Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Explanation-Based Approximate Weighted Model Counting for Probabilistic Logics

Joris Renkens, Angelika Kimmig,
Guy Van den Broeck, Luc De Raedt
KU Leuven
Celestijnenlaan 200A
3001 Heverlee - Belgium

Abstract

Probabilistic inference can be realized using weighted model counting. Despite a lot of progress, computing weighted model counts exactly is still infeasible for many problems of interest, and one typically has to resort to approximation methods. We contribute a new bounded approximation method for weighted model counting based on probabilistic logic programming principles. Our bounded approximation algorithm is an anytime algorithm that provides lower and upper bounds on the weighted model count. An empirical evaluation on probabilistic logic programs shows that our approach is effective in many cases that are currently beyond the reach of exact methods.

1 Introduction

During the last few years there has been a significant interest in statistical relational learning (Getoor and Taskar 2007; De Raedt et al. 2008) and probabilistic databases (Dalvi and Suciu 2004), leading to the definition of many expressive probabilistic logical languages such as Markov Logic (Richardson and Domingos 2006), PRISM (Sato 1995) and ProbLog (De Raedt, Kimmig, and Toivonen 2007). It is by now well-known that probabilistic inference for such languages corresponds to weighted model counting (WMC) on weighted logical formulae (Darwiche 2009; Fierens et al. 2011). Knowledge compilation (Darwiche and Marquis 2002) is an effective and popular technique for WMC. It essentially compiles the weighted logical formulae into logical circuits on which inference can be performed efficiently.

Despite a lot of progress in knowledge compilation and other probabilistic inference techniques, inference remains the bottleneck when facing real-life problems and doing applications. One often has to resort to approximate inference methods, typically employing some form of sampling. While sampling is often effective, current sampling methods do not give strong guarantees (in the form of bounds on the computed probabilities).

The key contribution of this paper is the introduction of a novel anytime approximate inference method for WMC using knowledge compilation that provides bounds on the computed probabilities. The technique is inspired by probabilistic logic programming techniques due to (Poole 1993; De Raedt, Kimmig, and Toivonen 2007; Renkens, Van den Broeck, and Nijssen 2012), where bounds were derived based on a subset of proofs (or explanations) instead of all proofs, with the actual subset depending on the search strategy employed. Rather than taking a logic programming perspective, however, this paper takes a weighted model counting and knowledge compilation perspective. This potentially opens the way to develop WMC approximation techniques for other types of probabilistic formalisms such as Markov Logic and Bayesian networks.

Our technique is empirically evaluated on a number of benchmark problems. The experiments provide evidence that it is effective for probabilistic languages working under Sato's distribution semantics (Sato 1995), whose probability distribution over possible worlds is defined using a basic distribution over extensional probabilistic facts (sometimes called atomic choices) which then, together with a logic program, induces a distribution over the intensionally defined atoms. Thus, under Sato's distribution semantics, the truth-values of a limited number of extensional probabilistic facts determine the probability of the possible worlds. This is unlike Markov Logic, where most formulae are soft and the truth-value of one atom therefore typically depends on that of almost all other atoms.

The structure of the paper is as follows. We first formalize the problem and discuss the basic solution idea in Section 2. Section 3 defines the central concept of *explanation*, and Section 4 introduces the algorithm as used in the experiments in Section 7. Section 5 positions our approach with respect to WMC in general, and Section 6 discusses related work.

2 Background

Many probabilistic programming languages, including PRISM (Sato 1995), ICL (Poole 1997), ProbLog (De Raedt, Kimmig, and Toivonen 2007) and LPADs (Vennekens, Verbaeten, and Bruynooghe 2004), are based on Sato's distribution semantics (Sato 1995). They start from a set of probabilistic independent choices (ground atoms) and define further derived predicates using logic programs. In such a program, each ground atom corresponds to a random variable. We will call the probabilistic choices extensional variables E and the derived atoms intensional variables I.

