Component Caching in Hybrid Domains with Piecewise Polynomial Densities

Vaishak Belle*
KU Leuven, Belgium
vaishak@cs.kuleuven.be

Guy Van den Broeck
University of California, Los Angeles
guyvdb@cs.ucla.edu

Andrea Passerini
University of Trento, Italy
passerini@disi.unitn.it

Abstract

Counting the models of a propositional formula is an important problem: for example, it serves as the backbone of probabilistic inference by weighted model counting. A key algorithmic insight is component caching (CC), in which disjoint components of a formula, generated dynamically during a DPLL search, are cached so that they only have to be solved once. In the recent years, driven by SMT technology and probabilistic inference in hybrid domains, there is an increasing interest in counting the models of linear arithmetic sentences. To date, however, solvers for these are block-clause implementations, which are nonviable on large problem instances. In this paper, as a first step in extending CC to hybrid domains, we show how propositional CC systems can be leveraged when limited to piecewise polynomial densities. Our experiments demonstrate a large gap in performance when compared to existing approaches based on a variety of block-clause strategies.

Introduction

Counting the models of a propositional formula, also referred to as #SAT, is an important problem in AI: for example, it serves as the backbone of probabilistic inference by weighted model counting (Chavira and Darwiche 2008). #SAT appears to be more computationally challenging than SAT, in that #SAT is complete for the class #P which is at least as hard as the polynomial-time hierarchy. Thus, the task of counting all assignments calls for novel algorithmic ideas.

One such powerful algorithmic idea is *component caching* (CC), in which disjoint components of a formula, generated dynamically during a DPLL search, are cached so that they only have to be solved once. Investigations into CC revealed that not only is it a clever practical insight for harnessing an existing DPLL trace for model counting, but it also achieves significant time-space tradeoffs (Bacchus, Dalmao, and Pitassi 2009). In a probabilistic context, CC closely matches the competitiveness of approaches such as *recursive conditioning* (Darwiche 2001) and *bucket elimination* (Dechter 1996). Going further, SAT-based techniques can also reason about logical equivalence and deterministic (hard) constraints in a principled way. Overall, CC is the dominant approach for solving #SAT exactly, directly by DPLL search or through knowledge compilation (Darwiche 2004). (For approximate methods, which are the not the focus here, see, for example, (Chakraborty et al. 2014).)

In the recent years, driven by SMT technology (Barrett et al. 2009), diverse applications from verification in hybrid systems (Chistikov, Dimitrova, and Majumdar 2015) to privacy (Fredrikson and Jha 2014) to inference in hybrid graphical models (Belle, Passerini, and Van den Broeck 2015) resort to counting the models of sentences from richer logical languages, especially the first-order fragment of linear arithmetic. This fragment includes sentences of the form $((x + y) > z) \vee p$ and $(0.3 \leq x \leq 8.2)$, that is, the language includes both binary and real-valued (or continuous) variables, and so can concisely capture many complex domains. Existing exact solvers here, however, perform model counting by means of a *block-clause strategy*: in each iteration of the search procedure, if a satisfying interpretation $\alpha \doteq b_1 \wedge \ldots \wedge b_k$ is found, where b_i is a literal, then $\neg\alpha$ is added as a constraint for the subsequent iteration. In the propositional context, such strategies are well-known to be nonviable on all but small problem instances (Sang, Beame, and Kautz 2005). To that end, it is natural to ask: can CC be generalized to hybrid domains? Given the maturity of propositional model counting technology, it is perhaps more immediate to consider the conditions under which we can leverage existing CC systems.

Our concern in this paper is the fundamental problem of *probabilistic inference in hybrid graphical models*. In previous work (Belle, Passerini, and Van den Broeck 2015), we proposed a formulation of this problem as a model counting task over linear arithmetic formulas followed by an integration over the weights of models, referred to as *weighted model integration* (WMI). As a first step in understanding effective modeling counting for hybrid domains, we develop ideas to leverage propositional CC systems for WMI. The key restriction needed is that the continuous random variables are assumed to have *piecewise polynomial densities.*

Early work by Curds (1997) and recent investigations by Shenoy and West (2011) show that polynomial densities are not only natural for certain distributions (*e.g.,* piecewise linear and uniform), but can also effectively approximate dif-

*Supported by the Research Foundation-Flanders (FWO-Vlaanderen).
Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

ferentiable families (*e.g.,* Gaussians). Such representations are widely used, for example, in computer graphics and numerical analysis (Shenoy and West 2011), and have received considerable attention in the recent years in the inference endeavor (cf. the subsequent section). Under this assumption, propositional CC algorithms can be leveraged using three simple but powerful ideas:

1. piecewise structure can be encoded in propositional logic;
2. DPLL search can be made to return densities rather than the probability mass;
3. the integration of densities can be performed as a last step in an effective manner.

We prove a number of formal properties about this approach, and then turn to empirical evaluations. For our evaluations, we compare this approach to two other WMI realizations: the straightforward block-clause implementation from our prior work (Belle, Passerini, and Van den Broeck 2015), and one that builds on the `allsat` model enumerator implemented in SMT solvers, which is based on a sophisticated integration of linear arithmetic and SAT solvers. To rigorously compare these realizations, we convert challenging discrete graphical models from the literature to hybrid ones, and our experiments demonstrate a large gap in performance.

