

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

# Weighted Model Counting Without Parameter Variables

**Citation for published version:**
Dilkas, P & Belle, V 2021, Weighted Model Counting Without Parameter Variables. in Theory and Applications of Satisfiability Testing – SAT 2021. Lecture Notes in Computer Science, vol. 12831, Springer, pp. 134-151, 24th International Conference on Theory and Applications of Satisfiability Testing, Barcelona, Spain, 5/07/21. https://doi.org/10.1007/978-3-030-80223-3_10

**Digital Object Identifier (DOI):**
10.1007/978-3-030-80223-3_10

**Link:**
Link to publication record in Edinburgh Research Explorer

**Document Version:**
Peer reviewed version

**Published In:**
Theory and Applications of Satisfiability Testing – SAT 2021

**General rights**
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

**Take down policy**
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.



Download date: 28. Mar. 2024

# Weighted Model Counting Without Parameter Variables

Paulius Dilkas[1(✉)] and Vaishak Belle[1,2]

[1] University of Edinburgh, Edinburgh, UK
p.dilkas@sms.ed.ac.uk, vaishak@ed.ac.uk
[2] Alan Turing Institute, London, UK

**Abstract.** Weighted model counting (WMC) is a powerful computational technique for a variety of problems, especially commonly used for probabilistic inference. However, the standard definition of WMC that puts weights on literals often necessitates WMC encodings to include additional variables and clauses just so each weight can be attached to a literal. This paper complements previous work by considering WMC instances in their full generality and using recent state-of-the-art WMC techniques based on pseudo-Boolean function manipulation, competitive with the more traditional WMC algorithms based on knowledge compilation and backtracking search. We present an algorithm that transforms WMC instances into a format based on pseudo-Boolean functions while eliminating around 43 % of variables on average across various Bayesian network encodings. Moreover, we identify sufficient conditions for such a variable removal to be possible. Our experiments show significant improvement in WMC-based Bayesian network inference, outperforming the current state of the art.

**Keywords:** Weighted model counting · Probabilistic inference · Bayesian networks

## 1 Introduction

Weighted model counting (WMC), i.e., a generalisation of propositional model counting that assigns weights to literals and computes the total weight of all models of a propositional formula [11], has emerged as a powerful computational framework for problems in many domains, e.g., probabilistic graphical models such as Bayesian networks and Markov networks [3, 8, 9, 15, 32], neuro-symbolic artificial intelligence [37], probabilistic programs [26], and probabilistic logic programs [21]. It has been extended to support continuous variables [6], infinite domains [4], first-order logic [24, 36], and arbitrary semirings [5, 27]. However, as the definition of WMC puts weights on literals, additional variables often need to be added for the sole purpose of holding a weight [3, 8, 9, 15, 32]. This can be particularly detrimental to WMC algorithms that rely on variable ordering heuristics.

One approach to this problem considers weighted clauses and probabilistic semantics based on Markov networks [22]. However, with a new representation

comes the need to invent new encodings and inference algorithms. Our work is similar in spirit in that it introduces a new representation for computational problems but can reuse recent WMC algorithms based on pseudo-Boolean function manipulation, namely, ADDMC [19] and DPMC [20]. Furthermore, we identify sufficient conditions for transforming a WMC instance into our new format. As many WMC inference algorithms [16, 29] work by compilation to tractable representations such as arithmetic circuits, deterministic, decomposable negation normal form [14], and sentential decision diagrams (SDDs) [17], another way to avoid parameter variables could be via direct compilation to a more convenient representation. Direct compilation of Bayesian networks to SDDs has been investigated [13]. However, SDDs only support weights on literals, and so are not expressive enough to avoid the issue. To the best of the authors' knowledge, neither approach [13, 22] has a publicly available implementation.

In this work, we introduce a way to transform WMC problems into a new format based on pseudo-Boolean functions—*pseudo-Boolean projection* (PBP). We formally show that every WMC problem instance has a corresponding PBP instance and identify conditions under which this transformation can remove parameter variables. Four out of the five known WMC encodings for Bayesian networks [3, 8, 9, 15, 32] can indeed be simplified in this manner. We are able to eliminate 43 % of variables on average and up to 99 % on some instances. This transformation enables two encodings that were previously incompatible with most WMC algorithms (due to using a different definition of WMC [8, 9]) to be run with ADDMC and DPMC and results in a significant performance boost for one other encoding, making it about three times faster than the state of the art. Finally, our theoretical contributions result in a convenient algebraic way of reasoning about two-valued pseudo-Boolean functions and position WMC encodings on common ground, identifying their key properties and assumptions.

## 2 Weighted Model Counting

We begin with an overview of some notation and terminology. We use $\wedge$, $\vee$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$ to denote conjunction, disjunction, negation, material implication, and material biconditional, respectively. Throughout the paper, we use set-theoretic notation for many concepts in logic. A *clause* is a set of literals that are part of an implicit disjunction. Similarly, a *formula* in CNF is a set of clauses that are part of an implicit conjunction. We identify a *model* with a set of variables that correspond to the positive literals in the model (and all other variables are the negative literals of the model). We can then define the *cardinality* of a model as the cardinality of this set. For example, let $\phi = (\neg a \vee b) \wedge a$ be a propositional formula over variables $a$ and $b$. Then an equivalent set-theoretic representation of $\phi$ is $\{\{\neg a, b\}, \{a\}\}$. Any subset of $\{a, b\}$ is an interpretation of $\phi$, e.g., $\{a, b\}$ is a model of $\phi$ (written $\{a, b\} \models \phi$) of cardinality two, while $\emptyset$ is an interpretation but not a model. We can now formally define WMC.

**Definition 1 (WMC).** *A* WMC instance *is a tuple* $(\phi, X_I, X_P, w)$*, where* $X_I$ *is the set of* indicator variables*,* $X_P$ *is the set of* parameter variables *(with*

*$X_I \cap X_P = \emptyset$), $\phi$ is a propositional formula in CNF over $X_I \cup X_P$, and $w\colon X_I \cup X_P \cup \{\neg x \mid x \in X_I \cup X_P\} \to \mathbb{R}$ is a* weight function *such that $w(x) = w(\neg x) = 1$ for all $x \in X_I$. The* answer *of the instance is $\sum_{Y \models \phi} \prod_{Y \models l} w(l)$.*

That is, the answer to a WMC instance is the sum of the weights of all models of $\phi$, where the weight of a model is defined as the product of the weights of all (positive and negative) literals in it. Our definition of WMC is largely based on the standard definition [11], but explicitly partitions variables into indicator and parameter variables. In practice, we identify this partition in one of two ways. If an encoding is generated by Ace[3], then variable types are explicitly identified in a file generated alongside the encoding. Otherwise, we take $X_I$ to be the set of all variables $x$ such that $w(x) = w(\neg x) = 1$. Next, we formally define a variation of the WMC problem used by some of the Bayesian network encodings [8, 9].