Example 1. Let us define two extensional variables `head1`

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

and `head2`, representing two biased coin flips. The game is won when the two coin flips have the same outcome. We use the ProbLog notation where the extensional variables are ground facts annotated with their probability and the intensional variables are defined using a logic program.

```
0.4::head1.
0.7::head2.
twoHeads :- head1, head2.
twoTails :- not head1, not head2.
win :- twoHeads; twoTails.
```

The probability of each truth assignment to the extensional variables is obtained as the product of the probability of the individual assignments. If the fact is true, it contributes the corresponding probability; if it is false it contributes 1 minus the probability of the fact. The truth values for the intensional variables are derived by logical deduction using the logic program. This process results in a model of the theory. A typical query in probabilistic logic programming languages asks for the marginal probability of a query atom q given the theory Δ: $P(\mathrm{q} \mid \Delta)$, which is defined as the sum of the weights of the models of the theory in which the query is true. It is well known that this probability can be computed using weighted model counting.

Definition 1. (Weighted Model Count) Let Φ be a propositional logic formula over variables V, w be a weight function that assigns a positive (real) weight value to each literal v and $\neg v$ (with $v \in \mathrm{V}$), and $\mathcal{I}(\mathrm{V})$ the set of interpretations of V, that is, the set of all possible truth value assignments to variables in V. The *weighted model count (WMC)* of Φ is

$$\mathrm{WMC}_{\mathrm{V}}(\Phi) = \sum_{i \in \mathcal{I}(\mathrm{V}), i \models \Phi} \prod_{l \in i} \mathrm{w}(l).$$

The probability $P(\mathrm{q} \mid \Delta)$ can be computed as $\mathrm{WMC}(\Phi)$, where Φ is a propositional representation of $\mathrm{q} \wedge \Delta$, and the weight function is $w(v) = p_v$ and $w(\neg v) = 1 - p_v$ for the variables $v \in \mathrm{E}$, and $w(v) = w(\neg v) = 1$ for the variables $v \in \mathrm{I}$, cf. (Fierens et al. 2011) for more details on this transformation. In the remainder of this paper, when the set of variables over which to do WMC is omitted, we assume it consists of the variables present in the formula.

Example 2. In our example, we obtain the following propositional formula (where we abbreviate names of propositional variables based on the names of logical atoms) and weight function for WMC.

$$\begin{aligned} th &\leftrightarrow h1 \wedge h2 \\ tt &\leftrightarrow \neg h1 \wedge \neg h2 \\ win &\leftrightarrow th \vee tt \end{aligned}$$

	true	*false*
h1	0.4	0.6
h2	0.7	0.3
th	1	1
tt	1	1
win	1	1

$\psi \leftarrow \mathit{false}$
while $\mathit{time} < \mathit{maxTime}$ **do**
  $\mathrm{e} \leftarrow \mathrm{nextExpl}(\Delta \wedge \mathrm{q})$
  $\psi \leftarrow \psi \vee \mathrm{e}$
end
return $\mathrm{WMC}_{V(\psi)}(\psi)$

Algorithm 1: Computing a lower bound on $\mathrm{WMC}(\Delta \wedge \mathrm{q})$ given a time budget $\mathit{maxTime}$.

A popular and effective approach to WMC is by knowledge compilation: Φ is compiled into a graph structure on which the WMC can be computed efficiently (Darwiche and Marquis 2002). However, since WMC is a #P-complete problem, compilation often is infeasible in practice, and approximation methods are needed. The approximation technique proposed here is sketched in Algorithm 1. It is based on (De Raedt, Kimmig, and Toivonen 2007; Renkens, Van den Broeck, and Nijssen 2012) where a formula ψ in disjunctive normal form (DNF) is used to approximate $\mathrm{WMC}(\Delta \wedge \mathrm{q})$. Each explanation e in this algorithm is a conjunction of extensional variables that satisfies $\Delta \wedge \mathrm{q}$.

Since the intensional variables do not influence the WMC, they can be excluded from the calculations. And since every model of an explanation e represents a model in which q is true, its WMC has to be a lower bound: $\mathrm{WMC}_{\mathrm{E}}(\mathrm{e}) \leq \mathrm{WMC}(\Delta \wedge \mathrm{q})$. Furthermore, taking the disjunction of two explanations is equivalent to taking the union of the models of the two explanations. This means that the WMC of ψ will never decrease by adding an explanation and since the number of explanations is finite, it will converge to the actual WMC.