Related Work

A number of methods are known for performing exact inference in discrete graphical networks, such as bucket elimination (Dechter 1996), and weighted model counting (WMC) that extends #SAT in according weights to the models of a propositional formula (Chavira and Darwiche 2008). For hybrid graphical models, however, most inference algorithms are either approximate, *e.g.,* (Murphy 1999; Gogate and Dechter 2005; Lunn et al. 2000), or they make strong assumptions on the form of the densities, *e.g.,* Gaussians (Lauritzen and Jensen 2001). This led to considerable interest in graphical models with polynomial densities, *e.g.,* (Salzmann 2013; Wang, Schwing, and Urtasun 2014; Shenoy and West 2011; Sanner and Abbasnejad 2012), where inference is often addressed by generalizations of the join-tree algorithm or variable elimination. In contrast, WMI accords weights to the models of linear arithmetic formulas, and so is a strict generalization of WMC. Like WMC, it allows us to reason about logical equivalence and deterministic constraints over logical connectives in a general way.

While the focus of this paper is on exact inference, approximate inference for WMI is investigated in (Belle, Van den Broeck, and Passerini 2015), which is then related to approximate model counting (Chakraborty et al. 2014). In that vein, Chistikov, Dimitrova, and Majumdar (2015) introduced an approximate algorithm for counting the models of linear arithmetic.

Background

Probabilistic Models and Piecewise Polynomials

Let $\mathcal{V} = \{x_1, \ldots, x_k, b_1, \ldots, b_m\}$ be a finite set of random variables, where x_i take values from $\mathbb{R}$ and b_i take values from $\{0, 1\}$. We let $(x_1, \ldots, x_k, b_1, \ldots, b_m)$ be an element of the probability space $\mathbb{R}^k \times \{0, 1\}^m$, which denotes a particular assignment to the random variables from their respective domains. The joint probability density function is denoted by $\Pr(x_1, \ldots, x_k, b_1, \ldots, b_m)$, and the partition function is:

$$Z = \int_{\{x_1,\ldots,x_k\}} \sum_{\{b_1,\ldots,b_m\}} \Pr(x_1, \ldots, x_k, b_1, \ldots, b_m).$$

In this paper, the density is assumed to be of the form:

$$\Pr(x_1, \ldots, b_m) = \begin{cases} \mathrm{P}(x_1, \ldots, x_m) & \text{if } x_1 \in A_1, b_1 = \theta_1, \ldots \\ \ldots & \ldots \\ 0 & \text{otherwise} \end{cases}$$

where A_i is an interval of $\mathbb{R}$, $\theta_i \in \{0, 1\}$, and $\mathrm{P}(x_1, \ldots, x_m)$ is any polynomial over $\{x_1, \ldots, x_m\}$. We then say that the distribution has *piecewise polynomial densities.*

For ease of presentation, however, we center discussions around a definition from (Shenoy and West 2011), which can be seen as a special case of what we handle. For the sequel:

Definition 1: A one-dimensional function $f : \mathbb{R} \mapsto \mathbb{R}$ is said to be MOP if it is a function of the form:

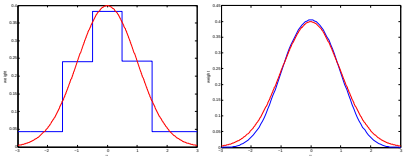
$$f(x) = \begin{cases} a_{0i} + a_{1i}x + \ldots + a_{ni}x^n & \text{if } x \in A_i, i = 1, \ldots, m \\ 0 & \text{otherwise} \end{cases}$$

where $A_1, \ldots, A_m$ are disjoint intervals in $\mathbb{R}$ that do not depend on x, and a_{ji} is a constant for all i, j.

A k-dimensional function $f : \mathbb{R}^k \mapsto \mathbb{R}$ is a MOP function if it is of the form:

$$f(x_1, \ldots, x_k) = f_1(x_1) \times f_2(x_2) \times \cdots \times f_k(x_k),$$

where $f_i(x_i)$ is a one-dimensional piecewise polynomial function as defined above.



(a) degree 0 (b) degree 3

Figure 1: approximations to a univariate Gaussian

As argued in Shenoy and West (2011), MOPs natively support distributions such as uniform and piecewise linear. By further appealing to Taylor expansions (Shenoy and West 2011), differentiable distributions can also be effectively approximated in terms of MOPs. Henceforth, although our definition of piecewise polynomial densities is strictly more general than MOPs, we simply use the term piecewise polynomial densities everywhere.

In Definition 1, we often refer to the intervals A_i as "pieces," and by definition, every density function is characterized in terms of a finite number of pieces. The degree of the polynomial representation corresponds to the granularity

of the approximation, in the sense that increasing the number of pieces and the corresponding degree of the polynomials can lead to better approximations. For example (Belle, Passerini, and Van den Broeck 2015), suppose x is a random variable with univariate Gaussian density. Its approximation in terms of 0 degree polynomials would be as in Figure 1a, whereas its approximation in terms of polynomials of degree 3 would be as in Figure 1b; in the former, an interval, say, $[1.5, 3]$ has the density 0.043, in the latter, an interval, say, $[1, 2]$ has the density $(2 - x)^3/6$.

