Testing Probabilistic Circuits

Anonymous Author(s)
Affiliation
Address
email

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 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 100 PC benchmarks. We find that our test correctly decides the closeness of all 100 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 [27, 28] to weather forecasting [3]. Probabilistic models have a multitude of representations, such as probabilistic circuits (PCs) [8], graphical models [18], generative networks [15], and determinantal point processes [19]. 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, 23]. In this work, we will focus on PCs that are fragments of the Negation Normal Form (NNF), specifically DNNFs, d-DNNFs, SDNNFs, and PIs [12]. We refer to the survey by Choi et al. [8] 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 [20, 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 [12], bioinformatics [29, 31, 33] and probabilistic program verification [14, 21].

Equivalence testing is a special case of closeness testing, where one tests if $d_{TV}(P, Q) = 0$. Darwiche and Huang [12] 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 [9] and sampling-based learning of PCs [24, 25]. As a concrete example, consider PCs that are learned via approximate methods such as stochastic gradient descent [25]. 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

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

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 (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 atmost $O((\eta-\varepsilon)^{-2}\log(\delta^{-1}))$ 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 [30]. 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 [10] and Arenas et al. [1]. 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 $\varphi : \{0,1\}^n \to \{0,1\}$ 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.

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 $\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P}$ [32]. 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 $\mathtt{w} : \{0,1\}^n \to \mathbb{Q}^+$ 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 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 [22]. Log-linear models are represented as literal-weighted functions, defined as:

Definition 1. *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.

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

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

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

Our closeness testing algorithm 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 as follows:

Definition 2. $\mathsf{Awct}(\alpha, \beta, \varphi, \mathtt{w})$ *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 3. $\mathsf{Samp}(\alpha, \beta, \varphi, \mathtt{w})$ *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, whenever Samp samples σ:*

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

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

In the rest of the paper $[m]$ denotes the set $\{1, 2, \ldots m\}$, $\mathbb{1}(e)$ 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 [12] 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.

Probabilistic circuits: In this paper, we study fragments of the Negation Normal Form (NNF). A circuit 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 [13] and the paper [26].

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 $\mathtt{w}_1, \mathtt{w}_2$ 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.

The 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.

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

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: $k_2 \leftarrow \mathsf{Awct}(\sqrt{1+\gamma/4} - 1, \delta/8, \varphi_2, \mathtt{w}_2)$
6: **for** $i \in \{1, 2 \ldots, m\}$ **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

The following theorem asserts the correctness of Teq.

Theorem 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* $(1-\delta)$.

The following theorem states the running time of the algorithm,

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

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

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.3.2 and Section A.3.3, respectively.

For the purpose of the proof, we will first define events $\mathtt{Pass}_1, \mathtt{Pass}_2$ and $\mathtt{Good}$. Events $\mathtt{Pass}_1, \mathtt{Pass}_2$ 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 1). $\mathtt{Pass}_1$ 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 $\mathtt{Fail}_\mathtt{i}$ denote the event that Samp (Algorithm 1, line 7) returns the symbol $\perp$ in the ith iteration of the loop. By the definition of Samp we know that $\forall_{i\in[m]} \Pr[\mathtt{Fail}_\mathtt{i}] < \delta/4m$.

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

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

We will now state a minor proposition which we will prove in the supplementary Section A.3.

Proposition 1. *For a non empty set* U *and real valued functions* $f(\cdot), g(\cdot)$, *if* $\sum_{x\in U}(f(x)-g(x)) = 0$, *then* $\frac{1}{2}\sum_{x\in U}|f(x)-g(x)| = \sum_{\{x: f(x)-g(x)>0\}}(f(x)-g(x))$

We will use the following lemma for our proof.

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

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

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

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,

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

Using the above lemmas, we prove the lemma critical for our proof of correctness of Teq.

