Approximate Probabilistic Inference with Bounded Error for Hybrid Probabilistic Logic Programming

Steffen Michels[a] and **Arjen Hommersom**[a,b] and **Peter J.F. Lucas**[a,c]
[a]Institute for Computing and Information Sciences, Radboud University, the Netherlands
[b]Faculty of Management, Science and Technology, Open University of the Netherlands
[c]LIACS, Leiden University, the Netherlands
s.michels@science.ru.nl,{arjenh,peterl}@cs.ru.nl

Abstract

Probabilistic logics, especially those based on *logic programming* (LP), are gaining popularity as modelling and reasoning tools, since they combine the power of logic to represent knowledge with the ability of probability theory to deal with uncertainty. In this paper, we propose a hybrid extension for probabilistic logic programming, which allows for exact inference for a much wider class of continuous distributions than existing extensions. At the same time, our extension allows one to compute approximations with bounded and arbitrarily small error. We propose a novel anytime algorithm exploiting the logical and continuous structure of distributions and experimentally show that our algorithm is, for typical relational problems, competitive with state-of-the-art sampling algorithms and outperforms them by far if rare events with deterministic structure are provided as evidence, despite the fact that it provides much stronger guarantees.

1 Introduction

Probabilistic logics are gaining popularity as modelling and reasoning tools, since they combine the power of logic to represent knowledge with the ability of probability theory to deal with uncertainty. Especially, probabilistic languages based on *logic programming* (LP) have become popular, as they enable knowledge representation by employing similar structure as the popular *Bayesian Networks* (BNs) [Pearl, 1988], but lift this from a propositional to a first-order level, thus dramatically increasing expressive power. Examples of such languages are *ICL* [Poole, 2008] and *ProbLog* [Fierens *et al.*, 2015]. The need for such expressive languages emerged from the fact that in many areas more and more data become available, which does not only imply uncertainty, but often provides rich structure in terms of relations between entities. Probabilistic logic methods have been applied to a wide range of problem domains. Examples include link and node prediction in metabolic networks [Kimmig and Costa, 2012], dealing with a potentially unknown number of relations between multiple objects by a robot [Moldovan *et al.*, 2012] and information fusion in the safety and security domain [Michels *et al.*, 2013].

Probabilistic inference is an inherently computationally hard problem. However, it is well known that by exploiting logical structure as present in many distributions, exact inference is rendered feasible for many realistic problems [Chavira and Darwiche, 2008]. This is achieved by viewing the problem as a *weighted model counting* (WMC) problem, utilising previous work on SAT solving and model counting. These inference methods are especially suitable for probabilistic LP, as here models always have a rich logical structure (e.g. [Fierens *et al.*, 2015]). Unfortunately, the above-mentioned structure-sensitive inference methods are usually restricted to discrete, finite distributions. Many real-world problems, however, also require continuous variables, i.e. they are best modelled using a hybrid distribution. Inference for hybrid distributions is however either made possible by significantly limiting the expressivity of the languages, e.g. [Gutmann *et al.*, 2010; Chistikov *et al.*, 2015], or by resorting to approximate inference methods, e.g. [Gutmann *et al.*, 2011].

Sampling algorithms – the most widely used methods for approximate inference – are however known to converge slowly when dealing with (near) deterministic probabilities and cannot handle observed rare events well. Also inference methods based on the approximate representation of the actual continuous distribution, e.g. [Sanner and Abbasnejad, 2012; Belle *et al.*, 2015a], suffer from observed rare events. All such methods furthermore share the drawback that they only provide weak guarantees about the quality of the estimates. In domains such as diagnosis and crisis management, where wrong decisions may have a huge impact, guarantees about the quality of approximations are highly desirable. The need to deal with rare events is also characteristic for such domains.

In this paper, we extend probabilistic LP with continuous distributions in a way that poses little restriction on the definition and use of continuous variables. Although the inference task is not computable, we show that we can provide an approximation with bounded error, which essentially provides more information about the quality of the result than the approximation methods mentioned above. The algorithm iteratively computes approximations with decreasing error, adopting a *coarse-to-fine* approach, making use of information from previous runs and also exploiting the logical structure of the problem to effectively decrease the error of an approximation. A comparison is made with sampling methods,

showing that in spite of the much stronger guarantees on the quality of the result, our method is competitive and even superior for hybrid problems that possess a rich logical structure and concern observed rare events.

2 Hybrid Probabilistic Logic Programming

We here introduce the language used, which is similar to *Hybrid ProbLog* [Gutmann *et al.*, 2010], but allows for more expressive constraints on continuous variables.