By extension, let:

$$\Pr(x_1, \ldots, b_m) = f(x_1, \ldots, x_k) \times g_1(b_1) \times \cdots \times g_m(b_m) \quad (1)$$

where f is a piecewise polynomial function and $g_i \colon \{0, 1\} \to \mathbb{R}$ is the form:

$$g_i(b_i) = \begin{cases} c_{i0} & \text{if } b_i = 0 \\ c_{i1} & \text{otherwise} \end{cases}$$

where c_{i0} and c_{i1} are constants.

Logical Background

In SAT, given a formula ϕ in propositional logic, we decide whether there is an assignment (or model) M that satisfies ϕ, written $M \models \phi$. We write $l \in M$ to denote the literals that are satisfied at M.

A generalization to this decision problem is that of *Satisfiability Modulo Theories* (SMT). In SMT, we are interested in deciding the satisfiability of a (typically quantifier-free) first-order formula with respect to some decidable background theory, such as linear real arithmetic $\mathcal{LRA}$. Standard first-order models can be used to formulate SMT; see (Barrett et al. 2009) for a treatment. We use p, q and r to range over propositional letters, and x, y and z to range over constants, *i.e.,* 0-ary functions, of the language. So, ground atoms are of the form q, $\neg p$ and $x + 1 \leq y$. For convenience, we also use a ternary version of $\leq$ written $y \leq x \leq z$ to capture intervals and treat them as literals.

For our purposes, we also need the notion of *formula abstraction* and *refinement* (Barrett et al. 2009). Here, a bijection is established between ground atoms and a propositional vocabulary that is homomorphic with regards to logical operations; propositions are mapped to themselves and ground $\mathcal{LRA}$ atoms are mapped to fresh propositional symbols. Abstraction proceeds by replacing the atoms by propositions, and refinement replaces the propositions with the atoms. We denote the abstraction of an SMT formula ϕ by ϕ^- and the refinement of a propositional formula ϕ by ϕ^+. For example, $[p \vee (x \leq 10)]^-$ is $p \vee q$, and $[p \vee q]^+$ is $p \vee (x \leq 10)$.

Weighted Model Counting and Integration

WMC extends #SAT in that the weight of a formula is given in terms of the total weight of its models, factorized in terms of the literals true in a model:

Definition 2: Given a formula Δ in propositional logic over literals $\mathcal{L}$, and a *weight function* $w : \mathcal{L} \to \mathbb{R}^{\geq 0}$, the *weighted model count* (WMC) is defined as:

$$\text{WMC}(\Delta, w) = \sum_{M \models \Delta} \text{WEIGHT}(M, w)$$

$$\text{WEIGHT}(M, w) = \prod_{l \in M} w(l)$$

WMI generalizes WMC in labeling $\mathcal{LRA}$ literals (Belle, Passerini, and Van den Broeck 2015) :

Definition 3 : Suppose Δ is a $\mathcal{LRA}$ sentence over binary variables $\mathcal{B}$ and continuous variables $\mathcal{R}$, and literals $\mathcal{L}$. Suppose $\text{POLY}(\mathcal{R})$ is the set of polynomial expressions over $\mathcal{R}$. Suppose $w \colon \mathcal{L} \to \text{POLY}(\mathcal{R})$. Then:

$$\text{WMI}(\Delta, w) = \sum_{M \models \Delta^-} \text{VOL}(M, w)$$

$$\text{VOL}(M, w) = \int_{\{l^+ : l \in M\}} \text{WEIGHT}(M, w)\, d\mathcal{X}.$$

In English: WMI is defined in terms of the models of the abstraction Δ^-, that are accorded a *volume,* obtained on integrating the refinements.[1]

To illustrate WMI, consider this example:

Example 4: Suppose $\Delta = p \vee (0 \leq x \leq 10)$. Letting $q = [0 \leq x \leq 10]^-$, suppose $w(p) = .1$, $w(\neg p) = 2x$, $w(q) = 1$ and $w(\neg q) = 0$. Roughly, x is a uniform distribution on $[0, 10]$ when p holds; otherwise it is characterized by a polynomial density of degree one. There are three interpretations of Δ^-. The model $\{\neg p, q\}$ is accorded a volume given by:[2]

$$\text{VOL}(\{\neg p, q\}, w) = \int_{\{\neg p^+,\ q^+\}} 2x\, dx = \int_{0 \leq x \leq 10} 2x\, dx = 100.$$

Analogously, $\text{VOL}(\{p, q\}, w) = 1$ and $\text{VOL}(\{p, \neg q\}, w) = 0$. Therefore, $\text{WMI}(\Delta, w) = 101$.

Exact Inference with Component Caching

As mentioned earlier, we appeal to three simple but powerful ideas for speeding up WMI using CC: (a) a propositional representation for the piecewise structure of densities, (b) letting DPLL traces return densities, and (c) handling integration compactly at the end. We justify these steps formally after briefly recapping the CC methodology.

