Variance Computation for Weighted Model Counting with Knowledge Compilation Approach

Kengo Nakamura, Masaaki Nishino, Norihito Yasuda

Communication Science Laboratories, NTT, Inc., Kyoto, Japan
{kengo.nakamura,masaaki.nishino,norihito.yasuda}@ntt.com

Abstract

One of the most important queries in knowledge compilation is weighted model counting (WMC), which has been applied to probabilistic inference on various models, such as Bayesian networks. In practical situations on inference tasks, the model's parameters have uncertainty because they are often learned from data, and thus we want to compute the degree of uncertainty in the inference outcome. One possible approach is to regard the inference outcome as a random variable by introducing distributions for the parameters and evaluate the *variance* of the outcome. Unfortunately, the tractability of computing such a variance is hardly known. Motivated by this, we consider the problem of computing the variance of WMC and investigate this problem's tractability. First, we derive a polynomial time algorithm to evaluate the WMC variance when the input is given as a structured d-DNNF. Second, we prove the hardness of this problem for structured DNNFs, d-DNNFs, and FBDDs, which is intriguing because the latter two allow polynomial time WMC algorithms. Finally, we show an application that measures the uncertainty in the inference of Bayesian networks. We empirically show that our algorithm can evaluate the variance of the marginal probability on real-world Bayesian networks and analyze the impact of the variances of parameters on the variance of the marginal.

Code — https://github.com/nttcslab/variance-wmc

Introduction

Knowledge compilation is a technique that represents a propositional formula, a.k.a., a Boolean function, as a compressed and tractable form. Once Boolean functions are compiled into certain representations, we can solve various queries in polynomial time in the sizes of the representations (Darwiche and Marquis 2002). Among various queries, the most prominent one is *weighted model counting* (*WMC*), which is the problem of counting the (weighted) number of satisfying assignments of a Boolean function. WMC has been applied to various probabilistic inference tasks on, e.g., Bayesian networks (Chavira and Darwiche 2008; Dilkas and Belle 2021), factor graphs (Choi, Kisa, and Darwiche 2013), and probabilistic programming (Fierens et al. 2011; Holtzen, Van den Broeck, and Millstein 2020).

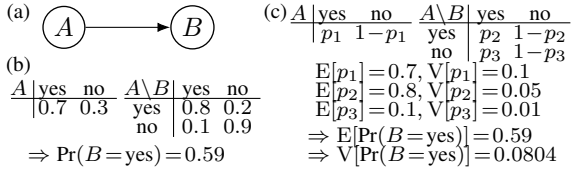

Figure 1: (a) A Bayesian network. (b) Example of ordinal inference, where parameters are fixed. (c) Example of our situation where parameters have variances.

In practical situations, the parameters of such probabilistic models are often obtained by learning from data (Cozman 2000; Heckerman 2008). When we lack sufficient data, they may suffer from uncertainty. Perhaps such an uncertainty leads to unreliable inference results. However, ordinal inference methods (including methods using WMC) disregard uncertainty in parameters. Thus, we want to compute the degree of uncertainty in the inference outcome when the parameters are imprecise. A Bayesian statistical approach regards the inference outcome as a random variable by considering the distributions for the parameters and computes the *variance* of the outcome. For example, for the Bayesian network in Fig. 1(a), we consider the variance of the outcome when parameters follow distributions, as in Fig. 1(c). By introducing the expectation and variance of parameters, the outcome's expectation equals the marginal, and we can also obtain its variance. The computed variance affects the decision-making that depends on the inference outcome; when the computed variance is too large, we should regard the inference result as unreliable. However, the tractability of computing the variance of the inference outcome remains unknown; although the variance computation is expected to be at least as difficult as the ordinal inference, we do not know the extent of its difficulty.

In the applications of inference, the WMC value typically equals the inference outcome. Thus, motivated by the variance computation of the inference outcome, we consider the query of computing the *variance of WMC value* when the weights associated with Boolean variables have variances. As explained later, a previous study (Nakamura et al. 2022) treated the variance of WMC value with knowledge compilation in a special case of network analysis. However, this work is specialized to network analysis and does not con-

sider general WMC tasks. Moreover, it only uses ordered binary decision diagrams (OBDDs) (Bryant 1986), one of the most restricted representations in knowledge compilation. Therefore, this work hardly reveals the tractability of variance computation including inference tasks. Thus, we formalize variance computation query for the WMC of general Boolean function and investigate the tractability of it with various knowledge compilation representations.

Our contributions are three-fold. First, we propose a polynomial time algorithm that computes the variance of WMC of a Boolean function represented as a structured d-DNNF (Pipatsrisawat and Darwiche 2008). This result is meaningful since structured d-DNNFs subsume sentential decision diagrams (SDDs) (Darwiche 2011), which have been widely used in many applications, as a subset. Second, we prove that we cannot compute the WMC's variance in polynomial time unless P=NP when the Boolean function is represented as a structured DNNF, a d-DNNF (Darwiche 2001), or an FBDD (Gergov and Meinel 1994), all of which are strict supersets of structured d-DNNFs. The results for d-DNNFs and FBDDs are interesting because the WMC itself can be computed in polynomial time for these representations. Third, we present an application for the inference of Bayesian networks and show that the variance of the marginal probability can be obtained in polynomial time for a Bayesian network with a constant treewidth. We also empirically demonstrate the tractability of the proposed algorithm with real-world Bayesian networks and showcase an example of uncertainty analysis on Bayesian networks with variance computation. Particularly, we demonstrated that we can find parameters of a Bayesian network whose variances have greater impact on the variance of the marginal probability, a useful result for the additional learning of parameters that effectively reduce the uncertainty of the inference.

The URL of the full version of this paper is announced on https://github.com/nttcslab/variance-wmc.

Related Work

Knowledge compilation is regarded as a key technique for tackling computationally difficult propositional reasoning tasks. Thus, as well as the succinctness of representations, the tractability for various operations is the central research subject. Knowledge compilation map (Darwiche and Marquis 2002), which summarizes the succinctness and tractability of various representations, have been extended by subsequent studies. For example, the tractability of standard operations has been studied for recently proposed representations (Illner 2025; Onaka et al. 2025) and the tractability of the generalization of WMC such as algebraic model counting (AMC) and two-level AMC (2AMC) was recently investigated (Kiesel, Totis, and Kimmig 2022; Wang et al. 2024). Our study broadens the application of knowledge compilation by proposing a new query related to probabilistic inference, which is a major application of knowledge compilation, and investigating this query's position on the knowledge compilation map.

In probabilistic inference, it is crucial to deal with uncertainty in parameters. A typical approach to incorporate uncertainty is a fully Bayesian approach, where we regard every parameter as drawn from a distribution, as in the Introduction. However, to the best of our knowledge, no study has considered the variance of the marginal in a Bayesian network with this approach. Another line of research for incorporating uncertainty in Bayesian networks is credal networks (Cozman 2000), where imprecise probabilities are modeled as sets of distributions called credal sets. Credal networks enable robust inferences by computing the bounds of the marginal probability when the parameters have fluctuated within given bounds. However, marginal inference for credal networks is NP-hard even for networks with constant treewidth (De Campos and Cozman 2005). In contrast, our approach can compute the variance of the marginal in polynomial time for networks with constant treewidth.

WMC has also been applied to the reliability analysis on communication networks where links are stochastically failed (Duenas-Osorio et al. 2017). For this purpose, Boolean function f', which indicates the connectivity in sub-networks, is considered and the reliability equals the WMC of f' (Hardy, Lucet, and Limnios 2007). Nakamura et al. (2022) proposed an algorithm that computes the variance of reliability in polynomial time in the size of the OBDD (Bryant 1986) representing f' when the existential probability of each link in the network has variance. We extend their problem setting and algorithm to handle WMC's variance computation of a general Boolean function. Moreover, we extended their algorithm to work on structured d-DNNFs, a strict superset of OBDDs; here, our algorithm's key technical difference is its management of variable sets and variable decompositions guided by a vtree, as described later. As a byproduct, we can prove that the variance of network reliability on networks with constant treewidth can be computed in polynomial time. This theoretically improves the previous result (Nakamura et al. 2022) stating that it can be computed in polynomial time for networks with constant *pathwidth*, since the treewidth subsumes the pathwidth but not vice versa; details are in the full version.

There exist studies to represent a probability distribution of a random variable X as a tractable circuit in a spirit of knowledge compilation: probabilistic circuits (Choi, Vergari, and Van den Broeck 2020) represent probability mass functions, while probabilistic generating circuits (Zhang, Juba, and Van den Broeck 2021) and characteristic circuits (Yu, Trapp, and Kersting 2023) represent probability generating and characteristic functions. These circuits admit polytime moment computation, including the variance, of the random variable X under certain structural restrictions. In contrast, our work regards the *probability* $\Pr(X = a)$ as a random variable and computes the variance of it, where X is a random variable appearing in, e.g., Bayesian networks. To derive the variance of the inference outcome, our work is needed because we currently have no approach to compute the variance of the probability value seen as a random variable with probabilistic circuits.