Example 1 *We consider a diagostic problem. Suppose there are three reasons why a component i can fail: the component breaks with some probability, the temperature got too high, or a sub-component fails. Additionally, there is a cooler that cools down the temperature of all components by an amount that is uncertain as well. This cooler can also fail, indicated by the variable* $\mathbf{NoC}$*. We can express this with the following rules:*

$$
\begin{aligned}
&\mathit{fails}(i) \leftarrow \mathbf{Break}(i) = \mathit{true}\\
&\mathit{fails}(i) \leftarrow \mathbf{Temp} - \mathbf{Cooling} > \mathbf{Limit}(i)\\
&\mathit{fails}(i) \leftarrow \mathbf{Temp} > \mathbf{Limit}(i),\ \mathbf{NoC} = \mathit{true}\\
&\mathit{fails}(i) \leftarrow i \neq 0,\ \mathit{fails}(i-1)
\end{aligned}
$$

For $i = 0$, the component does not depend on other components, so it can only fail because of the first two reasons. We can use arbitrary distributions for the continuous variables, for example:

$$
\begin{aligned}
\mathbf{Temp} &\sim \mathcal{N}(20.0, 5.0) \qquad \mathbf{Limit}(\_) \sim \mathcal{N}(30.0, 5.0)\\
\mathbf{Break}(\_) &\sim \{0.0001\colon \mathit{true}, 0.9999\colon \mathit{false}\}\\
\mathbf{Cooling} &\sim \Gamma(30.0, 18.0)\\
\mathbf{NoC} &\sim \{0.01\colon \mathit{true}, 0.99\colon \mathit{false}\}
\end{aligned}
$$

The distributions for $\mathbf{Break}(i)$ and $\mathbf{Limit}(i)$ are assumed to be the same for all components i. Assuming that a given component fails, a natural query is what is the probability that the first component also fails, e.g. $P(\mathit{fails}(0)|\mathit{fails}(9))$.

Formally, we consider a finite number of *random variables* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \ldots$, which have a *support*, i.e. the range of values on which the distribution is defined. These ranges are in principle arbitrary; in this paper we consider binary and real-valued random variables as typical representatives of discrete and continuous variables. For the continuous variables, we assume that their *cumulative distributions functions* (CDFs) can be computed exactly, thereby ignoring the potential error of typical implementations of such functions as such errors are typically negligible and predictable. We assume all random variables to be independent initially, and express dependencies by the logical structure, as common in probabilistic LP.

We use LP rules with constraints on the random variables as building blocks for the bodies, with the only restriction that satisfiability of those constraints remains decidable. Concretely, for real-numbered random variables we allow inequalities on combinations of variables. These combinations are not necessarily linear as for instance required for *Conditional Linear Gaussians* [Lauritzen, 1992], because constraints that consist of multiplications with real numbers are decidable. Note that this is a much weaker requirement than e.g. made for *Hybrid ProbLog*, which only allows a single random variable in a constraint, such that not only the constraint is decidable, but the probability of the constraint is computable as well.

As usual for probabilistic LP, the inference task we consider is the computation of the probability that a grounded predicate holds, given a conjunction of other grounded predicates.

3 Iterative Hybrid Probabilistic Model Counting

We here present a novel algorithm, the *iterative hybrid probabilistic model counting* (IHPMC) algorithm; it is an anytime algorithm that provides approximations with bounded error for the language outlined above.

3.1 Hybrid Probability Trees

Queries are converted to propositional formulas as is often done for probabilistic LP inference. Details of this transformation are not discussed in this paper as it is essentially the same as for ProbLog [Fierens *et al.*, 2015]. The only difference is that the propositional formulas also include constraints on random variables. For a fixed $i \neq 0$, the rules in Example 1 can for instance be represented as: $\mathit{fails}_i \leftrightarrow \mathbf{Break}_i = \mathit{true} \vee \mathbf{Temp} - \mathbf{Cooling} > \mathbf{Limit}_i \vee (\mathbf{Temp} > \mathbf{Limit}_i \wedge \mathbf{NoC} = \mathit{true}) \vee \mathit{fails}_{i-1}$.

To perform inference on such formulas we introduce the concept of *Hybrid Probability Trees* (HPTs). They are similar to the principles used in solvers for binary weighted model counting. Whereas in previous work the concept is applied to discrete, finite ranges, we extend the concept to continuous random variables. The basic idea is to split the variable's range into two parts at each edge and condition the formula on the split made. To support continuous variables, we allow further splits of the same variable at deeper levels. Then, for each choice taken at a certain child, the probability is assigned that the random variable takes a value in the chosen partition, conditioned on the range chosen for the random variable earlier on the path. Such probabilities can be computed for continuous random variables by their CDFs, because we assume independence of the random variables. The splitting is repeated until the formula simplifies to $\bot$ or $\top$, which we for now assume to happen in a finite number of steps. By attaching a corresponding probability to each branch, this can be used to compute the probability of the event represented by the formula.