Lemma 2. *Assuming the event* Good*, let* $A = \sum_{\sigma \in \{0,1\}^n} \mathbb{1}\left(r(\sigma) < 1\right)\left(1 - r(\sigma)\right) P_1(\sigma)$ *, 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. First we use Proposition 1,

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

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

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

Using our assumption of the event Good and Lemma 1, $|B| \leq \sum_{\sigma \in \{0,1\}^n} \gamma/4 \cdot P(\varphi_1, \mathtt{w_1})(\sigma) \leq \gamma/4$ 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 $A \geq \eta - \gamma/4$. □

3.1.1 Proof of Theorem 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:

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

According to definition 3, 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 2, 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 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 $\texttt{Good}$:

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

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

4 Evaluation

To evaluate the performance of Teq, we implemented a prototype in Python. The prototype uses WAPS[1] [16] as a weighted sampler to sample over the input $\mathsf{d\text{-}DNNF}$ 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?*

Our experiments were conducted on a high performance computing 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. 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 Peq 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.

Our results indicate that Teq can decide the closeness of circuits efficiently and reliably. Teq terminated with the correct result within 3600 seconds on every PC from the dataset.

Dataset We conducted experiments on a range of publicly available benchmarks arising from sampling and counting tasks. Our dataset contained 100 $\mathsf{d\text{-}DNNF}$ 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 $\mathsf{d\text{-}DNNF}$s 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.

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

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 1: 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 farness parameter is $\eta = 0.2$.

The number of samples required for the test (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 1. The first column indicates the name of the benchmark, the second and third indicate the result and runtime of Teq when presented with an ε-close PC 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 detailed results are presented in the appendix 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 2. 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 [12](presented in supplementary Section A.1 for completeness).

5.1 Upper bounds

In Table 2 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 [12] 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 [10], and a weighted sampler was provided by [16]. Arenas et al. [1] provided the first approximate counting and uniform sampling algorithm for SDNNFs. In the supplementary Section A.4), we show that with the use of chain formulas, the uniform sampling and counting algorithms extend to log-linear SDNNF distributions as well.

5.2 Hardness

In Table 2, we claim that the pairs of classes of PCs labeled with symbols E and C, cannot be tested in poly-time for equivalence and closeness, 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 [13].
- 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 Propo-

	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 2: 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.

sitions 2, 3 and 4. These propositions and their proofs can be found in the supplementary Section A.4.

- We then connect the existence of poly-time algorithms for equivalence to the collapse of PH via a complexity result of Karp and Lipton [17].

The NP-hardness of deciding the equivalence of pairs of DNNFs and pairs of SDNNFs was first shown by Pipatsrisawat and Darwiche [26]. We recast their proofs in the language of distribution testing for the sake of completeness in the supplementary Section A.4.

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 we 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 2, 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4
   (b) Did you mention the license of the assets? [Yes] License for WAPS can be accessed by following the footnote link in Section 4
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

[1] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. When is approximate counting for conjunctive queries tractable? *In STOC 2021, to appear*, 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] Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *Proc. of IJCAI*, 2015.

[8] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. 2020.

[9] 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.

[10] 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.

[11] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 2003.

[12] Adnan Darwiche and Jinbo Huang. Testing equivalence probabilistically. In *Technical Report*, 2002.

[13] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17, 2002.

[14] Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic. Testing probabilistic programming systems. In *Proc. of Joint Meeting on ESE and FSE*, 2018.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[16] Rahul Gupta, Shubham Sharma, Subhajit Roy, and Kuldeep S Meel. Waps: Weighted and projected sampling. In *TACAS*, 2019.

[17] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. of STOC*, 1980.

[18] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[19] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

[20] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *NeurIPS*, 2018.

[21] Andrzej S. Murawski and Joël Ouaknine. On probabilistic program equivalence and refinement. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, 2005.

[22] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[23] Agnieszka Oniśko, Marek J Druzdzel, and Hanna Wasyluk. Extension of the hepar ii model to multiple-disorder diagnosis. In *Intelligent Information Systems*. 2000.

[24] 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.

[25] 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.

[26] Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008.

[27] Arthur R Pope and David G Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 2000.

[28] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[29] Yasir Rahmatallah, Frank Emmert-Streib, and Galina Glazko. Gene sets net correlations analysis (gsnca): a multivariate differential coexpression test for gene sets. *Bioinformatics*, 2014.

[30] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2), 1996.

[31] Nicolas Städler and Sach Mukherjee. Multivariate gene-set testing based on graphical models. *Biostatistics*, 2015.

[32] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 1976.

[33] 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.