JMLR: Workshop and Conference Proceedings vol 52, 333-344, 2016 PGM 2016

The Effect of Combination Functions on the Complexity of Relational Bayesian Networks

Denis Deratani Mauá DDM@IME.USP.BR
Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil

Fabio Gagliardi Cozman FGCOZMAN@USP.BR
Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil

Abstract

We study the complexity of marginal inference with Relational Bayesian Networks as parameterized by their probability formulas. We show that without combination functions, inference is #P-equivalent, displaying the same complexity as standard Bayesian networks (this is so even when relations have unbounded arity and when the domain is succinctly specified in binary notation). By allowing increasingly more expressive probability formulas using only maximization as combination, we obtain inferential complexity that ranges from #P-equivalent to FPSPACE-complete to EXP-hard. In fact, by suitable restrictions to the number of nestings of combination functions, we obtain complexity classes in all levels of the counting hierarchy. Finally, we investigate the use of arbitrary combination functions and obtain that inference is FEXP-complete even under a seemingly strong restriction.

Keywords: Relational Bayesian networks; complexity theory; probabilistic inference.

1. Introduction

Bayesian networks provide a compact and intuitive probabilistic description of a concrete domain (Koller and Friedman, 2009). Jaeger's Relational Bayesian Networks, here referred to as RBNs, extend Bayesian networks to allow the modeling of relational, context-specific, deterministic and temporal knowledge (Jaeger, 1997, 2001). These networks are based on a small number of constructs: probability formulas, combination functions, and equality constraints. There are other languages that extend Bayesian networks into relational representations (Poole, 1993; Koller and Pfeffer, 1997, 1998; Getoor and Taskar, 2007; de Raedt, 2008; de Raedt et al., 2008); RBNs offer a particularly general and solid formalism.

In this paper, we examine the complexity of computing marginal inferences with RBNs. We first argue that, without combination functions, RBNs simply offer a language for *plate models* (Gilks et al., 1993; Lunn et al., 2012). Hence, recent results on plate models extend to RBNs (Cozman and Mauá, 2015b); in essence, marginal inference without combination functions is #P-equivalent, and thus matches the complexity of standard (propositional) Bayesian networks.[1] When we allow combination functions and the associated equality constraints into the language, matters complicate considerably. Without additional assumptions, inference is #EXP-equivalent even when the only combination function is maximization. We argue that most of this complexity originates from the use of relations of arbitrary arity. We show that when the arity of relations is bounded, and maximization is the only combination function used, inference is FPSPACE-complete. We prove that by restricting the number of nestings of combination functions, we obtain complexity classes in

1. Technical reasons discussed in Section 3 do not allow us to establish #P-completeness of inference.

all levels of the counting hierarchy, under the previous assumptions. To conclude, we look at the complexity of RBNs with arbitrary combination functions given as part of the input. The challenge here is to constrain language so as to obtain non-trivial complexity results. We show that requiring polynomial combination functions is too weak a condition in that it leads to FEXP-complete inference. On the other hand, requiring polynomial probability formulas brings inference down to #P-equivalence.

The paper begins with a brief review of RBNs (Section 2), and key concepts from complexity theory (Section 3). Our contributions regarding inferential complexity appear in Section 4. A summary of our contributions and open questions are presented in Section 5.

2. Relational Bayesian Networks

A Bayesian network is a compact description of a probabilistic model over a propositional language (Koller and Friedman, 2009; Cozman and Mauá, 2015a). It consists of two parts: an acyclic directed graph $G = (V, A)$ over a finite set of (categorical) random variables $X_1, \ldots, X_n$, and a set of conditional probability tables $\{\mathbb{P}(X|\mathrm{pa}(X)\}_{X \in V}$, where $\mathrm{pa}(X)$ denotes the parents of X in G. The semantics of the model is obtained by the directed Markov property, which states that every variable is conditionally independent of its non-descendants given its parents. This induces a single joint probability distribution by

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | \mathrm{pa}(X_i) = \pi_i)\,,$$

where π_i is the vector of values for $\mathrm{pa}(X_i)$ induced by assignments $\{X_1 = x_1, \ldots, X_n = x_n\}$. Bayesian networks can represent complex propositional domains, but lack the ability to represent relational knowledge.