Definition 1 (Hybrid Probability Trees (HPT)) *An HPT is a binary tree with at each node n a propositional formula φ_n and for all random variables $\mathbf{X}$ a range denoted by $\mathit{range}(n, \mathbf{X})$. If r is the root node, it holds that $\mathit{range}(r, \mathbf{X})$ equals the support of $\mathbf{X}$. Now let n be some node with children $\{c_1, c_2\}$. To each edge $n \rightarrow c_i$, $i \in \{1, 2\}$, a particular range τ_{ni} is associated to a fixed random variable $\mathbf{Y}$ such that $\tau_{n1} \cup \tau_{n2} = \mathit{range}(n, \mathbf{Y})$ and $\tau_{n1} \cap \tau_{n2} = \varnothing$. It holds that $\mathit{range}(c_i, \mathbf{Y}) = \tau_{ni}$ and $\mathit{range}(c_i, \mathbf{Z}) = \mathit{range}(n, \mathbf{Z})$ if $\mathbf{Y} \neq \mathbf{Z}$, with $i \in \{1, 2\}$. Furthermore, the formula φ_i associated to c_i equals φ_n simplified by the restrictions on the*

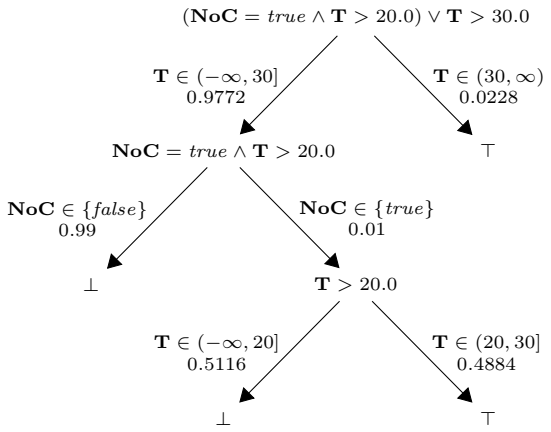


Figure 1: Example HPT

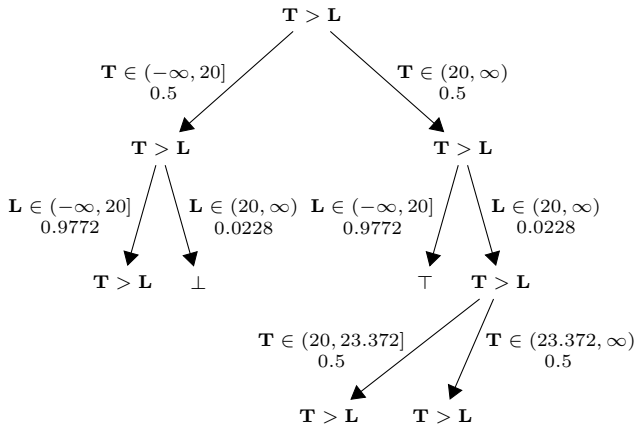


Figure 2: Example PHPT

range on $\mathbf{Y}$ *at* c_i*, i.e. by observing that some primitive constraints can be replaced by* $\top$ *or* $\bot$*. Furthermore, to each edge* $n \to c_i$*, with* $i \in \{1, 2\}$*, a probability* p_{ni} *is assigned, such that* $p_{ni} = P(\mathbf{Y} \in \tau_{ni} \mid \mathbf{Y} \in range(n, \mathbf{Y}))$*. Finally, if* l *is a leaf node, then it holds that* $\varphi_l = \top$ *or* $\varphi_l = \bot$*.*

Given a particular HPT, it is straightforward to compute the probability of the event represented by the formula at the root node. First, the probability of a leaf is by definition 0 or 1. Furthermore, due to the properties of the tree it is easy to show that for each non-leaf node n the probability of φ_n given the ranges of random variables can be computed by $p_n = p_{n1} \cdot p_1 + p_{n2} \cdot p_2$, where p_1 and p_2 are the probabilities associated to n's children.

Example 2 *Consider a simplified diagnosis problem:*

$$fail \leftarrow \mathbf{T} > 20.0, \mathbf{NoC} = true \quad fail \leftarrow \mathbf{T} > 30.0$$

Here $\mathbf{NoC}$ *represents that the cooling fails, implying a lower allowed temperature for the component. The temperature is modelled by* $\mathbf{T} \sim \mathcal{N}(20.0, 5.0)$*. The event that the component fails is then represented by the formula* $(\mathbf{NoC} = true \wedge \mathbf{T} > 20.0) \vee \mathbf{T} > 30.0$*.*

A possible HPT is given in Figure 1. At each node we simplify the formula as much as possible given the choices made. Note that the probability assigned to the edge with choice $\mathbf{T} \in (-\infty, 20]$ *is not* $P\big(\mathbf{T} \in (-\infty, 20]\big) = 0.5$*, as the range of* $\mathbf{T}$ *is already restricted on the path before. Therefore, it is* $P\big(\mathbf{T} \in (-\infty, 20] \mid \mathbf{T} \in (-\infty, 30]\big) \approx 0.5116$*. From this tree, the probability of component failure can be computed by* $0.9772 \cdot 0.01 \cdot 0.4884 + 0.0228 \approx 0.0276$*.*

3.2 Partially Evaluated Hybrid Probability Trees

The concept of HPT is restricted to problems for which exact inference is possible. For continuous events, which cannot be represented by hyperrectangles, we can never simplify all leaves to $\bot$ or $\top$. This problem is tackled by using *partially evaluated HPTs* (PHPTs). For PHPTs we drop the requirement that all leaves must contain $\bot$ or $\top$ and allow arbitrary formulas. One can therefore further extend a PHPT either ad infinitum or until one finds an HPT. In the next section we show how PHPTs can be used to compute approximations with known maximal error.

