Efficient Search-Based Weighted Model Integration

Zhe Zeng and **Guy Van den Broeck**
Computer Science Department
University of California, Los Angeles
{zhezeng, guyvdb}@cs.ucla.edu

Abstract

Weighted model integration (WMI) extends weighted model counting to integration in mixed discrete-continuous domains. It has shown tremendous promise for solving probabilistic inference problems in graphical models and probabilistic programs. Yet, state-of-the-art tools for WMI have limited performance and ignore the independence structure that is crucial to improving efficiency. To address this limitation, we propose an efficient model integration algorithm for theories with tree primal graphs. We exploit the sparse graph structure by using search to performing integration. Our algorithm greatly improves the computational efficiency on such problems and exploits context-specific independence between variables. Experimental results show dramatic speedups compared to existing WMI solvers on problems with tree-shaped dependencies.

1 INTRODUCTION

Weighted model counting (WMC) is the task of counting the weighted sum of all satisfying assignments of a propositional logic theory. In recent years, WMC was shown to be an effective solution for addressing probabilistic inference in a wide spectrum of formalisms (Sang et al., 2005; Chakraborty et al., 2014; Ermon et al., 2013; Chavira and Darwiche, 2008; Choi et al., 2013; Van den Broeck and Suciu, 2017; Fierens et al., 2015).

An inherent limitation of WMC is that it can only deal with discrete distributions. In order to overcome this restriction, weighted model integration (WMI) (Belle et al., 2015a) was introduced as a generalization of WMC towards hybrid domains, characterized by both discrete and continuous variables. The formalism relies on satisfiability modulo theory (SMT) (Barrett and Tinelli, 2018) technology, which permits reasoning about the satisfiability of theories involving, for example, linear constraints over reals. WMI works by summing a simple weight function over solutions to Boolean variables and integrating over solutions to the real variables of an SMT theory. Weight functions play the role of (unnormalized) densities, whereas the logic theory captures the structure of the distribution. WMI (or closely related formulations) has recently been applied to several probabilistic graphical model and programming tasks (Chistikov et al., 2015; Albarghouthi et al., 2017; Morettin et al., 2017; Belle, 2017; de Salvo Braz et al., 2016).

Both WMI and WMC are sum-of-product problems (Bacchus et al., 2009). In discrete domains, such problems are amenable to a divide-and-conquer approach called search-based inference, where variables are instantiated recursively until the inference problem decomposes. Solving WMC by search, exploiting problem-specific structure, has been shown to be highly effective, in particular on graphical models that exhibit sparsity (Chavira and Darwiche, 2008). However, progress in WMI is far from its Boolean counterpart, and currently does not exploit independence. More generally, exact inference algorithms for hybrid graphical models do not exploit sparsity and structure as much as discrete graphical model inference algorithms.

As a first approach to leverage structure, in this paper, we propose a search-based inference procedure for exact model integration that leverages decomposition to speed up inference. We demonstrate how local structure encoded in SMT theories gives rise to context-specific decomposition during search, reducing the number of models to be generated and integrated over. The integration problem is decomposed into sub-problems by instantiating shared variables and recursing independently on the resulting simplified SMT theories. We show how to choose finitely many values to instantiate continuous variables with, and subsequently do polynomial interpo-

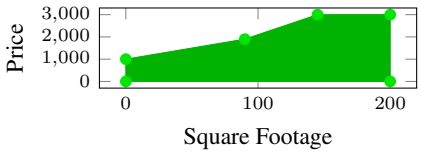


Figure 1: Feasible region of SMT theory γ_i from Example 2.1

lation to recover exact answers to WMI problems. Our complexity analysis proves the first tractability result for a non-trivial class of WMI problems. Moreover, our experimental evaluation confirms that the approach is drastically faster than existing alternatives on WMI problems with sparse, tree-shaped primal graphs.

2 BACKGROUND

We assume that the reader is familiar with propositional logic and the SAT problem (Biere et al., 2009). Model counting (#SAT) is the task of counting the number solutions (models) to a given SAT problem (Gomes, 2009). Weighted model counting (WMC) generalizes this task by summing weights associated with individual SAT solutions. It is widely used as tool for probabilistic reasoning (Sang et al., 2005; Chavira and Darwiche, 2008; Ermon et al., 2013; Chakraborty et al., 2014; Fierens et al., 2015; Van den Broeck and Suciu, 2017).

Satisfiability Modulo Theories (SMT) generalizes SAT to determining the satisfiability of a formula with respect to a decidable background theory. In particular, we will consider quantifier-free SMT formulas in the theory of linear arithmetic over the reals, or SMT($\mathcal{LRA}$). Here, formulas are Boolean combinations of atomic propositions (e.g., a, b), and of atomic $\mathcal{LRA}$ formulas over real variables (e.g., $x < y + 5$). Variable instantiations are denoted as $b^\star$ or $x^\star$. Sets are denoted in boldface.

Example 2.1. *For a house i, let $price_i$ be its price and $sqft_i$ its square footage. We can build a simple SMT($\mathcal{LRA}$) formula of the relationship between these real variables, with the corresponding solution space depicted in Figure 1. That is, SMT($\mathcal{LRA}$) formula γ_i is*

$$(price_i < 10 \cdot sqft_i + 1000) \vee (price_i < 20 \cdot sqft_i + 100)$$
$$(0 < price_i < 3000) \wedge (0 < sqft_i < 200).$$

Weighted model integration (WMI) generalizes WMC to support SMT($\mathcal{LRA}$) formulas and real variables (Belle et al., 2015a). In its simplest form, model integration (MI) or #SMT (Chistikov et al., 2015) computes the volume of the solution space. For example, the green area in Figure 1 is 430,250. General WMI is defined as follows (Belle et al., 2015a; Morettin et al., 2017).

Definition 2.2. *Suppose we have n real variables $\boldsymbol{x}$, m Boolean variables $\boldsymbol{b}$, an SMT($\mathcal{LRA}$) formula $\theta(\boldsymbol{x}, \boldsymbol{b})$, ranging over $\boldsymbol{x}$ and $\boldsymbol{b}$, and a weight function $w(\boldsymbol{x}, \boldsymbol{b})$ that maps variable instantiations to real weights. Then, weighted model integration (WMI) computes*

$$\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \sum\nolimits_{\boldsymbol{b}^\star \in \mathbb{B}^m} \int_{\theta(\boldsymbol{x}, \boldsymbol{b}^\star)} w(\boldsymbol{x}, \boldsymbol{b}^\star)\, d\boldsymbol{x}.$$

That is, the WMI is obtained by summing over every instantiation (total truth assignment) $\boldsymbol{b}^\star$ to the Boolean variables, and integrating $w(\boldsymbol{x}, \boldsymbol{b}^*)$ over the set of solutions $\{\boldsymbol{x}^* \mid \theta(\boldsymbol{x}^*, \boldsymbol{b}^\star) \text{ is SAT}\}$.

Weight functions w are usually defined as products of literal weights (Belle et al., 2015a; Chavira and Darwiche, 2008). That is, for some set of literals $\mathcal{L}$ we are given a set of per-literal weight functions $\mathcal{P} = \{p_\ell(\boldsymbol{x})\}_{\ell \in \mathcal{L}}$. When literal ℓ is satisfied in a world, denoted $\boldsymbol{x} \wedge \boldsymbol{b} \models \ell$, that world's weight is multiplied by $p_\ell(\boldsymbol{x})$. Formally,

$$w(\boldsymbol{x}, \boldsymbol{b}) = \prod\nolimits_{\ell \in \mathcal{L}, \boldsymbol{x} \wedge \boldsymbol{b} \models \ell} p_\ell(\boldsymbol{x}).$$

When all variables are Boolean (i.e., $\boldsymbol{x} = \emptyset$), the per-literal weights $p_\ell(\boldsymbol{x})$ are constants and we retrieve the original definition of WMC as a special case of WMI (Chavira and Darwiche, 2008). In this paper, we assume that all per-literal weights are polynomials. This setting is expressive enough to approximate any continuous distribution (Belle et al., 2015a).

Example 2.3. *Consider a formula $(b \vee \neg b) \wedge \gamma_i$ where b is a Boolean variable and γ_i is as defined in Example 2.1. Consider the set of literals $\mathcal{L} = \{b, (0 < price_i < 3000)\}$ and per-literal weight functions $\mathcal{P} = \{p_b, p_{(0<price_i<3000)}\}$, with $p_b(\boldsymbol{x}) = 1.5$ and $p_{(0<price_i<3000)}(\boldsymbol{x}) = price_i^2$. Then, in worlds where both literals in $\mathcal{L}$ are satisfied, our weight function is $p_b(price_i, sqft_i) \cdot p_{(0<price_i<3000)}(price_i, sqft_i) = 1.5 \cdot price_i^2$. In worlds where b is false and only $(0 < price_i < 3000)$ is satisfied, the weight function is $price_i^2$.*

Moreover, we will show that this class of weight functions is well-behaved. In particular, it allows for a natural reduction to unweighted model integration and is amenable to efficient integration.