Component Caching

Any propositional theory over variables $\mathcal{B}$ and CNF formulas $\mathcal{F}$ can be naturally seen as a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ where the vertices $\mathcal{V}$ correspond to the variables $\mathcal{B}$ and each clause in $\mathcal{F}$ corresponds to a hyperedge in $\mathcal{E}$. That is, a hyperedge $e \in \mathcal{E}$ connecting $a, b, c \in \mathcal{V}$ means that there is a clause in $\mathcal{F}$ over those three variables. Recall that #SAT is #P-complete (Valiant 1979), and so all known algorithms require exponential time in the worst case. Nonetheless, for theories whose hypergraphs have a small *treewidth*

[1] The original formulation in (Belle, Passerini, and Van den Broeck 2015) is not limited to polynomials, but we take the liberty of doing so for the sake of consistency in presentation.

[2] Propositions are ignored for integration; see (Belle, Passerini, and Van den Broeck 2015) for the general definition on how refinements are handled wrt integration.

Algorithm 1 #DPLLCache(Δ, w) returns WMC(Δ, w)

1: **if** InCache(Δ) **then** return GetValue(Δ)
2: Θ = RemoveCachedComponents(Δ)
3: choose variable v from some component $\beta \in \Theta$
4: #DPLLCache($\Delta - \{\beta\} \cup \Theta_0, w$) // Θ_0 = ToComponents($\beta|_{v=0}$)
5: #DPLLCache($\Delta - \{\beta\} \cup \Theta_1, w$) // Θ_1 = ToComponents($\beta|_{v=1}$)
6: AddToCache($\beta, w(v)\times$GetValue(Θ_1)$+w(\neg v)\times$GetValue(Θ_0))
7: return GetValue(Δ)

(Bacchus, Dalmao, and Pitassi 2009), techniques like recursive conditioning are very efficient. In contrast, the obvious modification of DPLL for counting that backtracks fully is always exponential-time. (Consider, for example, a k-CNF formula over $k \cdot n$ variables and n clauses that share no variables.) CC is an algorithm that decomposes the input formula into independent components that are cached and solved only once. For example, $(p \vee q) \wedge (q \vee r) \wedge (t \vee s)$ decomposes into components $\{p \vee q, q \vee r\}$ and $\{t \vee s\}$, whereas $(p \vee q) \wedge (q \vee r) \wedge (t \vee r)$ corresponds to a single component. CC solves #SAT with time complexity that is at least as good any other exact algorithm, but can also achieve the best known time-space tradeoff (Bacchus, Dalmao, and Pitassi 2009). By further appealing to more flexible variable orderings, significant speedups over other algorithms is possible.

A full description of the algorithm can be found in (Bacchus, Dalmao, and Pitassi 2009), but a basic weighted version is given in Algorithm 1. The algorithm takes as input a formula Δ as a set of components. If the formula is already present in the cache, then its weighted model count is returned via GetValue(Δ). If not, Δ is first restricted to the set of non-cached components $\{\beta_1, \ldots, \beta_n\}$. Then, choosing some β, and some variable v mentioned in β, we compute (or retrieve from cache) the weighted model counts of: (a) $\beta|_{v=1}$, and (b) $\beta|_{v=0}$, where $\phi|_{v=b}$ denotes the logical simplification of ϕ by setting v to b. The weighted model count of β is then obtained in line 6, which amounts to multiplying (a) by the weight of v and (b) by that of $\neg v$.

Step 1: Eager Encodings

Observe that if we were to directly apply standard CC on a propositional abstraction of a $\mathcal{LRA}$ sentence, we are doomed to obtain inconsistent components.

Example 5: Consider $\Delta = ((p \vee x \leq 10) \wedge (q \vee x \geq 11))$. On abstraction, suppose we have $(p \vee s) \wedge (q \vee t)$, where $s^+ = (x \leq 10)$ and $t^+ = (x \geq 11)$. This sentence can be decomposed into components $\{p \vee s\}$ and $\{q \vee t\}$, but the truth values of the literals $\{x \leq 10, x \geq 11\}$ are not independent in $\mathcal{LRA}$.

What is required, then, is a way to reason about $\mathcal{LRA}$ consistency explicitly in the propositional formula. For example, clearly $s \Leftrightarrow \neg t$ in the above example, and if that sentence is added to Δ^-, the resulting formula would be treated as a single component, as desired. Thus, in general, by appealing to $\mathcal{LRA}$ theorems, we can generate a purely propositional *equisatisfiable* formula for a given Δ. This would come with a singular benefit: with some effort, a propositional CC system could be applied to the $\mathcal{LRA}$ setting.

The above desiderata comprises what is often called the *eager encoding* of SMT sentences, the details of which are not necessary here (Barrett et al. 2009). Unfortunately, eager encodings of arbitrary sentences in many first-order fragments, including $\mathcal{LRA}$, incurs the cost of a significant blow-up in the translation. Fortunately, for the particular case of piecewise polynomial densities, the encoding is *small* and *elegant*.[3]

Leveraging the piecewise structure The key observation from Definition 1 is that one can provide an encoding of a joint distribution in a manner that limits the $\mathcal{LRA}$ sentences to disjoint intervals only, whose weights are given using single variable polynomials:

Theorem 6: *Suppose* $\Pr(x_1, \ldots, b_m)$ *is a joint distribution of the form* (1)*, whose partition function is Z. Then there is an* $\mathcal{LRA}$ *sentence* Δ *and w such that* $\text{WMI}(\Delta, w) = Z$*, where (a) for every continuous variable x in* Δ*, only literals of the form* $\alpha \leq x \leq \beta$ *appear in* Δ*, where* $\alpha, \beta \in \mathbb{R}$*, (b) given literals* $\alpha \leq x \leq \beta$ *and* $\alpha' \leq x \leq \beta'$ *in* Δ*,* $[\alpha, \beta]$ *and* $[\alpha', \beta']$ *are disjoint intervals in* $\mathbb{R}$*, and (c) w maps* $\alpha \leq x \leq \beta$ *to a polynomial of the form* $a_0 + a_1 x + \ldots + a_n x^n$.

