algorithm Teq and the exact equivalence test of Darwiche and Huang [13](presented in supplementary Section A.1 for completeness).

5.1 Upper bounds

In Table 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 [13] provided an equivalence test for d-DNNF s. From Theorem 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 [11], and a weighted sampler was provided by [17]. Arenas et al. [1] 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 3. *Given a SDNNF formula φ (with a v-tree T), and a weight function w , $\text{Samp}(\varphi, w)$ requires polynomial time in the size of φ .*

The proof is provided in the supplementary Section A.5.

	NNF	PI	DNNF	SDNNF	d-DNNF
NNF	<i>EC</i>				
PI	<i>EC</i>	<i>UU</i>			
DNNF	<i>EC</i>	<i>EU</i>	<i>EU</i>		
SDNNF	<i>EC</i>	<i>EU</i>	<i>EU</i>	<i>EC</i>	
d-DNNF	<i>EC</i>	<i>UU</i>	<i>EU</i>	<i>EC</i>	<i>EC</i>

Table 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.

5.2 Hardness

In Table 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 $DNF \subseteq SDNNF \subseteq DNNF$ [14].
- 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 1, 2 and 3. These propositions and their proofs can be found in the supplementary Section A.5.
- 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 [18].

The NP-hardness of deciding the equivalence of pairs of DNNFs and pairs of SDNNFs was first shown by Pipatsrisawat and Darwiche [28]. We recast their proofs in the language of distribution testing for the sake of completeness in the supplementary Section A.5.

6 Conclusion and future work

In this paper, we studied the problem of closeness testing of PCs. Before our work, poly-time algorithms were known only for the special case of equivalence testing of PCs; and, no poly-time closeness test was known for any PC. We provided the first such test, called Teq, that used ideas from the field of distribution testing to design a novel algorithm for testing the closeness of PCs. We then implemented a prototype for Teq, and tested it on publicly available benchmarks to determine the runtime performance. Experimental results demonstrate the effectiveness of Teq in practice.

We also characterized PCs with respect to the complexity of deciding equivalence and closeness. We combined known hardness results, reductions, and our proposed algorithm Teq to classify pairs of PCs according to closeness and equivalence testing complexity. Since the characterization is incomplete, as seen in Table 3, there are questions left open regarding the existence of tests for certain PCs, which we leave for future work.

Broader Impact

Recent advances in probabilistic modeling techniques have led to increased adoption of the said techniques in safety-critical domains, thus creating a need for appropriate verification and testing methodologies. This paper seeks to take a step in this direction and focuses on testing properties of probabilistic models likely to find use in safety-critical domains. Since our guarantees are probabilistic, practical adoption of such techniques still requires careful design to handle failures.

Acknowledgments and Disclosure of Funding

We are grateful to the anonymous reviewers of UAI 2021 and NeurIPS 2021 for their constructive feedback that greatly improved the paper. We would also like to thank Suwei Yang and Lawqueen Kanesh for their useful comments on the earlier drafts of the paper. This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004] and AI Singapore Programme [AISG-RP-2018-005], and NUS ODPRT Grant [R-252-000-685-13]. The computational work for this article was performed on resources of the National Supercomputing Centre, Singapore (<https://www.nsc.sg>).

References

- [1] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. When is Approximate Counting for Conjunctive Queries Tractable? In *STOC*, 2021.
- [2] Dominik Aronsky and Peter J Haug. Diagnosing community-acquired pneumonia with a bayesian network. In *Proceedings of the AMIA Symposium*, 1998.
- [3] Rafael Cano, Carmen Sordo, and José M Gutiérrez. Applications of bayesian networks in meteorology. In *Advances in Bayesian networks*. 2004.
- [4] Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *Automata, Languages, and Programming*, 2014.
- [5] Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? *Theory of Computing*, 2020.
- [6] Clément L. Canonne, Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Testing bayesian networks. *IEEE Transactions on Information Theory*, 2020.
- [7] Sourav Chakraborty and Kuldeep S. Meel. On testing of uniform samplers. In *AAAI*, 2019.
- [8] Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *IJCAI*, 2015.
- [9] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. 2020.
- [10] Karine Chubarian and György Turán. Interpretability of bayesian network classifiers: Obdd approximation and polynomial threshold functions. In *ISAIM*, 2020.
- [11] 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.
- [12] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 2003.
- [13] Adnan Darwiche and Jinbo Huang. Testing equivalence probabilistically. In *Technical Report*, 2002.
- [14] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 2002.
- [15] Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic. Testing probabilistic programming systems. In *Proc. of Joint Meeting on ESE and FSE*, 2018.
- [16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
- [17] Rahul Gupta, Shubham Sharma, Subhajit Roy, and Kuldeep S Meel. Waps: Weighted and projected sampling. In *TACAS*, 2019.
- [18] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *STOC*, 1980.
- [19] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [20] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.
- [21] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *NeurIPS*, 2018.
- [22] Kuldeep S. Meel[Ⓕ], Yash Pote[Ⓕ], and Sourav Chakraborty. On Testing of Samplers. In *NeurIPS*, 2020.