Preliminaries

A *Boolean function* takes a set of Boolean variables each valued *true* or *false* as an input and outputs either *true* or *false*. An *assignment* a on variable set $\mathcal{V}$ is a mapping $\mathcal{V} \to$

$\{$*true*,*false*$\}$. Assignment a is called a *model* of Boolean function f if f is evaluated to *true* under a.

A rooted directed acyclic graph is called a *negation normal form* (*NNF*) if the leaf nodes are labeled with *true*, *false*, x, or $\neg x$, where x is a Boolean variable, and the internal nodes are labeled with either $\wedge$ or $\vee$. The size of the NNF is defined as the number of arcs. For node α of an NNF, Boolean function f_α represented by α is defined as follows. For leaf node α, $f_\alpha = \alpha$; *true* and *false* stand for identity functions that always evaluate to *true* and *false*. For internal node α, let $\alpha_1, \ldots, \alpha_k$ be the child nodes of α. If α is a $\wedge$-node, $f_\alpha = \bigwedge_j f_{\alpha_j}$. If α is a $\vee$-node, $f_\alpha = \bigvee_j f_{\alpha_j}$. The Boolean function represented by an NNF is that represented by its root node. We often abuse a symbol for NNF node α to represent the whole NNF rooted at α.

Next, we define several restrictions on NNFs, which induce subsets of NNFs. For NNF node α, let $\mathrm{Var}(\alpha)$ be the set of Boolean variables that appear as the labels of the descendant nodes of α, called the *scope* of α. In the following, let $\alpha_1, \ldots, \alpha_k$ be the child nodes of internal node α.

Definition 1. An NNF is called *decomposable* if every $\wedge$-node α satisfies $\mathrm{Var}(\alpha_i) \cap \mathrm{Var}(\alpha_j) = \emptyset$ for any $i \neq j$. An NNF is called *deterministic* if every $\vee$-node α satisfies $f_{\alpha_i} \wedge f_{\alpha_j} =$ *false* for any $i \neq j$. An NNF is called *decision* if every $\vee$-node only appears in the form: $(x \wedge \alpha) \vee (\neg x \wedge \beta)$, where $x, \neg x$ are leaf nodes.

A *d-DNNF* is a decomposable and deterministic NNF, and *FBDD* is a decomposable and decision NNF with the following additional restriction; for every $\vee$-node, α, β in the decision property must be either a leaf node or a $\vee$-node. We also define structured decomposability as follows.

Definition 2. A *vtree* T on variable set $\mathcal{V}$ is a rooted binary tree, where each leaf node is labeled with a Boolean variable in $\mathcal{V}$ and each internal node v has exactly two child nodes $\mathsf{v}^l, \mathsf{v}^r$. Here, any Boolean variable $x \in \mathcal{V}$ must appear as a label exactly once. To distinguish them from NNF nodes, we call the nodes of a vtree a *vnode*. The *scope* $\mathrm{Var}(\mathsf{v})$ of vnode v is the set of the labels of the descendants of v. For NNF node α, its *decomposition vnode* $\mathsf{d}(\alpha)$ of vtree T is the deepest vnode v in T satisfying $\mathrm{Var}(\alpha) \subseteq \mathrm{Var}(\mathsf{v})$.

Definition 3. We say an NNF *respects vtree* T if every $\wedge$-node α has exactly two child nodes α^l, α^r and they satisfy $\mathrm{Var}(\alpha^l) \subseteq \mathrm{Var}(\mathsf{v}^l)$ and $\mathrm{Var}(\alpha^r) \subseteq \mathrm{Var}(\mathsf{v}^r)$ for some vnode v of T. An NNF is called *structured decomposable* if it respects some vtree.

A *structured DNNF* (*st-DNNF*) is a structured decomposable NNF. A *structured d-DNNF* (*st-d-DNNF*) is a structured decomposable and deterministic NNF.

Example 4. Let $f = (\neg a \wedge b \wedge \neg c \wedge d) \vee (a \wedge b \wedge \neg c \wedge d) \vee (a \wedge b \wedge c \wedge \neg d)$. Fig. 2(a) depicts an st-d-DNNF of f and the respected vtree. Fig. 2(b) is a d-DNNF of f; however, it is not structured decomposable because the left child of the root decomposes the variables into $\{a, d\}$ and $\{b, c\}$ while the right child decomposes them into $\{a, b\}$ and $\{c, d\}$.

Here, we assume that, for any $\wedge$-node α of an st-d-DNNF with child nodes α^l, α^r, $\mathrm{Var}(\alpha^l) \neq \emptyset$ and $\mathrm{Var}(\alpha^r) \neq \emptyset$. We can easily transform an st-d-DNNF to satisfy the above

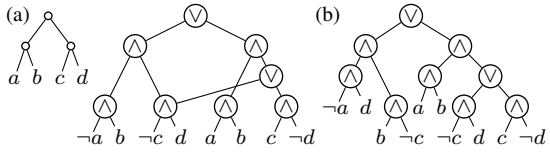

Figure 2: (a) A vtree and an st-d-DNNF. (b) A d-DNNF that is not structured decomposable.

assumption. If $\mathrm{Var}(\alpha^l) = \emptyset$, f_{α^l} is either *true* or *false*. When $f_{\alpha^l} =$ *true*, we can replace α with α^r; i.e., we eliminate α and redirect the incoming arcs of α to α^r. Otherwise, we can replace α with *false*. We can perform the same transformation when $\mathrm{Var}(\alpha^r) = \emptyset$. Under this assumption, the vnode v appeared in Definition 3 is determined as $\mathsf{v} = \mathsf{d}(\alpha)$. This can be proved as follows. We have $\mathrm{Var}(\alpha) = \mathrm{Var}(\alpha^l) \cup \mathrm{Var}(\alpha^r) \subseteq \mathrm{Var}(\mathsf{v}^l) \cup \mathrm{Var}(\mathsf{v}^r) = \mathrm{Var}(\mathsf{v})$. Also, we have $\mathrm{Var}(\alpha) \nsubseteq \mathrm{Var}(\mathsf{v}^l)$ following from $\mathrm{Var}(\alpha) \setminus \mathrm{Var}(\mathsf{v}^l) = \mathrm{Var}(\alpha^r) \neq \emptyset$. Similarly, $\mathrm{Var}(\alpha) \nsubseteq \mathrm{Var}(\mathsf{v}^r)$. Therefore, v is the deepest vnode such that $\mathrm{Var}(\alpha) \subseteq \mathrm{Var}(\mathsf{v})$.

Variance of Weighted Model Counting

We first define the WMC. We denote the set of models of f on variable set $\mathcal{V}$ by $\mathcal{A}_f^{\mathcal{V}}$. For each variable x in variable set $\mathcal{V}$, we assign *positive weight* P_x and *negative weight* N_x. Then we define the WMC $W_f^{\mathcal{V}}$ of f on variable set $\mathcal{V}$ by

$$W_f^{\mathcal{V}} \coloneqq \sum_{a \in \mathcal{A}_f^{\mathcal{V}}} W_a^{\mathcal{V}}, \quad W_a^{\mathcal{V}} \coloneqq \prod_{\substack{x \in \mathcal{V} \\ a(x) = \textit{true}}} P_x \cdot \prod_{\substack{x \in \mathcal{V} \\ a(x) = \textit{false}}} N_x. \tag{1}$$

Note that the value of $W_f^{\mathcal{V}}$ changes by modifying $\mathcal{V}$. Thus, when considering the WMC, we must care about the variable set. If $\mathcal{V}$ is clear from the context, we omit the superscripts.

In existing studies, P_x and N_x are given as real values without uncertainty. In this paper, for every variable $x \in \mathcal{V}$, P_x and N_x are regarded as random variables with bounded expectation and variance. Then, WMC W_f, defined by (1), is also a random variable with bounded expectation and variance. This virtually considers the variance of the inference outcome in the applications since the WMC value typically equals the outcome, as described in Introduction.

We assume that (P_x, N_x) and (P_y, N_y) are independent for $x \neq y$, while P_x and N_x for the same x are not necessarily independent. Then the expectation $\mathrm{E}[W_f]$ is equivalent to the ordinal WMC: $\mathrm{E}[W_f^{\mathcal{V}}] = \sum_{a \in \mathcal{A}_f^{\mathcal{V}}} \mathrm{E}[W_a^{\mathcal{V}}] = \sum_{a \in \mathcal{A}_f^{\mathcal{V}}} \prod_{x \in \mathcal{V}: a(x) = \textit{true}} \mathrm{E}[P_x] \cdot \prod_{x \in \mathcal{V}: a(x) = \textit{false}} \mathrm{E}[N_x]$. This assumption is reasonable for some applications, and later we slightly relax it for a specific application; see the Application section. Now we formally define the variance computation queries.

Problem 5. We are given expectations μ_{P_x}, μ_{N_x} and variances $\sigma^2_{P_x}, \sigma^2_{N_x}$ of P_x, N_x and covariance $\sigma_{P_x N_x}$ of P_x and N_x for every $x \in \mathcal{V}$. We define *variance computation* query **VC** as the computation of variance $\mathrm{V}[W_f^{\mathcal{V}}]$ of WMC of input Boolean function f. As a related one, we define *covariance computation* query **CVC** as the computation of covariance $\mathrm{Cov}[W_f^{\mathcal{V}}, W_g^{\mathcal{V}}]$ of WMCs of input Boolean functions f, g.