WMI was introduced as a tool for hybrid probabilistic reasoning. Indeed, the weight of each world can be interpreted as an unnormalized density, and the WMI is its partition function subject to the logical constraints. Under these semantics, suppose that we are interested in the probability of query $q = (price_i < 2000)$ in house price model γ_i. That probability can be computed as the ratio of two WMI problems: $\Pr(q) = \mathrm{WMI}(\gamma_i \wedge q)/\,\mathrm{WMI}(\gamma_i) = 350{,}250/430{,}250 = 81.4\%$.

Exact WMI Solvers The first solver for exact WMI (Belle et al., 2015a) was a proof-of-concept relying

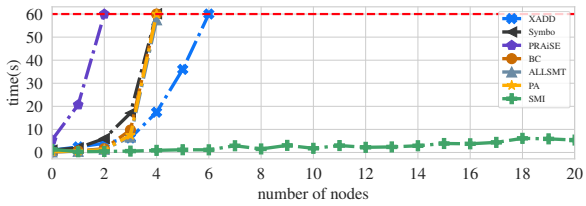
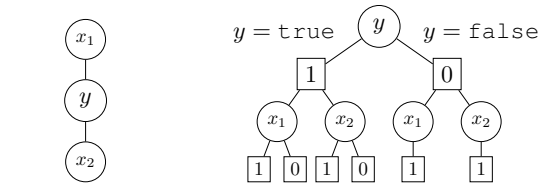


Figure 2: WMI runtime on independent model in Example 3.1.



(a) Primal graph (b) Discrete And/Or Search Tree

Figure 3: Primal graph and search tree for $(y \vee x_1) \wedge (y \vee x_2)$.

on a simple block-clause strategy (BC). It iteratively generates new models of a Boolean abstraction of the SMT formula. Each model individually is easily integrated using tools such as LATTE (Baldoni et al., 2011; De Loera et al., 2013). Belle et al. (2016) proposed an all-satisfying-assignments-based solver (ALLSMT). Unfortunately, enumerating models of the SMT abstraction is prohibitive in practice – there are exponentially many models, and enumerating them does not exploit structural properties of the SMT theory such as independence. Improvements to this algorithm include predicate-abstraction solvers (Belle et al., 2016; Morettin et al., 2017) (PA) and knowledge-compilation solvers (Kolb et al., 2018) (XADD) and Symbo (Zuidberg Dos Martires et al., 2019). The PRAiSE solver (de Salvo Braz et al., 2016) performs search-based inference on literals of SMT models (not theory variables) and can also be used to solve WMI problems. Nevertheless, WMI solvers come with no tractability guarantees and still enumerate Boolean models even when there is abundant independence structure, as we will show next.

3 STRUCTURE IN WMI PROBLEMS

This section shows how to reduce WMI to model integration (MI) problems whose structural independence properties can be captured by graph abstractions.

3.1 INDEPENDENCE

We begin by motivating why we want to exploit independence structure during probabilistic reasoning.

Example 3.1. *Consider n houses, and conjoin the theory γ_i from Example 2.1 n times, once for each house, into a larger SMT theory $\gamma = \wedge_{i=1}^{n} \gamma_i$. The n houses are independent since no formula in γ connects properties of different houses. Thus, the WMI of γ can be computed by multiplying the WMI of each individual theory γ_i.*
Figure 2 takes a trivial weight function and compares existing WMI solvers on this simple problem. None is able to exploit the extreme independence structure in γ. Our proposed method SMI, however, runs in linear time, as expected by the trivial factorization.

This explosion in runtime is due to the fact that existing solvers ignore independence between variables in the SMT($\mathcal{LRA}$) theory. However, in discrete graphical models and WMC, leveraging independence to decompose problems is at the core of all exact inference methods, and search-based algorithms in particular (Darwiche, 2009; Dechter and Mateescu, 2007). Specifically, exact discrete inference methods *create* independence even when it is not immediately present, by performing a case analysis on selected discrete variables, instantiating them to all values, and simplifying the model. Through this process, search-based inference algorithms induce and exploit context-specific independence (Boutilier et al., 1996). The decompositions afforded by (conditional and context-specific) independence vastly reduce the computational cost of inference. Example 3.1 illustrated that this intuition carries over to WMI problems.

In what follows, we first describe the graph abstraction of SMT theories that characterizes dependencies between variables. These form the basis of our algorithm. Second, we show how WMI in hybrid domains can be reduced to unweighted MI in real domains. Hence, the solver we develop in this paper will target MI problems.

3.2 GRAPH ABSTRACTION OF SMT

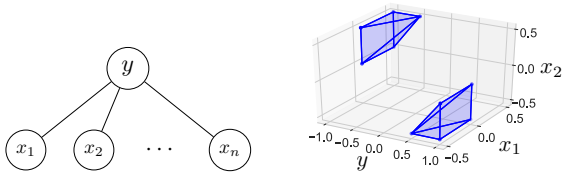
Primal graphs are often used to characterize variable dependencies. For the example Boolean CNF formula $\theta_B = (y \vee x_1) \wedge (y \vee x_2)$ the primal graph is shown in Figure 3a. Its edges encode that variable pairs (y, x_1) and (y, x_2) appear in the same clause, while (x_1, x_2) never appear together, and are thus independent given y. Similarly, we will use primal graphs for SMT theories to capture variable dependency information as follows.

Definition 3.2. ***(Primal graph of SMT)*** *The primal graph of an SMT($\mathcal{LRA}$) CNF is an undirected graph whose vertices are all variables and whose edges connect any two variables that appear in the same clause.*

Example 3.3. *Consider the following theory θ_n.*

$$\theta_n = \begin{cases} (-1 \leq y \leq 1) \wedge (-0.5 \leq x_1, \cdots, x_n \leq 0.5) \\ (x_i + 1 \leq y) \vee (y \leq x_i - 1), \text{ for all } i \in [n] \end{cases}$$

Figure 4 shows its primal graph and solution space.



(a) Primal graph of theory θ_n (b) Feasible region for θ_2

Figure 4: Primal graph and feasible region from Example 3.3.

While there are many flavors of *search-based* exact inference, including recursive conditioning (Darwiche, 2001), DPLL model counting (Sang et al., 2005), knowledge compilation (Chavira and Darwiche, 2008), and SumProd algorithms (Bacchus et al., 2009), we use the And/Or-search framework to illustrate the required concepts (Nilsson, 1982; Dechter and Mateescu, 2007).

The And/Or search algorithm for WMC problems recursively simplifies a discrete counting problem by alternating between two steps. The first (OR) step selects a Boolean variable and tries to instantiate it to both true and false (we will later see how to choose the variable). The second (AND) step finds ways of partitioning the WMC problem into independent sub-problems that can be solved separately. Such sub-problems are introduced by instantiating variables in the OR step in a way that creates independence. The OR step is also called the Shannon expansion. The AND step is also referred to as component caching (Sang et al., 2005) or detecting decomposability (Chavira and Darwiche, 2008).

This process is illustrated in Figure 3b for the earlier Boolean CNF θ_B. Circles denote OR-step variables whose square-node children are its instantiations. After instatiating y, the search tree creates independent problems for x_1 and x_2. This independence can be read off directly from the primal graph in Figure 3a. Search-based algorithms (with caching) are known to run efficiently on WMC problems with a tree or tree-like primal graph (Darwiche, 2009; Bacchus et al., 2009).

3.3 MODEL INTEGRATION IS ALL YOU NEED

This section casts hybrid WMI problems into MI problems over only real variables. We consider the case where per-literal weight functions are monomials – functions of the form $\beta x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over real variables x_i where $\beta \in \mathbb{R}$ and $\alpha_i \in \mathbb{N}$. We further assume that literals in $\mathcal{L}$ also appear in the theory, and that literals and their weights range over the same real variables.

We first show that any WMI problem with Boolean variables can be reduced to a WMI problem without Booleans. Then we show that WMI problems with per-literal weights can be reduced to an unweighted MI problem where the weight function is 1.

Proposition 3.4. *For each problem* $\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b})$ *there exists an equivalent problem* $\mathrm{WMI}(\theta', w' \mid \boldsymbol{x}')$ *without Boolean variables* $\boldsymbol{b}$ *such that*

$$\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \mathrm{WMI}(\theta', w' \mid \boldsymbol{x}')$$

and the primal graphs of θ *and* θ' *are isomorphic.*

This reduction encodes Boolean variables using fresh real variables and replaces each Boolean atom and its negation by two exclusive $\mathcal{LRA}$ atoms over those real variables. Proposition 3.4 allows us to focus on WMI problems without Boolean variables involved. Certain weight functions can also be reduced, as we show next.