- [23] Andrzej S. Murawski and Joël Ouaknine. On probabilistic program equivalence and refinement. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, 2005.
- [24] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [25] Agnieszka Oniśko, Marek J Druzdzal, and Hanna Wasyluk. Extension of the hepar ii model to multiple-disorder diagnosis. In *Intelligent Information Systems*. 2000.
- [26] 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.
- [27] 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*. PMLR, 2020.
- [28] Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008.
- [29] Arthur R Pope and David G Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 2000.
- [30] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [31] Yasir Rahmatallah, Frank Emmert-Streib, and Galina Glazko. Gene sets net correlations analysis (gsnca): a multivariate differential coexpression test for gene sets. *Bioinformatics*, 2014.
- [32] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2), 1996.
- [33] Nicolas Städler and Sach Mukherjee. Multivariate gene-set testing based on graphical models. *Biostatistics*, 2015.
- [34] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 1976.
- [35] 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.

A Proofs omitted from the paper

A.1 A test for equivalence

For the sake of completeness we recast the d-DNNF circuit equivalence test of Darwiche and Huang [13] into an equivalence test for log-linear probability distributions.

Algorithm 2 $\text{Peq}(\varphi_1, \mathbf{w}_1, \varphi_2, \mathbf{w}_2, \delta)$

```

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

```

The algorithm: The pseudocode for Peq is shown in Algorithm 2. Peq takes as input two satisfiable circuits φ_1, φ_2 defined over n Boolean variables, a pair of weight functions $\mathbf{w}_1, \mathbf{w}_2$ and a tolerance parameter $\delta \in (0, 1)$. Recall that a circuit φ and a weight function \mathbf{w} together define the probability distribution $P(\varphi, \mathbf{w})$. Peq returns Accept with confidence 1 if the two probability distributions $P(\varphi_1, \mathbf{w}_1)$ and $P(\varphi_2, \mathbf{w}_2)$ are equivalent, i.e. $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) = 0$. If $d_{TV}(P(\varphi_1, \mathbf{w}_1), P(\varphi_2, \mathbf{w}_2)) > 0$, 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 2 (in Section A.2), Peq computes the values $\pi(\varphi_1, \mathbf{w}_1)(\theta)$ and $\pi(\varphi_2, \mathbf{w}_2)(\theta)$, where $\pi(\varphi, \mathbf{w})$ is the network polynomial [12]. $\pi(\varphi, \mathbf{w})$ defined as:

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

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 $P(\varphi_1, \mathbf{w}_1)$ and $P(\varphi_2, \mathbf{w}_2)$ are equivalent, the polynomials $\pi(\varphi_1, \mathbf{w}_1)$ and $\pi(\varphi_2, \mathbf{w}_2)$ are also equivalent, however when they are not equivalent, the polynomials disagree on at least $1 - \delta$ fraction of assignments from the set $[m]^n$.

We formally claim and prove the correctness of Peq in Lemma 4 in the Section A.2.

A.2 An analysis for Algorithm 2

In this section, we present the theoretical analysis of Algorithm 2 (Peq) and the proof of the following lemma.

Lemma 4. *Given two satisfiable probabilistic circuits φ_1, φ_2 and weight functions $\mathbf{w}_1, \mathbf{w}_2$, along with confidence parameter $\delta \in (0, 1)$.*

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

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

For the proof of Lemma 4(B) we will first define some notation, and then we show (in Lemma 5) 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 $\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma)$.

Definition 4. $\pi|_{x_i=1}(\varphi, \mathbf{w})$ is a polynomial over $n - 1$ variables, obtained by setting the variable x_i to 1. Similarly $\pi|_{x_i=0}(\varphi, \mathbf{w})$ is obtained by setting the variable x_i to 0, thus:

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

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

Proposition 1. *If $\pi(\varphi_1, \mathbf{w}_1) \not\equiv \pi(\varphi_2, \mathbf{w}_2)$ then for all x_i , at least one of the following must be true:*

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

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

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

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

To show that the polynomial $\pi(\varphi, \mathbf{w})$ can be computed in time polynomial in the size of the representation, we will adapt the procedure given by [13].

Proposition 2. *Let φ be a circuit over the set $X = \{x_1, \dots, x_n\}$ of n variables, that admits poly-time WMC. Let $\mathbf{w} : X \rightarrow \mathbb{Q}^+$ 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 .
- $S(\eta) = \begin{cases} 0, & \text{if } \eta \text{ is a leaf node false} \\ 1, & \text{if } \eta \text{ is a leaf node true} \\ \mathbf{w}(x)\theta(x), & \text{if } \eta \text{ is a leaf node } x \in X \\ (1 - \mathbf{w}(x))(1 - \theta(x)), & \text{if } \eta \text{ is a leaf node } \neg x, x \in X \end{cases}$
- $\pi(\varphi, \mathbf{w}) = S(\eta)/\mathbf{w}(\varphi)$, where η is the root node

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

Lemma 5. *For a random assignment $\sigma \sim [m]^n$, $\Pr[\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma) \mid P(\varphi_1, \mathbf{w}_1) \not\equiv P(\varphi_2, \mathbf{w}_2)] > 1 - \delta$*

Proof. For $n = 1$, σ is an assignment to a single variable x . The polynomial on the single variable x can be parameterised as $\pi(\varphi, \mathbf{w})(x) = \alpha x + (1 - \alpha)(1 - x)$ where parameter $\alpha = \frac{\mathbf{w}(x)}{\sum_{\theta \in R_\varphi} \mathbf{w}(\theta)}$.