We have $\mathrm{Cov}[W_f, W_g] = \sum_{a\in\mathcal{A}_f}\sum_{b\in\mathcal{A}_g}\mathrm{Cov}[W_a, W_b]$, each term of which can be computed in $O(|\mathcal{V}|)$ time. Thus, if the models of Boolean functions are explicitly enumerated, we can compute $\mathrm{V}[W_f] = \mathrm{Cov}[W_f, W_f]$ in $O(|\mathcal{V}||\mathcal{A}_f|^2)$ time and $\mathrm{Cov}[W_f, W_g]$ in $O(|\mathcal{V}||\mathcal{A}_f||\mathcal{A}_g|)$ time.

Example 6. Let f be the Boolean function in Example 4. Let $\mu_{P_x} = \mu$, $\mu_{N_x} = 1-\mu$, $\sigma^2_{P_x} = \sigma^2_{N_x} = \sigma^2$, and $\sigma_{P_xN_x} = -\sigma^2$ for any $x \in \mathcal{V} = \{a, b, c, d\}$. Then $W_f = N_aP_bN_cP_d + P_aP_bN_cP_d + P_aP_bP_cN_d$, and thus $\mathrm{E}[W_f] = \mu^2 - \mu^4$. Similarly, $\mathrm{V}[W_f] = (2\mu^2 - 2\mu^3 - 2\mu^4 + 4\mu^6)\sigma^2 + (1 - 2\mu + 2\mu^2 + 6\mu^4)\sigma^4 + (2 + 4\mu^2)\sigma^6 + \sigma^8$.

However, since $|\mathcal{A}_f|$ and $|\mathcal{A}_g|$ are generally exponential in $|\mathcal{V}|$, this solution causes a prohibitively long running time. Therefore, we consider how to solve these queries when Boolean functions are represented as NNFs.

Tractability Results

The goal of this section is to prove the following theorem.

Theorem 7. *When f, g are given as st-d-DNNFs α, β respecting the same vtree,* ***CVC*** *can be solved in $O(|\alpha||\beta| + |\mathcal{V}|^2)$ time. Thus, when f is given as an st-d-DNNF α,* ***VC*** *can be solved in $O(|\alpha|^2 + |\mathcal{V}|^2)$ time.*

We first introduce some fundamental formulas that are frequently used. Given random variables A, B, C, X, Y, suppose that (A, B) and (X, Y) are independent. Then,

$$\mathrm{Cov}[A + B, C] = \mathrm{Cov}[A, C] + \mathrm{Cov}[B, C], \tag{2}$$

$$\mathrm{Cov}[AX, BY] = \mathrm{Cov}[A, B]\mathrm{Cov}[X, Y] + \mathrm{Cov}[A, B]\mathrm{E}[X]\mathrm{E}[Y] + \mathrm{E}[A]\mathrm{E}[B]\mathrm{Cov}[X, Y]. \tag{3}$$

Eq. (2) is a well-known formula derived from the linearity of covariances. Eq. (3) is analogous to the formula for the variance of the product of independent random variables; the proof of (3) can be found in (Nakamura et al. 2022).

Using these formulas, we design an algorithm to compute $\mathrm{Cov}[W_{f_\alpha}, W_{f_\beta}]$ for given st-d-DNNFs α, β respecting the same vtree. The proposed algorithm computes $\mathrm{Cov}[W_{f_\alpha}, W_{f_\beta}]$ by recursively decomposing it into the sums and products of $\mathrm{Cov}[W_{f_{\alpha'}}, W_{f_{\beta'}}]$s, where α', β' are the child nodes of α, β. To avoid redundant recursive calls, the value of $\mathrm{Cov}[W_{f_{\alpha'}}, W_{f_{\beta'}}]$ is cached once it is computed. However, since WMC value $W^{\mathcal{V}}_f$ is altered by changing variable set $\mathcal{V}$, we must track the variable set in decomposing the covariance. We manage the variable set by fully using the vtree. More specifically, let $\mathsf{anc} = \mathsf{LCA}(\mathsf{d}(\alpha), \mathsf{d}(\beta))$ be the *lowest common ancestor* (*LCA*) of $\mathsf{d}(\alpha)$ and $\mathsf{d}(\beta)$, which is the deepest vnode v such that it is the ancestor of both $\mathsf{d}(\alpha)$ and $\mathsf{d}(\beta)$. Our algorithm recursively computes $\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}]$, where $\mathcal{V} = \mathrm{Var}(\mathsf{anc})$. In the following, for convenience, we define $\mathsf{d}(\mathit{true}) = \mathsf{d}(\mathit{false}) = \bot$, which is an imaginary vnode satisfying $\mathrm{Var}(\bot) = \emptyset$, $\mathsf{LCA}(\bot, \bot) = \bot$, and $\mathsf{LCA}(\mathsf{v}, \bot) = \mathsf{v}$ for any other vnode v. In other words, $\bot$ is a vnode that is a descendant of any other vnodes.

Decomposition Lemmas

To derive the algorithm, we must determine how the covariance is decomposed into the covariances of child nodes. We derive decomposition formulas by conducting a comprehensive case analysis: (I) $\mathsf{d}(\alpha)$ and $\mathsf{d}(\beta)$ have no ancestor-descendant relation, (II) $\mathsf{d}(\alpha)$ is an ancestor of $\mathsf{d}(\beta)$ and α is a $\vee$-node, and (III) $\mathsf{d}(\alpha)$ is an ancestor of $\mathsf{d}(\beta)$ and α is a $\wedge$-node. Here, (II) and (III) allow $\mathsf{d}(\alpha) = \mathsf{d}(\beta)$. Note that when $\mathsf{d}(\beta)$ is an ancestor of $\mathsf{d}(\alpha)$, we can swap α and β to satisfy (II) or (III). In the following, we derive decomposition formulas for each case.

In case (I), i.e., both $\mathsf{anc} \neq \mathsf{d}(\alpha)$ and $\mathsf{anc} \neq \mathsf{d}(\beta)$ hold, the following decomposition holds by considering how f_α and f_β can be represented on variable set $\mathrm{Var}(\mathsf{anc})$.

Lemma 8. *In case (I), by letting $\mathcal{V} := \mathrm{Var}(\mathsf{anc})$, $\mathcal{V}^l := \mathrm{Var}(\mathsf{anc}^l)$, and $\mathcal{V}^r := \mathrm{Var}(\mathsf{anc}^r)$, we have*

$$\begin{aligned}\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}] &= \mathrm{Cov}[W^{\mathcal{V}^l}_{f_\alpha}, W^{\mathcal{V}^l}_{\mathit{true}}]\mathrm{Cov}[W^{\mathcal{V}^r}_{\mathit{true}}, W^{\mathcal{V}^r}_{f_\beta}] \\ &\quad + \mathrm{Cov}[W^{\mathcal{V}^l}_{f_\alpha}, W^{\mathcal{V}^l}_{\mathit{true}}]\mathrm{E}[W^{\mathcal{V}^r}_{\mathit{true}}]\mathrm{E}[W^{\mathcal{V}^r}_{f_\beta}] \\ &\quad + \mathrm{E}[W^{\mathcal{V}^l}_{f_\alpha}]\mathrm{E}[W^{\mathcal{V}^l}_{\mathit{true}}]\mathrm{Cov}[W^{\mathcal{V}^r}_{\mathit{true}}, W^{\mathcal{V}^r}_{f_\beta}].\end{aligned} \tag{4}$$

Proof. Since $\mathrm{Var}(\alpha) \subseteq \mathcal{V}^l$, $\mathcal{V}^l \cup \mathcal{V}^r = \mathcal{V}$, and $\mathcal{V}^l \cap \mathcal{V}^r = \emptyset$, f_α on variable set $\mathcal{V}$ can be represented as $f_\alpha \wedge \mathit{true}^{\mathcal{V}^r}$, where $\mathit{true}^{\mathcal{V}^r}$ is a *true* function on variable set $\mathcal{V}^r$. Thus, we have $W^{\mathcal{V}}_{f_\alpha} = W^{\mathcal{V}^l}_{f_\alpha} W^{\mathcal{V}^r}_{\mathit{true}}$. Similarly, $W^{\mathcal{V}}_{f_\beta} = W^{\mathcal{V}^l}_{\mathit{true}} W^{\mathcal{V}^r}_{f_\beta}$. Since $(W^{\mathcal{V}^l}_{f_\alpha}, W^{\mathcal{V}^l}_{\mathit{true}})$ and $(W^{\mathcal{V}^r}_{\mathit{true}}, W^{\mathcal{V}^r}_{f_\beta})$ are independent, the lemma follows from (3). □

In case (II), we have the following decomposition.