Example 3. In our running example, there are two possible explanations when calculating $\mathrm{WMC}(\Delta \wedge \mathit{win})$: $\mathrm{e}_1 = \mathit{h1} \wedge \mathit{h2}$ and $\mathrm{e}_2 = \neg \mathit{h1} \wedge \neg \mathit{h2}$. Both explanations entail win relative to theory Δ, that is, $\Delta \wedge \mathrm{e}_i \models \Delta \wedge \mathit{win}$. Therefore, $\mathrm{WMC}(\Delta \wedge \mathrm{e}_i) \leq \mathrm{WMC}(\Delta \wedge \mathit{win})$. By taking the disjunction $\mathrm{e}_1 \vee \mathrm{e}_2$ of these two explanations, all models of $\Delta \wedge \mathit{win}$ are accounted for and we have $\mathrm{WMC}(\Delta \wedge (\mathrm{e}_1 \vee \mathrm{e}_2)) = \mathrm{WMC}(\Delta \wedge \mathit{win})$. A further important observation is that the weight of all the intensional variables is identical (here equal to 1) in all models, which allows us to simplify $\mathrm{WMC}(\Delta \wedge (\mathrm{e}_1 \vee \mathrm{e}_2)) = c \cdot \mathrm{WMC}_{\mathrm{E}}(\mathrm{e}_1 \vee \mathrm{e}_2)$ (with $c = 1$).

The WMC of a disjunction of explanations is typically much easier to compute than that of the original formula since it has fewer variables. In the next section we present a formal framework for explanations that guarantees that the WMC can be simplified and bounded.

3 Explanations

Let us now formally introduce the notion of an *explanation* used in the algorithm sketched above. Our definitions are motivated by probabilistic logic programming languages (PLP) and databases. In such languages, the truth values of the extensional variables (or predicates) determine the truth values of the intensional ones (through some fixpoint computation), and the probability of a world is a function of only

its extensional variables. In this and the next section, we focus on WMC problems in this context; a discussion of the general case can be found in Section 5.

For WMC problems generated by probabilistic logic programs, this separation of variables translates into the notion of an intensional split. Although the formulae for WMC are generally provided in CNF, we will use equivalences for the sake of readability. It is trivial to transform them into CNF.

Definition 2 (Intensional Split). An *intensional split* of a WMC problem with theory Φ over variables V is a tuple $(\mathrm{E}, \mathrm{I}, \Delta, \Gamma)$, consisting of an ordered set of *intensional variables* $\mathrm{I} = \{d_1, \ldots, d_k\} \subseteq \mathrm{V}$, a set of *extensional variables* $\mathrm{E} = \mathrm{V} \setminus \mathrm{I}$, a *defining theory* Δ and a *constraining theory* Γ, such that

(i) $\Phi \equiv \Delta \wedge \Gamma$,

(ii) $\Delta = \bigwedge_{j=1}^{k} (d_j \leftrightarrow \alpha_j)$

(iii) all α_j range over the variables $\mathrm{E} \cup \{d_{j+1}, \ldots, d_k\}$, and

(iv) all d_i have uniform weights $\mathrm{w}(d_i) = \mathrm{w}(\neg d_i) = 1$.

The intuition is that for every value assignment to the extensional variables that satisfies Γ, there is exactly one possible truth-assignment to the intensional variables that satisfies the formula Φ. In addition, the weight of a world is only a function of the extensional variables.

Our algorithm will assume that an intensional split of the theory is given. Starting from a probabilistic logic program, the intensional split of its WMC problem is immediately clear: the extensional variables are the groundings of probabilistic facts, and the intensional variables are the groundings of the heads of clauses. The constraining theory consists of queries and constraints, and the defining theory represents the grounding of Prolog clauses.

Example 4. In our running example, we have Δ exactly as used so far, and $\Gamma \equiv \mathit{win}$.

Definition 3. (Explanation) An *explanation* e of a formula Φ with extensional split $(\mathrm{E}, \mathrm{I}, \Delta, \Gamma)$ is a conjunction of literals, containing no variables from I, and at most one of v and $\neg v$ for every variable $v \in \mathrm{E}$, such that

$$\Delta \wedge \mathrm{e} \models \Gamma \tag{1}$$

Intuitively, an explanation fixes truth values of a subset of the extensional variables that is sufficient to ensure that the constraints Γ are satisfied by all models of the explanation together with the definitions, where the latter propagate truth values from those extensional variables to the intensional variables.

