Importance Sampling on Relational Bayesian Networks

Manfred Jaeger

Institut for Datalogi, Aalborg Universitet, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø
jaeger@cs.aau.dk

Abstract. We present techniques for importance sampling from distributions defined by Relational Bayesian Networks. The methods operate directly on the abstract representation language, and therefore can be applied in situations where sampling from a standard Bayesian Network representation is infeasible. We describe experimental results from using standard, adaptive and backward sampling strategies. Furthermore, we use in our experiments a model that illustrates a fully general way of translating the recent framework of Markov Logic Networks into Relational Bayesian Networks.

1 Introduction

Numerous representation languages have been proposed over the last 10 years that integrate logical or relational representations with probabilistic graphical models [21, 14, 15, 19, 10, 7, 13, 20]. For many of these languages the proposed method for probabilistic inference is *knowledge-based model construction (kbmc)* [1]: for a particular instance of the generic, logical model a Bayesian network is constructed on which inference then is performed with standard Bayesian network algorithms. Unfortunately, Bayesian networks representing such model instances tend to become very large and complex, and so are often intractable for exact inference algorithms.

To address this problem, several different approaches have been investigated. The most important breakthrough for the inference problem could be achieved by methods that perform inference directly on the abstract level of the logical representation, instead of propositionalizing, or grounding, every concrete model instance. Complexity results given in [11] establish some limitations to the possible success of such methods. Poole [18], on the other hand, presents a first-order level inference technique, but the practical applicability of this technique is as yet untested. In [2] a method is described that expands the scope of problems manageable with kbmc: instead of applying standard inference techniques on the constructed Bayesian network, the network is compiled into an arithmetic circuit, which is an often much more compact and efficient support structure for inference than a junction tree [5].

In view of the inherent complexity of exact inference for logical models, the most promising approach for practical problems lies in approximate inference techniques. Approximate, sampling-based, inference techniques for probabilistic-logical models have first been systematically studied in [4, 16]. In the present paper we present an approximate inference method for *relational Bayesian Networks (RBNs)* based on importance sampling. The distinguishing feature of this method is that while it operates on a Bayesian network structure, it utilizes the underlying logical representation language for the specification of the conditional probability tables (cpts), and by that means can operate on models for which methods based on standard cpt representations are not applicable.

The methods we here develop are intended for inclusion in the public domain *Primula* system (www.cs.aau.dk/∼jaeger/Primula). An important design goal, therefore, is that they are robustly applicable to the wide variety of models that can be specified in the RBN language, and that they do not require special purpose techniques or heuristics for every particular model.

In the following section we give a more detailed review of previous work, and re-introduce a motivating example from [16].

2 Sampling on the BN

Based on a logical-relational representation language that supports kbmc one can directly run sampling algorithms on the constructed Bayesian networks. However, this approach finds its limitations when the Bayesian networks becomes intractably large due to nodes with a large number of parents. The following example, adopted from Pasula and Russell [16] and going back to [17] illustrates the difficulties.

Example 21 Consider a domain consisting of *professors* and *students*. A professor has attributes *fame* and *funding*, with *funding* depending probabilistically on *fame*. A student has the attribute *success*, which depends probabilistically on the fame of his *advisor*, where *advisor* is a probabilistic relation between students and professors. Exact specifications of this probabilistic model can be given in a variety of representation languages. When kbmc is applied to the generic model and a concrete domain consisting of 3 professors and 1 student, then a Bayesian network with the structure shown in figure 1 will be built. The node *advisor(s1)* has as possible values the possible advisors of student *s1*, i.e. professors *p1,p2,p3*.

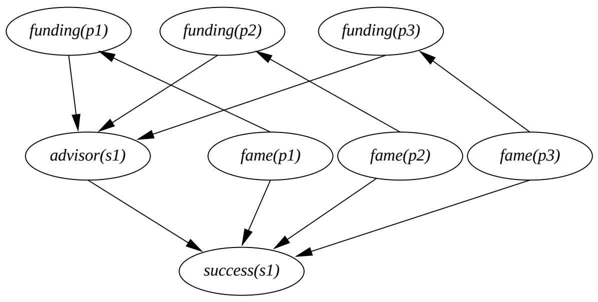


Fig. 1. Bayesian Network from KBMC

The problem with this network construction is evident: the number of parents of the *advisor* and *success* nodes increases with the number of professors in the domain, and so the Bayesian network representation becomes exponential in the domain size when explicit cpt representations are used for the specifications of the conditional distributions at the nodes. Thus, it is not possible to use generic Bayesian network sampling algorithms for approximate inference for this model.

Pasula and Russell bypass this problem for this particular model using two strategies: first, they observe that e.g. *fame(p2)* and *fame(p3)* can be removed as parents from *success(s1)* when *advisor(s1)* is instantiated to *p1*. Given an instantiation of all other nodes, the value of *success(s1)* can thus be always efficiently re-sampled from a low-dimensional distribution. Secondly, they assume a particular functional form (a *softmax* distribution) of the dependence of *advisor(s1)* on *funding(p1)-funding(p3)* , so that no extensive cpt representations are needed for the re-sampling of *advisor(s1)*.