Example 3 *Suppose the limit of the temperature the component can bear is uncertain as well, modelled by* $\mathbf{L} \sim \mathcal{N}(30.0, 5.0)$*. A PHPT for the event that the temperature is above the limit is depicted in Figure 2. It is impossible to find a finite HPT for this problem, as the event does not have the shape of a hyperrectangle. Note that the formula* $\mathbf{T} > \mathbf{L}$ *cannot be simplified directly, in case the range of only one of the variables is split. This is in contrast to binary versions of model counting algorithms, for which the choices of variable values are completely reflected in the simplified formula.*

3.3 Approximating Probabilities by Partially Evaluated Hybrid Probability Trees

Bounds on Event Probabilities

Each leaf of a PHPT with a formula which is not $\bot$ or $\top$ corresponds to an area of the sample space which is only partially part of the event associated to the query. While this does not provide an exact probability of this event, it does provide a bound on this probability by assuming that the probability of such areas is 0.0 or 1.0 respectively. In this way, PHPTs can be used to compute probability bounds, denoted by $\underline{P}$ and $\overline{P}$.

Example 4 *The PHPT in Figure 2 contains only one leaf with* $\top$*, whereas all other leaves contribute at most partially to the event's probability. The lower bound of the event's probability is therefore* $\underline{P}(\mathbf{T} > \mathbf{L}) = 0.5 \cdot 0.9772 = 0.4886$*. Analogously, as there is only one path which certainly does not contribute to the probability, the probability's upper bound is* $\overline{P}(\mathbf{T} > \mathbf{L}) = 1 - 0.5 \cdot 0.0228 = 0.9886$*.*

Proposition 1 *For any event* e *and PHPT for* e*:* $\underline{P}(e) \leq P(e) \leq \overline{P}(e)$*.*

All proofs are given in [Michels, 2016, Chapter 6, Appendix A]. Furthermore, we can always achieve arbitrary precision, by evaluating the PHPT sufficiently deep.

Proposition 2 *For every event* e *and maximal error* ϵ*, there is a PHPT, such that:* $P(e) - \underline{P}(e) \leq \epsilon \wedge \overline{P}(e) - P(e) \leq \epsilon$*.*

Bounds on Conditional Event Probabilities

Conditional probabilities are defined as $P(e \mid o) = P(e \wedge o)/P(o)$. As this definition requires the non-conditional

Input: Events e and o, maximal error ϵ
Result: Approximation of $P(e \mid o)$ with max. error ϵ

hpt_p = PHPT with single root node $e \wedge o$
hpt_n = PHPT with single root node $\neg e \wedge o$
$\underline{p} = 0.0; \overline{p} = 1.0$
while $(\overline{p} - \underline{p})/2 > \epsilon$ **do**
  $eval\_hpt = choose\_hpt(hpt\_p, hpt\_n)$
  $eval\_node = choose\_node(eval\_hpt)$
  $branch\_rv = choose\_rv(eval\_node)$
  $(rv\_l, rv\_r) = choose\_part(branch\_rv, eval\_node)$
  add children to $eval\_node$ according to (rv_l, rv_r)
  $\underline{p} = lower(hpt\_p)/(lower(hpt\_p) + upper(hpt\_n))$
  $\overline{p} = upper(hpt\_p)/(upper(hpt\_p) + lower(hpt\_n))$
end
return $(\underline{p} + \overline{p})/2$

Algorithm 1: IHPMC Algorithm

probabilities of two events, we cannot compute bounds on this probability using a single PHPT, but have to use a second one. We can furthermore compute tighter bounds by making use of the fact that the bounds of $e \wedge o$ and e are not independent: an area cannot at the same time be outside of e, but within $e \wedge o$. This can be exploited by expressing the bounds of conditional probabilities in terms of bounds on two non-conditional probabilities [Michels *et al.*, 2015]:

Proposition 3

$$\underline{P}(e \mid o) = \underline{P}(e \wedge o)/\big(\underline{P}(e \wedge o) + \overline{P}(\neg e \wedge o)\big)$$
$$\overline{P}(e \mid o) = \overline{P}(e \wedge o)/\big(\overline{P}(e \wedge o) + \underline{P}(\neg e \wedge o)\big)$$

Also for conditional probabilities we can therefore achieve arbitrary precisions.