Let polynomials $\pi(\varphi_1, \mathbf{w}_1), \pi(\varphi_2, \mathbf{w}_2)$ be parameterised with α_1, α_2 , respectively. Our assumption that $P(\varphi_1, \mathbf{w}_1) \not\equiv P(\varphi_2, \mathbf{w}_2)$ immediately leads to the fact that $\pi(\varphi_1, \mathbf{w}_1) \not\equiv \pi(\varphi_2, \mathbf{w}_2)$ 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,

$$\begin{aligned} \pi(\varphi_1, \mathbf{w}_1)(x) &= \pi(\varphi_2, \mathbf{w}_2)(x) \\ \alpha_1 x + (1 - \alpha_1)(1 - x) &= \alpha_2 x + (1 - \alpha_2)(1 - x) \\ 2(\alpha_1 - \alpha_2)x &= \alpha_1 - \alpha_2 \\ x &= 1/2 \end{aligned}$$

From the initial assumption we know that x can only take integer values, hence there are no inputs in the set $[m]$ for which $\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma)$. Thus, for $n = 1$, and any σ , $\Pr[\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma) \mid P(\varphi_1, \mathbf{w}_1) \not\equiv P(\varphi_2, \mathbf{w}_2)] = 0$

We now assume that the hypothesis holds for $n - 1$ variables. Consider polynomials $\pi(\varphi_1, \mathbf{w}_1) \not\equiv \pi(\varphi_2, \mathbf{w}_2)$ over n variables. From Prop 1 we know that at least one of the following holds:

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

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

$$\forall \sigma \in \Sigma, \pi|_{x_n=0}(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi|_{x_n=0}(\varphi_2, \mathbf{w}_2)(\sigma)$$

The set of assignments σ for which $\pi(\varphi_1, \mathbf{w}_1)(\sigma) = \pi(\varphi_2, \mathbf{w}_2)(\sigma)$ is given by,

$$\begin{aligned} & \pi(\varphi_1, \mathbf{w}_1)(\sigma) = \pi(\varphi_2, \mathbf{w}_2)(\sigma) \\ (1-x_n)\pi|_{x_n=0}(\varphi_1, \mathbf{w}_1)(\sigma) + x_n\pi|_{x_n=1}(\varphi_1, \mathbf{w}_1)(\sigma) &= (1-x_n)\pi|_{x_n=0}(\varphi_2, \mathbf{w}_2)(\sigma) + x_n\pi|_{x_n=1}(\varphi_2, \mathbf{w}_2)(\sigma) \\ x_n(\pi|_{x_n=1}(\varphi_1, \mathbf{w}_1)(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathbf{w}_1)(\sigma) - \pi|_{x_n=1}(\varphi_2, \mathbf{w}_2)(\sigma) + \pi|_{x_n=0}(\varphi_2, \mathbf{w}_2)(\sigma)) & \\ &= \pi|_{x_n=0}(\varphi_2, \mathbf{w}_2)(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathbf{w}_1)(\sigma) \end{aligned}$$

From the assumptions we know that there are at least $(m-1)^{n-1}$ assignments σ s.t. $\pi|_{x_n=0}(\varphi_2, \mathbf{w}_2)(\sigma) - \pi|_{x_n=0}(\varphi_1, \mathbf{w}_1)(\sigma) \neq 0$, 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 $\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma)$.

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

$$\begin{aligned} \Pr[\pi(\varphi_1, \mathbf{w}_1)(\sigma) \neq \pi(\varphi_2, \mathbf{w}_2)(\sigma) \mid P(\varphi_1, \mathbf{w}_1) \neq P(\varphi_2, \mathbf{w}_2)] &\geq \frac{(m-1)^n}{m^n} \geq \left(1 - \frac{\delta}{n}\right)^n \\ &> 1 - \delta \quad (\text{using } m \text{ from Algorithm 2}) \end{aligned}$$

□

A.3 Omitted proof from the analysis of Algorithm 1

In this subsection, we present the proof of Theorem 1(B), and Theorem 2. Recall that we use P_1 and P_2 to refer to $P(\varphi_1, \mathbf{w}_1)$ and $P(\varphi_2, \mathbf{w}_2)$, respectively.

A.4 Proof of Lemma 1

Proof. The quantity $r(\sigma)$ (line 10 from Algorithm 1) conditioned on the event $\overline{\text{Fail}}_i \subset \text{Good}$:

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

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

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

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,

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

A.4.1 Proof of Theorem 1(A)

Proof. We assume the event Good. Let σ_i be the sample returned by the sampler Samp in the i th 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 $\Gamma[i] = \mathbb{1}(r(\sigma_i) < 1)(1 - r(\sigma_i))$. The expectation of $\Gamma[i]$ is:

$$\mathbb{E}[\Gamma[i]] = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}(r(\sigma) < 1)(1 - r(\sigma)) \cdot \Pr[\text{Samp}(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathbf{w}_1) = \sigma] \quad (2)$$

According to definition 3, and our assumption of $\overline{\text{Fail}}_i \subset \text{Good}$, we know that with probability 1 $\Pr[\text{Samp}(\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathbf{w}_1) = \sigma] \leq (1 + \gamma/(4\eta - 2\gamma))P_1(\sigma)$. Thus we have,

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

Recall that in Lemma 2, we define $A = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}(r(\sigma) < 1)(1 - r(\sigma))P_1(\sigma)$. 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 2-1 to find a bound on the expectation,

$$\mathbb{E}[\Gamma[i]] \leq (1 + \gamma/(4\eta - 2\gamma))(\varepsilon + \gamma/4) \leq \varepsilon + \gamma/2$$