Proposition 3.5. *For each problem* $\mathrm{WMI}(\theta, w \mid \boldsymbol{x})$ *with per-literal weights* w *as defined in this section, there exists an equivalent unweighted problem* $\mathrm{MI}(\theta' \mid \boldsymbol{x}')$ *s.t.*

$$\mathrm{WMI}(\theta, w \mid \boldsymbol{x}) = \mathrm{MI}(\theta' \mid \boldsymbol{x}').$$

Moreover, theories θ *and* θ' *have identical primal graph treewidth (Robertson and Seymour, 1986).*

This reduction encodes weights using auxiliary parameter variables. For each literal over which a weight function is defined, two clauses will be appended: if the literal holds, the MI over the auxiliary real variables equals the monomial weight function; otherwise, it equals one.

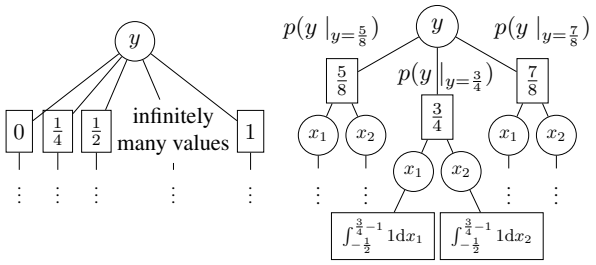
Crucially, both reductions can be constructed in polynomial time. Similar efficient reductions exist for arbitrary polynomial weight functions, but can slightly increase treewidth. Detailed descriptions of these reduction processes are included in Appendix A.3.

Example 3.6. *Consider SMT($\mathcal{LRA}$) theory* $(b \vee \neg b) \wedge \gamma_i$ *with literal set* $\mathcal{L}$ *and per-literal weight functions* $\mathcal{P}$ *as defined in Example 2.3. There exists an equivalent MI problem* $\mathrm{MI}(\delta \mid \boldsymbol{x} \cup \{\lambda_b, z_b, z_i^{(1)}, z_i^{(2)}\})$ *with a weight function of* 1 *and without Boolean variables. Its SMT($\mathcal{LRA}$) theory* δ *is shown below. Note that its primal graph remains a tree.*

$$\delta = \begin{cases} \gamma_i \wedge (-1 < \lambda_b < 1) \wedge_{j=1,2} \ (0 < z_i^{(j)} < \mathit{price}_i) \\ \quad \lambda_b > 0 \ \Rightarrow (0 < z_b < 1.5) \\ \neg(\lambda_b > 0) \Rightarrow (0 < z_b < 1). \end{cases}$$

4 SEARCH-BASED MI

The goal of our work is to take advantage of the independence structure in SMT($\mathcal{LRA}$) theories to reduce the computational cost of model integration. Our solution is to exploit context-specific independence by search.



(a) Infinite Search Tree (b) Our Finite Search Tree

Figure 5: Continuous search trees for θ_2 from Example 3.3.

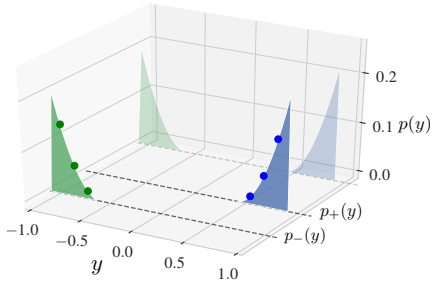


Figure 6: Piecewise polynomial $p(y)$ as defined in Proposition 4.1 for theory θ_2 from Example 3.3, whose integration is $\mathrm{MI}(\theta_2)$. The two polynomials $p_-(y)$ and $p_+(y)$ are unknown, but we can recover them from a finite number of points.

One obstacle is that to introduce independence in discrete search, we instantiate a variable with all values in its domain. Unfortunately, when the variable has a real domain (e.g., $y \in [0, 1]$), we cannot instantiate it with every value in its domain, since there are uncountably many (see Figure 5a). This basic limitation has precluded the use of search-based inference in continuous graphical models.

We overcome this problem by observing that MI is an integration over a piecewise polynomial, which can be fully recovered from a finite number of points. Specifically, for real variable y in theory θ, if we instantiate the variable y with a value α, then the MI of theory $\theta \wedge (y = \alpha)$ is the density of $\mathrm{WMI}(\theta, w)$ at $y = \alpha$. Recall that a polynomial function $p(y)$ with degree d defined over an interval I is uniquely defined by its values at $d+1$ distinct points in I, and that a closed-form expression for $p(y)$ can be recovered exactly and efficiently.

Consider again the theory γ_i from Example 2.1. As shown in Figure 1, function $f(\alpha) = \mathrm{MI}(\gamma_i \wedge (sqft_i = \alpha))$ is a piecewise polynomial with three intervals. We can recover all three polynomials from a finite number of points, and thus obtain the integration of $f(\alpha)$, that is, the model integration $\mathrm{MI}(\gamma_i)$. This motivates the search-based model integration algorithm we develop next.

4.1 VARIABLE INSTANTIATION

We first show that when per-literal weight functions $\mathcal{P}$ are polynomials, WMI of theory θ can be obtained by doing search with finite instantiations on real variables.

Proposition 4.1. *Let y be a real variable in SMT($\mathcal{LRA}$) theory θ. Suppose that per-literal weight functions $\mathcal{P}$ are polynomials. Then WMI is an integration over a univariate piecewise polynomial $p(y)$, that is,*

$$\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \int_I p(y)\mathrm{d}y \tag{1}$$

where piecewise polynomial $p(y)$ is integrated over set $I = \{y^* \mid \exists \hat{\boldsymbol{x}}^*, \exists \boldsymbol{b}^* \ s.t. \ \theta(y^*, \hat{\boldsymbol{x}}^*, \boldsymbol{b}^*)$ is SAT$\}$ with $\hat{\boldsymbol{x}}$ being the remaining real variables.

The set I is a union of disjoint supports for piecewise polynomial $p(y)$. We refer to these intervals as "pieces". To describe our MI algorithm, we first assume in this section that these intervals and their polynomial degrees are given. Our method to explicitly find these intervals and degrees will be given in Section 4.2.

Although Proposition 4.1 holds for WMI problems with polynomial per-literal weight functions in general, we use the insights from Section 3.3 to only focus on MI problems. For interval set I defined in Proposition 4.1, suppose we are given the interval pieces $[l, u] \in I$ and degrees d of their associated polynomials. If we instantiate variable y with $d+1$ distinct values in each piece $[l, u]$ of degree d, and solve any sub-problems recursively, we can recover polynomial $p_{l,u}(y)$ defined on interval $[l, u]$ by performing interpolation on $d + 1$ points. Finally, MI of the full theory θ can be computed as follows.

$$\mathrm{MI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \sum\nolimits_{[l,u]\in I} \int_l^u p_{l,u}(y)\mathrm{d}y. \tag{2}$$

For example, consider theory θ_2 from Example 3.3. We can interpret $\mathrm{MI}(\theta_2)$ as an integration over piecewise polynomial $p(y)$ whose intervals $[-1, -0.5]$ and $[0.5, 1]$ both have associated degree two. After instantiating y to three values in each interval, we get two independent sub-MI problems that contain variable x_1 and variable x_2 respectively. By solving these sub-problems, we obtain three points fitted by each polynomial $p_-(y)$ and $p_+(y)$ as shown in Figure 6. Therefore, we can recover both by polynomial interpolation and can obtain $\mathrm{MI}(\theta_2)$ by Equation 2. Figure 5b depicts the search space of our algorithm on interval $[0.5, 1]$.

The above discussion has shown that for MI problems, we can instantiate a real variable to finitely many values, decompose the problem into independent parts, and then solve the sub-problems recursively. Algorithm 1 follows exactly this strategy for search-based model integration. The role of pseudo trees will be explained in Section 4.3.

Algorithm 1 SMI: Search-Based Model Integration

Input: T: pseudo tree, θ: SMT($\mathcal{LRA}$) theory
Output: p: MI of theory θ

if T is a forest of trees T' **then**
  $\theta' \leftarrow$ sub-theories containing variables in T'
  return $\prod_{T'} SMI(T', \theta')$
$p = 0, y = root(T)$, $ST_y =$ set of subtrees below y
$I =$ PE_NODE(θ, y)
for all polynomial piece $\{[l, u], d\} \in I$ **do**
  select $d + 1$ distinct values α_i's in $[l, u]$
  $p_i \leftarrow SMI(ST_r, \theta\,|_{(y=\alpha_i)})$
  $p_{l,u}(y) \leftarrow$ polynomial interpolation on (α_i, p_i)'s
  $p \leftarrow p + \int_l^u p_{l,u}(y)\mathrm{d}y$
return p

Details on caching to speed up the algorithm are included in Appendix B. The remaining problem is how to exactly obtain pieces $[l, u]$ and their associated degrees d in function PE_NODE. We address this problem next.