The techniques in [16] are not based on any specific, well-defined representation language, and often utilize specific features of the particular models under consideration. They therefore do not translate into a generic approximate inference technique for RBNs. Much closer to our needs, in that respect, are the sampling techniques for *Stochastic Logic Programs (SLPs)* described in [4]. These techniques are fully general for SLP models, and they operate on structures (proof trees) that directly utilize the SLP language. On the other hand, SLPs and RBNs are most naturally used for the representation of quite different types of models. Furthermore, the sampling methods for SLPs cannot handle evidence (except via straightforward rejection sampling), whereas sampling in the presence of unlikely evidence will be a key concern for our methods.

3 Relational Bayesian Networks

Relational Bayesian Networks [10] are a powerful representation language for for logical/relational models. The language is based on *probability formulas* that are associated with each probabilistic relation. These probability formulas define for each ground atom over the probabilistic relations both its dependence on other atoms, and the exact conditional probability distribution. We illustrate the use of probability formulas with the formula that defines the distribution for the probabilistic *success* relation of example 21:

```
       (advisor(s,p):
                (fame(p):
                    0.7,
n-or{                0.2        |p:professor(p)}
                 ),
                  0
        )
```
(1)

The structured format used in (1) just highlights the syntactic components of the formula, which, in reality, should be just seen as a linear string. The two dashed boxes in the formula show two (nested) occurrences of the *convex combination* probability formula construct. The solid box indicates an occurrence of the *combination function* construct, here used with the *noisy-or* combination function. These two constructs, together with the base constructs of *relational atoms* and *(numerical) constants* constitute the whole syntax of probability formulas (see [10, 12] for a full formal definition). Probability formulas can be seen as a probabilistic analog of predicate logic formulas, with the convex combination construct (basically a probabilistic if-then-else construct) corresponding to Boolean connectives, and the combination function construct corresponding to quantification. Different combination functions can be used; the noisy-or shown in our example is the probabilistic analog of existential quantification.

The probability formula for *success* contains atoms formed from the probabilistic relations *advisor,fame*, and an atom formed from the *predefined relation professor*. A predefined relation is required to be fully known for each concrete domain over which the general model is instantiated. For the model of example 21 we require the two predefined *type* or *class* predicates *professor* and *student* (the predefined relations in the RBN framework correspond to the *skeleton structures* of [7]).

The probability formula for *success*, together with probability formulas for *fame*, *funding* and *advisor* constitute a RBN representation of the model from example 21. Given the general RBN model, and a domain with three professors and one student, one can construct the Bayesian network shown in figure 2 (this kbmc procedure is implemented in the *Primula* system). The network in figure 2 differs from the one in figure 1 in that the node *advisor(s1)* with possible values $p1, p2, p3$ has been decomposed into three boolean nodes *advisor(s1,p1),...,advisor(s1,p3)*. This is because RBNs always define joint probability distributions over ground relational atoms (seen as boolean random variables). The specification of the joint distribution ensures that the three atoms *advisor(s1,p1),...,advisor(s1,p3)* behave like a functional selection, i.e. with probability one exactly one of the three atoms will be true.

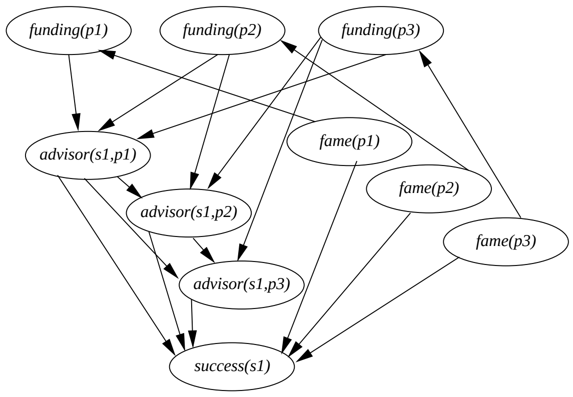


Fig. 2. Bayesian Network from RBN representation

In the kbmc process from the RBN representation the cpt for the node *success(s1)* is computed by substituting *s1* for the free variable s in (1), and then recursively evaluating the resulting ground probability formula. This recursive evaluation first requires that a list is produced of all objects *p* in the current domain that belong to the predicate *professor*. In general it also is allowed that combination functions quantify over boolean conditions involving several predefined relations, and therefore this processing step is essentially a database query. Next, the elements of the computed list *p1,p2,p3* are substituted for the variable p in the sub-formula of (1) that consists of the outer convex combination. This results in three different ground probability formulas, which are evaluated recursively. The recursion stops with the evaluation of probability formulas that are either constants or ground atoms. The set of all ground atoms encountered in the recursive evaluation of the probability formula for *success(s1)* is the set of parents of the *success(s1)* node in the constructed network.

4 Sampling Structure

We have seen in the preceding section how a ground probability formula compactly represents all the information needed to construct the cpt for a ground atom node in a Bayesian Network. When kbmc is performed to subsequently apply standard exact inference algorithms, such standard table representations are required. However, when we only want to sample from the network, then it is enough to represent the conditional distributions with the original ground probability formulas. When the value of a ground atom needs to be sampled, while truth values of its parents are given, then we only need to evaluate its probability formula, which gives us the relevant conditional probability for the truth of the current atom, given the existing instantiations.