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:

$$\Pr \left[\text{Teq returns Accept} \mid \text{Good} \right] = 1 - \Pr \left[\sum_{i \in [m]} \Gamma[i] > m(\varepsilon + \gamma) \right] \geq 1 - 2e^{-\gamma^2 m/2} \geq 1 - \delta/2$$

The value for m is taken from line 2 of Algorithm 1. Using (1), we see that the probability of Teq returning Accept is: $\Pr[\text{Teq returns Accept}] \geq \Pr[\text{Teq returns Accept} \mid \text{Good}] \Pr[\text{Good}] = (1 - \delta/2)(1 - \delta/2) \geq 1 - \delta \quad \square$

A.4.2 Proof of Theorem 1(B)

Proof. First we assume the event Good. Then according to definition 3, we know that with probability 1 (since we assume event $\overline{\text{Fail}}_i \subset \text{Good}$)

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

Thus substituting into (2), we get

$$\mathbb{E}[\Gamma[i]] \geq \sum_{\sigma \in \{0,1\}^n} \mathbb{1}(r(\sigma) < 1) (1 - r(\sigma)) \frac{P_1(\sigma_i)}{1 + \gamma/(4\eta - 2\gamma)} \quad (3)$$

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

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

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

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

Hence, the probability that Algorithm 1 returns Reject is

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

□

A.4.3 Proof of Theorem 2

Proof. Teq makes two calls to Awct on line 4 and 5 of Algorithm 1. According to definition 2, the runtime of the Awct($\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi, \mathbf{w}$) query is $T(\sqrt{1 + \gamma/4} - 1, \delta/8, \varphi) = \text{poly}((\sqrt{1 + \gamma/4} - 1)^{-1}, \log(\delta^{-1}), |\varphi|)$.

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

Teq makes $m = \lceil \log(2/\delta)/2\gamma^2 \rceil$ calls to Samp on lines 7 of Algorithm 1. According to definition 3, the runtime of the Samp($\gamma/(4\eta - 2\gamma), \delta/4m, \varphi_1, \mathbf{w}_1$) query is $T(\gamma/(4\eta - 2\gamma), \delta/4m, |\varphi_1|) = \text{poly}((\gamma/(4\eta - 2\gamma))^{-1}, \log((\delta/4m)^{-1}), |\varphi_1|)$. First we see that $\frac{4\eta - 2\gamma}{\gamma} < \frac{4}{\gamma}$, thus the algorithm remains in $\text{poly}(\gamma^{-1})$. We then see that $\log(4m/\delta) = \log(4m) + \log(\delta^{-1})$. Since $\log(m) \in \text{poly}(\log(\gamma^{-1}), \log \log(\delta^{-1}))$, we know that Samp queries run in $O(\text{poly}(\gamma^{-1}, \log(\delta^{-1}), \max(|\varphi_1|, |\varphi_2|)))$.

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 $\max(|\varphi_1|, |\varphi_2|)$. □

A.5 Proofs omitted from Section 5

For the following proofs, we assume a uniform weight function.

Proposition 1. *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 $\{x_1, \dots, x_n\}$ 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 $d_{TV}(P(\hat{\varphi}, \mathbf{w}), P(\psi, \mathbf{w})) = 0$ if and only if $|R_\varphi| = 0$. Thus the existence of a poly-time randomised algorithm for deciding whether $d_{TV}(P(\hat{\varphi}, \mathbf{w}), P(\psi, \mathbf{w})) = 0$ would imply $\text{NP} \subseteq \text{RP}$ and hence $\text{NP}=\text{RP}$. \square

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

Proposition 3. *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 NP=RP.*

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

Using Corollary 6.3 from [18] we see that PH collapses as a result of either of the above implications.

From the set inclusions $\text{DNF} \subseteq \text{SDNNF} \subseteq \text{DNNF}$ and $\text{CNF} \subseteq \text{NNF}$, 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 3.

Lemma 6. *Given a SDNNF formula φ (with a v-tree T), and a weight function \mathbf{w} , $\text{Samp}(\varphi, \mathbf{w})$ 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 [8], the problem of approximate weighted sampling over SDNNF can be reduced to approximate uniform sampling. Given a SDNNF φ , and a weight function \mathbf{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 $w(x_i) \times 2^p = d$ 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 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 $\text{var}(\varphi)$. The formula φ_w defined above has the property that if $\varphi(\sigma) = 1$, then $|\{\sigma' | \varphi_w(\sigma') = 1 \wedge \sigma'_{\downarrow \text{var}(\varphi)} = \sigma\}| / |R_{\varphi_w}| = \mathbf{w}(\sigma)$. Thus a uniform distribution on R_{φ_w} , when projected on $\text{var}(\varphi)$ induces the weighted distribution $P(\varphi, \mathbf{w})$. This property allows weighted sampling and counting on φ with the help of a uniform sampler for the generated formula φ_w . \square

B Experimental evaluation

In this section we will first discuss the method for generating the synthetic dataset, and then we present the extended table of results.