4.2 FINDING PIECES VIA CRITICAL POINTS

Recall that by Proposition 4.1, WMI of SMT($\mathcal{LRA}$) theory θ can be rewritten as $\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \int_I p(y)\mathrm{d}y$ where $p(y)$ is a piecewise polynomial, set I is a union of disjoint support of polynomials in $p(y)$, and each piece $[l, u] \in I$ is associated with a polynomial degree d. We hope that when a real variable y in theory θ is chosen to be instantiated, we can exactly find all pieces and their associated degrees for piecewise polynomial $p(y)$.

It turns out that this can be achieved. While integrating over satisfying assignments with respect to a certain variable given an SMT($\mathcal{LRA}$) theory, integration upper bounds and lower bonds are defined by its literals. Changes in integration bounds give rise to different pieces of integration and therefore result in the piecewise nature of the polynomial in Proposition 4.1. In our method we determine these pieces by collecting points where certain bounds meet. Further, by propagating polynomial piece and degree information in a bottom-up manner along the primal graph, we can obtain the pieces and degree for the chosen piecewise polynomial.

We will first describe our method in a basic case where there are only two real variables in the theory. Then we extend this approach to theories with tree primal graphs.

4.2.1 Base Case: Pieces of Two Real Variables

First we investigate a simple case where there are only two real variables x and y in SMT($\mathcal{LRA}$) theory θ. Recall that we are solving an unweighted MI problem. We would like to find pieces and associated degrees for real variable y such that we can instantiate y as in Section 4.1:

$$p(y) = \int_{\theta(x,y)} 1\,\mathrm{d}x = \sum_{[l(y),u(y)]\in I(y)} \int_{l(y)}^{u(y)} 1\,\mathrm{d}x$$
$$= \sum_{[l(y),u(y)]\in I(y)} u(y) - l(y)$$

where set $I(y)$ is defined as

$$\{[l(y), u(y)] \mid \forall x \in [l(y), u(y)], \theta(x, y) \text{ is SAT}\}. \quad (3)$$

That is, for any fixed value y^*, the set $I(y^*)$ consists of intervals of consistent values for variable x. For any $[l(y), u(y)] \in I(y)$, it gives a pair of integration bounds for variable x. Further by integrating over x we can obtain a polynomial with respect to variable y.

Each piece $[l, u]$ corresponds to a certain class of values that gives the same symbolic integration bounds to variable x. The two values $y = l$ and $y = u$ are endpoints of the piece only if integration bound set $I(y)$ changes at these points, since the piecewise polynomial $p(y)$ is defined by these bounds. That is, for arbitrarily small ϵ, we have $I(l-\epsilon) \neq I(l+\epsilon)$, and it also holds at point $y = u$. We formally define critical points below.

Definition 4.2. ***(Critical Point)*** *Let θ be an SMT($\mathcal{LRA}$) theory with two real variables, and denote one of the real variables by y. Let $I(y)$ be an integration bound set as defined in Equation 3. Then $y = \alpha$ is a critical point if for arbitrarily small ϵ, it holds that $I(\alpha-\epsilon) \neq I(\alpha+\epsilon)$.*

Remark. The comparison of set $I(y)$ is done symbolically. That is, for two distinct values α, β, we say $I(\alpha) = I(\beta)$ if they have the same set of symbolic integration bounds. For example, if at $y = \alpha$, $I(y) = \{[1, y]\}$ and at $y = \beta \neq \alpha$, $I(x) = \{[1, y]\}$, it holds that $I(\alpha) = I(\beta)$. However, if at $y = \alpha$, $I(y) = \{[1, y]\}$ and at $y = \beta$, $I(y) = \{[y, 2]\}$, then we say $I(\alpha) \neq I(\beta)$.

Our idea is that, if we can find all critical points $y = \alpha$ where the set $I(y)$ changes, then we can partition real domains of y into disjoint intervals, such that any support of piecewise polynomial $p(y)$ is either one of these intervals or a union of some intervals. For the resulting interval $[l, u]$, we can apply an SMT($\mathcal{LRA}$) solver to $\theta' = \theta \wedge (l < y < u)$ to check whether it is a satisfiable piece of function $p(y)$; if this is true, we can obtain the polynomial degree of $p_{l,u}(y)$ defined over this piece by simply traversing theory θ'. We summarize this procedure as PE_EDGE in Algorithm 2 in Appendix C.

4.2.2 General Case: Pieces of Tree Structures

Given an SMT($\mathcal{LRA}$) theory θ with a tree-shape primal graph G, our goal is to enumerate pieces and their associated degrees for the root variable y, building on the

algorithm we developed in the base case above. This can be done in a bottom-up manner with tree primal graphs.

Specifically, we first partition theory θ into sub-theories $\theta_{r,c}$ and θ_{G_c} for each c, such that $\theta = \bigwedge_c(\theta_{r,c} \wedge \theta_{G_c})$, where variables c are the child variables of root r, and graph G_c is the sub-tree rooted at variable c. Each theory $\theta_{r,c}$ contains only variables r and c, on which we can apply the enumeration for the base case above, and each theory θ_{G_c} contains only variables in sub-tree G_c. This is possible provided that the primal graph of theory θ has a tree structure, which is why our algorithm is restricted to SMT($\mathcal{LRA}$) theories with tree-shaped primal graphs.

For each child variable c, we first obtain its pieces with respect to theory θ_{G_c} in a recursive way. Then we can apply our enumeration algorithm for two-variable theory PE_EDGE to theory $\theta_{r,c}$ with the given pieces of variable c. What we would get are sets of pieces for each child variable c. To be consistent with theory θ, we need to take intersections of these sets which we refer to as the shattering operation. Finally, the resulting intersections are pieces and polynomial degrees for root variable r. We provide more details of this procedure called PE_NODE in Algorithm 2 in Appendix C.

As described above, our piece enumeration algorithm is applicable to MI problems for theories with tree primal graphs. Moreover, it is also applicable to WMI problems whose SMT theory has a tree primal graph and whose per-literal weights are monomials as described in Section 3.3, since our reduction process can preserve the tree structure of the primal graph.

4.3 COMPLEXITY ANALYSIS

Inference over networks involving real variables raises considerable challenges for inference, and network structures that are tractable in the discrete case, such as polytrees, give rise to NP-hard inference problems in the hybrid case (Koller and Friedman, 2009). We show that the complexity of our algorithm is mainly exponentially bounded by the tree height of the primal graph.

Our search algorithm for MI needs to choose which variables to instantiate first. This choice can be based on a tree data structure that orders the variables. Such a tree characterizes the computational complexity as it does for discrete And/Or search algorithms. We first formally defined the tree that helps guide our search.

Definition 4.3. ***(Pseudo Tree)*** *Given an undirected graph G with vertices and edges (V, E_G), a pseudo tree for G is a directed rooted tree T with vertices and edges (V, E_T), such that any edge e that is in G but not in T must connect a vertex in T to one of its ancestors.*

That is, edge $e = (v_1, v_2)$ such that $e \in E_G$ and $e \notin E_T$ implies that either vertex v_1 is an ancestor of vertex v_2 in T or vertex v_2 is an ancestor of vertex v_1 in T. Note that the pseudo tree has the same set of vertices as G. Such a pseudo tree guides SMI (Algorithm 1) in deciding which variable to instantiate, and when to decompose.

Next, we analyze the complexity of SMI. Since our algorithm performs search, its time and space complexity is characterized by the size of its search space. Our analysis does not take caching improvements into consideration.