**Definition 2.** *Let $\phi$ be a formula over a set of variables $X$. Then $Y \subseteq X$ is a* minimum-cardinality model *of $\phi$ if $Y \models \phi$ and $|Y| \leq |Z|$ for all $Z \models \phi$.*

**Definition 3 (Minimum-Cardinality WMC).** *A* minimum-cardinality WMC *instance consists of the same tuple as a WMC instance, but its* answer *is defined to be $\sum_{Y \models \phi,\ |Y|=k} \prod_{Y \models l} w(l)$ (where $k = \min_{Y \models \phi} |Y|$) if $\phi$ is satisfiable, and zero otherwise.*

*Example 1.* Let $\phi = (x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee p) \wedge (\neg y \vee q) \wedge x$, $X_I = \{x, y\}$, $X_P = \{p, q\}$, $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$. Then $\phi$ has two models: $\{x, p\}$ and $\{x, p, q\}$ with weights 0.2 and $0.2 \times 0.8 = 0.16$, respectively. The WMC answer is then $0.2 + 0.16 = 0.36$, and the minimum-cardinality WMC answer is 0.2.

### 2.1 Bayesian Network Encodings

A *Bayesian network* is a directed acyclic graph with random variables as vertices and edges as conditional dependencies. As is common in related literature [15, 32], we assume that each variable has a finite number of values. We call a Bayesian network *binary* if every variable has two values. If all variables have finite numbers of values, the probability function associated with each variable $v$ can be represented as a *conditional probability table* (CPT), i.e., a table with a row for each combination of values that $v$ and its parent vertices can take. Each row then also has a *probability*, i.e., a number in $[0, 1]$.

WMC is a well-established technique for Bayesian network inference, particularly effective on networks where most variables have only a few possible values [15]. Many ways of encoding a Bayesian network into a WMC instance have been proposed. We will refer to them based on the initials of the authors and the year of publication. Darwiche was the first to suggest the `d02` [15] encoding that, in many ways, remains the foundation behind most other encodings.

[3] Ace [11] implements most of the Bayesian network encodings and can also be used for compilation (and thus inference). It is available at http://reasoning.cs.ucla.edu/ace/.

He also introduced the distinction between *indicator* and *parameter variables*; the former represent variable-value pairs in the Bayesian network, while the latter are associated with probabilities in the CPTs. The encoding `sbk05` [32] is the only encoding that deviates from this arrangement: for each variable in the Bayesian network, one indicator variable acts simultaneously as a parameter variable. Chavira and Darwiche propose `cd05` [8] where they shift from WMC to minimum-cardinality WMC because that allows the encoding to have fewer variables and clauses. In particular, they propose a way to use the same parameter variable to represent all probabilities in a CPT that are equal and keep only clauses that 'imply' parameter variables (i.e., omit clauses where a parameter variable implies indicator variables).[4] In their next encoding, `cd06` [9], the same authors optimise the aforementioned implication clauses, choosing the smallest sufficient selection of indicator variables. A decade later, Bart et al. present `bklm16` [3] that improves upon `cd06` in two ways. First, they optimise the number of indicator variables used per Bayesian network variable from a linear to a logarithmic amount. Second, they introduce a scaling factor that can 'absorb' one probability per Bayesian network variable. However, for this work, we choose to disable the latter improvement since this scaling factor is often small enough to be indistinguishable from zero without the use of arbitrary precision arithmetic, making it completely unusable on realistic instances. Indeed, the reader is free to check that even a small Bayesian network with seven mutually independent binary variables, 0.1 and 0.9 probabilities each, is already big enough for the scaling factor to be exactly equal to zero (as produced by the `bklm16` encoder[5]). We suspect that this issue was not identified during the original set of experiments because the authors never looked at numerical answers.

*Example 2.* Let $\mathcal{B}$ be a Bayesian network with one variable $X$ which has two values $x_1$ and $x_2$ with probabilities $\Pr(X = x_1) = 0.2$ and $\Pr(X = x_2) = 0.8$. Let $x, y$ be indicator variables, and $p, q$ be parameter variables. Then Example 1 is both the `cd05` and the `cd06` encoding of $\mathcal{B}$. The `bklm16` encoding is $(x \Rightarrow p) \wedge (\neg x \Rightarrow q) \wedge x$ with $w(p) = w(\neg q) = 0.2$, and $w(\neg p) = w(q) = 0.8$. And the `d02` encoding is $(\neg x \Rightarrow p) \wedge (p \Rightarrow \neg x) \wedge (x \Rightarrow q) \wedge (q \Rightarrow x) \wedge \neg x$ with $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$. Note how all other encodings have fewer clauses than `d02`. While `cd05` and `cd06` require minimum-cardinality WMC to make this work, `bklm16` achieves the same thing by adjusting weights.[6]

## 3 Pseudo-Boolean Functions

In this work, we propose a more expressive representation for WMC based on pseudo-Boolean functions. A *pseudo-Boolean function* is a function of the form $\{0, 1\}^n \to \mathbb{R}$ [7]. Equivalently, let $X$ denote a set with $n$ elements (we will refer to

[4] Example 2 demonstrates what we mean by implication clauses.
[5] http://www.cril.univ-artois.fr/kc/bn2cnf.html
[6] Note that since `cd05` and `cd06` are minimum-cardinality WMC encodings, they are not supported by most WMC algorithms.

them as *variables*), and $2^X$ denote its powerset. Then a pseudo-Boolean function can have $2^X$ as its domain (then it is also known as a *set function*).

Pseudo-Boolean functions, most commonly represented as algebraic decision diagrams (ADDs) [2] (although a tensor-based approach has also been suggested [18, 20]), have seen extensive use in value iteration for Markov decision processes [25], both exact and approximate Bayesian network inference [10, 23], and sum-product network [30] to Bayesian network conversion [38]. ADDs have been extended to compactly represent additive and multiplicative structure [35], sentences in first-order logic [33], and continuous variables [34], the last of which was also applied to weighted model integration, i.e., the WMC extension for continuous variables [6, 28].

Since two-valued pseudo-Boolean functions will be used extensively henceforth, we introduce some new notation. For any propositional formula $\phi$ over $X$ and $p, q \in \mathbb{R}$, let $[\phi]_q^p \colon 2^X \to \mathbb{R}$ be the pseudo-Boolean function defined as

$$[\phi]_q^p(Y) \coloneqq \begin{cases} p & \text{if } Y \models \phi \\ q & \text{otherwise} \end{cases}$$

for any $Y \subseteq X$. Next, we define some useful operations on pseudo-Boolean functions. The definitions of multiplication and projection are equivalent to those in previous work [19, 20].