B.1 One variable perturbation

Consider two weight functions w_1 and w_2 that differ only in the weight assigned to the literals v^0 and v^1 . Then, from the definition of d_{TV} :

$$d_{TV}(P(\varphi, w_1), P(\varphi, w_2)) = \frac{1}{2} \sum_{\sigma \in \{0,1\}^n} \left| \frac{w_1(\sigma)}{w_1(\varphi)} - \frac{w_2(\sigma)}{w_2(\varphi)} \right|$$

Let $S \subseteq \{0, 1\}^n$ be the set of assignments for which $\frac{w_1(\sigma)}{w_1(\varphi)} > \frac{w_2(\sigma)}{w_2(\varphi)}$. Thus,

$$d_{TV}(P(\varphi, w_1), P(\varphi, w_2)) = \sum_{\sigma \in S} \left(\frac{w_1(\sigma)}{w_1(\varphi)} - \frac{w_2(\sigma)}{w_2(\varphi)} \right)$$

Lets assume wlog that w_1 assigns a larger weight to v^1 than 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,

$$d_{TV}(P(\varphi, w_1), P(\varphi, w_2)) = \frac{w_1(\varphi \wedge v^1)}{w_1(\varphi)} - \frac{w_2(\varphi \wedge v^1)}{w_2(\varphi)}$$

We can rewrite $w_1(\varphi \wedge v^1) = w'_1(\varphi) \times w_1(v^1)$, where w'_1 is w_1 with the weight of v^1 set to 1. Using a similar transformation on $w_2(\varphi \wedge v^1)$ we get

$$d_{TV}(P(\varphi, w_1), P(\varphi, w_2)) = \frac{w'_1(\varphi) \times w_1(v^1)}{w_1(\varphi)} - \frac{w'_2(\varphi) \times w_2(v^1)}{w_2(\varphi)}$$

We know that $w'_1(\varphi) = w'_2(\varphi)$ as w_1 and w_2 differed only on the one variable v^1 .

$$d_{TV}(P(\varphi, w_1), P(\varphi, w_2)) = w'_1(\varphi) \times \left(\frac{w_1(v^1)}{w_1(\varphi)} - \frac{w_2(v^1)}{w_2(\varphi)} \right)$$

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 $d_{TV}(P(\varphi, w_1), P(\varphi, w_2))$ can be computed in poly-time.

B.2 Extended table of results

The timeout for all our experiments was set to 7200 seconds.

B.2.1 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 4: The Extended Table

Benchmark	ε	η	Actual d_{TV}	Result	Expected Result
14_1	0.75	0.99	0.740	A	A
14_1	0.85	0.94	0.740	A	A
14_1	0.75	0.96	0.740	A	A

14_1	0.9	0.94	0.740	A	A
14_1	0.85	0.9	0.740	A	A
14_1	0.8	0.96	0.740	A	A
14_1	0.75	0.94	0.740	A	A
14_1	0.8	0.9	0.740	A	A
14_1	0.9	0.96	0.740	A	A
14_1	0.85	0.99	0.740	A	A
14_1	0.8	0.94	0.740	A	A
14_1	0.8	0.99	0.740	A	A
14_1	0.85	0.96	0.740	A	A
14_1	0.75	0.9	0.740	A	A
14_1	0.9	0.99	0.740	A	A
14_2	0.9	0.99	0.764	A	A
14_2	0.85	0.94	0.764	A	A
14_2	0.9	0.96	0.764	A	A
14_2	0.85	0.99	0.764	A	A
14_2	0.75	0.96	0.764	A	A/R
14_2	0.8	0.94	0.764	A	A
14_2	0.75	0.94	0.764	A	A/R
14_2	0.75	0.9	0.764	A	A/R
14_2	0.8	0.9	0.764	A	A
14_2	0.75	0.99	0.764	A	A/R
14_2	0.85	0.9	0.764	A	A
14_2	0.8	0.99	0.764	A	A
14_2	0.9	0.94	0.764	A	A
14_2	0.8	0.96	0.764	A	A
14_2	0.85	0.96	0.764	A	A
14_0	0.75	0.96	0.771	A	A/R
14_0	0.9	0.94	0.771	A	A
14_0	0.75	0.99	0.771	A	A/R
14_0	0.8	0.96	0.771	A	A
14_0	0.85	0.94	0.771	A	A
14_0	0.85	0.9	0.771	A	A
14_0	0.75	0.94	0.771	A	A/R
14_0	0.85	0.96	0.771	A	A
14_0	0.8	0.9	0.771	A	A
14_0	0.9	0.99	0.771	A	A
14_0	0.9	0.96	0.771	A	A
14_0	0.85	0.99	0.771	A	A
14_0	0.8	0.94	0.771	A	A