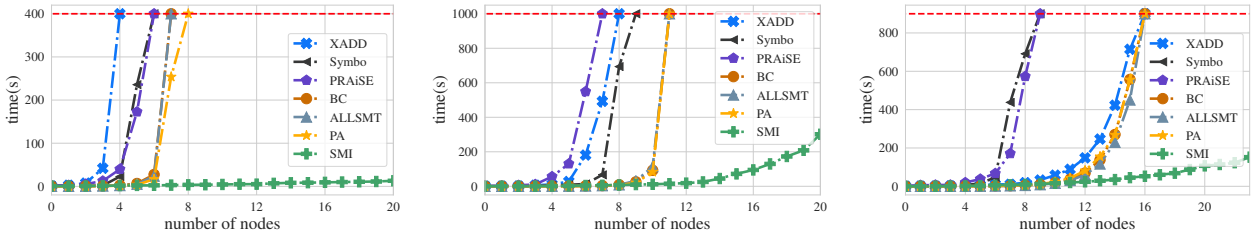
Theorem 4.4. ***(Size of Search Space)*** *Consider an SMT($\mathcal{LRA}$) theory θ with a tree-shaped primal graph with height h_p, and a pseudo tree T with l leaves and height h_t. Let m be the number of $\mathcal{LRA}$ literals in θ, and n be the number of real variables. Then the size of the SMI search space is $O(l \cdot (n^3 \cdot m^{h_p})^{h_t})$.*

Hence, we can conclude that the complexity of our algorithm is bounded exponentially by tree heights of both the primal graph and pseudo tree. In fact, for any tree-shaped primal graph, we can always choose a pseudo tree whose height h_t is $O(\log n)$ to guide the search (Dechter and Mateescu, 2007). Moreover, the number of leaves l in pseudo tree T is no larger than the number of nodes n. Thus, we have the following corollary.

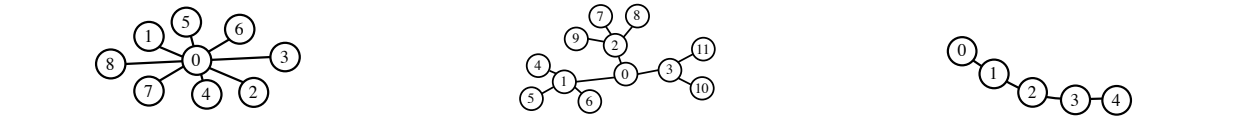
Corollary 4.5. *Following the notation in Theorem 4.4, with properly chosen pseudo tree T whose tree height h_t is $O(\log n)$, the size of the search space generated by SMI is $O\big(n^{1+3\log n + h_p \log m}\big)$.*

Therefore, the complexity of our algorithm is mainly decided by tree heights of primal graphs h_p. In the worst case when tree primal graphs have height $O(n)$ – for instance path graphs, whose tree height is n when rooted at the start node – then the worst-case complexity of our algorithm is $O(n^{n \log m})$ by Corollary 4.5. That is, the time complexity is worst-case super-exponential.

In cases when the tree primal graph has tree height of size $O(\log n)$, the complexity of our algorithm is $O(n^{1+(3+\log m)\log n})$ which is of quasi-polynomial complexity, and considered to be efficient. Trees with tree height in $O(\log n)$ are a general class of trees used in various models. Balancing trees like AVL trees and full k-ary trees are of tree height $O(\log n)$. Another example is a star graph, which has one internal node and all other nodes as leaves. This graph corresponds to the well-known naive Bayes structure for directed graphical models. It is the primal graph of a theory modeling independent variables predicting one and the same dependent (class) variable. The tree height of star graphs is constant 1 when choosing the internal node as root. Hence, our algorithm runs efficiently on such WMI problems.



(a) MI runtime on star primal graphs. (b) MI runtime on full 3-ary tree graphs. (c) MI runtime on path primal graphs.



(d) Star primal graph with nodes $n = 8$. (e) Full 3-ary tree with nodes $n = 12$. (f) Path primal graph with nodes $n = 5$.

Figure 7: (a)-(c) MI execution time on SMT($\mathcal{LRA}$) with tree primal graphs. (d)-(f) Example tree primal graphs.

5 EMPIRICAL EVALUATION

We analyze the performance of our search-based MI algorithm on SMT($\mathcal{LRA}$) theories with tree primal graphs. First, we show that our algorithm is efficient for theories whose primal graphs have constant tree heights, or tree heights of log scale w.r.t. the number of real variables n. For theories whose primal graph has tree height in $O(n)$ – the cases where our algorithm has super-exponential worst-case complexity in theory – empirical results show that our algorithm still runs efficiently. We also consider a more complex house price model where house sizes are dependent, as opposed to those in Example 3.1. Moreover, the house price model has non-trivial weight functions that our algorithm first reduces to a MI problem as outlined in Section 3.3. We compare our algorithm to alternative WMI solvers and conclude that it significantly outperforms existing solvers on these benchmarks.

Benchmarks We compare our algorithm (SMI) with other WMI solvers. The block-clause-strategy-based solver (BC) (Belle et al., 2015a) iteratively generates new models by adding the negation of the latest model to the formula for the following iteration. The all-satisfying-assignments-based solver (ALLSMT) (Belle et al., 2016) first generates the set of all $\mathcal{LRA}$-satisfiable total truth assignments on atoms that propositionally satisfy the theory. The implementation of de Salvo Braz et al. (2016) (PRAiSE) is a variable-elimination-based solver. The predicate-abstraction-based solver (PA) (Morettin et al., 2017) exploits the power of SMT-based predicate abstraction to reduce the number of models to be integrated over. Both the extended algebraic-decision-diagram-based solver (XADD) (Kolb et al., 2018) and sentential-decision-diagram-based solver (Symbo) (Zuidberg Dos Martires et al., 2019) use circuit-based compilation languages and exploits the circuit structures.

5.1 TREE PRIMAL GRAPHS

We investigate the performance of our algorithm on SMT($\mathcal{LRA}$) theories with three types of tree primal graphs: 1) star graphs, consisting of one center node connected to all other nodes, and no other connections; 2) full 3-ary trees, whose non-leaf vertices have exactly three children and all levels are full except for some rightmost position of the bottom level; 3) path graphs, consisting of linearly connected nodes. These structural constraints arise naturally in data and many probabilistic graphical modeling problems.

For each graph type, given a number of nodes n, we introduce n real variables $\boldsymbol{x} = \{x_0, x_1, \cdots, x_{n-1}\}$ with bounded domains $\forall i, (-1 \leq x_i \leq 1)$. Denote the graph by $G = (V, E)$ where $V = \{0, 1, \cdots, n-1\}$ is the vertex set and $E = \{(i, j), i, j \in V\}$ the edge set. We perform MI for the following theories and increasing n.

$$\theta(\boldsymbol{x}) = \begin{cases} \bigwedge_{i \in V} \ (-1 \leq x_i \leq 1) \\ \bigwedge_{(i,j) \in E} \ ((x_i + 1 \leq x_j) \vee (x_j \leq x_i - 1)) \end{cases}$$

Figure 7 shows example primal graphs and the execution time of experiments comparing SMI with baselines.

For MI over theories with all three types of tree primal graphs, our algorithm significantly outperform other WMI solvers in terms of execution time. The runtime curves of other solvers grow seemingly exponentially while our curve grows slowly with the number of real variables. For theories with star graphs and full 3-ary trees as primal graphs, the time curves of SMI are consistent with the complexity analysis in Section 4.3 stating that our algorithm has quasi-polynomial complexity. For theories with path graphs as primal graphs, which are

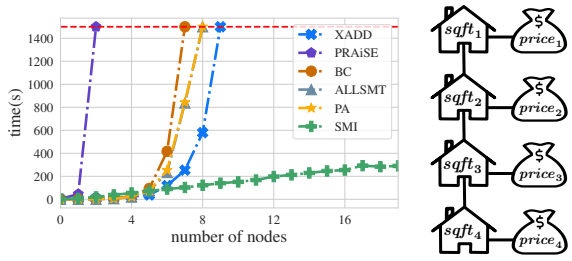


Figure 8: Runtime and primal graph for house price model.

still sparse graphs, we perform caching and the runtime curve grows slowly, even though our worst-case analysis allows for a super-exponential time complexity.

5.2 HOUSE PRICE SMT($\mathcal{LRA}$) MODEL

In Example 3.1 we performed MI for multiple houses based on extreme independence assumptions. Now we consider a more complicated case where houses are not independent and there are Boolean variables in the SMT($\mathcal{LRA}$) model. Moreover, we choose non-trivial per-literal weight functions in order to evaluate our algorithm for reducing WMI to unweighted MI problems.

Specifically, we consider n houses that are located along a street. Each house i has its price and square footage model as in Example 2.1. Also, we enforce the constraint that square footage between two neighboring houses should not vary too much and we use a Boolean variable b to indicate whether or not these houses are located in an urban area. This gives the following SMT theory.

$$\gamma_{street} = \begin{cases} (b \vee \neg b) \wedge \bigwedge_{i=1}^{n} \gamma_i \\ \bigwedge_{i=1}^{n-1} (sqft_i \leq sqft_{i+1} + offset) \end{cases}$$

with $offset$ a constant characterizing maximum difference in square footage between two neighboring houses. For weights w, consider the set of literals $\mathcal{L} = \{b\} \cup \{0 < price_i < 3000, i = 1, \cdots, n\}$ and per-literal weight functions $\mathcal{P} = \{p_b\} \cup \{p_{(0<price_i<3000)}, i = 1, \cdots, n\}$, with $p_b(\boldsymbol{x}) = 1.5$ and $p_{(0<price_i<3000)}(\boldsymbol{x}) = price_i^2$ for all i. Then, in worlds where all literals in $\mathcal{L}$ are satisfied, our weight function is $1.5 \prod_{i=1}^{n} price_i^2$. In worlds where b is false but other literals are satisfied, the weight function is $\prod_{i=1}^{n} price_i^2$. Figure 8 shows an example primal graph and WMI runtime for this house price model.