Example 5. In our example, an explanation has to satisfy $\Delta \wedge \mathrm{e} \models \mathit{win}$, as already illustrated in Example 3.

Theorem 1. *Given a theory Φ with intensional split $(\mathrm{E}, \mathrm{I}, \Delta, \Gamma)$ and a disjunction of explanations $\mathrm{e}_1 \vee \ldots \vee \mathrm{e}_m$ for Φ, we have*

$$\mathrm{WMC}_{\mathrm{V}}(\Phi \wedge (\mathrm{e}_1 \vee \ldots \vee \mathrm{e}_m)) = \mathrm{WMC}_{\mathrm{E}}(\mathrm{e}_1 \vee \ldots \vee \mathrm{e}_m) \tag{2}$$

For a single explanation e*,*

$$\mathrm{WMC}_{\mathrm{E}}(\mathrm{e}) = \prod_{l \in \mathrm{e}} \mathrm{w}(l) \cdot \prod_{v \in \mathrm{E} \setminus \mathrm{V}_{\mathrm{e}}} (\mathrm{w}(v) + \mathrm{w}(\neg v)) \tag{3}$$

with V_{e} denoting those variables occurring in e*.*

Proof outline. $\mathrm{WMC}_{\mathrm{V}}(\Phi \wedge \mathrm{e})$ sums the weights of all interpretations i for which $i \models \Phi \wedge \mathrm{e}$, or equivalently $i \models \Delta \wedge \Gamma \wedge \mathrm{e}$. Such an interpretation has to agree with e on the extensional variables fixed by the latter. By construction, for every assignment to the extensional variables, there is a single assignment to the intensional variables such that the combination of the two is a model of Δ. All literals for intensional variables have weight 1, and thus can be omitted from the product (and the sum). By (1), every model of $\Delta \wedge \mathrm{e}$ is also a model of Γ. Equation (3) then follows by distributivity. □

This shows that we can efficiently compute the WMC of $\Phi \wedge \mathrm{e}$ for any given explanation e. We will discuss how to find such explanations in the next section.

4 Algorithm

Given the notion of an explanation, we are now in a position to define the algorithm that finds explanations, and that iteratively computes bounds on our queries.

Explanation Finding

We now introduce the nextExpl method of Algorithm 1 to find the next best explanation, that is, the explanation maximizing the WMC in Equation (3). It is based on a reduction to a weighted partial max-SAT (WPMS) problem:

Given: a set of weighted clauses
$WC = \{w_i : c_i \mid \text{with } w_i \text{ the weight of the clause } c_i\}$

Find: $\arg\min_{i \in \mathcal{I}(\mathrm{V})} \sum_{i \not\models c_i} w_i$.

The goal of WPMS thus is to find an interpretation i with smallest (weighted) sum of violated clauses, where hard constraints can be imposed by setting $w = \infty$.

We transform a theory Φ with intensional split $(\mathrm{E}, \mathrm{I}, \Delta, \Gamma)$ into a WPMS $W(\Phi)$ as follows, where we introduce two WPMS variables v^+ and v^- for each variable $v \in \mathrm{V}$, indicating that the value of v is fixed to true or false, respectively:

1. assign weight ∞ to each formula in Φ;
2. for all variables $v \in \mathrm{V}$:
   - replace all positive literals v in Φ by v^+,
   - replace all negative literals $\neg v$ in Φ by v^-,
   - add $\infty : \neg v^- \vee \neg v^+$ to Φ;
3. for all variables $v \in \mathrm{I}$, replace $\infty : (v^+ \leftrightarrow \alpha)$ in Φ by $\infty : (v^+ \leftrightarrow \alpha) \vee (\neg v^+ \wedge \neg v^-)$
4. for all variables $v \in \mathrm{E}$, add the following to Φ:
   (a) $-\log(w(v)) : \neg v^+$
   (b) $-\log(w(\neg v)) : \neg v^-$
   (c) $-\log(w(v) + w(\neg v)) : v^+ \vee v^-$
5. rewrite Φ to clausal form