14_0	0.8	0.99	0.771	A	A
14_0	0.75	0.9	0.771	A	A/R
14_4	0.9	0.99	0.773	A	A
14_4	0.85	0.94	0.773	A	A
14_4	0.9	0.96	0.773	A	A
14_4	0.85	0.99	0.773	A	A
14_4	0.75	0.96	0.773	A	A/R
14_4	0.8	0.94	0.773	A	A
14_4	0.8	0.9	0.773	A	A
14_4	0.8	0.99	0.773	A	A
14_4	0.75	0.9	0.773	A	A/R
14_4	0.9	0.94	0.773	A	A
14_4	0.85	0.9	0.773	A	A
14_4	0.75	0.99	0.773	A	A/R
14_4	0.8	0.96	0.773	A	A
14_4	0.75	0.94	0.773	A	A/R
14_4	0.85	0.96	0.773	A	A
15_3	0.75	0.99	0.804	R	A/R
15_3	0.8	0.99	0.804	R	A/R
15_3	0.9	0.94	0.804	A	A
15_3	0.8	0.96	0.804	R	A/R
15_3	0.85	0.96	0.804	A	A
15_3	0.9	0.99	0.804	A	A
15_3	0.85	0.94	0.804	A	A
15_3	0.85	0.9	0.804	R	A/R
15_3	0.9	0.96	0.804	A	A
15_3	0.85	0.99	0.804	A	A
15_3	0.8	0.94	0.804	R	A/R
15_3	0.75	0.94	0.804	R	A/R
15_3	0.75	0.96	0.804	R	A/R
15_3	0.75	0.9	0.804	R	A/R
15_3	0.8	0.9	0.804	R	A/R
16_4	0.75	0.96	0.833	A	A/R
16_4	0.75	0.9	0.833	A	A/R
16_4	0.9	0.96	0.833	A	A
16_4	0.85	0.96	0.833	A	A
16_4	0.8	0.9	0.833	A	A/R
16_4	0.9	0.99	0.833	A	A
16_4	0.75	0.99	0.833	A	A/R
16_4	0.8	0.94	0.833	A	A/R

16_4	0.85	0.99	0.833	A	A
16_4	0.75	0.94	0.833	A	A/R
16_4	0.8	0.99	0.833	A	A/R
16_4	0.9	0.94	0.833	A	A
16_4	0.85	0.94	0.833	A	A
16_4	0.8	0.96	0.833	A	A/R
16_4	0.85	0.9	0.833	A	A
14_3	0.8	0.96	0.852	A	A/R
14_3	0.8	0.94	0.852	A	A/R
14_3	0.75	0.94	0.852	A	A/R
14_3	0.85	0.96	0.852	A	A/R
14_3	0.75	0.9	0.852	R	A/R
14_3	0.8	0.9	0.852	R	A/R
14_3	0.9	0.99	0.852	A	A
14_3	0.85	0.9	0.852	A	A/R
14_3	0.75	0.99	0.852	A	A/R
14_3	0.85	0.99	0.852	A	A/R
14_3	0.8	0.99	0.852	A	A/R
14_3	0.75	0.96	0.852	A	A/R
14_3	0.85	0.94	0.852	A	A/R
14_3	0.9	0.94	0.852	A	A
14_3	0.9	0.96	0.852	A	A
17_1	0.8	0.96	0.874	R	A/R
17_1	0.85	0.94	0.874	A	A/R
17_1	0.75	0.96	0.874	R	A/R
17_1	0.85	0.99	0.874	A	A/R
17_1	0.75	0.9	0.874	R	A/R
17_1	0.75	0.94	0.874	R	A/R
17_1	0.8	0.9	0.874	R	A/R
17_1	0.8	0.94	0.874	R	A/R
17_1	0.75	0.99	0.874	R	A/R
17_1	0.9	0.94	0.874	A	A
17_1	0.85	0.9	0.874	R	A/R
17_1	0.8	0.99	0.874	R	A/R
17_1	0.9	0.99	0.874	A	A
17_1	0.85	0.96	0.874	A	A/R
17_1	0.9	0.96	0.874	A	A
16_3	0.8	0.9	0.879	A	A/R
16_3	0.8	0.94	0.879	A	A/R
16_3	0.85	0.99	0.879	A	A/R

16_3	0.9	0.99	0.879	A	A
16_3	0.75	0.9	0.879	R	A/R
16_3	0.85	0.96	0.879	A	A/R
16_3	0.8	0.96	0.879	A	A/R
16_3	0.8	0.99	0.879	A	A/R
16_3	0.75	0.99	0.879	A	A/R
16_3	0.75	0.96	0.879	A	A/R
16_3	0.9	0.96	0.879	A	A
16_3	0.85	0.9	0.879	A	A/R
16_3	0.9	0.94	0.879	A	A
16_3	0.85	0.94	0.879	A	A/R
16_3	0.75	0.94	0.879	R	A/R
15_2	0.85	0.9	0.905	A	A/R
15_2	0.9	0.94	0.905	A	A/R
15_2	0.85	0.94	0.905	A	A/R
15_2	0.9	0.96	0.905	A	A/R
15_2	0.8	0.96	0.905	A	A/R
15_2	0.85	0.96	0.905	A	A/R
15_2	0.85	0.99	0.905	A	A/R
15_2	0.9	0.99	0.905	A	A/R
15_2	0.8	0.94	0.905	A	A/R
15_2	0.75	0.94	0.905	R	A/R
15_2	0.75	0.9	0.905	R	R
15_2	0.8	0.9	0.905	A	A/R
15_2	0.75	0.99	0.905	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
18_4	0.75	0.99	0.907	R	A/R
18_4	0.8	0.94	0.907	R	A/R
18_4	0.85	0.99	0.907	A	A/R
18_4	0.8	0.96	0.907	R	A/R
18_4	0.85	0.94	0.907	R	A/R
18_4	0.8	0.9	0.907	R	R
18_4	0.85	0.9	0.907	R	R
18_4	0.75	0.94	0.907	R	A/R
18_4	0.9	0.99	0.907	A	A/R
18_4	0.75	0.96	0.907	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_4	0.9	0.96	0.907	A	A/R