Thus, we can construct a computational structure that supports sampling procedures by constructing a standard Bayesian network structure as usual, but omitting the computation of standard cpt representations, and labeling the nodes with ground probability formulas instead. There is a problem with the *sampling network* constructed in this way, however: consider, for example, a node labeled with a ground instance of (1). Each time a value is sampled for this node, we need to perform the database query that gives us the list of all professors *p*. The result of this query does not depend on the instantiations of ground atoms in the current (partial) sample, and therefore the same, potentially very costly, computation will be performed over and over again. Clearly, such duplicated computations should be avoided. Obviously, any method for caching the results of the database queries will solve this problem. However, probability formulas provide a particularly simple solution: we can replace the original probability formula with a *pre-evaluated* formula, in which all required substitutions for the quantified variables are explicitly represented. Figure 3 shows the result of pre-evaluating (1) for $\mathtt{s} = s1$ over our example domain. These pre-evaluated formulas are still legitimate probability formulas according to the original syntax, and their use does not require any extensions or modifications of existing algorithms for handling probability formulas. Pre-evaluation can be applied recursively also to probability formulas containing nested occurrences of combination functions, and leads to probability formulas whose evaluation no longer requires querying any of the predefined relations. The size of the pre-evaluated formula can be much larger than the original formula, but always is polynomial in the size of the domain.

It can be possible that a (pre-evaluated) formula is relatively complex, and its evaluation in the sampling process time consuming, even though it depends only on a small number of ground atoms. In such cases it is more efficient to use a standard cpt representation for the conditional distribution at this node. We, thus, obtain three different levels of representation: the original ground probability formula, the pre-evaluated probability formula, and the standard cpt. When moving from a higher to a lower level we are trading compactness of representation for faster evaluation. Representations from any of the three levels can be the most useful ones under certain circumstances (in particular, when the database query required to evaluate the original probability formula is simple to compute, but gives a relatively large result, then pre-evaluation may not be advisable). In the current implementation, however, we use the following simple rule: when a node has no more than 3 parents, then we use a cpt representation for its conditional distribution; for nodes with more than 3 parents pre-evaluated probability formulas are used.

```
         (advisor(s1,p1):      (advisor(s1,p2):      (advisor(s1,p3):
                 (fame(p1):            (fame(p2):            (fame(p3):
                      0.7,                  0.7,                  0.7,
n−or{                 0.2                   0.2                   0.2      }
                   ),                    ),                    ),
                   0                     0                     0
         )                     )                     )
```

Fig. 3. Pre-evaluated Probability Formula

Typically we will want to sample from a network where some evidence nodes have been instantiated to a known truth value. In many networks constructed from logical/relational models we find a large amount of determinism, i.e. 0/1-valued conditional probabilities. These deterministic dependencies can be due to deterministic components in the distribution of interest. In addition, the strictly binary encodings defined by RBN representations generate further deterministic dependencies, e.g. the deterministic dependencies between the *advisor(s1,·)* nodes in the network of figure 2. Due to these deterministic dependencies, it will often be the case that the entered evidence strictly implies the value of some other nodes. To improve the efficiency of sampling, it is useful to first try to find as many as possible of such implied instantiations. For this reason, we apply to the constructed sampling network as a further pre-processing step the `propagate_deterministic` procedure shown in table 1 (slightly simplified).

The *is_locally_inconsistent(v,true)* method checks whether instantiating v to true is consistent with known instantiations for parents and children of v. The *add_neighbors_to_stack(v)* method adds all nodes to the stack that are not instantiated already, and which are either parents or children of v, or are parents of instantiated children of v. While clearly not complete (detecting all deterministically implied instantiations is an NP-hard problem), the `propagate_deterministic` is quite successful in propagating deterministic dependencies. It will, for example, set *advisor(s1,p1)* and *advisor(s1,p3)* to false when *advisor(s1,p2)* is instantiated to true.

5 Importance Sampling

The sampling network whose construction was described in last section can be used to support different sampling techniques. We focus here on different versions of *importance sampling*, because importance sampling may be expected to be more robustly

```
propagate_deterministic
    stack:= all instantiated nodes
    while stack is nonempty
        $v$:=stack.pop()
        if is_locally_inconsistent(v,true)
            $v$ := false; newinst:=true
        if is_locally_inconsistent(v,false)
            $v$ := true; newinst:=true
        if newinst
            add_neighbors_to_stack(v)
```

Table 1. Deterministic Propagation

applicable to a wide variety of models than Markov-Chain-Monte-Carlo sampling. This is because the often complex logical dependencies we find in our models make it very difficult to devise generic proposal distributions that will lead to irreducible Markov chains in Monte-Carlo sampling. When we are dealing with specific models, or restricted classes of models, however, it can be expected that MCMC sampling with carefully designed proposal distributions is more efficient than our generic importance sampling.

In this paper we investigate three different versions of importance sampling: basic importance forward sampling (also known as *likelihood weighting* [8]), *adaptive* forward sampling [3], and *backward* sampling [9]. The techniques and results of this section are not strictly linked to approximate inference for logical/relational models. They are applicable to any large Bayesian networks, especially those with many 0/1-valued parameters and/or nodes with many parents.