Lemma 9. *In case (II), by letting $\mathcal{V} := \mathrm{Var}(\mathsf{anc})$ and $\alpha_1, \ldots, \alpha_k$ be the child nodes of α, we have*

$$\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}] = \textstyle\sum_{j=1}^{k} \mathrm{Cov}[W^{\mathcal{V}}_{f_{\alpha_j}}, W^{\mathcal{V}}_{f_\beta}]. \tag{5}$$

Proof. By determinism of $f_\alpha = \bigvee_{j=1}^{k} f_{\alpha_j}$, we have $W^{\mathcal{V}}_{f_\alpha} = \sum_{j=1}^{k} W^{\mathcal{V}}_{f_{\alpha_j}}$. Eq. (5) follows by recursively applying (2). □

In case (III), two child nodes α^l, α^r satisfy $\mathrm{Var}(\alpha^l) \subseteq \mathrm{Var}(\mathsf{anc}^l)$ and $\mathrm{Var}(\alpha^r) \subseteq \mathrm{Var}(\mathsf{anc}^r)$ by structured decomposability. This leads to the following decomposition.

Lemma 10. *In case (III), suppose f_β can be decomposed as $f'_\beta \wedge f''_\beta$, where $\mathrm{Var}(f'_\beta) \subseteq \mathrm{Var}(\mathsf{anc}^l) =: \mathcal{V}^l$ and $\mathrm{Var}(f''_\beta) \subseteq \mathrm{Var}(\mathsf{anc}^r) =: \mathcal{V}^r$. Then, by letting $\mathcal{V} := \mathrm{Var}(\mathsf{anc})$,*

$$\begin{aligned}\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}] &= \mathrm{Cov}[W^{\mathcal{V}^l}_{f_{\alpha^l}}, W^{\mathcal{V}^l}_{f'_\beta}]\mathrm{Cov}[W^{\mathcal{V}^r}_{f_{\alpha^r}}, W^{\mathcal{V}^r}_{f''_\beta}] \\ &\quad + \mathrm{Cov}[W^{\mathcal{V}^l}_{f_{\alpha^l}}, W^{\mathcal{V}^l}_{f'_\beta}]\mathrm{E}[W^{\mathcal{V}^r}_{f_{\alpha^r}}]\mathrm{E}[W^{\mathcal{V}^r}_{f''_\beta}] \\ &\quad + \mathrm{E}[W^{\mathcal{V}^l}_{f_{\alpha^l}}]\mathrm{E}[W^{\mathcal{V}^l}_{f'_\beta}]\mathrm{Cov}[W^{\mathcal{V}^r}_{f_{\alpha^r}}, W^{\mathcal{V}^r}_{f''_\beta}].\end{aligned} \tag{6}$$

Proof. We have $W^{\mathcal{V}}_{f_\alpha} = W^{\mathcal{V}^l}_{f_{\alpha^l}} W^{\mathcal{V}^r}_{f_{\alpha^r}}$ and $W^{\mathcal{V}}_{f_\beta} = W^{\mathcal{V}^l}_{f'_\beta} W^{\mathcal{V}^r}_{f''_\beta}$, where $(W^{\mathcal{V}^l}_{f_{\alpha^l}}, W^{\mathcal{V}^l}_{f'_\beta})$ and $(W^{\mathcal{V}^r}_{f_{\alpha^r}}, W^{\mathcal{V}^r}_{f''_\beta})$ are independent. The lemma follows from (3). □

We can decompose $f_\beta = f'_\beta \wedge f''_\beta$ for the following cases. If $\mathsf{anc} \neq \mathsf{d}(\beta)$, either $\mathrm{Var}(\mathsf{d}(\beta)) \subseteq \mathcal{V}^l$ or $\mathrm{Var}(\mathsf{d}(\beta)) \subseteq \mathcal{V}^r$ holds. We can take $(f'_\beta, f''_\beta) = (f_\beta, \mathit{true})$ for the former and $(f'_\beta, f''_\beta) = (\mathit{true}, f_\beta)$ for the latter. If $\mathsf{anc} = \mathsf{d}(\beta)$ and β is a $\wedge$-node, child nodes β^l, β^r satisfy $\mathrm{Var}(\beta^l) \subseteq \mathcal{V}^l$ and $\mathrm{Var}(\beta^r) \subseteq \mathcal{V}^r$ by structured decomposability.

Procedure and Complexity

We can recursively decompose $\mathrm{Cov}[W_{f_\alpha}, W_{f_\beta}]$ into the covariances and expectations of the WMCs of child nodes with Lemmas 8–10. The base cases of the recursion, e.g., the case where both are literals with the same Boolean variable, can be resolved using the input (co)variances $\sigma^2_{P_x}, \sigma^2_{N_x}, \sigma_{P_x N_x}$ of weights. Also, we pre-compute $\mathrm{E}[W_{f_\gamma}]$ for every node γ in st-d-DNNFs α, β. Since this procedure is identical to a standard one for computing WMC with st-d-DNNFs, the details of computing expectations are in the full version.

Algorithm 1 is the proposed covariance computation algorithm. This algorithm outputs a pair $(\mathsf{anc}, \mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}])$, where $\mathcal{V} = \mathrm{Var}(\mathsf{anc})$. We cache the output for $\mathrm{Cov}[\alpha, \beta]$ in $\mathsf{c}[\alpha, \beta]$ once computed. $\mathsf{e}[\gamma]$ stores a pair of $\mathsf{v} = \mathsf{d}(\gamma)$ and $\mathrm{E}[W^{\mathrm{Var}(\mathsf{v})}_{f_\gamma}]$. As stated above, these can be pre-computed with a standard WMC algorithm; we defer the details to the full version. Lines 3 and 4 deal with the base cases and lines 5–10 use Lemma 8. Lines 11–16 deal with the remaining base cases involving literals. Lines 19 and 20 use Lemma 9 and lines 21–31 use Lemma 10. To ensure that f_β can be decomposed into $f'_\beta \wedge f''_\beta$ as in Lemma 10, α, β are swapped in line 18, if needed.

We must care about the variable set during the computation. For this purpose, we implement two auxiliary functions ADJEXP and ADJCOV. ADJEXP receives vnode w and $\mathsf{e}[\alpha']$, where $\mathrm{Var}(\mathsf{d}(\alpha')) \subseteq \mathrm{Var}(\mathsf{w})$, and returns $\mathrm{E}[W^{\mathrm{Var}(\mathsf{w})}_{f_{\alpha'}}]$. ADJCOV receives vnode w, the output of $\mathtt{COV}(\alpha', \beta')$, $\mathsf{e}[\alpha']$, and $\mathsf{e}[\beta']$, where $\mathrm{Var}(\mathsf{LCA}(\mathsf{d}(\alpha'), \mathsf{d}(\beta'))) \subseteq \mathrm{Var}(\mathsf{w})$, and returns $\mathrm{Cov}[W^{\mathrm{Var}(\mathsf{w})}_{f_{\alpha'}}, W^{\mathrm{Var}(\mathsf{w})}_{f_{\beta'}}]$. Using these functions, we adjust the variable sets. With a preprocessing taking $O(|\mathcal{V}|^2)$ time, these functions can be computed in constant time; see the full version. The correctness of Algorithm 1, i.e., that $\mathtt{COV}(\alpha, \beta)$ returns $\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}]$, follows from the fact that cases (I), (II), and (III) are comprehensive and recursive decomposition follows Lemmas 8–10 for each case. We now move to the proof of Theorem 7.

Proof of Theorem 7. Preprocessing requires $O(|\mathcal{V}|^2)$ time, and computing expectations takes $O(|\alpha| + |\beta|)$ time. Computing the LCA (line 2) needs $O(1)$ time with a data structure that is built in $O(|\mathcal{V}|)$ time (Bender and Farach-Colton 2000). Thus, other than recursion, $\mathtt{COV}(\alpha', \beta')$ requires at most $O(k_{\alpha'} k_{\beta'})$ time, where $k_{\alpha'}, k_{\beta'}$ are the number of child nodes of α', β'. Since the answer is cached in $\mathsf{c}[\alpha', \beta']$ once $\mathtt{COV}(\alpha', \beta')$ is computed, the overall complexity of $\mathtt{COV}(\alpha, \beta)$ is bounded by $O(|\alpha||\beta|)$. □

We finally give a brief note on the assumption that two st-d-DNNFs share the same vtree in solving **CVC** query. Such an assumption is also imposed on some queries that take multiple st-d-DNNFs as an input (Pipatsrisawat and Darwiche 2008); e.g., sentential entailment and bounded conjunction defined in (Darwiche and Marquis 2002). Although we do not prove the tractability of **CVC** for the case where two st-d-DNNFs do not respect the same vtree, we believe it is intractable because st-d-DNNFs do not admit polytime sentential entailment unless P=NP when they do not share the vtree. Note that, for **VC**, such an assumption is not imposed because we have a single input st-d-DNNF for **VC**.