**Definition 4 (Operations).** *Let $f, g \colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions, $x, y \in X$, $Y = \{y_i\}_{i=1}^n \subseteq X$, and $r \in \mathbb{R}$. Operations such as addition and multiplication are defined pointwise, i.e., $(f+g)(Y) \coloneqq f(Y)+g(Y)$, and likewise for multiplication. Note that properties such as associativity and commutativity are inherited from $\mathbb{R}$. By regarding a real number as a constant pseudo-Boolean function, we can reuse the same definitions to define* scalar *operations as $(r + f)(Y) = r + f(Y)$, and $(r \cdot f)(Y) = r \cdot f(Y)$.*

Restrictions $f|_{x=0}, f|_{x=1} \colon 2^X \to \mathbb{R}$ *of $f$ are defined as $f|_{x=0}(Y) \coloneqq f(Y \setminus \{x\})$, and $f|_{x=1}(Y) \coloneqq f(Y \cup \{x\})$ for all $Y \subseteq X$.*

Projection $\exists_x$ *is an endomorphism $\exists_x \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ defined as $\exists_x f \coloneqq f|_{x=1} + f|_{x=0}$. Since projection is commutative (i.e., $\exists_x \exists_y f = \exists_y \exists_x f$) [19, 20], we can define $\exists_Y \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ as $\exists_Y \coloneqq \exists_{y_1} \exists_{y_2} \ldots \exists_{y_n}$. Throughout the paper, projection is assumed to have the lowest precedence (e.g., $\exists_x fg = \exists_x (fg)$).*

Below we list some properties of the operations on pseudo-Boolean functions discussed in this section that can be conveniently represented using our syntax. The proofs of all these properties follow directly from the definitions.

**Proposition 1 (Basic Properties).** *For any propositional formulas $\phi$ and $\psi$, and $a, b, c, d \in \mathbb{R}$,*

- $[\phi]_b^a = [\neg\phi]_a^b$;
- $c + [\phi]_b^a = [\phi]_{b+c}^{a+c}$;
- $c \cdot [\phi]_b^a = [\phi]_{bc}^{ac}$;
- $[\phi]_b^a \cdot [\phi]_d^c = [\phi]_{bd}^{ac}$;

– $[\phi]_0^1 \cdot [\psi]_0^1 = [\phi \wedge \psi]_0^1$.

*And for any pair of pseudo-Boolean functions* $f, g\colon 2^X \to \mathbb{R}$ *and* $x \in X$, $(fg)|_{x=i} = f|_{x=i} \cdot g|_{x=i}$ *for* $i = 0, 1$.

*Remark 1.* Note that our definitions of binary operations assumed equal domains. For convenience, we can assume domains to shrink whenever a function is independent of some of the variables (i.e., $f|_{x=0} = f|_{x=1}$) and expand for binary operations to make the domains of both functions equal. For instance, let $[x]_0^1, [\neg x]_0^1\colon 2^{\{x\}} \to \mathbb{R}$ and $[y]_0^1\colon 2^{\{y\}} \to \mathbb{R}$ be pseudo-Boolean functions. Then $[x]_0^1 \cdot [\neg x]_0^1$ has $2^\emptyset$ as its domain. To multiply $[x]_0^1$ and $[y]_0^1$, we expand $[x]_0^1$ into $\left([x]_0^1\right)' : 2^{\{x,y\}} \to \mathbb{R}$ which is defined as $\left([x]_0^1\right)'(Z) \coloneqq [x]_0^1(Z \cap \{x\})$ for all $Z \subseteq \{x, y\}$ (and equivalently for $[y]_0^1$).

## 4 Pseudo-Boolean Projection

We introduce a new type of computational problem called *pseudo-Boolean projection* based on two-valued pseudo-Boolean functions. While the same computational framework can handle any pseudo-Boolean functions, two-valued functions are particularly convenient because DPMC can be easily adapted to use them as input. Since we will only encounter functions of the form $[\phi]_b^a$, where $\phi$ is a conjunction of literals, we can represent it in text as `w` $\langle\phi\rangle$ `a b` where $\langle\phi\rangle$ is a representation of $\phi$ analogous to the representation of a clause in the DIMACS CNF format.

**Definition 5 (PBP Instance).** *A PBP instance is a tuple* $(F, X, \omega)$*, where* $X$ *is the set of variables,* $F$ *is a set of two-valued pseudo-Boolean functions* $2^X \to \mathbb{R}$*, and* $\omega \in \mathbb{R}$ *is the scaling factor.*[7] *Its* answer *is* $\omega \cdot \left(\exists_X \prod_{f \in F} f\right)(\emptyset)$.

### 4.1 From WMC to PBP

In this section, we describe an algorithm for transforming WMC instances to the PBP format while removing all parameter variables. We chose to transform existing encodings instead of creating a new one to reuse already-existing techniques for encoding each CPT to its minimal logical representation such as prime implicants and limited forms of resolution [3, 8, 9]. The transformation algorithm works on four out of the five Bayesian network encodings: `bklm16` [3], `cd05` [8], `cd06` [9], and `d02` [15]. There is no obvious way to adjust it to work with `sbk05` because the roles of indicator and parameter variables overlap [32].

The algorithm is based on several observations that will be made more precise in Section 4.2. First, all weights except for $\{w(p) \mid p \in X_P\}$ are redundant

[7] Adding scaling factor $\omega$ to the definition allows us to remove clauses that consist entirely of a single parameter variable. The idea of extracting some of the structure of the WMC instance into an external multiplicative factor was loosely inspired by the `bklm16` encoding, where it is used to subsume the most commonly occurring probability of each CPT [3].

**Algorithm 1:** WMC to PBP transformation

**Data:** WMC (or minimum-cardinality WMC) instance $(\phi, X_I, X_P, w)$
**Result:** PBP instance $(F, X_I, \omega)$

$F \leftarrow \emptyset$;
$\omega \leftarrow 1$;
**foreach** *clause* $c \in \phi$ **do**
  **if** $c \cap X_P = \{p\}$ *for some* $p$ **and** $w(p) \neq 1$ **then**
    **if** $|c| = 1$ **then**
      $\omega \leftarrow \omega \times w(p)$;
    **else**
      $F \leftarrow F \cup \left\{ \left[ \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} \right\}$;
  **else if** $\{p \mid \neg p \in c\} \cap X_P = \emptyset$ **then**
    $F \leftarrow F \cup \{[c]_0^1\}$;
**foreach** $v \in X_I$ *such that* $\{[v]_1^p, [\neg v]_1^q\} \subseteq F$ *for some* $p$ *and* $q$ **do**
  $F \leftarrow F \setminus \{[v]_1^p, [\neg v]_1^q\} \cup \{[v]_q^p\}$;

as they either duplicate an already-defined weight or are equal to one. Second, each clause has at most one parameter variable. Third, if the parameter variable is negated, we can ignore the clause (this idea first appears in the `cd05` paper [8]). Note that while we formulate our algorithm as a sequel to the WMC encoding procedure primarily because the implementations of Bayesian network WMC encodings are all closed-source, as all transformations in the algorithm are local, it can be efficiently incorporated into a WMC encoding algorithm with no slowdown.

The algorithm is listed as Algorithm 1. The main part of the algorithms is the first loop that iterates over clauses. If a clause consists of a single parameter variable, we incorporate it into $\omega$. If a clause is of the form $\alpha \Rightarrow p$, where $p \in X_P$, and $\alpha$ is a conjunction of literals over $X_I$, we transform it into a pseudo-Boolean function $[\alpha]_1^{w(p)}$. If a clause $c \in \phi$ has no parameter variables, we reformulate it into a pseudo-Boolean function $[c]_0^1$. Finally, clauses with negative parameter literals are omitted.