Proposition 4 *For all events e and o and every maximal error ϵ, there are PHPTs, such that: $P(e \mid o) - \underline{P}(e \mid o) \leq \epsilon \wedge \overline{P}(e \mid o) - P(e \mid o) \leq \epsilon$.*

3.4 Anytime Inference by Iterative Hybrid Probability Tree Evaluation

The anytime approximation IHPMC algorithm based on PHPTs is given in Algorithm 1. We start with two PHPTs with initial root nodes $e \wedge o$ and $\neg e \wedge o$, which we use to compute in each iteration an approximation according to Proposition 3. In case no evidence is present, we can use a single PHPT with formula e. In the general case, at each iteration we first choose one of the two PHPTs and then, in this PHPT, we choose a non-evaluated node to evaluate further. From the formula of this node we choose a random variable that will be used for splitting and finally we choose a partitioning for this random variable.

Note that in the algorithm we could also compute components of the formula independently if they do not share variables, as is usually done in WMC solvers. For brevity this is left out here. While the basic algorithm is straightforward, there are four non-deterministic choices at each iteration which largely determines the effectiveness of the algorithm. In the following, we describe general heuristics which seem to be very effective for most problems.

As the first choice, we pick the PHPT which has the highest difference between lower and upper bound, i.e. , the largest error. Second, we choose the non-evaluated node which represents the sub-space with the highest probability mass, as these nodes have the most potential for reducing the error. A more sophisticated heuristic, that tries to predict how much effort is required to reduce the error of the node's formula, offered no improvement in experiments. The reason is that an accurate prediction takes as much time as just trying to reduce the error and continuing with another branch if the error for the firstly chosen one cannot be reduced quickly.

In contrast to the choice of the node, the heuristic for choosing a random variable must be a good predictor for how much this simplifies the formula, as this choice influences all descendants of the node in the tree. We use a heuristics preferring variables occurring more frequently, as also common for binary model counting [Sang *et al.*, 2005, Section 3.2]. For continuous variables we however have to adapt the heuristic to make sure that continuous variables are not selected repeatedly without an opportunity to eliminate a primitive constraint, and thereby simplifying the formula. For instance, in Example 3, simplification of the constraint can only be achieved if both variables are selected in an alternating order.

For the binary variables there is obviously only a single partitioning possible ($\{true\}$ and $\{false\}$). The partitioning of a continuous random variable amounts to picking a point, on which the range is split further. Initially, we find points that can lead to a simplified sub-formula. For instance, in Figure 2 the choice $\mathbf{L} \in (20, \infty)$ in the left branch can be used to simplify the formula to $\bot$, given the choice for $\mathbf{T}$ above in the tree. In case there are multiple of those points, we count how many sub-formulas can be simplified by choosing this point. There are also cases in which there is no such point, as at the top or the rightmost evaluated node in Figure 2. In this case we use a heuristic trying to maximise the probability of the path eliminating the constraint.

Proposition 5 *IHPMC (Algorithm 1) terminates, i.e. it always finds an approximation with the desired error bound in finite time, given the heuristics as described above.*

4 Experiments

4.1 Set-up

The algorithm was implemented in the functional programming language Haskell[1]. A comparison was made with the sampler implementations of *BLOG* 0.8[2] [Milch *et al.*, 2005] and *Distributional clauses*[3] [Gutmann *et al.*, 2011] (DC); both are optimised for problems with logical structure. We compare to the Likelihood weighted (BLOG LW) and the naive rejection sampler (BLOG RJ) of *BLOG*; the MCMC sampler fails completely on the problem. To allow for a comparison, we ignore the bounded error guarantee of IHPMC and just consider the error of approximations. We measure the squared error and for the sampling approaches the mean

[1] http://www.steffen-michels.de/ihpmc
[2] https://bayesianlogic.github.io
[3] https://github.com/davidenitti/DC

squared error over 50 runs versus inference time. All experiments were run on a Laptop with a Intel Core i3 2.4 GHz processor and 4GB of RAM. We use the following diagnostic problem:

$$fails(i) \leftarrow \mathbf{Break}(i) = true$$
$$fails(i) \leftarrow \mathbf{Temp} > \mathbf{Limit}(i)$$
$$fails(i) \leftarrow i \neq 0, fails(i-1)$$

The number of components (n) and probability of $\mathbf{Break}(i)$ (p) are varied in the experiments. For the continuous variables we use the following distributions: $\mathbf{Temp} \sim \mathcal{N}(20.0, 5.0)$ and $\mathbf{Limit}(i) \sim \mathcal{N}(\mu, 5.0)$, where μ is a parameter that we also vary during the experiments. Note that since we use a single variable $\mathbf{Temp}$ for all components, dependencies are created such that probabilities cannot be computed exactly in a straightforward way.