6 RELATED WORK

SMT (Barrett et al., 2010) has been one of the most prominent advances in automated reasoning and many efficient SMT solvers have been built (De Moura and Bjørner, 2008; Barrett et al., 2011; Cimatti et al., 2013; Dutertre, 2014). The counting version of SMT, that is #SMT, and in particular #SMT($\mathcal{LA}$) is a fundamental problem in quantitative program analysis (Liu and Zhang, 2011; Geldenhuys et al., 2012; Filieri et al., 2014; Phan et al., 2014; von Gleissenthall et al., 2015; Filieri et al., 2015). The #SMT($\mathcal{LA}$) problem is #P-hard, as is model counting (Valiant, 1979). Other first-order hybrid probability models have been proposed, usually based on sampling inference (Ravkic et al., 2015).

SGDPLL(T) is an algorithm for solving probabilistic inference modulo theories while also generating simpler sub-problems (de Salvo Braz et al., 2016). It performs case analysis on SMT literals, whereas SMI instead operates on continuous theory variables. Similar to our observation that WMI problems can be reduced to MI problems, Chakraborty et al. (2015) propose a method to reduce WMC to unweighted model counting. Although the focus of this paper is on exact inference, there also exist notable approximate solutions to #SMT($\mathcal{LA}$) and WMI (Ma et al., 2009; Belle et al., 2015b; Chakraborty et al., 2016; Chistikov et al., 2017).

Morettin et al. (2017) enumerate integrable spaces by predicate abstraction and allow general weight functions. Kolb et al. (2018) use case functions as weights, which still permits compilation into XADD circuits. Weight functions in these two cases are not consistent with the factorization structure of the SMT sentence. The factorization structure is a crucial aspect of efficient inference, and its isolation to the logical part of WMC/WMI is considered to be an advantage, facilitating solver building. Our definition of factorized weight functions is similar to Belle et al. (2015a) and Zuidberg Dos Martires et al. (2019). Belle et al. (2016) exploit independence in WMI problems that are exactly equivalent to WMC problems.

7 Conclusions

This paper proposed a search-based WMI algorithm that exploits structural independence properties to improve efficiency. For WMI on SMT($\mathcal{LRA}$) theories with tree primal graphs and piecewise polynomial weight functions, our algorithm decomposes WMI problems during search. A complexity analysis showed that for balanced tree primal graphs, our algorithm yields quasi-polynomial complexity. Experimental comparisons confirmed a drastic efficiency improvement over baselines.

Acknowledgements The authors would like to thank Brendan Juba, Andrea Passerini, and Roberto Sebastiani for valuable discussions. This work is partially supported by NSF grants #IIS-1657613, #IIS-1633857, #CCF-1837129, DARPA XAI grant #N66001-17-2-4032, NEC Research, and gifts from Intel and Facebook Research.

References

Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. Fairsquare: Probabilistic verification of program fairness. *Proc. ACM Program. Lang.*, (OOPSLA):80:1–80:30, 2017.

Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.

Velleda Baldoni, Nicole Berline, Jesus De Loera, Matthias Köppe, and Michèle Vergne. How to integrate a polynomial over a simplex. *Mathematics of Computation*, 80(273):297–325, 2011.

Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.

Clark Barrett, Leonardo de Moura, Silvio Ranise, Aaron Stump, and Cesare Tinelli. The smt-lib initiative and the rise of smt (hvc 2010 award talk). In *Proceedings of the 6th international conference on Hardware and software: verification and testing*, pages 3–3. Springer-Verlag, 2010.

Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. Cvc4. In *International Conference on Computer Aided Verification*, pages 171–177. Springer, 2011.

Vaishak Belle. Weighted model counting with function symbols. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2770–2776, 2015a.

Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. Hashing-based approximate probabilistic inference in hybrid domains. In *UAI*, pages 141–150, 2015b.

Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. Component caching in hybrid domains with piecewise polynomial densities. In *AAAI*, pages 3369–3375, 2016.

Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

Supratik Chakraborty, Daniel J Fremont, Kuldeep S Meel, Sanjit A Seshia, and Moshe Y Vardi. Distribution-aware sampling and weighted model counting for sat. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Supratik Chakraborty, Kuldeep S Meel, Rakesh Mistry, and Moshe Y Vardi. Approximate probabilistic inference via word-level counting. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. 2008.

Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in smt and value estimation for probabilistic programs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 320–334. Springer, 2015.

Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in smt and value estimation for probabilistic programs. *Acta Informatica*, 54 (8):729–764, 2017.

Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 121–132. Springer, 2013.

Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The mathsat5 smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer, 2013.

Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

Jesús A De Loera, Brandon Dutra, Matthias Koeppe, Stanislav Moreinis, Gregory Pinto, and Jianqiu Wu. Software for exact integration of polynomials over polyhedra. *Computational Geometry*, 46(3):232–252, 2013.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

Rodrigo de Salvo Braz, Ciaran O'Reilly, Vibhav Gogate, and Rina Dechter. Probabilistic inference modulo theories. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3591–3599. AAAI Press, 2016.

Rina Dechter and Robert Mateescu. And/or search spaces for graphical models. *Artificial intelligence*, 171(2-3):73–106, 2007.

Bruno Dutertre. Yices 2.2. In *International Conference on Computer Aided Verification*, pages 737–744. Springer, 2014.

Stefano Ermon, Carla P Gomes, Ashish Sabharwal, and Bart Selman. Embed and project: Discrete sampling with universal hashing. In *Advances in Neural Information Processing Systems*, pages 2085–2093, 2013.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

Antonio Filieri, Corina S Pasareanu, and Willem Visser. Reliability analysis in symbolic pathfinder: A brief summary. 2014.

Antonio Filieri, Corina S Pasareanu, and Guowei Yang. Quantification of software changes through probabilistic symbolic execution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 703–708. IEEE, 2015.

Jaco Geldenhuys, Matthew B Dwyer, and Willem Visser. Probabilistic symbolic execution. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pages 166–176. ACM, 2012.

Carla P. Gomes. Model counting. *Handbook of Satisfiability*, 20, 2009.

Samuel Kolb, Martin Mladenov, Scott Sanner, Vaishak Belle, and Kristian Kersting. Efficient symbolic integration for probabilistic inference. In *IJCAI*, pages 5031–5037, 2018.

Daphne Koller and Nir Friedman. Probabilistic graphical models. 2009.

Sheng Liu and Jian Zhang. Program analysis: from qualitative analysis to quantitative analysis (nier track). In *2011 33rd International Conference on Software Engineering (ICSE)*, pages 956–959. IEEE, 2011.

Feifei Ma, Sheng Liu, and Jian Zhang. Volume computation for boolean combination of linear arithmetic constraints. In *International Conference on Automated Deduction*, pages 453–468. Springer, 2009.

Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Efficient weighted model integration via smt-based predicate abstraction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 720–728. AAAI Press, 2017.

Nils J Nilsson. Principles of artificial intelligence. *Symbolic Computation, Berlin: Springer, 1982*, 1982.

Quoc-Sang Phan, Pasquale Malacaria, Corina S Păsăreanu, and Marcelo d'Amorim. Quantifying information leaks using reliability analysis. In *Proceedings of the 2014 International SPIN Symposium on Model Checking of Software*, pages 105–108. ACM, 2014.

Irma Ravkic, Jan Ramon, and Jesse Davis. Learning relational dependency networks in hybrid domains. *Machine Learning*, 100(2):217–254, 2015.

Neil Robertson and P.D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986. ISSN 0196-6774.

Tian Sang, Paul Beame, and Henry A Kautz. Performing bayesian inference by weighted model counting. In *AAAI*, volume 5, pages 475–481, 2005.

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

Guy Van den Broeck and Dan Suciu. *Query Processing on Probabilistic Data: A Survey*. Foundations and Trends in Databases. Now Publishers, August 2017.

Klaus von Gleissenthall, Boris Köpf, and Andrey Rybalchenko. Symbolic polytopes for quantitative interpolation and verification. In *International Conference on Computer Aided Verification*, pages 178–194. Springer, 2015.

Pedro Miguel Zuidberg Dos Martires, Anton Dries, and Luc De Raedt. Exact and approximate weighted model integration withprobability density functions using knowledge compilation. In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI Press, 2019.