As all 'weighted' pseudo-Boolean functions produced by the first loop are of the form $[\alpha]_1^p$ (for some $p \in \mathbb{R}$ and formula $\alpha$), the second loop merges two functions into one whenever $\alpha$ is a literal. Note that taking into account the order in which clauses are typically generated by encoding algorithms allows us to do this in linear time (i.e., the two mergeable functions will be generated one after the other).

### 4.2 Correctness Proofs

In this section, we outline key conditions that a (WMC or minimum-cardinality WMC) encoding has to satisfy for Algorithm 1 to output an equivalent PBP instance. We divide the correctness proof into two theorems: Theorem 2 for

WMC encodings (i.e., `bklm16` and `d02`) and Theorem 3 for minimum-cardinality WMC encodings (i.e., `cd05` and `cd06`). We begin by listing some properties of pseudo-Boolean functions and establishing a canonical transformation from WMC to PBP.

**Theorem 1 (Early Projection [19, 20]).** *Let $X$ and $Y$ be sets of variables. For all pseudo-Boolean functions $f\colon 2^X \to \mathbb{R}$ and $g\colon 2^Y \to \mathbb{R}$, if $x \in X \setminus Y$, then $\exists_x (f \cdot g) = (\exists_x f) \cdot g$.*

**Lemma 1.** *For any pseudo-Boolean function $f\colon 2^X \to \mathbb{R}$, we have that $(\exists_X f)(\emptyset) = \sum_{Y \subseteq X} f(Y)$.*

*Proof.* If $X = \{x\}$, then $(\exists_x f)(\emptyset) = (f|_{x=1} + f|_{x=0})(\emptyset) = f|_{x=1}(\emptyset) + f|_{x=0}(\emptyset) = \sum_{Y \subseteq \{x\}} f(Y)$. This easily extends to $|X| > 1$ by the definition of projection on sets of variables.

**Proposition 2.** *Let $(\phi, X_I, X_P, w)$ be a WMC instance. Then*

$$\left(\left\{[c]_0^1 \mid c \in \phi\right\} \cup \left\{[x]_{w(\neg x)}^{w(x)} \;\middle|\; x \in X_I \cup X_P\right\}, X_I \cup X_P, 1\right) \tag{1}$$

*is a PBP instance with the same answer (as defined in Definitions 1 and 5).*

*Proof.* Let $f = \prod_{c \in \phi} [c]_0^1$, and $g = \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}$. Then the WMC answer of (1) is $(\exists_{X_I \cup X_P} fg)(\emptyset) = \sum_{Y \subseteq X_I \cup X_P} (fg)(Y) = \sum_{Y \subseteq X_I \cup X_P} f(Y)g(Y)$ by Lemma 1. Note that

$$f(Y) = \begin{cases} 1 & \text{if } Y \models \phi, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad g(Y) = \prod_{Y \models l} w(l),$$

which means that $\sum_{Y \subseteq X_I \cup X_P} f(Y)g(Y) = \sum_{Y \models \phi} \prod_{Y \models l} w(l)$ as required.

**Theorem 2 (Correctness for WMC).** *Algorithm 1, when given a WMC instance $(\phi, X_I, X_P, w)$, returns a PBP instance with the same answer (as defined in Definitions 1 and 5), provided* either *of the two conditions is satisfied:*

1. *for all $p \in X_P$, there is a non-empty family of literals $(l_i)_{i=1}^n$ such that*
   (a) *$w(\neg p) = 1$,*
   (b) *$l_i \in X_I$ or $\neg l_i \in X_I$ for all $i = 1, \ldots, n$,*
   (c) *and $\{c \in \phi \mid p \in c \text{ or } \neg p \in c\} = \{p \vee \bigvee_{i=1}^n \neg l_i\} \cup \{l_i \vee \neg p \mid i = 1, \ldots, n\}$;*

2. *or for all $p \in X_P$,*
   (a) *$w(p) + w(\neg p) = 1$,*
   (b) *for any clause $c \in \phi$, $|c \cap X_P| \leq 1$,*
   (c) *there is no clause $c \in \phi$ such that $\neg p \in c$,*
   (d) *if $\{p\} \in \phi$, then there is no clause $c \in \phi$ such that $c \neq \{p\}$ and $p \in c$,*
   (e) *and for any $c, d \in \phi$ such that $c \neq d$, $p \in c$ and $p \in d$, $\bigwedge_{l \in c \setminus \{p\}} \neg l \wedge \bigwedge_{l \in d \setminus \{p\}} \neg l$ is false.*

Condition 1 (for `d02`) simply states that each parameter variable is equivalent to a conjunction of indicator literals. Condition 2 is for encodings that have implications rather than equivalences associated with parameter variables (which, in this case, is `bklm16`). It ensures that each clause has at most one positive parameter literal and no negative ones, and that at most one implication clause per any parameter variable $p \in X_P$ can 'force $p$ to be positive'.

*Proof.* By Proposition 2,

$$\left(\left\{[c]_0^1 \mid c \in \phi\right\} \cup \left\{[x]_{w(\neg x)}^{w(x)} \;\middle|\; x \in X_I \cup X_P\right\}, X_I \cup X_P, 1\right) \tag{2}$$

is a PBP instance with the same answer as the given WMC instance. By Definition 5, its answer is $\left(\exists_{X_I \cup X_P} \left(\prod_{c\in\phi}[c]_0^1\right) \prod_{x \in X_I \cup X_P}[x]_{w(\neg x)}^{w(x)}\right)(\emptyset)$. Since both Conditions 1 and 2 ensure that each clause in $\phi$ has at most one parameter variable, we can partition $\phi$ into $\phi_* \coloneqq \{c \in \phi \mid \texttt{Vars}(c) \cap X_P = \emptyset\}$ and $\phi_p \coloneqq \{c \in \phi \mid \texttt{Vars}(c) \cap X_P = \{p\}\}$ for all $p \in X_P$. We can then use Theorem 1 to reorder the answer into $\left(\exists_{X_I} \left(\prod_{x\in X_I}[x]_{w(\neg x)}^{w(x)}\right) \left(\prod_{c\in\phi_*}[c]_0^1\right) \prod_{p\in X_P} \exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c\in\phi_p}[c]_0^1\right)(\emptyset)$.