In the following we denote with $\boldsymbol{X}$ the set of all random variables (i.e. all ground atoms of probabilistic relations) in an instance of a logical/relational model. $P(\boldsymbol{x})$ is the probability according to the model of an instantiation $\boldsymbol{x}$ of $\boldsymbol{X}$. With $\boldsymbol{E} = \boldsymbol{e}$ we denote an instantiation of variables $\boldsymbol{E}$ to observed values $\boldsymbol{e}$. In all versions of importance sampling that we use, joint instantiations $\boldsymbol{x}$ are generated according to some distribution $Q(\boldsymbol{x})$. Q is such that $Q(\boldsymbol{x}) > 0$ only if $\boldsymbol{x}$ is consistent with $\boldsymbol{E} = \boldsymbol{e}$, and for $\boldsymbol{x}$ that is consistent with this evidence $P(\boldsymbol{x}) > 0$ implies $Q(\boldsymbol{x}) > 0$. The empirical mean of the *importance weights* $P(\boldsymbol{x})/Q(\boldsymbol{x})$ is an unbiased estimator for the marginal probability of $\boldsymbol{E} = \boldsymbol{e}$. For a non-instantiated variable Z, the empirical mean of $P(\boldsymbol{x})/Q(\boldsymbol{x})1_{Z=z}(\boldsymbol{x})$ is an estimator for the conditional probability of $Z = z$, given $\boldsymbol{E} = \boldsymbol{e}$, where $1_{Z=z}(\boldsymbol{x})$ is a function that returns value 1 or 0, according to whether $Z = z$ in the sample $\boldsymbol{x}$. This estimator can be biased (see [3] for more information on bias in importance sampling).

In *simple forward sampling* nodes are sampled such that all parents of a node are sampled before the node itself. An uninstantiated node is sampled according to P, whereas instantiated nodes E are deterministically set to their known values e.

In *adaptive forward sampling* one tries to learn in the course of the sampling process sampling distributions for uninstantiated nodes that approximate the posterior distribution of the node given the evidence. In our implementation of adaptive sampling we follow essentially [3]. The basic idea is to maintain for every node X a second *impor-*

tance conditional probability table (icpt). Like the original cpt, the icpt has one row for every configuration $\boldsymbol{y}$ of the parents $\mathit{Pa}(X)$ of X. The table entry for $\boldsymbol{y}$ is an approximation for $P(X = \mathit{true} \mid \mathit{Pa}(X) = \boldsymbol{y}, \boldsymbol{E} = \boldsymbol{e})$. It is maintained as

$$\frac{\mathit{atw}^k(\boldsymbol{y})}{\mathit{atw}^k(\boldsymbol{y}) + \mathit{afw}^k(\boldsymbol{y})}$$

where $\mathit{atw}^k(\boldsymbol{y})$ is the average importance weight among those of the first k samples that instantiated $\mathit{Pa}(X) = \boldsymbol{y}$ and $X = \mathit{true}$; similarly for $\mathit{afw}^k(\boldsymbol{y})$ with $X = \mathit{false}$. The conditional sampling distribution $Q^k(X \mid \boldsymbol{y})$ in the kth sample is a weighted combination $(1 - \lambda(k, \boldsymbol{y}))\mathit{cpt}(\boldsymbol{y}) + \lambda(k, \boldsymbol{y})\mathit{icpt}(\boldsymbol{y})$. A key problem is a good choice of the factors $\lambda(k)$, which should give increasing weight to the icpt entries as sampling progresses. In our implementation we are currently using

$$\lambda(k, \boldsymbol{y}) = \frac{\mathit{atw}^k(\boldsymbol{y}) + \mathit{afw}^k(\boldsymbol{y})}{\mathit{atw}^k(\boldsymbol{y}) + \mathit{afw}^k(\boldsymbol{y}) + p},$$

where p is a positive parameter (in our experiments set to $p = 5$). Thus, $\lambda(k, \boldsymbol{y})$ converges to 1 as the total weight of samples with parent configuration $\boldsymbol{y}$ increases.

The adaptation of the sampling distribution as described so far suffers from the problem that we again require explicit representations of the icpt tables, and that these tables, again, grow exponentially in the number of parents. Since, thus, this adaptation technique is infeasible for nodes with many parents, we restrict it to nodes for which we maintain explicitly cpt representations also for the underlying distribution P, i.e. nodes with no more than three parents. Nodes whose conditional distribution is represented with a (pre-evaluated) probability formula are sampled according to $P(X \mid \mathit{Pa}(X) = \boldsymbol{y})$.

Forward sampling techniques, whether simple or adaptive, will face difficulties when we have many instantiated nodes near the leaves of the network. *Backward sampling* has been proposed as a possible alternative in such cases [9]. In our version of backward sampling we do not follow [9] very closely, neither in the construction of the sampling order, nor in the definition of the conditional sampling distribution. Main reason for this is that the method of [9] requires that in certain backward sampling steps all non-instantiated parents of an instantiated node are simultaneously instantiated according to likelihood of the current node values under different joint instantiations of the parents. This requires consideration of all joint instantiations of the parents, which is, yet again, infeasible in our networks.

We construct a sampling order for backward sampling as follows: first each node that is an ancestor of some instantiated node is assigned a *level*, which is the length of the shortest undirected path to the nearest instantiated node. These ancestor-of-evidence nodes are sampled first, in an arbitrary order consistent with the level assignment. The remaining nodes are then sampled in an arbitrary order consistent with the topological order of the network.

An ancestor-of-evidence node X is sampled as follows: first, it is determined whether both $X = \mathit{true}$ and $X = \mathit{false}$ are locally consistent as determined by the *is_locally_inconsistent* method we already used in the propagation of evidence (cf. table 1). If neither value for X is consistent, then the current sample is abandoned. If only one of the values is consistent, X is set to that value. If both values are consistent, then one is chosen randomly with probability 1/2. Thus, in these sampling operations we do not try to utilize at all the underlying distribution P. After each backward sampling step deterministic implications of the new instantiation are propagated with the method of table 1.