Theorem 2. *A solution of the WPMS $W(\Phi)$ encodes an explanation of Φ with maximal WMC.*

Proof outline. Step 1 turns Φ into hard constraints. Step 2 introduces the encoding which allows the WPMS solver to leave the truth value of a variable open (by setting both v^+ and v^- to false), but at the same time excludes a contradicting assignment (setting both v^+ and v^- to true). Step 3 states that for each intensional variable, if it has a truth value assigned, its definition needs to hold. All constraints up to here are hard, and thus have to be satisfied by every assignment. Step 4 adds the soft clauses influencing the minimization. If a clause of type (a) is violated, v^+ is true, and the corresponding positive literal v is part of the explanation, in which case the negative logarithm of its weight is added to the sum to be minimized. Clauses of type (b) achieve the same for negative literals. Clauses of type (c) cover variables that are not part of the explanation; these clauses are violated if the variable is set neither true nor false. Exactly one type of clause is violated for any legal assignment to v^+ and v^-, and the sum becomes the negative logarithm of Equation (3). Minimizing this logarithm is equivalent to maximizing Equation (3). □

The solutions of the WPMS problem contain intensional variables, which can simply be dropped since their truth values are implied by the extensional variables and the definitions.

This reduction is the basis of procedure nextExpl, which calls a weighted maxSAT solver on $W(\Phi)$ to produce the next explanation. Once an explanation $v_1 \wedge \ldots \wedge v_i \wedge \neg v_{i+1} \wedge \ldots \wedge \neg v_n$ has been found, we simply extend $W(\Phi)$ with the hard constraint $\infty : \neg v_1^+ \vee \ldots \vee \neg v_i^+ \vee \neg v_{i+1}^- \vee \ldots \vee \neg v_n^-$ in order to avoid that the same explanation is found again, and iterate. Note that this constraint does not guarantee that the found explanations are mutually exclusive.

Computing Bounds

For arbitrary WMC tasks, we can use Algorithm 1 with an implementation of nextExpl(.) based on the WPMS approach discussed above, that is, prior to each call, we add a new hard clause corresponding to the last solution.

We can do better when $\mathrm{WMC}(\Delta)$ represents a probability distribution ($\mathrm{WMC}(\Delta) = 1$), as is the case in the PLP setting. In this case we can calculate an upper bound by calculating a lower bound on $\mathrm{WMC}(\Delta \wedge \neg \mathrm{q})$ and by using $\mathrm{WMC}(\Delta \wedge \mathrm{q}) = \mathrm{WMC}(\Delta) - \mathrm{WMC}(\Delta \wedge \neg \mathrm{q})$. Algorithm 2 sketches an anytime algorithm based on these observations, using the WPMS reduction to implement nextExpl(.). It keeps updating the lower and upper bounds, in each iteration adding a new explanation to the bound whose last update resulted in the largest change in value.

```
improveTop ← 0.5
improveBot ← 0.5
top ← true
bot ← false
up ← 1
low ← 0
while time < maxTime do
    if improveTop > improveBot then
        e ← nextExpl(Δ ∧ ¬Γ)
        next ← WMC(top ∧ ¬e)
        improveTop ← up − next
        top ← top ∧ ¬e
        up ← next
    else
        e ← nextExpl(Δ ∧ Γ)
        next ← WMC(bot ∨ e)
        improveBot ← next − low
        bot ← bot ∨ e
        low ← next
    end
end
return [low, up]
```

Algorithm 2: Computing lower as well as upper bounds in the PLP setting given a time budget $maxTime$.

5 Beyond Probabilistic Logic Programs

Of all existing WMC algorithms, exact and approximate, our algorithm takes a rather unique position. Existing algorithms, such as `c2d` (Darwiche 2004) and `Cachet` (Sang et al. 2004), are intended to solve arbitrary WMC problems (in CNF). This is partly their strength, as this permits their application to Bayesian network, PLP, and Markov logic inference equally. Our algorithm applies to general WMC problems (as we discuss later). Yet, its main strength comes from exploiting the presence of intensional variables and definitions. This type of logical structure is found in WMC encodings of probabilistic logic programs. For example, in the formulae for the GENES dataset used in the experiments (Section 7), the number of intensional variables is on average three times that of extensional variables. This means that using the intensional split can allow significantly smaller explanations. However, intensional variables may not appear in arbitrary WMC problems, or encodings of other formalisms. The converse is also true: the generality of existing WMC solvers prohibits them from easily exploiting the presence of definitions and intensional variables, and our algorithm loses much of its appeal when applied to general WMC problems.