4.2 Results

Non-Conditional Probabilities With Varying Parameters
The results depicted in Figure 3a are based on a run with parameters, where the continuous cause has a much higher probability than for a realistic diagnostic problem, but is used to illustrate behaviour of the algorithm. In the first milliseconds the error of IHPMC is very unstable. Note that the actual error depicted in the figure is not the guaranteed error bound, which is usually much larger. Actually, it takes about 2.6 seconds before the error is guaranteed to be below 0.01. Decreasing the error bound can temporarily result in an increased error, which explains that the error does not decrease monotonically. In contrast the errors of the sampling algorithms decrease more or less monotonically, as the errors were averaged over several runs.

IHPMC can profit from the case in which the discrete cause has more impact, as shown by the experiment in Figure 3b. The error bound drops below 0.01 already after about 6 milliseconds. The sampling algorithms can also profit from the discrete structure, but are clearly outperformed by IHPMC. Similarly, IHPMC can profit if the event is more rare, which is more realistic for a diagnostic problem (Figure 3c). Here is takes about 28 milliseconds to get an error bound of 0.01. In this case, IHPMC cannot make use that much of the discrete structure, but in the first iterations a single path with a high probability is found that implies $\mathbf{Break}_0 = false \wedge \mathbf{Temp} \leq \mathbf{Limit}_0 \wedge \cdots \wedge \mathbf{Break}_{n-1} = false \wedge \mathbf{Temp} \leq \mathbf{Limit}_{n-1}$. This disproves $fails_{n-1}$, which means the upper probability bound jumps to a small value.

Scalability
To show scalability of IHPMC we performed an experiment with a larger set of components ($n = 100$), where it takes about 2.3 seconds to get a guaranteed error below 0.01. As the results in Figure 3d show, in the beginning two of the sampling implementations outperform IHPMC, as sampling algorithms by their nature do not suffer from a high number of dimensions. However, recall that IHPMC is outperformed by algorithms providing much less information, as no guarantees about the error are provided by the sampling algorithms. After about 1.5 second however the results of IHPMC become competitive.

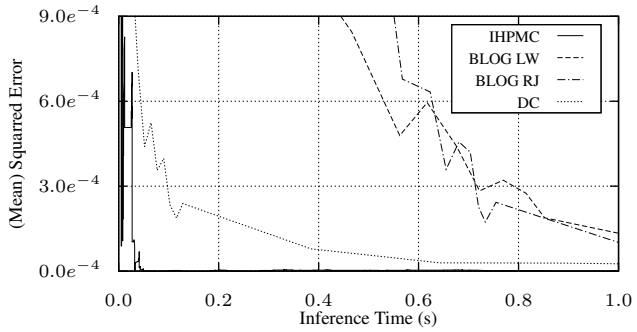
Computing Probabilities Conditioned on Rare-Event
Finally, in Figure 3e, we show that IHPMC is superior if probabilities conditioned on rare events are computed, which is quite natural for a diagnostic setting. The performance of IHPMC is also affected by conditioning on the rare event; it takes about 250 milliseconds to guarantee an error below 0.01, while for the probability of the evidence only, this takes about 15 milliseconds. The performance of the samplers however decreases significantly more. Error measures only tell half of the story as IHPMC provides a different kind of result as sampling approaches. This is illustrated in Figure 3f, where the bounds computed by IHPMC have been visualised together with point approximations produced by a sampler. Each point represents a single run of the sampler, which can or cannot be within the bounds computed by IHPMC. However, as a user of a sampling method, one only obtains a single point with, if at all, only a very weak guarantee about its error.

5 Related Work

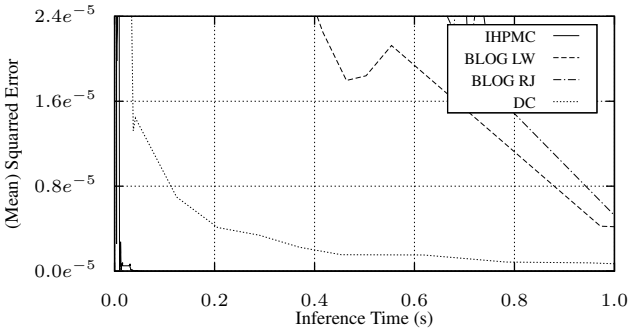
Probabilistic LP has been extended with continuous distributions, in a way that still allows for exact inference, by either restricting the events expressible [Gutmann *et al.*, 2010] or the distributions employable [Islam *et al.*, 2012]. There are also representations of continuous distributions that are powerful enough to approximate arbitrary distributions, but still allow for exact inference, e.g. *piecewise polynomials* [Sanner and Abbasnejad, 2012]. A recent algorithm for inference on such representation, which exploits the logical structure of distributions was presented in [Belle *et al.*, 2015a]. Such approximations can in principle resemble intended distributions very closely, but there is no guarantee about the approximation error of event probabilities, as a small probability of evidence can significantly increase the error made when computing conditional probabilities.