Algorithm 1: $\mathtt{COV}(\alpha, \beta)$: computing $\mathrm{Cov}[W_{f_\alpha}, W_{f_\beta}]$

Input : Two st-d-DNNFs α, β respecting the same vtree
Output : Pair of $\mathsf{anc} = \mathsf{LCA}(\mathsf{d}(\alpha), \mathsf{d}(\beta))$ and $\mathrm{Cov}[W^{\mathcal{V}}_{f_\alpha}, W^{\mathcal{V}}_{f_\beta}]$ $(\mathcal{V} = \mathrm{Var}(\mathsf{anc}))$

1 **if** $\mathsf{c}[\alpha, \beta] \neq$ *null* **then return** $\mathsf{c}[\alpha, \beta]$ *// Cache for* $\mathtt{COV}(\alpha, \beta)$
2 $\mathsf{anc} \leftarrow \mathsf{LCA}(\mathsf{d}(\alpha), \mathsf{d}(\beta))$
3 **if** $\alpha =$ *false or* $\beta =$ *false* **then return** $(\mathsf{anc}, 0)$
4 **if** $\alpha =$ *true and* $\beta =$ *true* **then return** $(\bot, 0)$
5 **if** $\mathsf{anc} \neq \mathsf{d}(\alpha)$ *and* $\mathsf{anc} \neq \mathsf{d}(\beta)$ **then** *// Let* $\mathrm{Var}(\alpha) \subseteq \mathrm{Var}(\mathsf{anc}^l)$ *and* $\mathrm{Var}(\beta) \subseteq \mathrm{Var}(\mathsf{anc}^r)$
6 $\quad \mathsf{el} \leftarrow \mathtt{ADJEXP}(\mathsf{anc}^l, \mathsf{e}[\alpha]) \cdot \mathtt{ADJEXP}(\mathsf{anc}^l, \mathsf{e}[\mathit{true}])$
7 $\quad \mathsf{er} \leftarrow \mathtt{ADJEXP}(\mathsf{anc}^r, \mathsf{e}[\mathit{true}]) \cdot \mathtt{ADJEXP}(\mathsf{anc}^r, \mathsf{e}[\beta])$
8 $\quad \mathsf{cl} \leftarrow \mathtt{ADJCOV}(\mathsf{anc}^l, \mathtt{COV}(\alpha, \mathit{true}), \mathsf{e}[\alpha], \mathsf{e}[\mathit{true}])$
9 $\quad \mathsf{cr} \leftarrow \mathtt{ADJCOV}(\mathsf{anc}^r, \mathtt{COV}(\mathit{true}, \beta), \mathsf{e}[\mathit{true}], \mathsf{e}[\beta])$
10 $\quad \mathsf{r} \leftarrow \mathsf{cl} \cdot \mathsf{cr} + \mathsf{cl} \cdot \mathsf{er} + \mathsf{el} \cdot \mathsf{cr}$ *// Eq. (4)*
11 **else if** α, β *are both leaf nodes* **then**
12 $\quad$ **if** $(\alpha, \beta) = (\mathit{true}, x), (x, \mathit{true})$ **then** $\mathsf{r} \leftarrow \sigma^2_{P_x} + \sigma_{P_x N_x}$
13 $\quad$ **else if** $(\alpha, \beta) = (\mathit{true}, \neg x), (\neg x, \mathit{true})$ **then** $\mathsf{r} \leftarrow \sigma^2_{N_x} + \sigma_{P_x N_x}$
14 $\quad$ **else if** $\alpha = \beta = x$ **then** $\mathsf{r} \leftarrow \sigma^2_{P_x}$
15 $\quad$ **else if** $\alpha = \beta = \neg x$ **then** $\mathsf{r} \leftarrow \sigma^2_{N_x}$
16 $\quad$ **else** $\mathsf{r} \leftarrow \sigma_{P_x N_x}$ *// $(\alpha, \beta) = (x, \neg x), (\neg x, x)$*
17 **else**
18 $\quad$ Swap α, β if (i) $\mathsf{anc} = \mathsf{d}(\beta) \neq \mathsf{d}(\alpha)$ or (ii) $\mathsf{d}(\alpha) = \mathsf{d}(\beta)$ and only β is a $\vee$-node
19 $\quad$ **if** α *is a* $\vee$*-node* **then** *// $\alpha_1, \ldots, \alpha_k$: the child nodes of α*
20 $\quad\quad \mathsf{r} \leftarrow \sum_{j=1}^{k} \mathtt{ADJCOV}(\mathsf{anc}, \mathtt{COV}(\alpha_i, \beta), \mathsf{e}[\alpha_i], \mathsf{e}[\beta])$ *// Eq. (5)*
21 $\quad$ **else** *// α is a $\wedge$-node*
22 $\quad\quad \alpha^l, \alpha^r \leftarrow$ (child nodes of α) s.t. $\mathrm{Var}(\alpha^l) \subseteq \mathrm{Var}(\mathsf{anc}^l)$ and $\mathrm{Var}(\alpha^r) \subseteq \mathrm{Var}(\mathsf{anc}^r)$
23 $\quad\quad$ **if** $\mathrm{Var}(\mathsf{d}(\beta)) \subseteq \mathrm{Var}(\mathsf{anc}^l)$ **then** $\beta^l \leftarrow \beta$, $\beta^r \leftarrow \mathit{true}$
24 $\quad\quad$ **else if** $\mathrm{Var}(\mathsf{d}(\beta)) \subseteq \mathrm{Var}(\mathsf{anc}^r)$ **then** $\beta^l \leftarrow \mathit{true}$, $\beta^r \leftarrow \beta$
25 $\quad\quad$ **else** *// $\mathsf{d}(\alpha) = \mathsf{d}(\beta)$; thus β is a $\wedge$-node due to line 18*
26 $\quad\quad\quad \beta^l, \beta^r \leftarrow$ (child nodes of β) s.t. $\mathrm{Var}(\beta^l) \subseteq \mathrm{Var}(\mathsf{anc}^l)$ and $\mathrm{Var}(\beta^r) \subseteq \mathrm{Var}(\mathsf{anc}^r)$
27 $\quad\quad \mathsf{el} \leftarrow \mathtt{ADJEXP}(\mathsf{anc}^l, \mathsf{e}[\alpha^l]) \cdot \mathtt{ADJEXP}(\mathsf{anc}^l, \mathsf{e}[\beta^l])$
28 $\quad\quad \mathsf{er} \leftarrow \mathtt{ADJEXP}(\mathsf{anc}^r, \mathsf{e}[\alpha^r]) \cdot \mathtt{ADJEXP}(\mathsf{anc}^r, \mathsf{e}[\beta^r])$
29 $\quad\quad \mathsf{cl} \leftarrow \mathtt{ADJCOV}(\mathsf{anc}^l, \mathtt{COV}(\alpha^l, \beta^l), \mathsf{e}[\alpha^l], \mathsf{e}[\beta^l])$
30 $\quad\quad \mathsf{cr} \leftarrow \mathtt{ADJCOV}(\mathsf{anc}^r, \mathtt{COV}(\alpha^r, \beta^r), \mathsf{e}[\alpha^r], \mathsf{e}[\beta^r])$
31 $\quad\quad \mathsf{r} \leftarrow \mathsf{cl} \cdot \mathsf{cr} + \mathsf{cl} \cdot \mathsf{er} + \mathsf{el} \cdot \mathsf{cr}$ *// Eq. (6)*
32 **return** $\mathsf{c}[\alpha, \beta] \leftarrow (\mathsf{anc}, \mathsf{r})$

Intractability Results

The goal of this section is to prove the following result.

Theorem 11. *When f, g are given as st-DNNFs, d-DNNFs, or FBDDs, **CVC** is intractable, i.e., it cannot be solved in polynomial time unless P=NP. When f is given as an st-DNNF, a d-DNNF, or an FBDD, **VC** is intractable.*

We prove this by first introducing some queries from the knowledge compilation map (Darwiche and Marquis 2002).

Problem 12. Given Boolean function f, *model counting* query **CT** computes the number of models of f, i.e., $|\mathcal{A}^{\mathcal{V}}_f|$. Given Boolean functions f, g, *sentential entailment* query **SE** asks whether $f \models g$, i.e., $\mathcal{A}^{\mathcal{V}}_f \subseteq \mathcal{A}^{\mathcal{V}}_g$.

We now show a polynomial time reduction from the **CT** and **SE** queries to the **VC** and **CVC** queries. It is known that **CT** is intractable when f is given as an st-DNNF (Pipatsrisawat and Darwiche 2008). It is also known that **SE** is intractable when f, g are given as d-DNNFs or FBDDs (Darwiche and Marquis 2002). Thus, the existence of the above reduction indicates the intractability of **VC** and **CVC** queries with such representations.

Let $n \coloneqq |\mathcal{V}|$. The key lemmas are as follows.