Let us first consider how the unfinished WMC instance $(F, X_I, \omega)$ after the loop on Lines 3 to 10 differs from (2). Note that Algorithm 1 leaves each $c \in \phi_*$ unchanged, i.e., adds $[c]_0^1$ to $F$. We can then fix an arbitrary $p \in X_P$ and let $F_p$ be the set of functions added to $F$ as a replacement of $\phi_p$. It is sufficient to show that

$$\omega \prod_{f\in F_p} f = \exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c\in\phi_p} [c]_0^1. \tag{3}$$

Note that under Condition 1, $\bigwedge_{c\in\phi_p} c \equiv p \Leftrightarrow \bigwedge_{i=1}^n l_i$ for some family of indicator variable literals $(l_i)_{i=1}^n$. Thus, $\exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c\in\phi_p}[c]_0^1 = \exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1$. If $w(p) = 1$, then

$$\exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1 = \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1\Bigg|_{p=1} + \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1\Bigg|_{p=0}. \tag{4}$$

Since for any input, $\bigwedge_{i=1}^n l_i$ is either true or false, exactly one of the two summands in Eq. (4) will be equal to one, and the other will be equal to zero, and so $\left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1\Big|_{p=1} + \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1\Big|_{p=0} = 1$, where 1 is a pseudo-Boolean function that always returns one. On the other side of Eq. (3), since $F_p = \emptyset$, and $\omega$ is unchanged, we get $\omega \prod_{f\in F_p} f = 1$, and so Eq. (3) is satisfied under Condition 1 when $w(p) = 1$.

If $w(p) \neq 1$, then $F_p = \left\{\left[\bigwedge_{i=1}^n l_i\right]_1^{w(p)}\right\}$, and $\omega = 1$, and so we want to show that $\left[\bigwedge_{i=1}^n l_i\right]_1^{w(p)} = \exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1$. Indeed, $\exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i\right]_0^1 = w(p) \cdot \left[\bigwedge_{i=1}^n l_i\right]_0^1 + \left[\bigwedge_{i=1}^n l_i\right]_1^0 = \left[\bigwedge_{i=1}^n l_i\right]_1^{w(p)}$. This finishes the proof of the correctness of the first loop under Condition 1.

Now let us assume Condition 2. We still want to prove Eq. (3). If $w(p) = 1$, then $F_p = \emptyset$, and $\omega = 1$, and so the left-hand side of Eq. (3) is equal to one. Then the right-hand side is $\exists_p [p]_0^1 \prod_{c \in \phi_p} [c]_0^1 = \exists_p \left[ p \wedge \bigwedge_{c \in \phi_p} c \right]_0^1 = \exists_p [p]_0^1 = 0 + 1 = 1$ since $p \in c$ for every clause $c \in \phi_p$.

If $w(p) \neq 1$, and $\{p\} \in \phi_p$, then, by Condition 2d, $\phi_p = \{\{p\}\}$, and Algorithm 1 produces $F_p = \emptyset$, and $\omega = w(p)$, and so $\exists_p [p]_{w(\neg p)}^{w(p)} [p]_0^1 = \exists_p [p]_0^{w(p)} = w(p) = \omega \prod_{f \in F_p} f$. The only remaining case is when $w(p) \neq 1$ and $\{p\} \notin \phi_p$. Then $\omega = 1$, and $F_p = \left\{ \left[ \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} \;\middle|\; c \in \phi_p \right\}$, so we need to show that $\prod_{c \in \phi_p} \left[ \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} = \exists_p [p]_{1-w(p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1$. We can rearrange the right-hand side as

$$\begin{aligned}
\exists_p [p]_{1-w(p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1 &= \exists_p [p]_{1-w(p)}^{w(p)} \left[ p \vee \bigwedge_{c \in \phi_p} c \setminus \{p\} \right]_0^1 \\
&= w(p) + (1 - w(p)) \left[ \bigwedge_{c \in \phi_p} c \setminus \{p\} \right]_0^1 \\
&= \left[ \bigwedge_{c \in \phi_p} c \setminus \{p\} \right]_{w(p)}^1 = \left[ \bigvee_{c \in \phi_p} \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)}.
\end{aligned}$$

By Condition 2e, $\bigwedge_{l \in c \setminus \{p\}} \neg l$ can be true for at most one $c \in \phi_p$, and so $\left[ \bigvee_{c \in \phi_p} \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} = \prod_{c \in \phi_p} \left[ \bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)}$ which is exactly what we needed to show. This ends the proof that the first loop of Algorithm 1 preserves the answer under both Condition 1 and Condition 2. Finally, the loop on Lines 11 to 12 of Algorithm 1 replaces $[v]_1^p [\neg v]_1^q$ with $[v]_q^p$ (for some $v \in X_I$ and $p, q \in \mathbb{R}$), but, of course, $[v]_1^p [\neg v]_1^q = [v]_1^p [v]_q^1 = [v]_q^p$, i.e., the answer is unchanged.

**Theorem 3 (Minimum-Cardinality Correctness).** *Let $(\phi, X_I, X_P, w)$ be a minimum-cardinality WMC instance that satisfies Conditions 2b to 2e of Theorem 2 as well as the following:*

1. *for all parameter variables $p \in X_P$, $w(\neg p) = 1$.*
2. *all models of $\{c \in \phi \mid c \cap X_P = \emptyset\}$ (as subsets of $X_I$) have the same cardinality;*
3. *$\min_{Z \subseteq X_P} |Z|$ such that $Y \cup Z \models \phi$ is the same for all $Y \models \{c \in \phi \mid c \cap X_P = \emptyset\}$.*

*Then Algorithm 1, when applied to $(\phi, X_I, X_P, w)$, outputs a PBP instance with the same answer (as defined in Definitions 3 and 5).*

In this case, we have to add some assumptions about the cardinality of models. Condition 2 states that all models of the indicator-only part of the formula have the same cardinality. Bayesian network encodings such as `cd05` and

cd06 satisfy this condition by assigning an indicator variable to each possible variable-value pair and requiring each random variable to be paired with exactly one value. Condition 3 then says that the smallest number of parameter variables needed to turn an indicator-only model into a full model is the same for all indicator-only models. As some ideas duplicate between the proofs of Theorems 2 and 3, the following proof is slightly less explicit and assumes that $\omega = 1$.

*Proof.* Let $(F, X_I, \omega)$ be the tuple returned by Algorithm 1 and note that $F = \left\{[c]_0^1 \mid c \in \phi,\, c \cap X_P = \emptyset\right\} \cup \left\{ \left[\bigwedge_{l \in c \setminus \{p\}} \neg l\right]_1^{w(p)} \;\middle|\; p \in X_P,\, p \in c \in \phi,\, c \neq \{p\} \right\}$. We split the proof into two parts. In the first part, we show that there is a bijection between minimum-cardinality models of $\phi$ and $Y \subseteq X_I$ such that $\left(\prod_{f \in F} f\right)(Y) \neq 0$.[8] Let $Y \subseteq X_I$ and $Z \subseteq X_I \cup X_P$ be related via this bijection. Then in the second part we will show that

$$\prod_{Z \models l} w(l) = \left(\prod_{f \in F} f\right)(Y). \tag{5}$$

On the one hand, if $Z \subseteq X_I \cup X_P$ is a minimum-cardinality model of $\phi$, then $\left(\prod_{f \in F}\right)(Z \cap X_I) \neq 0$ under the given assumptions. On the other hand, if $Y \subseteq X_I$ is such that $\left(\prod_{f \in F}\right)(Y) \neq 0$, then $Y \models \{c \in \phi \mid c \cap X_P = \emptyset\}$. Let $Y \subseteq Z \subseteq X_I \cup X_P$ be the smallest superset of $Y$ such that $Z \models \phi$ (it exists by Condition 2c of Theorem 2). We need to show that $Z$ has minimum cardinality. Let $Y'$ and $Z'$ be defined equivalently to $Y$ and $Z$. We will show that $|Z| = |Z'|$. Note that $|Y| = |Y'|$ by Condition 2, and $|Z \setminus Y| = |Z' \setminus Y'|$ by Condition 3. Combining that with the general property that $|Z| = |Y| + |Z \setminus Y|$ finishes the first part of the proof.