6 Experiments

6.1 Models used

We used three different RBN models for our experiments. The first is the *students_and-_professors* model from example 21. The other two are described below. The full RBN encodings of the models that were used in the experiments can be found at www.cs.aau.dk/∼jaeger/Primula/Examples.

Example 61 *(Linkage)* Our model *linkage_1locus* is a model for genetic linkage, which describes the inheritance of genetic variations in a pedigree. In addition to random variables that describe the genotypes of the people in the pedigree, the model also contains random variables that encode the exact trace of the gene variations in the pedigree. The model is instantiated over specific pedigrees, which are given by predefined *father* and *mother* relations. The Bayesian networks constructed for specific instances are similar in structure to the Bayesian networks used in the *Superlink* system [6]. Instances of the *linkage_1locus* model contain a high degree of inherent deterministic dependencies, due to deterministic constraints on genotypes of related individuals.

Example 62 *(Agent communication)* This probabilistic toy model for communication in a multi-agent system is instantiated by domains consisting of agents and and a time structure. Figure 4 shows an example of a domain with eight agents and a time structure consisting of five points in time. Some agents are connected with direct communication channels. In figure 4 these are shown by the connecting lines. An agent can have the special attribute of being a *source* agent. In Figure 4 agent 1 is the single source agent. A domain, thus, is specified by a number of objects, and the predefined relations *agent,timepoint,connected,source,timeord*, where *timeord* defines a linear order on timepoints. The probabilistic model, now, is as follows: at time t_1 a message is given to all source agents in the system. At each point in time, each agent sets each of its communication channels randomly (with equal probability) into *receive-mode* or *send-mode*. If at some time point t_i agent k sets its communication channel with agent j into *receive-mode*, agent j sets that channel into *send-mode*, and agent j had the message at time t_{i-1}, then agent k will have the message at time t_i. Once received, the message is never forgotten by an agent, i.e. it can be transmitted to connected agents at all subsequent time points. This probabilitstic model is encoded by a relational Bayesian network via the two probabilistic relations *receivemode*(a_1, a_2, t) and *hasmessage*(a, t), where *receivemode*(a_1, a_2, t)*=true* means that a_1 has set its channel with a_2 into receive-mode, *receivemode*(a_1, a_2, t)*=false* means that a_1 has set its channel with a_2 into send-mode, and *hasmessage*(a, t)*=true* means that a has the message at time t.

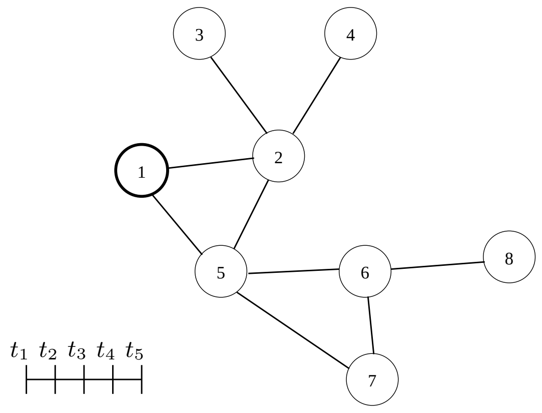


Fig. 4. Agent communication example

The next example is presented at greater length, because it includes a translation of *Markov Logic Networks* int RBNs, which is of independent interest.

Example 63 The *Markov Logic Networks* of Richardson and Domingos [20] are an intuitive and modular representation language that combines first-order logic and probabilistic modeling in a very direct way. The language simply consists of weighted first-order predicate logic formulas. Table 2 gives an example taken from [20]. The model describes uncertainty over the relations *Fr(iends),Sm(okes)*, and *Ca(ncer)*. The weight attached to a formula specifies a bias for ground instances of the formula to be satisfied in a (logical) model of the knowledge base. More precisely, for a given finite domain $C = \{c_1, \ldots, c_n\}$, the MLN of table 2 defines a probability distribution over all possible worlds ω with domain C and relations *Fr,Sm,Ca* via

$$P(\omega) = \frac{1}{Z} \prod_i e^{n_i(\omega) w_i} \tag{2}$$

where the index i ranges over the formulas in the MLN, w_i is the weight of the ith formula, $n_i(\omega)$ is the number of true ground instances in ω of the ith formula, and Z is a normalization constant. This distribution is equivalently specified by the log-odds of pairs of possible worlds:

$$\textit{log}(P(\omega)/P(\omega')) = \sum_i w_i(n_i(\omega) - n_i(\omega')). \tag{3}$$

To obtain an RBN encoding of this distribution, consider the Bayesian network of figure 5. The node Ω represents a random variable ranging over possible worlds ω. The nodes $W_i(\boldsymbol{c})$ are boolean random variables ($i = 1, \ldots, 4$; $\boldsymbol{c}$ ranging over all k_i-tuples of elements from C, where k_i is the number of free variables of F_i). Their conditional distribution given Ω is defined by the conditional probability table as shown in table 3. Thus, the probability of $W_i(\boldsymbol{c})$ being true only depends on whether ω satisfies the ground formula $F_i(\boldsymbol{c})$, and is either 1 or $1/e^{w_i}$, accordingly.