Proof: By assumption, $\Pr(x_1, \ldots, x_k) = f_1(x_1) \times \cdots \times f_k(x_k) \times g_1(b_1) \times \cdots \times g_m(b_m)$ where $f_i(x_i)$ are piecewise polynomials. By Definition 1, any instantiation of the random variable x_i must be in one of the disjoint intervals $A_1, \ldots, A_m$, where $A_j = [s_j, t_j]$ such that $s_j, t_j \in \mathbb{R}$. Moreover, the density accorded to $x_i \in A_j$ is of the form $P_{ij}(x_i) = a_{0j} + a_{1j} x_i + \ldots + a_{lj} x_i^l$. For variable i, then, let $\phi_i \doteq \vee_j (s_j \leq x_i \leq t_j)$ with $w(s_j \leq x_i \leq t_j) = P_{ij}(x_i)$. Finally, let $\Delta = \wedge_i \phi_i$. For the binary variables, let $w(b_i) = c_{i1}$ and $w(\neg b_i) = c_{i0}$. It is now not hard to see that $\text{WMI}(\Delta, w) = Z$. ∎

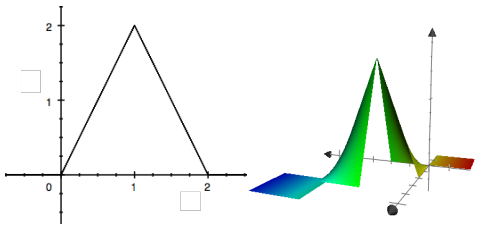
Intuitively, then, for a full instantiation of the random variables $x_1 \in A_1, \ldots, x_k \in A_k$, associated with densities $P_1(x_1), \ldots, P_k(x_k)$, any model satisfying $\bigwedge_i (x_i \in A_i)$ is accorded the multi-variate density $P_1(x_1) \times \cdots \times P_k(x_k)$.

Example 7: For illustration, Figure 2a is a coarse approximation of a univariate Gaussian distribution given by $\Delta = (0 \leq x \leq 1) \vee (1 < x \leq 2)$, $w(0 \leq x \leq 1) = 2x$ and $w(1 < x \leq 2) = 2(2 - x)$ and 0 everywhere else.

Analogously, Figure 2b is a coarse approximation for a bivariate Gaussian distribution, where $\Delta' = \Delta \wedge ((0 \leq y \leq 1) \vee (1 < y \leq 2))$, and w additionally assigns: $w(0 \leq y \leq 1) = 2y$ and $w(1 < y \leq 2) = 2(2 - y)$ and 0 everywhere else for y's values. So, for example, any model that satisfies $\psi = (0 \leq x \leq 1) \wedge (0 \leq y \leq 1)$ would be accorded the density $2x \times 2y = 4xy$.

Finally, consider a proposition p and suppose w assigns: $w(p) = .9$ and $w(\neg p) = .1$. Then any model satisfying $\psi \wedge p$ is accorded the density $3.6xy$.

[3] Strictly speaking, it is the piecewise structure of the density function that turns out to be crucial to our methodology. In other words, much of what we discuss in this paper would change little if the density was, say, piecewise exponential. However, we are not aware of any efficient methodologies to integrate products of exponentials, outside of the usual case for conjugate distributions.



(a) 1d (b) 2d

Figure 2: coarse approximations

From $\mathcal{LRA}$ to propositional logic From Theorem 6, a restricted $\mathcal{LRA}$ fragment is sufficient for encoding the joint distribution, by means of which an elegant eager encoding is possible.

Definition 8: Suppose (Δ, w) is as in Theorem 6. For any variable x, suppose literals $\alpha_1 \leq x \leq \beta_1, \ldots, \alpha_k \leq x \leq \beta_k$ are the only ones mentioning x in Δ. Let Γ be obtained by adding the following clauses to Δ: $\neg(\alpha_i \leq x \leq \beta_i) \vee \neg(\alpha_j \leq x \leq \beta_j)$ for $i \neq j$. Then, let $\Omega = \Gamma^{-}$ and $w'([\alpha_i \leq x \leq \beta_i]^{-}) = w(\alpha_i \leq x \leq \beta_i)$. We call (Ω, w') the *eager encoding* of (Δ, w).

Basically, the additional constraint disallows x from taking on values from multiple intervals.[4] To see that, suppose $\Delta = ((0 \leq x \leq 1) \vee (1 < x \leq 2))$. If Ω is defined as $\Delta^{-} = (p \vee q)$, then there is a model of Ω that makes both p and q true, which corresponds to a $\mathcal{LRA}$-inconsistent assignment. The new constraint $(\neg p \vee \neg q)$ is trivially entailed by Δ in $\mathcal{LRA}$, but is needed in Ω for the equisatisfiability of Ω and Δ.