This distinction between general-purpose solvers and solvers that exploit structural properties of the WMC encoding opens up new questions. Can we develop WMC solvers that exploit structural properties of Bayesian network encodings, or encodings of Markov logic networks? For Bayesian networks alone, three different WMC encodings have been proposed in the literature (Darwiche 2009), each with their own intrinsic properties. In the future, we anticipate to see more WMC solvers that exploit the properties of a specific WMC encoding. To bridge the gap in the other direction, we will now investigate the application of our algorithm to general WMC problems.

Finding Intensional Splits

So far, we have assumed that an intensional split can be read off directly from the probabilistic logic program that generated the WMC problem at hand. When presented with an arbitrary WMC problem, not necessarily encoding a prob-

abilistic logic program, we can still use our algorithm. We only need to provide an intensional split, which can be found as follows. Starting with $\Delta = true$ and $\mathrm{I} = \emptyset$, repeatedly delete a subformula $(d \leftrightarrow \alpha)$ from theory Φ for which $\mathrm{w}(d) = \mathrm{w}(\neg d)$ and neither d nor any of the variables in α belong to I, conjoin this subformula to Δ and add the variable d to I. Proceed until no further definitions can be found in Φ. Then $\Gamma = \Phi$ and $\mathrm{E} = \mathrm{V} \setminus \mathrm{I}$.

It is desirable to obtain the split that maximizes $|\,\mathrm{I}\,|$, so as to minimize the length of our explanations. In this regard, the effectiveness of the split-finding procedure above strongly depends on the order in which subformulae are selected. In fact, we can show the following.

Theorem 3. *Given a WMC problem, finding its intensional split that maximizes $|\,\mathrm{I}\,|$ is NP-hard.*

Proof outline. The proof is by reduction from the maximum independent set problem. Given a graph $G = (V, \mathcal{E})$, an independent set $S \subseteq V$ contains no two vertices that are connected by an edge in $\mathcal{E}$. The maximum independent set problem is concerned with finding the independent set that maximizes $|S|$, and is NP-hard (Karp 1972).

We can encode a maximum independent set problem into the problem of finding an optimal intensional split as follows. Let there be a set of variables $\mathrm{V} = \{v_1, \ldots, v_n\}$ denoting the vertices V. Suppose that for every variable v whose neighbors in G are denoted by variables $n_1, \ldots, n_k$, theory Φ contains the formula $v \leftrightarrow n_1 \vee \cdots \vee n_k$. Suppose further that $\mathrm{w}(\ell) = 1$ for all literals ℓ. There is now a one-to-one mapping between intensional splits of Φ and independent sets of G, where variables I of the split denote an independent set of vertices. Indeed, no two can have an edge between them, as this would violate property (iii) of Definition 2. Conversely, every independent set has a corresponding intensional split whose $\mathrm{I} = S$ and $\mathrm{E} = \mathrm{V} \setminus S$, which can be verified by checking properties (i)-(iv) of Definition 2. In this mapping, $|S| = |\,\mathrm{I}\,|$, and we can solve the NP-hard maximum independent set problem by finding the intensional split that maximizes $|\,\mathrm{I}\,|$. □

Nonetheless, our algorithm works with any intensional split, not only the one minimizing $|\,\mathrm{I}\,|$, and approximately optimal splits can be used in practice.

6 Related Work

Our approximate WMC algorithm is closely related, and inspired by several approximation techniques developed in the PLP community (Poole 1993; Kimmig et al. 2008; Renkens, Van den Broeck, and Nijssen 2012). Their key observation is that the probability of a probabilistic Prolog query equals the probability of its proofs, and that this probability can be approximated, and bounded below by a subset of the proofs. Whereas Kimmig et al. (2008) search for the k proofs with the highest individual probability, Renkens, Van den Broeck, and Nijssen (2012) search for the k proofs whose joint probability is maximized. They show that this optimization problem is NP-hard. Luckily it is submodular, and therefore easy to optimize greedily within bounds.