i	Formula F_i	k_i	Weight w_i
1	*Fr*$(x,y) \wedge$ *Fr*$(y,z) \Rightarrow$ *Fr*(x,z)	2	0.7
2	$\neg\exists y$*Fr*$(x,y) \Rightarrow$ *Sm*(x)	1	2.3
3	*Sm*$(x) \Rightarrow$ *Ca*(x)	1	1.5
4	*Fr*$(x,y) \Rightarrow ($*Sm*$(x) \Leftrightarrow$ *Sm*$(y))$	2	1.1

Table 2. MLN: friends and smokers

Now consider the conditional distribution on Ω, given that all $W_i(\boldsymbol{c})$-variables are instantiated to true:

$$\begin{aligned} P(\omega \mid \boldsymbol{W}=\boldsymbol{t}) &= \frac{P(\omega)}{P(\boldsymbol{W}=\boldsymbol{t})} P(\boldsymbol{W}=\boldsymbol{t} \mid \omega) \\ &= \frac{P(\omega)}{P(\boldsymbol{W}=\boldsymbol{t})} \prod_i (1/e^{w_i})^{n^{k_i}-n_i(\omega)} \end{aligned}$$

The log-odds according to this conditional distribution are

$$\begin{aligned} &\mathit{log}(P(\omega \mid \boldsymbol{W}=\boldsymbol{t})/P(\omega' \mid \boldsymbol{W}=\boldsymbol{t})) \\ &\quad = \mathit{log}(P(\omega)/P(\omega')) + \sum_i w_i(n_i(\omega) - n_i(\omega')), \end{aligned}$$

and thus are equal to (3) if the prior distribution on Ω is uniform.

A joint distribution of Ω and random variables W_i of the desired form can easily be encoded in a RBN. First, one defines a uniform distribution for Ω, which is equivalent to a uniform joint distribution of the ground relational atoms *Fr*(c_1,c_1), *Fr*(c_1,c_2),…,*Sm*(c_n). This is achieved with the three probability formulas

```
friends(x,y)=0.5
smokes(x)=0.5
cancer(x)=0.5
```

Next, we introduce for each formula F_i a probabilistic relation W_i. We illustrate the general method for constructing the probability formula associated with W_i using F_2 from table 2. The top-level structure of the formula for W_2 is

$$(\mathtt{PF}_2(\mathtt{x}) : \mathtt{1}, \mathtt{0.1002}),$$

where $0.1002 = 1/e^{2.3}$. $\mathtt{PF}_2(\mathtt{x})$ is a sub-formula, such that $\mathtt{PF}_2(c)$ evaluates to 1 for $c \in C$ exactly when the instantiations of the relations *Fr,Sm,Ca* make the formula $F_2(c)$ true. Otherwise $\mathtt{PF}_2(c)$ evaluates to 0. That it is generally possible to construct such *indicator formulas for first-order properties* was shown in [10]. It is a simple consequence of the close connection between the syntax of probability formulas and the syntax of predicate logic, and only requires the use of the noisy-or combination function. For our example formula F_2 we obtain

$$\mathtt{PF}_2(\mathtt{x}) = ((\mathtt{n\text{-}or}\{\mathtt{Fr}(\mathtt{x},\mathtt{y})|\mathtt{y}\}\mathtt{:}\ \mathtt{0},\mathtt{1}) : \mathtt{Sm}(\mathtt{x}), \mathtt{1}).$$

This formula is composed of one combination function corresponding to the existential quantification in F_2, and two convex combination constructs, which translate the negation and implication in F_2.

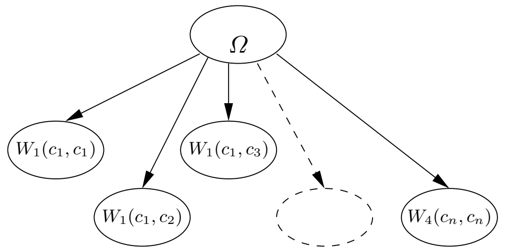


Fig. 5. Translating MLNs

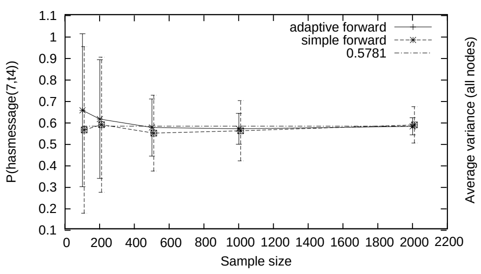
Ω	*true*	*false*
$\omega \models F_i(\boldsymbol{c})$	1	0
$\omega \not\models F_i(\boldsymbol{c})$	$1/e^{w_i}$	$1-1/e^{w_i}$

Table 3. CPT for MLN translations

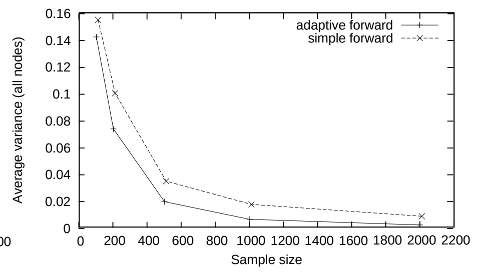
Similarly, one defines relations and probability formulas for the remaining formulas F_1, F_3, F_4. The joint distribution defined by the resulting RBN conditioned on all W_i atoms being true is the original distribution defined by the MLN. That we obtain the distribution of interest as a conditional distribution in a larger state space is of some theoretical significance. For practical purposes, it does not seem to make an important difference, as our additional random variables do not lead to larger computational structures for inference. In fact, the moral graphs of the Bayesian networks we obtain by kbmc from the RBN encoding are basically isomorphic to the Markov networks Richardson and Domingos construct for inference from the MLN representation.