18_4	0.85	0.96	0.907	R	A/R
18_4	0.9	0.94	0.907	A	A/R
17_3	0.9	0.94	0.914	A	A/R
17_3	0.8	0.99	0.914	A	A/R
17_3	0.75	0.99	0.914	R	A/R
17_3	0.8	0.96	0.914	R	A/R
17_3	0.85	0.96	0.914	A	A/R
17_3	0.8	0.94	0.914	R	A/R
17_3	0.85	0.9	0.914	A	A/R
17_3	0.9	0.99	0.914	A	A/R
17_3	0.85	0.94	0.914	A	A/R
17_3	0.75	0.94	0.914	R	A/R
17_3	0.8	0.9	0.914	R	R
17_3	0.75	0.96	0.914	R	A/R
17_3	0.9	0.96	0.914	A	A/R
17_3	0.85	0.99	0.914	A	A/R
17_3	0.75	0.9	0.914	R	R
16_1	0.85	0.9	0.918	R	R
16_1	0.8	0.99	0.918	R	A/R
16_1	0.8	0.9	0.918	R	R
16_1	0.9	0.94	0.918	R	A/R
16_1	0.85	0.94	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
16_1	0.9	0.99	0.918	R	A/R
16_1	0.75	0.9	0.918	R	R
16_1	0.85	0.96	0.918	R	A/R
16_1	0.9	0.96	0.918	R	A/R
16_1	0.75	0.94	0.918	R	A/R
16_1	0.8	0.96	0.918	R	A/R
16_1	0.75	0.99	0.918	R	A/R
16_1	0.75	0.96	0.918	R	A/R
18_2	0.75	0.99	0.918	R	A/R
18_2	0.9	0.96	0.918	R	A/R
18_2	0.8	0.99	0.918	R	A/R
18_2	0.9	0.94	0.918	R	A/R
18_2	0.85	0.9	0.918	R	R
18_2	0.85	0.94	0.918	R	A/R
18_2	0.9	0.99	0.918	R	A/R
18_2	0.8	0.96	0.918	R	A/R

18_2	0.85	0.96	0.918	R	A/R
18_2	0.75	0.94	0.918	R	A/R
18_2	0.8	0.9	0.918	R	R
18_2	0.8	0.94	0.918	R	A/R
18_2	0.85	0.99	0.918	R	A/R
18_2	0.75	0.96	0.918	R	A/R
18_2	0.75	0.9	0.918	R	R
15_1	0.85	0.96	0.927	R	A/R
15_1	0.85	0.9	0.927	R	R
15_1	0.9	0.99	0.927	R	A/R
15_1	0.75	0.94	0.927	R	A/R
15_1	0.8	0.9	0.927	R	R
15_1	0.9	0.96	0.927	R	A/R
15_1	0.8	0.94	0.927	R	A/R
15_1	0.75	0.96	0.927	R	A/R
15_1	0.8	0.99	0.927	R	A/R
15_1	0.9	0.94	0.927	R	A/R
15_1	0.75	0.9	0.927	R	R
15_1	0.85	0.99	0.927	R	A/R
15_1	0.8	0.96	0.927	R	A/R
15_1	0.75	0.99	0.927	R	A/R
15_1	0.85	0.94	0.927	R	A/R
18_3	0.75	0.96	0.930	R	A/R
18_3	0.8	0.99	0.930	R	A/R
18_3	0.85	0.9	0.930	R	R
18_3	0.9	0.96	0.930	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
18_3	0.75	0.94	0.930	R	A/R
18_3	0.8	0.9	0.930	R	R
18_3	0.85	0.99	0.930	R	A/R
18_3	0.9	0.99	0.930	R	A/R
18_3	0.8	0.96	0.930	R	A/R
18_3	0.75	0.9	0.930	R	R
18_3	0.85	0.96	0.930	R	A/R
18_3	0.8	0.94	0.930	R	A/R
18_3	0.75	0.99	0.930	R	A/R
15_4	0.9	0.94	0.941	R	R
15_4	0.85	0.94	0.941	R	R
15_4	0.8	0.96	0.941	R	A/R

15_4	0.8	0.9	0.941	R	R
15_4	0.85	0.9	0.941	R	R
15_4	0.8	0.99	0.941	R	A/R
15_4	0.75	0.94	0.941	R	R
15_4	0.75	0.9	0.941	R	R
15_4	0.9	0.96	0.941	R	A/R
15_4	0.85	0.96	0.941	R	A/R
15_4	0.75	0.99	0.941	R	A/R
15_4	0.85	0.99	0.941	R	A/R
15_4	0.75	0.96	0.941	R	A/R
15_4	0.9	0.99	0.941	A	A/R
15_4	0.8	0.94	0.941	R	R
17_4	0.75	0.99	0.941	R	A/R
17_4	0.75	0.96	0.941	R	A/R
17_4	0.9	0.96	0.941	R	A/R
17_4	0.8	0.99	0.941	R	A/R
17_4	0.85	0.96	0.941	R	A/R
17_4	0.8	0.94	0.941	R	R
17_4	0.85	0.99	0.941	R	A/R
17_4	0.75	0.94	0.941	R	R
17_4	0.9	0.94	0.941	R	R
17_4	0.85	0.94	0.941	R	R
17_4	0.75	0.9	0.941	R	R
17_4	0.8	0.96	0.941	R	A/R
17_4	0.8	0.9	0.941	R	R
17_4	0.85	0.9	0.941	R	R
17_4	0.9	0.99	0.941	A	A/R
16_0	0.85	0.9	0.954	R	R
16_0	0.8	0.99	0.954	R	A/R
16_0	0.9	0.94	0.954	A	A/R
16_0	0.9	0.99	0.954	A	A/R
16_0	0.9	0.96	0.954	A	A/R
16_0	0.85	0.96	0.954	A	A/R
16_0	0.8	0.96	0.954	A	A/R
16_0	0.85	0.94	0.954	A	A/R
16_0	0.8	0.94	0.954	A	A/R
16_0	0.85	0.99	0.954	A	A/R
16_0	0.75	0.96	0.954	R	A/R
16_0	0.75	0.9	0.954	R	R
16_0	0.75	0.94	0.954	R	R