The key difference between the PLP techniques and our approach is that we search for explanations of WMC problems, not for proofs. This has several advantages. Firstly, while the PLP techniques work with probabilistic logic programs that have no negation or loops, our approach is oblivious to which program generated the WMC problem it operates on. The reductions from PLP to WMC of Fierens et al. (2011) support negation and loops, and so does our work. In fact, the need for approximate inference with such programs is what motived us. Moreover, we can directly benefit from any future extensions of these conversions, for example supporting new PLP language concepts. Secondly, support for negation directly allows us to approximate the negation of a query. Our algorithm can therefore also compute upper bounds on queries, which is not a feature of the earlier algorithms. Thirdly, our approach obviates the need to find individual Prolog proofs. Instead, it requires a PLP-to-WMC conversion, which is very different in nature from proof-finding, and an off-the-shelf WPMS solver.

This work follows a tradition of *approximate knowledge compilation* and *theory approximation* (Selman and Kautz 1996). There, the goal is to approximate a theory Φ below and above such that $\Phi_{lb} \models \Phi \models \Phi_{ub}$, and Φ_{lb}, Φ_{ub} allow efficient logical query answering. We follow a similar approach to approximate WMC problems.

Finally, by reducing approximate inference to a WMC task, our algorithm exploits local structure and determinism. As such, it follows a long line of research on *approximate model counting* (Wei and Selman 2005; Gomes, Sabharwal, and Selman 2006) and more general approximate probabilistic inference algorithms that exploit local structure and determinism (Poon and Domingos 2006; Gogate and Dechter 2011).

7 Experiments

We experimentally evaluate our approximation algorithm as given in Algorithm 2, focusing on two questions:

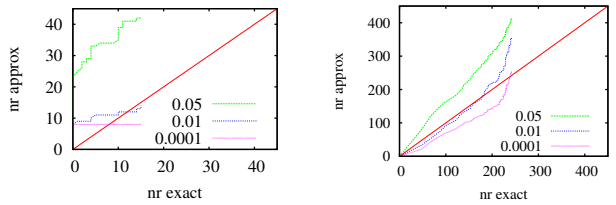
Q1: How does approximate inference compare to exact inference in general?

Q2: How does approximate inference perform when exact inference is not feasible?

Both approximate and exact inference use the PLP-to-WMC conversion of Fierens et al. (2011) to obtain the theory Φ, and compilation to SDDs (Darwiche 2011) to compute WMCs. The WPMS solver used in approximate inference is WPM1 (Ansótegui, Bonet, and Levy 2009). Experiments are run on a 3.4 GHz machine with 16 GB of memory.

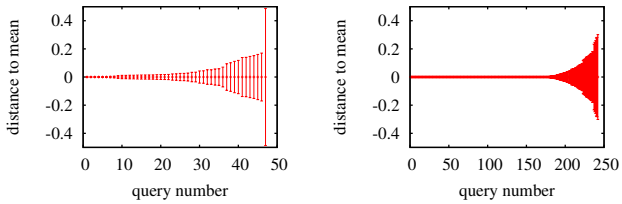
Our comparison uses two datasets. The first is WEBKB[1], where university webpages have to be tagged with classes. The tag of a page depends on both its textual content as well as the classes of the pages it links to. Following Fierens et al. (2011), we select 150 pages, and we use the 100 most frequent words only. Probabilities are chosen randomly from $[0.1, 0.4]$. This dataset contains 62 queries, and we give each algorithm a time budget of five minutes per query. The second dataset is the biological application of (Ourfali et al.

[1] http://www.cs.cmu.edu/~webkb/



(a) WEBKB (b) GENES

Figure 1: Number of queries solved by exact inference (x-axis) and by approximate inference with different thresholds on approximation error (y-axis) over time.



(a) WebKb (b) GENES

Figure 2: Interval sizes obtained by approximate inference on queries not solved by exact inference within the time budget.

2007), where the regulatory effect of a gene on another gene has to be predicted based on combinations of different types of molecular interactions. We refer to this dataset as GENES. This dataset contains 500 queries, and we give each algorithm a time budget of ten minutes per query.