6.2 Results

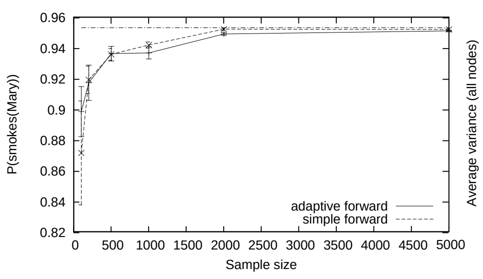
The plots in figure 6 show results from sampling experiments on our four example models. In the first experiment, the agent communication model was instantiated with the domain shown in figure 4. The resulting model instance has 130 random variables) As evidence was entered *hasmessage*$(6,t_3)$ = *true*, *hasmessage*$(2,t_3)$ = *true*, *hasmessage*$(8,t_5)$ = *true*. Plot (a) shows the convergence of the sample estimate of the posterior probability for *hasmessage*$(7,t_4)$. This model is still accessible to exact inference (it has , and the correct probability is computed as 0.5781. The plot shows the estimates for five different sample sizes, where each estimate is averaged over 100 sample runs (thus, in effect, the mean value shown for sample size 200 is, in effect, the estimate obtained in a sample of size 20000). The error bars show the standard deviation of the estimate in the 100 samples. The results indicate a not dramatic, but still significant, advantage of adaptive sampling, as here we observe a similar convergence of the mean value, but a lower standard deviation. This advantage is further confirmed by plot (b), which shows the average variance in the estimate for all (uninstantiated) nodes. Backward sampling did not succeed on this model: even for large sample sizes no samples with nonzero probability were produced.



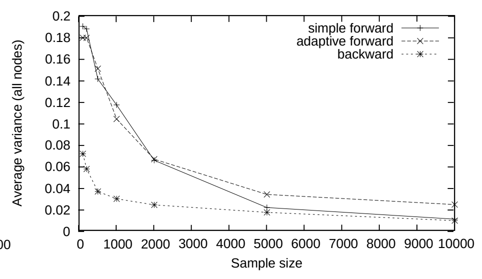
(a)



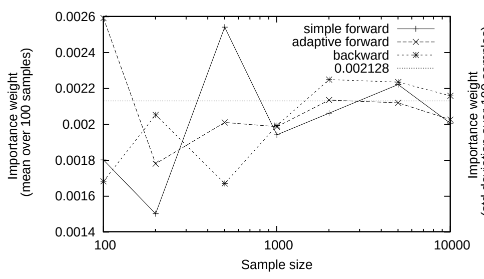
(b)



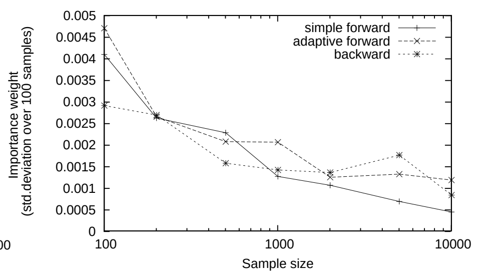
(c)



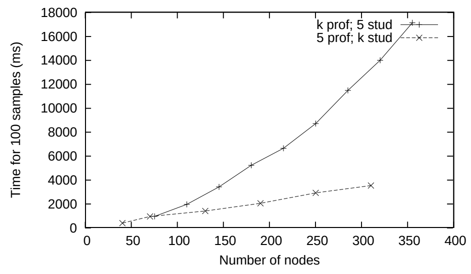
(d)



(e)



(f)



(g)

Fig. 6. Experimental results

In the second experiment, the RBN for the friends and smokers model from example 63 was instantiated over a domain domain of five individuals, leading to a model instance with 195 nodes. Evidence was entered for five nodes. We are approximating the posterior probability of *smokes(mary)*. Again, this model instance is still amenable to exact inference, and the true value of the target probability is computed as 0.953. Plot (c) in figure 6 shows mean values and standard deviations obtained for 100 samples of different. The results indicate that estimates are biased for small sample sizes, but with increasing sample size the bias becomes negligible. In this experiment standard and adaptive sampling show almost identical performance (also when the average variance for all nodes is considered – not shown here). Because of the special 2-layer structure of the network in this experiment, backward sampling was not used.

In the third experiment, the linkage model was instantiated with a pedigree of 31 individuals, and genotype information was entered for 5 individuals in the lower part of the pedigree. Plot (d) shows the average variance in the estimates for the posterior probabilities of the 226 un-instantiated nodes. In this model backward sampling produces much better results for small sample sizes than either version of forward sampling. The advantage is less pronounced when the higher time complexity of backward sampling is taken into account, but also when plotting time against average variance (not shown here) do we observe an advantage of backward sampling for smaller sample sizes. The good performance of backward sampling, at first, is quite surprising in view of its failure in the experiment with the agent communication model. The two models appear to be rather similar in structure, basically describing how information (genotypes, respectively messages) propagates through a network. The crucial difference appears to be that the linkage model permits more local propagations of implied instantiations: when it is known, for example, that a person has genotype AA, then this implies that neither of its parents has genotype aa, and this propagation is possible based on the local information at the affected nodes. In contrast, in the agent communication example, the entered evidence *hasmessage*$(6, t_3) =$ *true* implies that *hasmessage*$(5, t_2) =$ *true*, but this implication cannot be detected by considering only the local neighborhood of the instantiated node *hasmessage*$(6, t_3)$. One has to consider the whole structure of the communication network, from which it follows that the only way agent 6 could have received the message at t_3 is via agent 5.