Lemma 13. *Let $\mu_{P_x} = \mu_{N_x} = 1$, $\sigma^2_{P_x} = \sigma^2_{N_x} = 3$, and $\sigma_{P_x N_x} = -1$ for every variable $x \in \mathcal{V}$. Then, for any assignment a of $\mathcal{V}$, $\mathrm{V}[W_a] = 4^n - 1$. In addition, for any assignments a, b $(a \neq b)$ of $\mathcal{V}$, $\mathrm{Cov}[W_a, W_b] = -1$.*

Proof. We can decompose $W^{\mathcal{V}}_a = Q^a_x W^{\mathcal{V}\setminus\{x\}}_a$, where $Q^a_x = P_x$ if $a(x) = \mathit{true}$ or $Q^a_x = N_x$ otherwise. Similar decomposition can be derived for $W^{\mathcal{V}}_b$. Thus, by (3),

$$\begin{aligned}&\mathrm{Cov}[W^{\mathcal{V}}_a, W^{\mathcal{V}}_b] = \\ &(\mathrm{Cov}[Q^a_x, Q^b_x] + \mathrm{E}[Q^a_x]\mathrm{E}[Q^b_x])\mathrm{Cov}[W^{\mathcal{V}\setminus\{x\}}_a, W^{\mathcal{V}\setminus\{x\}}_b] \\ &+ \mathrm{Cov}[Q^a_x, Q^b_x]\mathrm{E}[W^{\mathcal{V}\setminus\{v\}}_a]\mathrm{E}[W^{\mathcal{V}\setminus\{x\}}_b]. \qquad (7)\end{aligned}$$

Here, $\mathrm{E}[Q^a_x] = \mathrm{E}[Q^b_x] = \mathrm{E}[W^{\mathcal{V}\setminus\{x\}}_a] = \mathrm{E}[W^{\mathcal{V}\setminus\{x\}}_b] = 1$ because $\mu_{P_x} = \mu_{N_x} = 1$ for any $x \in \mathcal{V}$. When $a = b$, (7) becomes $\mathrm{V}[W^{\mathcal{V}}_a] = 4\mathrm{V}[W^{\mathcal{V}\setminus\{x\}}_a] + 3$ because $\mathrm{V}[Q^a_x] = 3$ regardless of whether Q^a_x equals P_x or N_x. By applying this formula recursively for every Boolean variable x, we have $\mathrm{V}[W^{\mathcal{V}}_a] = 3(1 + 4 + \cdots + 4^{n-1}) = 4^n - 1$. When $a \neq b$, by letting x be a Boolean variable satisfying $a(x) \neq b(x)$, we have $\mathrm{Cov}[Q^a_x, Q^b_x] = \mathrm{Cov}[P_x, N_x] = -1$. By substituting $\mathrm{Cov}[Q^a_x, Q^b_x]$ in (7), $\mathrm{Cov}[W^{\mathcal{V}}_a, W^{\mathcal{V}}_b] = -1$. □

Lemma 14. *Let f, g be Boolean functions on variable set $\mathcal{V}$. Then, under identical settings of expectations and (co)variances as Lemma 13, $|\mathcal{A}^{\mathcal{V}}_f| = \lceil \mathrm{V}[W^{\mathcal{V}}_f]/(4^n - 1) \rceil$ and $|\mathcal{A}^{\mathcal{V}}_{f\wedge g}| = \lceil \mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g]/(4^n - 1) \rceil$.*

Proof. By recursively applying (2), we have

$$\begin{aligned}\mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g] &= \textstyle\sum_{a\in\mathcal{A}^{\mathcal{V}}_f}\sum_{b\in\mathcal{A}^{\mathcal{V}}_g}\mathrm{Cov}[W^{\mathcal{V}}_a, W^{\mathcal{V}}_b] \\ &= \textstyle\sum_{a\in\mathcal{A}^{\mathcal{V}}_f\cap\mathcal{A}^{\mathcal{V}}_g}\mathrm{V}[W^{\mathcal{V}}_a] \\ &\quad + \textstyle\sum_{(a,b)\in\mathcal{A}^{\mathcal{V}}_f\times\mathcal{A}^{\mathcal{V}}_g : a\neq b}\mathrm{Cov}[W^{\mathcal{V}}_a, W^{\mathcal{V}}_b],\end{aligned}$$

The first term is $(4^n - 1)|\mathcal{A}^{\mathcal{V}}_{f\wedge g}|$ and the second term is $-|\{(a, b) \in \mathcal{A}^{\mathcal{V}}_f \times \mathcal{A}^{\mathcal{V}}_g \mid a \neq b\}|$ by Lemma 13. The latter can be lower bounded by $-|\{(a, b) \in \mathcal{A}^{\mathcal{V}}_{\mathit{true}} \times \mathcal{A}^{\mathcal{V}}_{\mathit{true}} \mid a \neq b\}| = -2^n(2^n - 1) = -(4^n - 2^n) > -(4^n - 1)$. Thus, we have

$$(4^n - 1)(|\mathcal{A}^{\mathcal{V}}_{f\wedge g}| - 1) < \mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g] \leq (4^n - 1)|\mathcal{A}^{\mathcal{V}}_{f\wedge g}|,$$

indicating $|\mathcal{A}^{\mathcal{V}}_{f\wedge g}| = \lceil \mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g]/(4^n - 1) \rceil$. By setting $g = f$, we have $|\mathcal{A}^{\mathcal{V}}_f| = \lceil \mathrm{V}[W^{\mathcal{V}}_f]/(4^n - 1) \rceil$. □

Lemma 14 indicates the reductions from **CT** to **VC** and from **SE** to **CVC**. Given Boolean function f, we can answer **CT** by computing $\mathrm{V}[W_f]$ under the settings of Lemma 13.

Query	st-d-DNNF	st-DNNF	d-DNNF	FBDD
VC	✓(Thm. 7)	○ (Thm. 11)	○ (Thm. 11)	○ (Thm. 11)
CVC	✓* (Thm. 7)	○ (Thm. 11)	○ (Thm. 11)	○ (Thm. 11)
CT	✓	○	✓	✓
SE	✓*	○	○	○

*Assuming that two st-d-DNNFs respect the same vtree.

Table 1: Tractability of queries. ✓ indicates that this query can be answered in polynomial time in the sizes of NNFs, and ○ indicates that it cannot be answered in polynomial time unless P=NP.

Given Boolean functions f, g, we can answer **SE** as follows. We compute $\mathrm{V}[W_f]$ and $\mathrm{Cov}[W_f, W_g]$ under the settings of Lemma 13 and obtain $|\mathcal{A}^{\mathcal{V}}_f|$ and $|\mathcal{A}^{\mathcal{V}}_{f\wedge g}|$ by Lemma 14. Then $f \models g$ if and only if $|\mathcal{A}^{\mathcal{V}}_f| = |\mathcal{A}^{\mathcal{V}}_{f\wedge g}|$. Combined with the intractability results of **CT** and **SE**, the intractability of **VC** for st-DNNFs and that of **CVC** for d-DNNFs and FBDDs follow. Note that **CVC** is also intractable for st-DNNFs since **VC** can be reduced to **CVC** with $g = f$.

The remaining is to show the intractability of **VC** for d-DNNFs and FBDDs. We use the following lemma.

Lemma 15. *Let f, g be Boolean functions on variable set $\mathcal{V}$, $z \notin \mathcal{V}$ be a Boolean variable, and $h = (z \wedge f) \vee (\neg z \wedge g)$. We set $\mu_{P_z} = \mu_{N_z} = 1$ and $\sigma^2_{P_z} = \sigma^2_{N_z} = -\sigma_{P_z N_z} = 3$, and N_x and P_x $(x \in \mathcal{V})$ have identical settings as Lemma 13. Then, $\mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g] = \mathrm{V}[W^{\mathcal{V}}_f] + \mathrm{V}[W^{\mathcal{V}}_g] - \mathrm{V}[W^{\mathcal{V}\cup\{z\}}_h]/4 + 3(\mathrm{E}[W^{\mathcal{V}}_f] - \mathrm{E}[W^{\mathcal{V}}_g])^2/4$.*

Proof. Since $W^{\mathcal{V}\cup\{z\}}_h = P_z W^{\mathcal{V}}_f + N_z W^{\mathcal{V}}_g$, $\mathrm{V}[W^{\mathcal{V}\cup\{z\}}_h] = \mathrm{V}[P_z W^{\mathcal{V}}_f] + \mathrm{V}[N_z W^{\mathcal{V}}_g] + 2\mathrm{Cov}[P_z W^{\mathcal{V}}_f, N_z W^{\mathcal{V}}_g]$ by (2). We have $\mathrm{V}[P_z W^{\mathcal{V}}_f] = 4\mathrm{V}[W^{\mathcal{V}}_f] + 3(\mathrm{E}[W^{\mathcal{V}}_f])^2$, $\mathrm{V}[N_z W^{\mathcal{V}}_g] = 4\mathrm{V}[W^{\mathcal{V}}_g] + 3\mathrm{E}[W^{\mathcal{V}}_g]^2$, and $\mathrm{Cov}[P_z W^{\mathcal{V}}_f, N_z W^{\mathcal{V}}_g] = -2\mathrm{Cov}[W^{\mathcal{V}}_f, W^{\mathcal{V}}_g] - 3\mathrm{E}[W^{\mathcal{V}}_f]\mathrm{E}[W^{\mathcal{V}}_g]$ by using (3). Substituting each term leads to the equation in Lemma 15. □

Lemma 15 demonstrates that we can obtain $\mathrm{Cov}[W_f, W_g]$ by computing the variances of W_f,W_g, and W_h and the expectations of W_f and W_g. If f, g are given as FBDDs, we can easily construct the FBDD of h by simply adding decision node $\vee$ at the root: $(z \wedge f) \vee (\neg z \wedge g)$, which does not break the restrictions of FBDDs. This construction is also valid for d-DNNFs when f, g are given as d-DNNFs. Thus, **CVC** can be answered by solving **VC** when f, g are given as d-DNNFs or FBDDs; they also admit the expectation computation because it amounts to ordinal WMC. This indicates the intractability of **VC** for d-DNNFs and FBDDs, proving Theorem 11. Table 1 summarizes the tractability of the queries.