For **Q1**, we focus on the queries for which exact inference did return a result within the allocated time. On WEBKB, exact inference solved 15 of the 62 queries, and approximate inference converged to the exact value for four of these. On GENES, exact inference solved 258 of the 500 queries, and approximate inference converged to the exact value for 173 of these. We further consider three different thresholds $(0.05, 0.01, 0.0001)$ on the quality of approximation as measured by the maximal error (i.e., half of the difference between the bounds). For each threshold, in Figures 1a (WEBKB) and 1b (GENES), we plot the number of queries solved by exact inference at any point within the time budget against the number of queries approximated to at least the threshold quality at that time. For a maximal error of 0.05, in both cases the approximate inference outperforms the exact inference for any timeout. For smaller error thresholds, the results for the two datasets differ. In the case of WEBKB, approximate inference outperforms exact inference initially, but exact inference solves more queries if allowed more time. On GENES, exact inference solves more queries initially, but is outperformed by approximate inference if more time is given. As an answer to **Q1**, we thus conclude that although approximate inference is competitive with exact inference, even when the allowed error is small, it is hard to predict when approximate inference will do better. A lot depends on the structure of the given program.

For **Q2**, we focus on the queries on which exact inference did not return a result within the allocated time (47 out of 62 for WEBKB, 242 out of 500 for GENES). Figures 2a (WEBKB) and 2b (GENES) show for each such query the returned interval plotted around the average of the lower and upper bound, ordered by increasing difference between the bounds. Here, the trend is the same on both datasets. Our algorithm achieves close approximations for most queries, though there are also queries that are hard for this approach. As an answer to **Q2**, we thus conclude that approximate inference performs well on most of the queries, even when exact inference is not feasible.

8 Conclusions

We have proposed an anytime approximation algorithm for WMC motivated by approximation techniques from probabilistic logic programming. The explanation-based approach searches for conjunctions of literals whose WMC is easy to calculate and can be used to approximate the WMC of the original problem. Furthermore, explanations can be combined to obtain a better approximation. The approach has been used in an approximation algorithm that can calculate hard bounds on the marginal probability for probabilistic logic programs which has been tested empirically on two datasets. The results show that the approximate inference is competitive with exact inference and performs well, even when exact inference is not feasible.

9 Acknowledgments

Joris Renkens is supported by the IWT (agentschap voor Innovatie door Wetenschap en Technologie). Angelika Kimmig and Guy Van den Broeck are supported by the Research Foundation-Flanders (FWO-Vlaanderen).

References

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. Solving (weighted) partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.

Dalvi, N. N., and Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*.

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

Darwiche, A. 2004. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI)*.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press. Chapter 12.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*.

De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic Inductive Logic Programming — Theory and Applications*, volume 4911 of *Lecture Notes in Artificial Intelligence*. Springer.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: a probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNFs. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Dechter, R. 2011. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2006. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer. 85–103.

Kimmig, A.; Santos Costa, V.; Rocha, R.; Demoen, B.; and De Raedt, L. 2008. On the efficient execution of ProbLog programs. In *Proceedings of the 24th International Conference on Logic Programming (ICLP)*.

Ourfali, O.; Shlomi, T.; Ideker, T.; Ruppin, E.; and Sharan, R. 2007. SPINE: a framework for signaling-regulatory pathway inference from cause-effect experiments. *Bioinformatics* 23(13):i359–366.

Poole, D. 1993. Logic programming, abduction and probability. *New Generation Computing* 11:377–400.

Poole, D. 1997. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence* 94(1-2):7–56.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

Renkens, J.; Van den Broeck, G.; and Nijssen, S. 2012. k-Optimal: a novel approximate inference algorithm for ProbLog. *Machine Learning* 89:215–231.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP)*.

Selman, B., and Kautz, H. 1996. Knowledge compilation and theory approximation. *Journal of the ACM* 43(2):193–224.

Vennekens, J.; Verbaeten, S.; and Bruynooghe, M. 2004. Logic programs with annotated disjunctions. In *Proceedings of the 20th International Conference on Logic Programming (ICLP)*.

Wei, W., and Selman, B. 2005. A new approach to model counting. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.