16_0	0.8	0.9	0.954	R	R
16_0	0.75	0.99	0.954	R	A/R
17_0	0.85	0.94	0.968	A	A/R
17_0	0.85	0.96	0.968	A	A/R
17_0	0.8	0.94	0.968	A	A/R
17_0	0.85	0.99	0.968	A	A/R
17_0	0.75	0.9	0.968	R	R
17_0	0.75	0.94	0.968	R	R
17_0	0.8	0.96	0.968	A	A/R
17_0	0.8	0.9	0.968	R	R
17_0	0.75	0.99	0.968	A	A/R
17_0	0.75	0.96	0.968	R	R
17_0	0.85	0.9	0.968	A	A/R
17_0	0.8	0.99	0.968	A	A/R
17_0	0.9	0.96	0.968	A	A/R
17_0	0.9	0.94	0.968	A	A/R
17_0	0.9	0.99	0.968	A	A/R
15_0	0.8	0.9	0.984	R	R
15_0	0.9	0.96	0.984	R	R
15_0	0.85	0.96	0.984	R	R
15_0	0.9	0.99	0.984	R	A/R
15_0	0.8	0.94	0.984	R	R
15_0	0.8	0.99	0.984	R	A/R
15_0	0.75	0.9	0.984	R	R
15_0	0.75	0.99	0.984	R	A/R
15_0	0.85	0.99	0.984	R	A/R
15_0	0.9	0.94	0.984	R	R
15_0	0.85	0.94	0.984	R	R
15_0	0.8	0.96	0.984	R	R
15_0	0.85	0.9	0.984	R	R
15_0	0.75	0.96	0.984	R	R
15_0	0.75	0.94	0.984	R	R
16_2	0.9	0.99	0.987	A	A/R
16_2	0.85	0.96	0.987	A	A/R
16_2	0.75	0.94	0.987	R	R
16_2	0.8	0.9	0.987	R	R
16_2	0.8	0.94	0.987	A	A/R
16_2	0.85	0.99	0.987	A	A/R
16_2	0.8	0.96	0.987	A	A/R
16_2	0.75	0.96	0.987	R	R

16_2	0.75	0.9	0.987	R	R
16_2	0.8	0.99	0.987	A	A/R
16_2	0.9	0.96	0.987	A	A/R
16_2	0.75	0.99	0.987	A	A/R
16_2	0.9	0.94	0.987	A	A/R
16_2	0.85	0.94	0.987	A	A/R
16_2	0.85	0.9	0.987	A	A/R
18_1	0.9	0.96	0.993	R	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
18_1	0.8	0.99	0.993	R	R
18_1	0.75	0.9	0.993	R	R
18_1	0.85	0.96	0.993	R	R
18_1	0.75	0.99	0.993	R	R
18_1	0.75	0.96	0.993	R	R
18_1	0.9	0.94	0.993	R	R
18_1	0.85	0.94	0.993	R	R
18_1	0.85	0.9	0.993	R	R
18_1	0.75	0.94	0.993	R	R
18_1	0.8	0.9	0.993	R	R
18_1	0.8	0.96	0.993	R	R
18_0	0.75	0.94	0.994	R	R
18_0	0.8	0.9	0.994	R	R
18_0	0.9	0.99	0.994	R	R
18_0	0.9	0.96	0.994	R	R
18_0	0.85	0.96	0.994	R	R
18_0	0.8	0.94	0.994	R	R
18_0	0.85	0.99	0.994	R	R
18_0	0.75	0.96	0.994	R	R
18_0	0.8	0.99	0.994	R	R
18_0	0.75	0.9	0.994	R	R
18_0	0.9	0.94	0.994	R	R
18_0	0.75	0.99	0.994	R	R
18_0	0.8	0.96	0.994	R	R
18_0	0.85	0.94	0.994	R	R
18_0	0.85	0.9	0.994	R	R
17_2	0.75	0.99	0.998	R	R
17_2	0.75	0.96	0.998	R	R
17_2	0.9	0.94	0.998	R	R

17_2	0.85	0.9	0.998	R	R
17_2	0.85	0.94	0.998	R	R
17_2	0.85	0.96	0.998	R	R
17_2	0.8	0.94	0.998	R	R
17_2	0.75	0.94	0.998	R	R
17_2	0.8	0.96	0.998	R	R
17_2	0.8	0.9	0.998	R	R
17_2	0.85	0.99	0.998	R	R
17_2	0.75	0.9	0.998	R	R
17_2	0.9	0.96	0.998	R	R
17_2	0.9	0.99	0.998	R	R
17_2	0.8	0.99	0.998	R	R

B.2.2 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 5: The Extended Table

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