Reasoning about evidence and queries is an important concern, which requires some extra work with eager encodings. Given $((0 \leq x \leq 1) \vee (1 < x \leq 2)) \in \Delta$, the query/evidence $0 \leq x \leq 1$ can be easily handled by simply considering its abstraction with Ω. A query such as $(0 \leq x \leq .5)$ needs the addition of the following $\mathcal{LRA}$ theorem to Ω: $(0 \leq x \leq .5) \Rightarrow (0 \leq x \leq 1)$. Finally, a query of the form $(.5 \leq x \leq 1.5)$ is less trivial. We can, however, meaningfully split intervals. Using the mathematical property that if the density of the interval $\alpha \leq x \leq \beta$ is δ, then the density of both $\alpha \leq x \leq (\beta - \alpha)/2$ and $(\beta - \alpha)/2 < x \leq \beta$ are also δ, we can convert the original specification to one that handles arbitrary interval queries.

Step 2: DPLL Traces with Densities

The construction (Ω, w') from Definition 8 differs from a standard WMC task in that w' maps literals to polynomial densities. In fact, these densities are defined for specific intervals, which determine the bounds of the integral in Definition 3. The key observation we now make is that Algorithm 1 can be modified to carry over information about the bounds in line 6 by letting it be AddToCache(β, γ) where γ is:

$$[\mathbb{I}_{v^+} \times w'(v) \times \textsc{GetValue}(\Theta_1) + \mathbb{I}_{\neg v^+} \times w'(\neg v) \times \textsc{GetValue}(\Theta_0)]$$

[4] Readers may observe significant similarities between this encoding scheme and that of multi-valued discrete Bayesian networks (Sang, Beame, and Kautz 2005; Chavira and Darwiche 2008).

and $\mathbb{I}_e$ is the indicator event that e holds. That is, γ returns a sum of product of densities, as applicable for the interval determined by $(\neg)v^+$, which is enabled by $\mathbb{I}_{(\neg)v^+}$.

For illustration, continuing Example 7, let $[0 \leq x \leq 1]^{-} = p$ and $[0 \leq y \leq 1]^{-} = q$. Suppose $\beta = \{p, q\}$ is a component considered for line 3 of Algorithm 1. Then γ becomes

$$\mathbb{I}_{p^+} \times w'(p) \times \textsc{GetValue}(\beta|_{p=1}) + \mathbb{I}_{\neg p^+} \times w'(\neg p) \times \textsc{GetValue}(\beta|_{p=0}).$$

The first term, for example, simplifies to $\mathbb{I}_{0 \leq x \leq 1} \times 2x \times \textsc{GetValue}(\beta|_{p=1})$, which is understood as saying that $(0 \leq x \leq 1)$ is the interval for the density $2x$.

Step 3: Delayed Integration

We now observe that the volume of a model in Definition 3 can be defined in terms of indicator events:

$$\int_{\{l^+ : l \in M\}} \text{WEIGHT}(M, w) = \int_{\mathbb{R}^n} \mathbb{I}_{\{l^+ : l \in M\}} \times \text{WEIGHT}(M, w).$$

For example, $\int_0^5 \phi dx = \int_{\mathbb{R}} \mathbb{I}_{0 \leq x \leq 5} \phi dx$. More generally, this simplification can be applied to Definition 3 to yield:

$$\begin{aligned} \text{WMI}(\Delta, w) &= \sum_{M \models \Delta^{-}} \int_{\mathbb{R}^n} \mathbb{I}_{\{l^+ : l \in M\}} \times \text{WEIGHT}(M, w)\, d\mathcal{X} \\ &= \int_{\mathbb{R}^n} \sum_{M \models \Delta^{-}} \prod_{l \in M} \mathbb{I}_{l^+} \times w(l^+)\, d\mathcal{X} \end{aligned}$$

Indeed, the modified CC algorithm essentially returns expressions of the form $\mathbb{I}_{l^+} \times w(l^+)$. This means that only a final integration step is needed wrt piecewise densities for computing WMI. More precisely, we obtain:

Theorem 9: *Suppose Δ, w, Ω, w' are as in Definition 8. Then*

$$\text{WMI}(\Delta, w) = \int_{\mathbb{R}^n} \#\text{DPLLCache}(\Omega, w').$$

Complexity

There are two key computations in the WMI framework, the first of which is the integration when computing the volume in Definition 3. It is known that computing the volume of polytopes of varying dimension is #P-hard (Dyer and Frieze 1988), and that integrating arbitrary polynomials over a simplex is NP-hard (Baldoni et al. 2011). However, Baldoni *et al.* (2011) further show that when the polynomial, of a degree at most d, uses a fixed number of variables, integration over a simplex can be done in polynomial time. Under those same assumptions, this result can be extended to general polytopes because all polyhedral computation, such as computing triangulations, is efficient (Baldoni et al. 2011). This allows us to obtain the following:

Theorem 10: *Suppose $d, n \in \mathbb{N}$, (Δ, w) is as before, where Δ mentions literals $\mathcal{L}$. Suppose $|\mathcal{L}| = n$ and for any $L \subseteq \mathcal{L}$, $\prod_{l \in L} w(l)$ is of degree at most d. Then for any $L \subseteq \mathcal{L}$, given the polynomial $P = \prod_{l \in L} w(l)$, there is a polynomial-time algorithm for the integration of P.*