A PROOFS

A.1 PROOF OF PROPOSITION 3.4

Proof. **(Proof of Proposition 3.4)**

Consider the most basic case when there is only one Boolean variable b in theory θ. Let θ' be an SMT($\mathcal{LRA}$) theory defined as follow

$$\theta' = \theta\{b : \lambda_b\} \wedge (-1 \leq \lambda_b \leq 1)$$

where $\theta\{b : \lambda_b\}$ is obtained by replacing all atom b by $0 < \lambda_b$ and replacing all its negation $\neg b$ by $\lambda_b < 0$ in theory θ.

Recall that weight functions are defined by a set of literals $\mathcal{L}$ and a set of per-literal weight functions $\mathcal{P} = \{p_\ell(\boldsymbol{x})\}_{\ell \in \mathcal{L}}$. When a literal ℓ is satisfied in a world, denoted by $\boldsymbol{x} \wedge \boldsymbol{b} \models \ell$, weights are defined as follows

$$w(\boldsymbol{x}, \boldsymbol{b}) = \prod_{\substack{\ell \in \mathcal{L} \\ \boldsymbol{x} \wedge \boldsymbol{b} \models \ell}} p_\ell(\boldsymbol{x}).$$

Let $\mathcal{L}'$ be a set of literals obtained by replacing Boolean literal b by $0 < \lambda_b$ and replacing its negation $\neg b$ by $\lambda_b < 0$ in theory θ as we do for theory. For the set of per-literal weight functions $\mathcal{P}'$, we define it for introduced real variable λ_b by $p_{(\lambda_b > 0)} = p_b$ and $p_{(\lambda_b < 0)} = p_{\neg b}$.

Then we have that for any $\boldsymbol{x}^*$,

$$w'(\boldsymbol{x}^*, \lambda_b) = \begin{cases} w(\boldsymbol{x}^*, b), & 1 > \lambda_b > 0 \\ w(\boldsymbol{x}^*, \neg b), & -1 < \lambda_b < 0 \end{cases}$$

By definition of WMI, we write $\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b})$ in its integration form as follows.

$$\begin{aligned} &\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) \\ &= \int_{\theta(\boldsymbol{x}, b)} w(\boldsymbol{x}, b) d\boldsymbol{x} + \int_{\theta(\boldsymbol{x}, \neg b)} w(\boldsymbol{x}, \neg b) d\boldsymbol{x} \end{aligned}$$

For the first term in the above equation, we can rewrite it s.t. Boolean variable b is replaced by real variable λ_b in the following way.

$$\begin{aligned} \int_{\theta(\boldsymbol{x}, b)} w(\boldsymbol{x}, b) d\boldsymbol{x} &= \int_0^1 \int_{\theta(\boldsymbol{x}, b)} w(\boldsymbol{x}, b) d\boldsymbol{x} d\lambda_b \\ &= \int_{\theta'(\boldsymbol{x}, \lambda_b)} w'(\boldsymbol{x}, \lambda_b) d\boldsymbol{x} d\lambda_b \end{aligned}$$

By doing this to the other integration term of $\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b})$, and also by the definition of WMI, we finally obtain that

$$\mathrm{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \mathrm{WMI}(\theta', w' \mid \boldsymbol{x}')$$

where $\boldsymbol{x}' = \boldsymbol{x} \cup \{\lambda_b\}$ is a set of real variables. The proof above can be easily adapted to multiple Boolean variable cases, which proves our proposition. □

A.2 PROOF OF PROPOSITION 3.5

Proof. **(Proof of Proposition 3.5)** To start with, we consider SMT($\mathcal{LRA}$) theory θ with no Boolean variables with a simple weight function w where the set of literal $\mathcal{L} = \{\ell\}$ has only one literal and literal weight function $p_\ell(\boldsymbol{x}) = \prod_{i=0}^n x_i^{p_i}$.

Claim A.1. *For a monomial function $f(\boldsymbol{x}) = \prod_{i=0}^n x_i^{p_i}$, let $\theta_f = \bigwedge_{i=0}^n \bigwedge_{j=1}^{p_i} (0 \leq z_j^i \leq x_i)$. Then we have the monomial $f(\boldsymbol{x}) = MI(\theta_f \mid \boldsymbol{z}; \boldsymbol{x})$, where $\boldsymbol{z}$ is the set of real variables z_j^i in theory θ_f, and $\boldsymbol{x}$ is parameters of theory θ_f.*

Let $\theta' = \theta \wedge (\ell \Rightarrow \theta_p) \wedge (\neg\ell \Rightarrow \hat{\theta}_p)$ where $p = p_\ell$ for brevity, θ_p is as defined in Claim A.1 and $\hat{\theta}_p := \bigwedge_{i=0}^n \bigwedge_{j=1}^{p_i} (0 \leq z_j^i \leq 1)$. Then we can rewrite $\mathrm{WMI}(\theta, w \mid \boldsymbol{x})$ as MI problem by Claim A.1 as follows.

$$\begin{aligned} &\mathrm{WMI}(\theta, w \mid \boldsymbol{x}) = \int_{\theta(\boldsymbol{x})} w(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} p(\boldsymbol{x}) d\boldsymbol{x} + \int_{\theta(\boldsymbol{x}) \wedge \neg\ell(\boldsymbol{x})} 1 d\boldsymbol{x} \\ &= \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} MI(\theta_p \mid \boldsymbol{z}; \boldsymbol{x}) d\boldsymbol{x} + \int_{\theta(\boldsymbol{x}) \wedge \neg\ell(\boldsymbol{x})} 1 d\boldsymbol{x} \\ &= \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} \int_{\theta_p(\boldsymbol{z})} 1 d\boldsymbol{z} d\boldsymbol{x} + \int_{\theta(\boldsymbol{x}) \wedge \neg\ell(\boldsymbol{x}) \wedge \hat{\theta}_p} 1 d\boldsymbol{x} d\boldsymbol{z} \\ &= \mathrm{MI}(\theta \wedge (\ell \Rightarrow \theta_p) \wedge (\neg\ell \Rightarrow \hat{\theta}_p) \mid \boldsymbol{x}, \boldsymbol{z}) \end{aligned}$$

Take $\boldsymbol{x}' = \boldsymbol{x} \cup \boldsymbol{z}$ then the proposition holds. The proof can be easily adapted for monomials with non-trivial coefficient by inducing more real variables z. It also holds for more general weight functions with literal set $\mathcal{L} = \{\ell_i\}_{i=1}^k$ and set of monomial per-literal weight functions $\mathcal{P} = \{p_{\ell_i}\}_{i=1}^k$, by taking theory θ' as follows which completes the proof of proposition.

$$\theta' = \theta \wedge \bigwedge_{i=1}^k (\ell_i \Rightarrow \theta_{p_{\ell_i}}) \wedge \bigwedge_{i=1}^k (\neg\ell_i \Rightarrow \hat{\theta}_{p_{\ell_i}}).$$

□

Proof. **(Proof of Claim A.1)** By definition of theory θ_f,

$$\begin{aligned} MI(\theta_f \mid \boldsymbol{z}; \boldsymbol{x}) &= \int_{\theta_f(\boldsymbol{z})} 1 d\boldsymbol{z} \\ &= \prod_{i=1}^{n} \prod_{j=1}^{p_i} \int_0^{x_i} 1 dz_j^i \\ &= \prod_{i=1}^{n} \prod_{j=1}^{p_i} x_i = \prod_{i=1}^{n} x_i^{p_i} = f(\boldsymbol{x}). \end{aligned}$$

□

A.3 REDUCTION TO MI WITH POLYNOMIAL WEIGHTS

The reduction from WMI problems to MI problems in Proposition 3.5 can also be done for arbitrary polynomial weight functions but can increase treewidth of primal graphs. We give a formal description on this reduction as follows.

Let θ be an SMT($\mathcal{LRA}$) theory with no Boolean variables with weight functions where the set of literal $\mathcal{L} = \{\ell\}$ has only one literal and literal weight function is a polynomial, denoted by $p(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i f_i(\boldsymbol{x})$ with each f_i a monomial function.

It has been shown in the proof of Proposition 3.5 in Section A.2 that for each monomial function f_i, there exist two SMT($\mathcal{LRA}$) theories θ_i and $\hat{\theta}_i$ such that $\text{MI}(\theta_i \mid \boldsymbol{z}_i; \boldsymbol{x}) = f_i(\boldsymbol{x})$ and $\text{MI}(\hat{\theta}_i \mid \boldsymbol{z}_i; \boldsymbol{x}) = 1$.