Application for Bayesian Networks

We introduce an application that considers the uncertainty in the inference of Bayesian networks. A discrete Bayesian network represents a joint distribution over categorical random variables $\mathcal{X} \coloneqq \{X_1, \ldots, X_n\}$, where the range of X_i is

$\{x_{i1}, \ldots, x_{ik_i}\}$. Each random variable X_i has parents $\mathcal{U}_i \subseteq \mathcal{X}$, and the dependence structure is assumed to be acyclic. The joint probability that X_i takes value x_i for $i = 1, \ldots, n$ is described as $\Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i|\mathbf{u}_i)$, where $\mathbf{u}_i \coloneqq \{u_{i1}, \ldots, u_{i\ell_i}\}$ is the set of values of parent variables $\mathcal{U}_i$. A marginal inference for a Bayesian network is to compute the marginal probability of *partial assignments* that are the values of some random variables.

In an ordinal setting, every conditional distribution is characterized by a set of fixed parameters. More specifically, distribution $\Pr(x_i|\mathbf{u}_i)$ for given parent values $\mathbf{u}_i$ is a Bernoulli (for a binary-valued X_i) or a categorical (for a general X_i) distribution with fixed parameters. Since these parameters are often learned from data, they may have uncertainty. As explained in the Introduction, a Bayesian statistical approach to model the uncertainty is to introduce distributions, e.g., beta or Dirichlet distributions, for the parameters and regard the marginal probability as a random variable. With our method, we can compute the variance of the marginal probability. Our main result here is as follows.

Theorem 16. *Given a Bayesian network with a constant treewidth, we can compute the variance of a marginal probability in polynomial time.*

This theorem can be proved by using the existing WMC encoding of Bayesian networks (Chavira and Darwiche 2008) and then compute the marginal probability's variance by our proposed algorithm. Since the method for the general case is complicated, as it requires the slight relaxation of the independence assumption, we defer the details of the general method and the proof of Theorem 16 to the full version. Instead, we here explain a simpler method for the case where every random variable is binary-valued, i.e., $k_i = 2$ for every X_i. We use the encoding of Sang, Bearne, and Kautz (2005), referred to as ENC2 by Chavira and Darwiche (2008).

Before incorporating uncertainty, we explain the method for ordinal marginal inference. For every random variable X_i, we prepare indicator variables $\lambda_{x_{i1}}, \lambda_{x_{i2}}$; $\lambda_{x_{ij}} = \mathit{true}$ when $X_i = x_{ij}$. We set the following clauses:

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}}, \quad \neg\lambda_{x_{i1}} \vee \neg\lambda_{x_{i2}}. \tag{8}$$

We also prepare parameter variable $\rho_{x_{i1}|\mathbf{u}_i}$ for every pattern on parent values $\mathbf{u}_i$ and set the following clauses:

$$\begin{aligned} \lambda_{u_{i1}} \wedge \cdots \wedge \lambda_{u_{i\ell_i}} \wedge \rho_{x_{i1}|\mathbf{u}_i} &\implies \lambda_{x_{i1}}, \\ \lambda_{u_{i1}} \wedge \cdots \wedge \lambda_{u_{i\ell_i}} \wedge \neg\rho_{x_{i1}|\mathbf{u}_i} &\implies \lambda_{x_{i2}}. \end{aligned} \tag{9}$$

In addition, we set $P_\rho = 1 - N_\rho = \Pr(x_{i1}|\mathbf{u}_i)$ for every $\rho = \rho_{x_{i1}|\mathbf{u}_i}$. Let f be a Boolean function that is a conjunction of all the clauses in (8) and (9). Then the marginal probability given partial assignment $\mathbf{x}$ can be obtained in two ways: (i) To prepare Boolean function $g_{\mathbf{x}}$, which is a conjunction of indicator variables corresponding to $\mathbf{x}$, and compute the WMC of $f \wedge g_{\mathbf{x}}$ with $P_\lambda = N_\lambda = 1$ for every $\lambda = \lambda_{ij}$. (ii) To set $P_\lambda = 0$ for $\lambda = \lambda_{x_{ij}}$ such that $\mathbf{x}$ contains $x_{ij'}$ $(j' \neq j)$, set all the other weights of the indicator variables to 1, and then compute the WMC of f. For example, when $\mathbf{x} = \{x_{11}, x_{32}\}$, method (i) prepares $g_{\mathbf{x}} = \lambda_{x_{11}} \wedge \lambda_{x_{32}}$, while method (ii) sets $P_\lambda = 0$ for $\lambda = \lambda_{x_{12}}, \lambda_{x_{31}}$.

To incorporate uncertainty, we regard P_x and N_x as random variables. Expectations μ_{P_x}, μ_{N_x} are set to the original weight values. Since the weights of indicator variables $\lambda = \lambda_{x_{ij}}$ are determined regardless of the probability values, we set $\sigma^2_{P_\lambda} = \sigma^2_{N_\lambda} = \sigma_{P_\lambda N_\lambda} = 0$. For parameter variables $\rho = \rho_{x_{i1}|\mathbf{u}_i}$, we consider variance $\sigma^2_{x_{i1}|\mathbf{u}_i}$ of probability parameter $\Pr(x_{i1}|\mathbf{u}_i)$. Since $P_\rho + N_\rho = 1$, we set $\sigma^2_{P_\rho} = \sigma^2_{N_\rho} = -\sigma_{P_\rho N_\rho} = \sigma^2_{x_{i1}|\mathbf{u}_i}$. By computing the variance of the WMC under this setting, we can compute the variance of the marginal. Note that we here assume that each parameter is independent of the others because (P_x, N_x) and (P_y, N_y) $(x \neq y)$ are independent, which is justified by the widely-adopted *parameter independence* assumption (Spiegelhalter and Lauritzen 1990) when parameters are learned from data; see also (Heckerman 2008). In the following experiments, we empirically validate the tractability of the proposed algorithm and showcase the usage of variance computation with this encoding.

We finally mention that the variance of the *conditional* probability of a Bayesian network can be *approximately* obtained by using the **CVC** query. The conditional probability of $\mathbf{x}$ given condition $\mathbf{c}$ equals $W_{h'}/W_h$ with $h' = f \wedge g_{\mathbf{x}} \wedge g_{\mathbf{c}}$ and $h = f \wedge g_{\mathbf{c}}$ using method (i), i.e., to prepare Boolean function $g_{\mathbf{x}}$. Although we cannot precisely determine $\mathrm{V}[W_{h'}/W_h]$, by Taylor expansion (van Kempen and van Vliet 2000) we have $\mathrm{V}[W_{h'}/W_h] \approx \mathrm{V}[W_{h'}]/\{\mathrm{E}[W_h]\}^2 - 2\mathrm{Cov}[W_{h'}, W_h]\mathrm{E}[W_{h'}]/\{\mathrm{E}[W_h]\}^3 + \mathrm{V}[W_h]\{\mathrm{E}[W_{h'}]\}^2/\{\mathrm{E}[W_h]\}^4$.

Experiments

In our experiment, we first confirmed the practical tractability of the proposed algorithm for st-d-DNNFs with an application for computing the variance of the marginal of Bayesian networks. We used Bayesian networks from bnRep (Leonelli 2025), which collects networks from recent academic literature in various areas. We retrieved all 70 binary Bayesian networks from bnRep. The number of random variables ranges from 3 to 122. We derived the CNF of f with ENC2 by Ace v3.0 (http://reasoning.cs.ucla.edu/ace/) and compiled every CNF into a SDD, which is a subset of an st-d-DNNF, by the SDD package (Choi and Darwiche 2013). Given $p = \Pr(x_{i1}|\mathbf{u}_i)$ in the data, we set $\sigma^2_{x_{i1}|\mathbf{u}_i} = p(1-p)/\theta$, which virtually considered $\Pr(x_{i1}|\mathbf{u}_i)$ follows $\mathrm{Beta}((\theta-1)p, (\theta-1)(1-p))$. We set $\theta = 10$; note that the value of θ does not affect the computational time. For parameters where $p = 0$ or 1, we set $\sigma^2_{x_{i1}|\mathbf{u}_i} = 0$ because $\Pr(x_{i1}|\mathbf{u}_i)$ should take a value within $[0, 1]$, and thus the variance must be 0 when the expectation is 0 or 1. We chose one random variable from a Bayesian network as a partial assignment and computed the variance of the marginal by method (ii). Note that the choices of the partial assignment and the expectations and (co)variances of weights do not affect the computational time since the size of the SDD representing f remains unchanged. The proposed method was implemented in C++ and compiled with g++-11.4.0. All experiments were performed on a single thread of a Linux server with AMD EPYC 7763 CPU and 2048 GB RAM;

Name	#rv	|SDD|	Compile (s)	Variance (s)
projectmanagement	26	3888	0.500	0.025
GDIpathway2	28	2755	0.784	0.021
grounding	36	3397	2.387	0.017
engines	12	1804	0.240	0.011
windturbine	122	2043	1.380	0.009

Table 2: Top-5 (out of 70) time-consuming networks. "#rv" is the number of random variables. "|SDD|" is the size of compiled SDD. "Compile" and "Variance" indicate the time required to compile SDD and compute variance.