For the second part, let us consider the multiplicative influence of a single parameter variable $p \in X_P$ on Eq. (5). If the left-hand side is multiplied by $w(p)$ (i.e., $p \in Z$), then there must be some clause $c \in \phi$ such that $Z \setminus \{p\} \not\models c$. But then $Y \models \bigwedge_{l \in c \setminus \{p\}} \neg l$, and so the right-hand side is multiplied by $w(p)$ as well (exactly once because of Condition 2e of Theorem 2). This argument works in the other direction as well.

## 5 Experimental Evaluation

We run a set of experiments, comparing all five original Bayesian network encodings (bklm16, cd05, cd06, d02, sbk05) as well as the first four with Algorithm 1 applied afterwards.[9] For each encoding e, we write e++ to denote the combination of encoding a Bayesian network as a WMC instance using e and transforming it into a PBP instance using Algorithm 1. Along with DPMC[10],

[8] For convenience and without loss of generality we assume that $w(p) \neq 0$ for all $p \in X_P$.

[9] Recall that cd05 and cd06 are incompatible with DPMC.

[10] https://github.com/vardigroup/DPMC

we also include WMC algorithms used in the papers that introduce each encoding: Ace for `cd05`, `cd06`, and `d02`; Cachet[11] [31] for `sbk05`; and `c2d`[12] [16] with `query-dnnf`[13] for `bklm16`. Ace is also used to encode Bayesian networks into WMC instances for all encodings except for `bklm16` which uses another encoder mentioned previously. We focus on the following questions:

- Can parameter variable elimination improve inference speed?
- How does DPMC combined with encodings without (and with) parameter variables compare with other WMC algorithms and other encodings?
- Which instances is our approach particularly successful on (compared to other algorithms and encodings and to the same encoding before our transformation)?
- What proportion of variables is typically eliminated?
- Do some encodings benefit from this transformation more than others?

### 5.1 Setup

DPMC is run with tree decomposition-based planning and ADD-based execution—the best-performing combination in the original set of experiments [20]. We use a single iteration of htd [1] to generate approximately optimal tree decompositions—we found that this configuration is efficient enough to handle huge instances, and yet the width of the returned decomposition is unlikely to differ from optimal by more than one or two. We also enabled DPMC's greedy mode. This mode (which was not part of the original paper [20]) optimises the order in which ADDs are multiplied by prioritising those with small representations.

For experimental data, we use Bayesian networks available with Ace and Cachet. We split them into the following groups: − DQMR (390 instances) and − Grid networks (450 instances) as described by Sang et al. [32]; − Mastermind (144 instances) and − Random Blocks (256 instances) by Chavira et al. [12]; − other binary Bayesian networks (50 instances) including Plan Recognition [32], Friends and Smokers, Students and Professors [12], and `tcc4f`; − non-binary classic networks (176 instances): `alarm`, `diabetes`, `hailfinder`, `mildew`, `munin1`–`4`, `pathfinder`, `pigs`, and `water`.

To perform Bayesian network inference with DPMC (or with any other WMC algorithm not based on compilation such as Cachet), one needs to select a probability to compute [20, 31]. If a network comes with an evidence file, we compute the probability of this evidence. Otherwise, let $X$ be the variable last mentioned in the Bayesian network file. If `true` is one of the values of $X$, then we compute $\Pr(X = \mathsf{true})$, otherwise we choose the first-mentioned value of $X$.

The experiments were run on a computing cluster with Intel Xeon E5-2630, Intel Xeon E7-4820, and Intel Xeon Gold 6138 processors with a 1000 s timeout separately on both encoding and inference, and a 32 GiB memory limit.[14]

[11] https://cs.rochester.edu/u/kautz/Cachet/
[12] http://reasoning.cs.ucla.edu/c2d/
[13] http://www.cril.univ-artois.fr/kc/d-DNNF-reasoner.html
[14] Each instance was run on the same processor across all algorithms and encodings.

### 5.2 Results

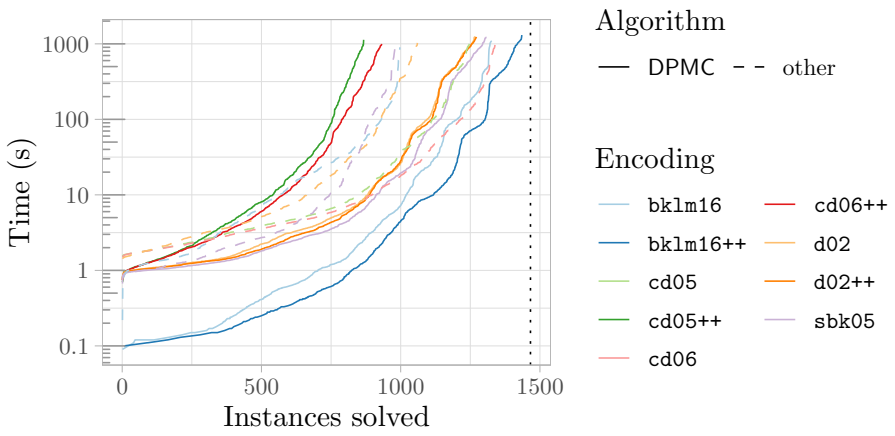



**Fig. 1.** Cactus plot of all algorithm-encoding pairs. The dotted line denotes the total number of instances used.

Figure 1 shows DPMC + `bklm16++` to be the best-performing combination across all time limits up to 1000 s with Ace + `cd06` and DPMC + `bklm16` not far behind. Overall, DPMC+`bklm16++` is 3.35 times faster than DPMC+`bklm16` and 2.96 times faster than Ace+`cd06`. Table 1 further shows that DPMC+`bklm16++` solves almost a hundred more instances than any other combination, and is the fastest in 69.1 % of them.

The scatter plots in Fig. 2 show that how DPMC + `bklm16++` (and perhaps DPMC more generally) compares to Ace + `cd06` depends significantly on the data set: the former is a clear winner on DQMR and Grid instances, while the latter performs well on Mastermind and Random Blocks. Perhaps because the underlying WMC algorithm remains the same, the difference between DPMC + `bklm16` with and without applying Algorithm 1 is quite noisy, i.e, with most instances scattered around the line of equality. However, our transformation does enable DPMC to solve many instances that were previously beyond its reach.