Table 1: characteristics of the datasets used in the experiments

Dataset	#vars	#clauses	#literals
ALARM	596	865	4167
CHILD	279	419	1655
INSURANCE	787	1380	6591
WATER	3661	3072	18145

Table 3: CC execution time (in seconds) over piecewise constant and polynomial densities

Dataset	CONST	POLY
ALARM	3.06	26.94
CHILD	0.02	1.64
INSURANCE	20.77	135.49
WATER	0.01	0.33

Table 2: comparing solvers execution times (in seconds) against increasing missing evidence, an X indicates that the solver did not terminate after 12 hours

Missing	5	10	15	20	25	30	50	75	100	150	200
	ALARM										
BC	1.09	261.74	X	X	X	X	X	X	X	X	X
ALL	0.48	4.88	162.89	2842.69	X	X	X	X	X	X	X
CC	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01
	CHILD										
BC	0.19	1.13	478.25	X	X	X	X	X	X	X	X
ALL	0.18	0.31	4.54	139.98	X	X	X	X	X	X	X
CC	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.01
	INSURANCE										
BC	1.47	344.41	X	X	X	X	X	X	X	X	X
ALL	0.67	6.71	238.33	X	X	X	X	X	X	X	X
CC	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00
	WATER										
BC	6.08	1700.70	X	X	X	X	X	X	X	X	X
ALL	2.73	34.54	2282.18	X	X	X	X	X	X	X	X
CC	0.00	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01	2.03	0.01

We also need to take into account the complexity of the counting operation, which, of course, runs in exponential-time in the worst case. However, based on the formal properties of CC (Bacchus, Dalmao, and Pitassi 2009), we obtain:

Theorem 11: *Suppose* (Δ, w) *is as before, and* (Ω, w') *is the eager encoding of* (Δ, w) *and suppose* Ω *uses n propositions. Then there is an execution of* #DPLLCACHE(Ω, w') *that runs in time bounded by* $n^{O(1)}2^{O(w)}$ *where w is the underlying treewidth of the instance.*

Evaluations

In this section, we demonstrate that the approach introduced here performs significantly better than known techniques for model counting of arithmetic constraints. Our techniques are built on the competitive WMC solver CACHET (Sang, Beame, and Kautz 2005). For integrating in fixed dimensions efficiently, we use LATTE v1.6.[5] All experiments were run using a 2.83 GHz Intel Core 2 Quad processor and 8GB RAM.

While there are a number of existing benchmarks for discrete graphical models (Chavira and Darwiche 2008), we are not aware of well-established and challenging benchmarks for mixed discrete-continuous graphical networks. On a related note, existing #SAT generalizations often resort to proof-of-concept demonstrations (Chistikov, Dimitrova, and Majumdar 2015), or small-size randomly generated problems (Belle, Passerini, and Van den Broeck 2015). In particular, as with (Sang, Beame, and Kautz 2005), we are deliberately after structured problems with many logical dependencies between variables and hard constraints.

Our observation here is that many standard Bayesian networks (see (Chavira and Darwiche 2005; 2008) for sources), including WATER, ALARM, MILDEW, MUNIN, among others, are discrete models of continuous random variables, often with a piecewise structure. Based on this, we propose the construction of hybrid networks where these properties are modeled using actual continuous variables. This allows us to generate a suite of challenging benchmark problems. To see the recipe using an example, consider the ALARM network where the measurement of a patient's blood pressure are: *low, normal* and *high.* Letting the continuous variable x denote blood pressure, we consider the intervals $70 \leq x \leq 90$, $90 < x \leq 120$ and $120 < x \leq 190$ to denote those states respectively. We associate *low* and *high* with triangular densities, that is, polynomials in x with degree 1. Finally, we associate *normal* with a piecewise polynomial approximation of a Gaussian. In the sequel, we consider such hybrid versions of ALARM, WATER, INSURANCE, and CHILD.[6] Table 1 briefly summarizes the characteristics of the datasets.

To demonstrate the benefits of WMI with component caching (denoted CC in the results), we consider a straightforward block-clause realization of WMI (denoted BC in the results) (Belle, Passerini, and Van den Broeck 2015). BC is implemented using MATHSAT v5.[7] In the recent years, the SMT community has developed their own variant of model enumeration called `allsat` (simply denoted ALL). ALL is based on a sophisticated and deep integration of $\mathcal{LRA}$-theory and SAT solvers.[8] To date, variants of such native model enumeration techniques and block-clause strategies are the most dominant in the literature. We implemented an ALL-based WMI system, also using MATHSAT v5. Our objective here is to show that for the *piecewise polynomial setting*, *CC via eager encodings provides huge savings*.

The overall recipe is this: given a hybrid graphical model N, we consider a simpler setting with piecewise constant

[5] `https://www.math.ucdavis.edu/~latte`

[6] The caveat here is that apart from the labels on the nodes in the graphical networks, we do not have access to natively hybrid versions of these networks. Thus, we randomly modeled some nodes as continuous variables and randomly chose polynomial densities.