Let's define theories $\theta_i' = \theta_i \wedge (0 < v_i < \alpha_i)$ and $\hat{\theta}_i' = \hat{\theta}_i \wedge (0 < v_i < 1)$ with parameter variables v_i. Also define an indicator variable λ with real domain $[0, k]$ and literals $\ell_i = i - 1 < \lambda < i$ with $i \in \{1, 2, \cdots, k\}$. Then we have that for an SMT($\mathcal{LRA}$) theory θ' defined as follows, it holds that $\text{WMI}(\theta, w \mid \boldsymbol{x}) = \text{MI}(\theta' \mid \boldsymbol{x}, \boldsymbol{z})$ with $\boldsymbol{z}$ denoting all auxiliary variables.

$$\theta' = \theta \wedge (\ell \iff \vee_{i=1}^{k} \ell_i) \bigwedge_{i=1}^{k} (\ell_i \Rightarrow \theta_i') \bigwedge_{i=1}^{k} (\neg \ell_i \Rightarrow \hat{\theta}_i')$$

Why the WMI problem and the MI problem are equal can be proved by the following observations.

$$\text{WMI}(\theta, w \mid \boldsymbol{x}) = \int_{\theta(\boldsymbol{x})} w(\boldsymbol{x}) d\boldsymbol{x} \tag{4}$$

$$= \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} p(\boldsymbol{x}) d\boldsymbol{x} + \int_{\theta(\boldsymbol{x}) \wedge \neg \ell(\boldsymbol{x})} 1 d\boldsymbol{x} \tag{5}$$

For the first term in Equation 5, we have that

$$\begin{aligned} \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} p(\boldsymbol{x}) d\boldsymbol{x} &= \sum_{i=1}^{k} \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x})} \alpha_i f_i(\boldsymbol{x}) d\boldsymbol{x} \\ &= \sum_{i=1}^{k} \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x}) \wedge \ell_i} \alpha_i f_i(\boldsymbol{x}) d\boldsymbol{x} d\lambda \\ &= \sum_{i=1}^{k} \int_{\theta(\boldsymbol{x}) \wedge \ell(\boldsymbol{x}) \wedge \ell_i \wedge \theta_i} 1 d\boldsymbol{x} d\boldsymbol{z} \\ &= \text{MI}(\theta' \wedge \ell \mid \boldsymbol{x}, \boldsymbol{z}) \end{aligned}$$

Also for the second term in Equation 5, it equates to $\text{MI}(\theta' \wedge \neg \ell \mid \boldsymbol{x}, \boldsymbol{z})$. Therefore, reduction from the WMI problem to the MI problem holds. Although the reduction process we show here is for theories with one polynomial weight function, this process can be generalized to theories with multiple polynomial weight functions with little modification.

A.4 PROOF OF PROPOSITION 4.1

Proof. **(Proof of Proposition 4.1)** It follows from definition of WMI. Denote the set of real variables $\boldsymbol{x} \backslash \{y\}$ by $\hat{\boldsymbol{x}}$. From the definition of WMI in Equation 2.2, we can obtain the following partial derivative of WMI of theory θ w.r.t. variable y.

$$\begin{aligned} &\frac{\partial}{\partial x} \text{WMI}(\theta, w \mid \boldsymbol{x}, b) \mid_{y=y^*} \\ &= \sum_{\mu \in \mathbb{B}^m} \int_{\theta(y^*, \hat{\boldsymbol{x}}, \mu)} w(y^*, \hat{\boldsymbol{x}}, \mu) d\hat{\boldsymbol{x}} \end{aligned}$$

where the variable y is fixed to value y^* in weight function, μ are total truth assignments to Boolean variables as defined before. The weight function is integrated over set $\{\hat{\boldsymbol{x}}^* \mid \theta(y^*, \hat{\boldsymbol{x}}^*, \mu) \text{ is } \texttt{true}\}$. We define $p(y)$ as follows

$$p(y) := \sum_{\mu \in \mathbb{B}^m} \int_{y, \theta(\hat{\boldsymbol{x}}, \mu)} w(y, \hat{\boldsymbol{x}}, \mu) d\hat{\boldsymbol{x}}$$

Since weight functions w are piecewise polynomial, function $p(y)$ is a univariate piecewise polynomial $p(y)$, and $\text{WMI}(\theta, w \mid \boldsymbol{x}, b)$ is an integration over $p(y)$, which finishes our proof. □

A.5 PROOF OF THEOREM 4.4

Claim A.2. *For each path in the primal graph that starts with the root and ends with a leaf, and each real variable in path with height i, its number of polynomial pieces is $O(n \cdot c^{i+1})$.*

Algorithm 2 Polynomial pieces and degree enumeration algorithms

a) PE_EDGE – For Two Variable Theory
Input: θ: SMT($\mathcal{LRA}$) theory with two real variables
I : interval and degree tuples of variable x
Output: I_y: pieces and degrees for variable y

$B \leftarrow$ collect integration bounds on variable x
$Y \leftarrow y$ values where two bounds in B meet
for all interval $[l, u]$ resulting from Y **do**
    $\theta' \leftarrow \theta \wedge (l \leq y \leq u)$
    if θ' is SAT **then**
        $\{l(y), u(y), d\} \leftarrow$ get_bound_degree(x,θ',I)
        $d' \leftarrow \text{argmax}_d$ get_degree($l(y), u(y), d$)})
        $I_y \leftarrow I_y \cup ([l, u], d')$
Return I_y

b) PE_NODE – For Tree Primal Graph
Input: θ: SMT theory with tree primal graph
G: primal graph for theory θ
Output: I_y: interval and degree tuples of root variable y

if root y has no child **then**
    $I_y \leftarrow$ get_bound_degree(θ)
return I_y
$\theta_{y,c}$'s, θ_{G_c}'s $\leftarrow$ partition SMT($\mathcal{LRA}$) theory θ
for all child c **do**
    $I_c \leftarrow \text{PE\_NODE}(\theta_c, G_c)$
    $I_y^c \leftarrow \text{PE\_EDGE}(\theta_{y,c}, I_c)$
Return $I_y = shatter\big(\{I_y^c\}_c\big)$

Proof. The proof can be done by mathematical induction. Denote the real variable with height i in the path by x_i For $i = 0$, since the number of $\mathcal{LRA}$ literals is c, then there are at most c critical points for real variable x_0 and therefore there are at most $c + 1$ polynomial pieces for x_0.

Suppose that the claim holds for i, that is, the number of polynomial pieces for x_i is $O(n \cdot c^{i+1})$. To obtain critical points for variable x_{i+1}, we collect integration bounds on variable x_i whose size is $O(n \cdot c^{i+1})$ by assumption. Since the critical points of variable x_{i+1} are obtained by solving $b_1 = b_2$ w.r.t. variable x_{i+1} for b_1, b_2 in bounds on variable x_i, where there are at most c bounds containing x_{i+1} and the rest bounds are numerical ones, there are at most $O(n \cdot c^{i+2})$ solutions. Therefore, the number of polynomial pieces for x_{i+1} is $O(n \cdot c^{i+2})$, which finishes our proof. □

Proof. **(Proof of Theorem 4.4)** Let p be an arbitrary path in the pseudo tree T that starts with the root and ends with a leaf. Denote the maximum polynomial degree in weight functions by d. By Claim A.2 for each variable, it has at most $O(n \cdot c^{h_p})$ polynomial pieces. Moreover from Prop. 4.1, polynomials defined over each pieces have at most $n(d + h)$ polynomial degree. Therefore the set of values chosen to do instantiation on a certain real variable has size $O(n^3 \cdot c^{h_p})$ and each path p induces a search space with size $O((n^3 \cdot c^{h_p})^{h_t})$ since length of each path is bounded by h_t.

The pseudo tree T is covered by l such directed paths. The union of their individual search spaces covers the whole search space, where every distinct full path in the search space appears exactly once. Therefore, the size of the search space is bounded by $O(l \cdot (n^3 \cdot c^{h_p})^{h_t})$. □

B CACHING

Our algorithm allows caching in two sense. The first is the caching of pieces, i.e. intervals and polynomial degrees obtained from child nodes, which can be considered as constraints from child nodes. The pieces of a certain nodes is decided both by instantiation values from its father node as well as pieces from child nodes. Although we instantiate root nodes with distinct values, the constraints from child nodes for a certain node remains unchanged as long as they have the same father-child relation in subtree.

Another case where caching is possible is values of $p(y)$ as defined in Prop. 4.1 at instantiations of variable x. This is possible because for a certain node, its pieces resulting from different instantiation values of its grandfather node might intersects. This is especially helpful when there is a long path in primal graphs and caching can save a lot computational effort.

C PIECE ENUMERATION ALGORITHM

We summarize piece enumeration algorithms for two variable theory and for theory with tree primal graphs as described in Section 4.2 in Algorithm 2. Both get_bound_degree and get_degree are trivial operations for specifying integration bounds and polynomial degree. They are applied when the magnitude order of integration bounds are fixed and thus they can be done by scanning through related theories.