We also record numbers of variables in each encoding before and after applying Algorithm 1. Figure 3 shows a significant reduction in the number of variables. For instance, the median number of variables in instances encoded with `bklm16` was reduced four times: from 1499 to 376. While `bklm16++` results in the overall lowest number of variables, the difference between `bklm16++` and `d02++` seems small. Indeed, the numbers of variables in these two encodings are equal for binary Bayesian networks (i.e., most of our data). Nonetheless, `bklm16++` is still much faster than `d02++` when run with DPMC.

It is also worth noting that there was no observable difference in the width of the project-join tree used by DPMC (which is equivalent to the treewidth of

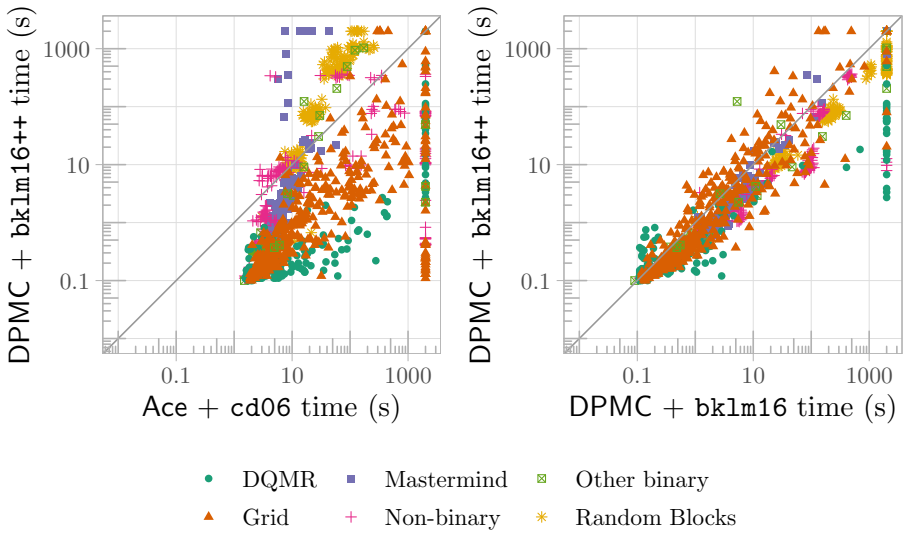



**Fig. 2.** An instance-by-instance comparison between DPMC + `bklm16++` (the best combination according to Fig. 1) and the second and third best-performing combinations: Ace + `cd06` and DPMC + `bklm16`.

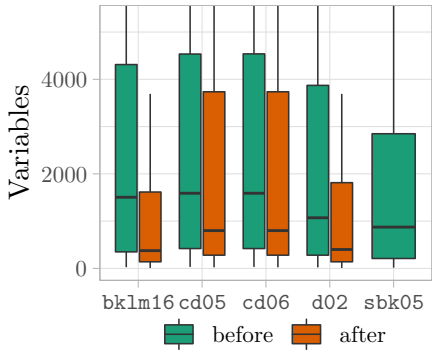



**Fig. 3.** Box plots of the numbers of variables in each encoding across all benchmark instances before and after applying Algorithm 1. Outliers and the top parts of some whiskers are omitted.

**Table 1.** The numbers of instances (out of 1466) that each algorithm and encoding combination solved faster than any other combination and in total.

| Combination | Fastest | Solved |
|---|---|---|
| Ace + `cd05` | 27 | 1247 |
| Ace + `cd06` | 135 | 1340 |
| Ace + `d02` | 56 | 1060 |
| DPMC + `bklm16` | 241 | 1327 |
| DPMC + `bklm16++` | **992** | **1435** |
| DPMC + `cd05++` | 0 | 867 |
| DPMC + `cd06++` | 0 | 932 |
| DPMC + `d02` | 1 | 1267 |
| DPMC + `d02++` | 7 | 1272 |
| DPMC + `sbk05` | 31 | 1308 |
| c2d + `bklm16` | 0 | 997 |
| Cachet + `sbk05` | 49 | 983 |

the primal/Gaifman graph of the input formula [20]) before and after applying Algorithm 1—the observed performance improvement is more likely related to the variable ordering heuristic used by ADDs.[15]

Overall, transforming WMC instances to the PBP format allows us to significantly simplify each instance. This transformation is particularly effective on `bklm16`, allowing it to surpass `cd06` and become the new state of the art. While there is a similarly significant reduction in the number of variables for `d02`, the performance of DPMC + `d02` is virtually unaffected. Finally, while our transformation makes it possible to use `cd05` and `cd06` with DPMC, the two combinations remain inefficient.

## 6 Conclusion

In this paper, we showed how the number of variables in a WMC instance can be significantly reduced by transforming it into a representation based on two-valued pseudo-Boolean functions. In some cases, this led to significant improvements in inference speed, allowing DPMC + `bklm16++` to overtake Ace + `cd06` as the new state of the art WMC technique for Bayesian network inference. Moreover, we identified key properties of Bayesian network encodings that allow for parameter variable removal. However, these properties were rather different for each encoding, and so an interesting question for future work is whether they can be unified into a more abstract and coherent list of conditions.

Bayesian network inference was chosen as the example application of WMC because it is the first and the most studied one [3, 8, 9, 15, 32]. While the distinction between indicator and parameter variables is often not explicitly described in other WMC encodings [21, 26, 37], perhaps in some cases variables could still be partitioned in this way, allowing for not just faster inference with DPMC or ADDMC but also for well-established WMC encoding and inference techniques (such as in the `cd05` and `cd06` papers [8, 9]) to be transferred to other application domains.

**Acknowledgments.** We thank the anonymous reviewers for their helpful comments. The first author was supported by the EPSRC Centre for Doctoral Training in Robotics and Autonomous Systems, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016834/1). The second author was supported by a Royal Society University Research Fellowship. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (http://www.ecdf.ed.ac.uk/).

## References

1. Abseher, M., Musliu, N., Woltran, S.: htd - A free, open-source framework for (customized) tree decompositions and beyond. In: Salvagnin, D., Lombardi, M.

[15] The data on this (along with the implementation of Algorithm 1) is available at https://github.com/dilkas/wmc-without-parameters.