Distributional clauses [Gutmann *et al.*, 2011] is a probabilistic logic programming language, that supports arbitrary continuous distributions and events, as also supported by various other probabilistic programming languages, such as *BLOG* [Milch *et al.*, 2005]. Inference for such languages is only possible by sampling approaches, usually *Markov chain Monte Carlo* (MCMC) methods, which can be very effective if they are hand-tailored for specific problems. Generic MCMC frameworks are however suboptimal and can provide poor results when rare events are observed. A significant difference with our work is that even under perfect circumstances, when detailed knowledge about the approximated distributions is available, guarantees for approximations obtained by MCMC algorithms can only be given in terms of for instance the *Monte Carlo standard error*, which gives a probabilistic, but no hard guarantee.

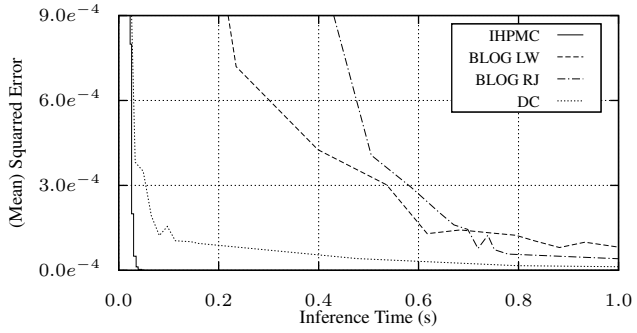
Recent hashing-based sampling methods, which exploit logical structure of distributions, have been extended for hybrid distributions as well [Belle *et al.*, 2015b; Chistikov *et al.*, 2015]. Such methods provide a probabilistic guarantee on the quality of approximations, but are restricted to exactly computable representations of continuous distributions (piecewise-polynomials [Belle *et al.*, 2015b] and uniform distributions [Chistikov *et al.*, 2015]). Moreover, this



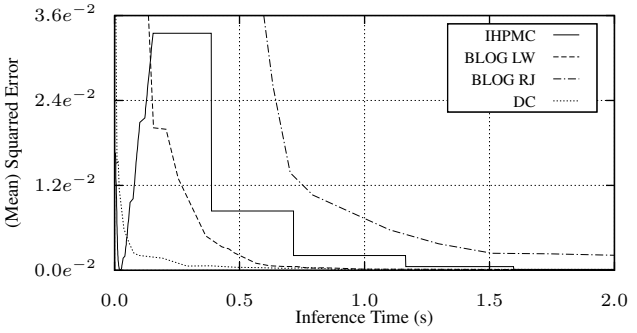
(a) $P(\mathit{fails}(9))$, $n = 10$, $p = 0.01$, $\mu = 30.0$



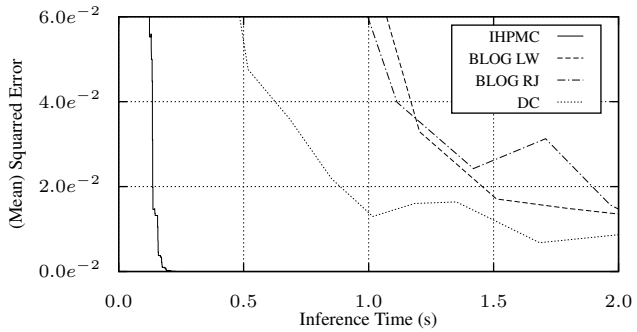
(b) $P(\mathit{fails}(9))$, $n = 10$, $p = 0.3$, $\mu = 30.0$



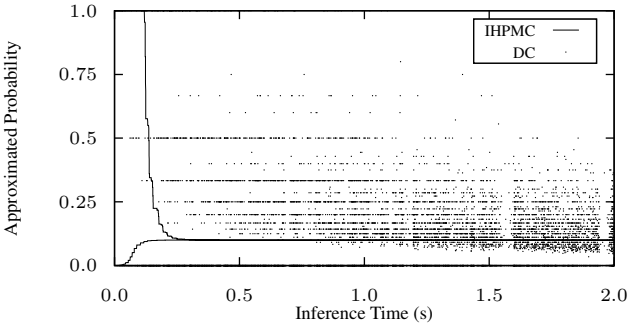
(c) $P(\mathit{fails}(9))$, $n = 10$, $p = 0.01$, $\mu = 60.0$



(d) $P(\mathit{fails}(99))$, $n = 100$, $p = 0.01$, $\mu = 60.0$



(e) $P(\mathit{fails}(9) \mid \mathit{fails}(0))$, $n = 10$, $p = 0.0001$, $\mu = 60.0$



(f) $P(\mathit{fails}(9) \mid \mathit{fails}(0))$, $n = 10$, $p = 0.0001$, $\mu = 60.0$

Figure 3: Experiment Results

remains a probabilistic guarantee on non-conditional probabilities, while IHPMC provides a hard guarantee for computed conditional probabilities.