[7] `http://mathsat.fbk.eu`

[8] It is worth noting that these model enumeration techniques handle arbitrary SMT problems, and thus, are more general than our CC methodology.

densities (denoted CONST) and a second setting with piecewise polynomial potentials (denoted POLY). A weighted $\mathcal{LRA}$ encoding of N is provided to BC and ALL, and a weighted propositional one to CC (following Definition 8).

Regarding CONST, ALL and BC were not able to complete the WMC task on all the networks considered, given a timeout of 12 hours. However, we can simplify the task for ALL and BC by providing a large amount of evidence. This can be done in a principled manner as follows. We first find a model M for the input sentence, which is a complete truth assignment to all the propositions. Suppose there are n propositions, and we would like to provide k propositions as evidence. Then the SMT and CNF files are modified to assert the truth values of k propositions as suggested by M. For extremely large k (or rather extremely small missing evidence $n - k$), ALL and BC successfully terminate in Table 2. As k is reduced in small iterations, ALL and BC are yet again infeasible. Overall, CC offers a large gap in performance.

For the more challenging case where no evidence is provided, the performance of CC on CONST and POLY are reported in Table 3. We observe that the effort needed for integration is reasonable, given the complex nature of the problems. (POLY is not reported for ALL and BC.)

Conclusions

In this paper, we considered the problem of performing exact inference in hybrid networks. By enabling component caching, realized in terms of eager encodings and delayed integration, we obtained a new WMI solver, with a large gap in performance when compared to existing implementations. Given that many real-world AI applications involve combinations of discrete and continuous variables, developing effective model counting techniques for hybrid domains is worthy of further investigation.

The efficiency of our eager encoding approach depends on the piecewise structure of the probability density function. Adapting component caching strategies to more general functions, like those obtained with the full language of linear arithmetic, is a challenging direction for future research. One possible strategy, for example, is to upgrade the notion of a component to not only range over propositions but also over the continuous variables. While this would be general and come with some benefits, it remains to be seen if it performs meaningfully in complex settings such as dynamic Bayesian networks (Boyen and Koller 1998) where variables become correlated very quickly.

References

Bacchus, F.; Dalmao, S.; and Pitassi, T. 2009. Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research* 34(1):391–442.

Baldoni, V.; Berline, N.; De Loera, J.; Köppe, M.; and Vergne, M. 2011. How to integrate a polynomial over a simplex. *Mathematics of Computation* 80(273):297–325.

Barrett, C.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*. chapter 26, 825–885.

Belle, V.; Passerini, A.; and Van den Broeck, G. 2015. Probabilistic inference in hybrid domains by weighted model integration. In *Proc. IJCAI*.

Belle, V.; Van den Broeck, G.; and Passerini, A. 2015. Hashing-based approximate probabilistic inference in hybrid domains. In *UAI*.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *UAI*, 33–42.

Chakraborty, S.; Fremont, D. J.; Meel, K. S.; Seshia, S. A.; and Vardi, M. Y. 2014. Distribution-aware sampling and weighted model counting for sat. *Proc. AAAI*.

Chavira, M., and Darwiche, A. 2005. Compiling Bayesian networks with local structure. In *IJCAI*, volume 19, 1306.

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

Chistikov, D.; Dimitrova, R.; and Majumdar, R. 2015. Approximate counting in smt and value estimation for probabilistic programs. In *TACAS*, volume 9035. 320–334.

Curds, R. M. 1997. *Propagation techniques in probabilistic expert systems*. Ph.D. Dissertation, Department of Statistical Science, University College London.

Darwiche, A. 2001. Recursive conditioning. *Artificial Intelligence* 126(1):5–41.

Darwiche, A. 2004. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of ECAI*, 328–332.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *UAI*, 211–219.

Dyer, M. E., and Frieze, A. M. 1988. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.* 17(5):967–974.

Fredrikson, M., and Jha, S. 2014. Satisfiability modulo counting: A new approach for analyzing privacy properties. In *LICS*, 42:1–42:10. New York, NY, USA: ACM.

Gogate, V., and Dechter, R. 2005. Approximate inference algorithms for hybrid bayesian networks with discrete constraints. *UAI*.

Lauritzen, S. L., and Jensen, F. 2001. Stable local computation with conditional gaussian distributions. *Statistics and Computing* 11(2):191–203.

Lunn, D. J.; Thomas, A.; Best, N.; and Spiegelhalter, D. 2000. Winbugs - a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and computing* 10(4):325–337.

Murphy, K. P. 1999. A variational approximation for bayesian networks with discrete and continuous latent variables. In *UAI*, 457–466.

Salzmann, M. 2013. Continuous inference in graphical models with polynomial energies. In *CVPR*, 1744–1751.

Sang, T.; Beame, P.; and Kautz, H. A. 2005. Performing bayesian inference by weighted model counting. In *AAAI*.

Sanner, S., and Abbasnejad, E. 2012. Symbolic variable elimination for discrete and continuous graphical models. In *AAAI*.

Shenoy, P., and West, J. 2011. Inference in hybrid bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning* 52(5):641–657.

Valiant, L. 1979. The complexity of enumeration and reliability problems. *SIAM Journal on Computing* 8(3):410–421.

Wang, S.; Schwing, A. G.; and Urtasun, R. 2014. Efficient inference of continuous markov random fields with polynomial potentials. In *NIPS*, 936–944.