(eds.) Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10335, pp. 376–386. Springer (2017). https://doi.org/10.1007/978-3-319-59776-8_30
2. Bahar, R.I., Frohm, E.A., Gaona, C.M., Hachtel, G.D., Macii, E., Pardo, A., Somenzi, F.: Algebraic decision diagrams and their applications. Formal Methods Syst. Des. **10**(2/3), 171–206 (1997). https://doi.org/10.1023/A:1008699807402
3. Bart, A., Koriche, F., Lagniez, J., Marquis, P.: An improved CNF encoding scheme for probabilistic inference. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 613–621. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-613
4. Belle, V.: Open-universe weighted model counting. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 3701–3708. AAAI Press (2017), http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/15008
5. Belle, V., De Raedt, L.: Semiring programming: A semantic framework for generalized sum product problems. Int. J. Approx. Reason. **126**, 181–201 (2020). https://doi.org/10.1016/j.ijar.2020.08.001
6. Belle, V., Passerini, A., Van den Broeck, G.: Probabilistic inference in hybrid domains by weighted model integration. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 2770–2776. AAAI Press (2015), http://ijcai.org/Abstract/15/392
7. Boros, E., Hammer, P.L.: Pseudo-Boolean optimization. Discret. Appl. Math. **123**(1-3), 155–225 (2002). https://doi.org/10.1016/S0166-218X(01)00341-9
8. Chavira, M., Darwiche, A.: Compiling Bayesian networks with local structure. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005. pp. 1306–1312. Professional Book Center (2005), http://ijcai.org/Proceedings/05/Papers/0931.pdf
9. Chavira, M., Darwiche, A.: Encoding CNFs to empower component analysis. In: Biere, A., Gomes, C.P. (eds.) Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4121, pp. 61–74. Springer (2006). https://doi.org/10.1007/11814948_9
10. Chavira, M., Darwiche, A.: Compiling Bayesian networks using variable elimination. In: Veloso, M.M. (ed.) IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007. pp. 2443–2449 (2007), http://ijcai.org/Proceedings/07/Papers/393.pdf
11. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artif. Intell. **172**(6-7), 772–799 (2008). https://doi.org/10.1016/j.artint.2007.11.002
12. Chavira, M., Darwiche, A., Jaeger, M.: Compiling relational Bayesian networks for exact inference. Int. J. Approx. Reason. **42**(1-2), 4–20 (2006). https://doi.org/10.1016/j.ijar.2005.10.001
13. Choi, A., Kisa, D., Darwiche, A.: Compiling probabilistic graphical models using sentential decision diagrams. In: van der Gaag, L.C. (ed.) Symbolic and

Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7958, pp. 121–132. Springer (2013). https://doi.org/10.1007/978-3-642-39091-3_11
14. Darwiche, A.: On the tractable counting of theory models and its application to truth maintenance and belief revision. J. Appl. Non Class. Logics **11**(1-2), 11–34 (2001). https://doi.org/10.3166/jancl.11.11-34
15. Darwiche, A.: A logical approach to factoring belief networks. In: Fensel, D., Giunchiglia, F., McGuinness, D.L., Williams, M. (eds.) Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002. pp. 409–420. Morgan Kaufmann (2002)
16. Darwiche, A.: New advances in compiling CNF into decomposable negation normal form. In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004. pp. 328–332. IOS Press (2004)
17. Darwiche, A.: SDD: A new canonical representation of propositional knowledge bases. In: Walsh, T. (ed.) IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 819–826. IJCAI/AAAI (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-143
18. Dudek, J.M., Dueñas-Osorio, L., Vardi, M.Y.: Efficient contraction of large tensor networks for weighted model counting through graph decompositions. CoRR **abs/1908.04381** (2019)
19. Dudek, J.M., Phan, V., Vardi, M.Y.: ADDMC: weighted model counting with algebraic decision diagrams. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 1468–1476. AAAI Press (2020), https://aaai.org/ojs/index.php/AAAI/article/view/5505
20. Dudek, J.M., Phan, V.H.N., Vardi, M.Y.: DPMC: weighted model counting by dynamic programming on project-join trees. In: Simonis, H. (ed.) Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12333, pp. 211–230. Springer (2020). https://doi.org/10.1007/978-3-030-58475-7_13
21. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D.S., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted Boolean formulas. Theory Pract. Log. Program. **15**(3), 358–401 (2015). https://doi.org/10.1017/S1471068414000076
22. Gogate, V., Domingos, P.M.: Formula-based probabilistic inference. In: Grünwald, P., Spirtes, P. (eds.) UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010. pp. 210–219. AUAI Press (2010)
23. Gogate, V., Domingos, P.M.: Approximation by quantization. In: Cozman, F.G., Pfeffer, A. (eds.) UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011. pp. 247–255. AUAI Press (2011)

24. Gogate, V., Domingos, P.M.: Probabilistic theorem proving. Commun. ACM **59**(7), 107–115 (2016). https://doi.org/10.1145/2936726
25. Hoey, J., St-Aubin, R., Hu, A.J., Boutilier, C.: SPUDD: stochastic planning using decision diagrams. In: Laskey, K.B., Prade, H. (eds.) UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999. pp. 279–288. Morgan Kaufmann (1999)
26. Holtzen, S., Van den Broeck, G., Millstein, T.D.: Scaling exact inference for discrete probabilistic programs. Proc. ACM Program. Lang. **4**(OOPSLA), 140:1–140:31 (2020). https://doi.org/10.1145/3428208
27. Kimmig, A., Van den Broeck, G., De Raedt, L.: Algebraic model counting. J. Appl. Log. **22**, 46–62 (2017). https://doi.org/10.1016/j.jal.2016.11.031
28. Kolb, S., Mladenov, M., Sanner, S., Belle, V., Kersting, K.: Efficient symbolic integration for probabilistic inference. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 5031–5037. ijcai.org (2018). https://doi.org/10.24963/ijcai.2018/698
29. Oztok, U., Darwiche, A.: A top-down compiler for sentential decision diagrams. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 3141–3148. AAAI Press (2015), http://ijcai.org/Abstract/15/443
30. Poon, H., Domingos, P.M.: Sum-product networks: A new deep architecture. In: Cozman, F.G., Pfeffer, A. (eds.) UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011. pp. 337–346. AUAI Press (2011)
31. Sang, T., Bacchus, F., Beame, P., Kautz, H.A., Pitassi, T.: Combining component caching and clause learning for effective model counting. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004), http://www.satisfiability.org/SAT04/programme/21.pdf
32. Sang, T., Beame, P., Kautz, H.A.: Performing Bayesian inference by weighted model counting. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. pp. 475–482. AAAI Press / The MIT Press (2005), http://www.aaai.org/Library/AAAI/2005/aaai05-075.php
33. Sanner, S., Boutilier, C.: Practical solution techniques for first-order MDPs. Artif. Intell. **173**(5-6), 748–788 (2009). https://doi.org/10.1016/j.artint.2008.11.003
34. Sanner, S., Delgado, K.V., de Barros, L.N.: Symbolic dynamic programming for discrete and continuous state MDPs. In: Cozman, F.G., Pfeffer, A. (eds.) UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011. pp. 643–652. AUAI Press (2011)
35. Sanner, S., McAllester, D.A.: Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005. pp. 1384–1390. Professional Book Center (2005), http://ijcai.org/Proceedings/05/Papers/1439.pdf
36. Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., De Raedt, L.: Lifted probabilistic inference by first-order knowledge compilation. In: Walsh, T. (ed.) IJCAI

2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 2178–2185. IJCAI/AAAI (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-363

37. Xu, J., Zhang, Z., Friedman, T., Liang, Y., Van den Broeck, G.: A semantic loss function for deep learning with symbolic knowledge. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 5498–5507. PMLR (2018), http://proceedings.mlr.press/v80/xu18h.html
38. Zhao, H., Melibari, M., Poupart, P.: On the relationship between sum-product networks and Bayesian networks. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 116–124. JMLR.org (2015), http://proceedings.mlr.press/v37/zhaoc15.html