Several approximation methods have been proposed that provide hard guarantees for discrete distributions [Poole, 1993]. Recently, very effective methods based on state-of-the-art knowledge compilation techniques have been proposed [Renkens *et al.*, 2014; 2015; Vlasselaer *et al.*, 2015]. In contrast to IHPMC, these algorithms are restricted to discrete problems and do not handle evidence, which are significant limitations in practice. In [Michels *et al.*, 2015], we provided a theoretical basis for approximate inference with continuous distributions, but did not provide an efficient and practical algorithm for actually computing good approximations.

6 Conclusions

In this paper, we propose a hybrid extension for probabilistic logic programming, which allows for a much wider class of continuous distributions than existing extensions allowing for exact inference. At the same time, our extension allows one to compute approximations with bounded and arbitrarily small error. We propose a novel anytime algorithm for computing such approximations, thereby exploring the logical and continuous structure of the problem by utilising information from previous iterations. We experimentally show that our algorithm is competitive with state-of-the-art sampling algorithms, for typical relational problems, which are characterised by a moderate number of dimensions and a significant amount of structure, in spite of the fact that it provides

much stronger guarantees. Our method outperforms sampling methods by far if rare events with deterministic structure are provided as evidence. Moreover, even when our algorithm converges slowly for a certain problem, this is clearly visible to the user, thereby preventing wrong conclusions. MCMC methods do not offer similar safeguards.

References

[Belle *et al.*, 2015a] 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)*, 2015.

[Belle *et al.*, 2015b] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. Hashing-based approximate probabilistic inference in hybrid domains. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

[Chavira and Darwiche, 2008] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

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

[Fierens *et al.*, 2015] 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(03):358–401, 2015.

[Gutmann *et al.*, 2010] Bernd Gutmann, Manfred Jaeger, and Luc De Raedt. Extending ProbLog with continuous distributions. In Paolo Frasconi and Francesca Alessandra Lisi, editors, *Proc. of the 20th International Conference on Inductive Logic Programming (ILP–10)*, Firenze, Italy, 2010.

[Gutmann *et al.*, 2011] Bernd Gutmann, Ingo Thon, Angelika Kimmig, Maurice Bruynooghe, and Luc De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11:663–680, 2011.

[Islam *et al.*, 2012] Muhammad Asiful Islam, C. R. Ramakrishnan, and I. V. Ramakrishnan. Inference in probabilistic logic programs with continuous random variables. *Theory and Practice of Logic Programming*, 12(4-5):505–523, 2012.

[Kimmig and Costa, 2012] Angelika Kimmig and Fabrizio Costa. Link and node prediction in metabolic networks with probabilistic logic. In *Bisociative Knowledge Discovery*, pages 407–426. Springer, 2012.

[Lauritzen, 1992] Steffen L Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

[Michels *et al.*, 2013] Steffen Michels, Marina Velikova, Arjen Hommersom, and Peter JF Lucas. A decision support model for uncertainty reasoning in safety and security tasks. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 663–668. IEEE, 2013.

[Michels *et al.*, 2015] Steffen Michels, Arjen Hommersom, Peter J. F. Lucas, and Marina Velikova. A new probabilistic constraint logic programming language based on a generalised distribution semantics. *Artif. Intell.*, 228:1–44, 2015.

[Michels, 2016] Steffen Michels. *Hybrid Probabilistic Logics: Theoretical Aspects, Algorithms and Experiments*. PhD thesis, Radboud University, May 2016.

[Milch *et al.*, 2005] Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: probabilistic models with unknown objects. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 1352–1359, 2005.

[Moldovan *et al.*, 2012] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, José Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4373–4378. IEEE, 2012.

[Pearl, 1988] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[Poole, 1993] David Poole. Logic programming, abduction and probability. *New Generation Computing*, 11(3-4):377–400, 1993.

[Poole, 2008] David Poole. The independent choice logic and beyond. In Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors, *Probabilistic inductive logic programming*, pages 222–243. Springer-Verlag, Berlin, Heidelberg, 2008.

[Renkens *et al.*, 2014] Joris Renkens, Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Explanation-based approximate weighted model counting for probabilistic logics. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[Renkens *et al.*, 2015] Joris Renkens, Angelika Kimmig, and Luc De Raedt. Lazy explanation-based approximation for probabilistic logic programming. *arXiv preprint arXiv:1507.02873*, 2015.

[Sang *et al.*, 2005] Tian Sang, Paul Beame, and Henry Kautz. Heuristics for fast exact model counting. In *Theory and Applications of Satisfiability Testing*, pages 226–240. Springer, 2005.

[Sanner and Abbasnejad, 2012] Scott Sanner and Ehsan Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *AAAI*, 2012.

[Vlasselaer *et al.*, 2015] Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt. Anytime inference in probabilistic logic programs with tp-compilation. 2015.