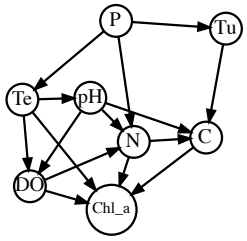

Figure 3: The "algalactivity2" network.

Parameter	Variance
$\mathrm{DO}\|\mathrm{pH}_0, \mathrm{Te}_0$	0.002887
$\mathrm{Chl\_a}\|\mathrm{C}_1, \mathrm{DO}_0, \mathrm{N}_0, \mathrm{Te}_1$	0.003532
$\mathrm{Te}\|\mathrm{P}_0$	0.003554
$\mathrm{pH}\|\mathrm{Te}_0$	0.003592
$\mathrm{Chl\_a}\|\mathrm{C}_1, \mathrm{DO}_1, \mathrm{N}_1, \mathrm{Te}_0$	0.003674
(none)	0.003904

Table 3: Variance of $\Pr(\mathrm{Chl\_a} = 0)$ when one parameter's variance is reduced to one-tenth. Top-5 parameters in ascending order of variance was exhibited.

note that we used less than 4 GB of memory during the experiments. We reported the average consumed time for SDD compilation and variance computation over 10 runs for each network.

As a result, after the SDD was compiled, the variance computation only took 0.025 sec at maximum, recorded for "projectmanagement" network whose SDD size was 3,888. Even if the SDD compilation is added to the computational time, it took only 10 sec to process the most time-consuming "propellant" network. Table 2 shows the top-5 networks in descending order of the variance computation times. This indicates the practical tractability of the proposed algorithm. More detailed results can be found in the full version.

Next, we showcased the usage of variance computations with the "algalactivity2" network from bnRep (Fig. 3). Each random variable in Fig. 3 is valued either 0 or 1. With the same setting as the above experiment, the mean and variance of $\Pr(\mathrm{Chl\_a} = 0)$ are computed as 0.5281 and 0.003904. Since the standard deviation is 0.06248, the variance will affect the decision-making that depends on, e.g., whether $\Pr(\mathrm{Chl\_a} = 0) \leq 0.55$. To conduct more robust decision-making, we want to reduce the variance of the marginal. One approach is to decrease the variance of the parameters by collecting more observations (data). However, since it is costly to collect observations corresponding to all the parameters, we want to find parameters that are effective for reducing the variance of the marginal. Thus, we additionally demonstrated how much the variance of this marginal is decreased by reducing the variance of one parameter to one-tenth. We conducted the above demonstration for each of the 43 parameters in the network. Table 3 shows the top-5 parameters in the reduction of the variance of $\Pr(\mathrm{Chl\_a} = 0)$. In other words, it shows the top-5 parameters having greater impact on the variance of the marginal. Here, RV_j stands for $\mathrm{RV} = j$; e.g., $\mathrm{DO}|\mathrm{pH}_0, \mathrm{Te}_0$ denotes parameter $\Pr(\mathrm{DO}|\mathrm{pH} = 0, \mathrm{Te} = 0)$. It is notable that, among the 43 parameters, those having greater impact on the variance of $\Pr(\mathrm{Chl\_a} = 0)$ are not only the conditional probabilities of Chl_a but also those of the other random variables. We realized that reducing the variance of the parameters in Table 3 efficiently decreased the variance of the marginal. These results suggest us that we cannot reveal what parameters have greater impact on the variance of the inference result without actually computing the variance. We give more examples of the variance computation in the full version.

Conclusion

We defined a query for computing the WMC's variance. We proved that this query is tractable for st-d-DNNFs and intractable for st-DNNFs, d-DNNFs, and FBDDs; the tractability was shown by presenting an algorithm to solve the query. We also showed an application for quantifying the uncertainty in the inference on Bayesian networks. Future directions include the computation of more involved WMC statistics, such as higher-order moments. We should also investigate the tractability of **VC** and **CVC** queries for emerging classes of representations, such as and-sum circuits (Onaka et al. 2025).

References

Bender, M. A.; and Farach-Colton, M. 2000. The LCA problem revisited. In *Proc. of the 4th Latin American Symposium on Theoretical Informatics (LATIN'00)*, 88–94.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8): 677–691.

Chavira, M.; and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6–7): 772–799.

Choi, A.; and Darwiche, A. 2013. Dynamic minimization of sentential decision diagrams. In *Proc. of the 27th AAAI Conference on Artificial Intelligence (AAAI'13)*, 187–194.

Choi, A.; Kisa, D.; and Darwiche, A. 2013. Compiling probabilistic graphical models using sentential decision diagrams. In *Proc. of the 12th European Conference on Symbolic and Quantiative Approaches to Resoning with Uncertainty (ECSQARU'13)*, 121–132.

Choi, Y.; Vergari, A.; and Van den Broeck, G. 2020. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, UCLA.

Cozman, F. G. 2000. Credal networks. *Artificial Intelligence*, 120(2): 199–233.

Darwiche, A. 2001. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2): 11–34.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *Proc. of the 22nd international Joint Conference on Artifical Intelligence (IJCAI'11)*, 819–826.

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1): 229–264.

De Campos, C. P.; and Cozman, F. G. 2005. The inferential complexity of Bayesian and credal networks. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, 1313–1318.

Dilkas, P.; and Belle, V. 2021. Weighted model counting with conditional weights for Bayesian networks. In *Proc. of the 37th Conference on Uncertainty in Artificial Intelligence (UAI'21)*, 386–396.

Duenas-Osorio, L.; Meel, K. S.; Paredes, R.; and Vardi, M. Y. 2017. Counting-based reliability estimation for power-transmission grids. In *Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 4488–4494.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and Raedt, L. D. 2011. Inference in probabilistic logic programs using weighted CNF's. In *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence (UAI'11)*, 211–220.

Gergov, J.; and Meinel, C. 1994. Efficient Boolean manipulation with OBDD's can be extended to FBDD's. *IEEE Transactions on Computers*, 43(10): 1197–1209.

Hardy, G.; Lucet, C.; and Limnios, N. 2007. K-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56(3): 506–515.

Heckerman, D. 2008. *A Tutorial on Learning with Bayesian Networks*, 33–82. Springer Berlin Heidelberg.

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

Illner, P. 2025. New compilation languages based on restricted weak decomposability. In *Proc. of the 39th AAAI Conference on Artificial Intelligence (AAAI'25)*, 14987–14996.

Kiesel, R.; Totis, P.; and Kimmig, A. 2022. Efficient knowledge compilation beyond weighted model counting. *Theory and Practice of Logic Programming*, 22(4): 505–522.

Leonelli, M. 2025. bnRep: A repository of Bayesian networks from the academic literature. *Neurocomputing*, 624: 129502.

Nakamura, K.; Inoue, T.; Nishino, M.; and Yasuda, N. 2022. Impact of link availability uncertainty on network reliability: analyses with variances. In *Proc. of the 2022 IEEE International Conference on Communications (ICC'22)*, 2713–2719.

Onaka, R.; Nakamura, K.; Nishino, M.; and Yasuda, N. 2025. An and-sum circuit with signed edges that is more succinct than SDD. In *Proc. of the 39th AAAI Conference on Artificial Intelligence (AAAI'25)*, 15100–15108.

Pipatsrisawat, K.; and Darwiche, A. 2008. New compilation languages based on structured decomposability. In *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, 517–522.

Sang, T.; Bearne, P.; and Kautz, H. 2005. Performing Bayesian inference by weighted model counting. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI'05)*, 475–481.

Spiegelhalter, D. J.; and Lauritzen, S. L. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5): 579–605.

van Kempen, G.; and van Vliet, L. 2000. Mean and variance of ratio estimators used in fluorescence ratio imaging. *Cytometry Part A - The Journal of Quantitative Cell Science*, 39(4): 300–305.

Wang, B.; Mauá, D. D.; den Broeck, G. V.; and Choi, Y. 2024. A compositional atlas for algebraic circuits. In *Proc. of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS'24)*.

Yu, Z.; Trapp, M.; and Kersting, K. 2023. Characteristic circuits. In *Proc. of the 37th Annual Conference on Neural Information Processing Systems (NeurIPS'23)*.

Zhang, H.; Juba, B.; and Van den Broeck, G. 2021. Probabilistic generating circuits. In *Proc. of the 38th International Conference on Machine Learning (ICML'21)*, 12447–12457.