The results discussed so far relate to the computation of posterior marginals for un-instantiated nodes. Another relevant quantity we compute by sampling is the mean importance weight, which is an estimate for the probability of the entered evidence. The marginal probability of the evidence in the experiment with the linkage model is 0.002128. Plots (e) and (f) show the mean and standard deviation of the estimate for this quantity in the same experiment as reported in plot (d). Here no advantage of backward sampling is observed. One, thus, sees that the choice of the best sampling strategy depends on whether a simultaneous estimate of all posterior marginals is required, or only the probability of the evidence is needed.

The last experiment uses the *students_and_professors* model to investigate the time complexity of sampling. Model instances are given by domains containing certain numbers of students and professors. In our experiment we are measuring the time for drawing a sample of size 100. This is done for two different types of domains: in the first type, the number of professors is held constant at 5, and the number of students increased. In the second type the number of students is fixed at 5, and the number of professor is varied. Note that in this model the number of ingoing edges into nodes, and, correspondingly, the size of pre-evaluated probability formulas, grows only in the number of professors. Plot (g) shows the time required for standard forward sampling as a function of the total number of nodes in the sampling network. For domains with a constant number of professors sampling time is linear in the number of nodes. For domains with a growing number of professors sampling time is quadratic, which corresponds to the total size of the model instance with sizes of pre-evaluated probability formulas taken into account. Neither in this nor in any other experiments did standard and adaptive sampling have a noticeably different time complexity.

Overall, our results indicate that adaptive sampling is a good default strategy. It tends to perform somewhat better than simple forward sampling, with no significantly higher time complexity. Backward sampling is a rather brittle method that can perform well in special situations, but also can fail completely.

6.3 Conclusion

We have developed importance sampling methods for RBNs. Main advantage of our methods is that they operate directly on the abstract representation language, and that they are directly applicable to every model given by an RBN representation. While, surely, there exist models and evidence scenarios for which our importance sampling methods will perform poorly, we have found that they perform well on a variety of models, which were not originally proposed for RBN representations or RBN-based sampling. As an additional result it was shown how to translate MLNs into RBNs.

References

1. J. S. Breese, R. P. Goldman, and M. P. Wellman. Introduction to the special section on knowledge-based construction of probabilistic decision models. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11), 1994.
2. M. Chavira, A. Darwiche, and M. Jaeger. Compiling relational Bayesian networks for exact inference. In P. Lucas, editor, *Proceedings of the Second European Workshop on Probabilistic Graphical Models (PGM'04)*, pages 49–56, 2004.
3. J. Cheng and M. Druzdzel. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *J. of Artifi cial Intelligence Research*, 13, 2000.
4. James Cussens. Stochastic logic programs: Sampling, inference and applications. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artifi cial Intelligence (UAI–00)*, pages 115–122, San Francisco, CA, 2000. Morgan Kaufmann Publishers.
5. Adnan Darwiche. A differential approach to inference in Bayesian networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artifi cial Intelligence (UAI–2000)*, 2000.
6. M. Fishelson and D. Geiger. Exact genetic linkage computations for general pedigrees. *Bioinformatics*, 18(Suppl. 1):S189–S198, 2002.
7. N. Friedman, Lise Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artifi cial Intelligence (IJCAI-99)*, 1999.

8. R. Fung and K-C Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In M. Henrion, R.D. Schachter, L.N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artifi cial Intelligence 5*, pages 209–219, 1990.
9. R. Fung and B. del Favero. Backward simulation in Bayesian networks. In *Proc. of UAI-94*, pages 227–234, 1994.
10. M. Jaeger. Relational bayesian networks. In Dan Geiger and Prakash Pundalik Shenoy, editors, *Proceedings of the 13th Conference of Uncertainty in Artifi cial Intelligence (UAI-13)*, pages 266–273, Providence, USA, 1997. Morgan Kaufmann.
11. M. Jaeger. On the complexity of inference about probabilistic relational models. *Artifi cial Intelligence*, 117:297–308, 2000.
12. M. Jaeger. Complex probabilistic modeling with recursive relational Bayesian networks. *Annals of Mathematics and Artifi cial Intelligence*, 32:179–220, 2001.
13. K. Kersting and L. de Raedt. Towards combining inductive logic programming and bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP-2001)*, Springer Lecture Notes in AI 2157, 2001.
14. S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
15. L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.
16. H. Pasula and S. Russell. Approximate inference for first-order probabilistic languages. In *Proceedings of IJCAI-01*, pages 741–748, 2001.
17. A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University, 2000.
18. D. Poole. First-order probabilistic inference. In *Proceedings of IJCAI-2003*, 2003.
19. David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artifi cial Intelligence*, 94(1-2):7–56, 1997.
20. M. Richardson and P. Domingos. Markov logic networks. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2004. Conditionally accepted for publication in *Machine Learning*. http://www.cs.washington.edu/-homes/pedrod/mln.pdf.
21. T. Sato. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 715–729, 1995.