We use standard relational (function-, constant- and quantifier-free) predicate logic with equality to describe relational knowledge (Enderton, 1972). Let $\mathcal{R}$ and $\mathcal{X}$ be disjoint sets of relation symbols and logical variables, respectively. We denote logical variables by capital letters (e.g., X, Y, Z), predicates by lowercase Latin letters (e.g., r, s, t), and formulas by Greek letters (e.g., α, β). Each relation symbol r is associated with a nonnegative integer $|r|$ describing its *arity*. An *atom* has the form $r(X_1, \ldots, X_{|r|})$, where $r \in \mathcal{R}$, and each $X_i \in \mathcal{X}$. An (atomic) equality is a expression $X = Y$, where $X, Y \in \mathcal{X}$. An *equality constraint* is any well-formed expression containing disjunctions, conjunctions and negations of equalities. For example, $((X = Y) \wedge \neg(X = Z)) \vee (Y = Z)$. A *probability formula* is either a rational number $q \in [0, 1]$; or an atom $r(X_1, \ldots, X_{|r|})$; or a convex combination $F_1 \cdot F_2 + (1 - F_1) \cdot F_3$, where each F_i is a probability formula; or a *combination expression*.

A combination expression has the form $\mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$, where comb is a word from a fixed vocabulary of combination functions, $F_1, \ldots, F_k$ are probability formulas, $Y_1, \ldots, Y_m$ is a (possibly empty) list of logical variables, and α is an equality constraint containing only these variables and the variables appearing in the subformulas. The variables $Y_1, \ldots, Y_m$ in a combination expression are said to be *bound* by that expression; there might be no variables bound by a particular combination expression, in which case we write $\mathsf{comb}\{F_1, \ldots, F_k | \emptyset; \alpha\}$. A logical variable is *free* if it appears in a relation. A variable can be both free and bound in a formula. In this case, we take *free* to mean its condition in the outermost expression. An example of a probability

formula is

$$\mathsf{mean}\{0.6 \cdot r(X) + 0.7 \cdot \mathsf{max}\{1 - s(X,Y)|X; X = X\}|Y, Z; Y \neq X \wedge Z \neq X\}\,.$$

In this formula X is free, and Y and Z are bound (even though X is bound by the inner combination expression involving max). We often write $F(X_1, \ldots, X_n)$ to denote that $X_1, \ldots, X_n$ are the free variables in F.

An RBN is an acyclic directed graph where each node is a relation symbol annotated with a probability formula. The probability formula F_r associated with a relation symbol r contains exactly $|r|$ free variables and mentions only the relation symbols $s \in \mathrm{pa}(r)$ that are parents of r in the graph.

To define the semantics of RBNs we need to introduce a few additional concepts.

An *interpretation* is a pair $(\mathcal{D}, \mu)$ such that $\mathcal{D}$ is a set of constants called the *domain*, and μ maps each relation symbol $r \in \mathcal{R}$ into a relation $r^\mu \subseteq \mathcal{D}^{|r|}$, and each equality constraint into its standard meaning. We assume here that domains are always finite.

A *combination function* is any function that maps a multiset of finitely many numbers in $[0, 1]$ into a single rational number in $[0, 1]$. In this paper we focus on the combination function max encoding maximization:

> $\mathsf{max}\{q_1, \ldots, q_k\} = q_i$, where i is the smallest integer such that $q_i \geq q_j$ for $j = 1, \ldots, k$, and $\mathsf{max}\{\} = 0$.

Other examples are: minimization, defined as $\mathsf{min}\{q_1, \ldots, q_k\} = 1 - \mathsf{max}\{1 - q_1, \ldots, 1 - q_k\}$, and arithmetic mean, defined as $\mathsf{mean}\{q_1, \ldots, q_k\} = \sum_{i=1}^{k} q_i/k$. Yet another example is Noisy-or: $\mathsf{noisy\text{-}or}\{q_1, \ldots, q_k\} = 1 - \prod_{i=1}^{k}(1 - q_i)$, and $\mathsf{noisy\text{-}or}\{\} = 0$.

An interpretation maps a probability formula F with n free variables into a function from $\mathcal{D}^n$ to $[0, 1]$ as follows. If $F = q$, then F is the constant function q. If $F = r(X_1, \ldots, X_n)$, then $F(a) = 1$ if $a \in r^\mu$ and $F(a) = 0$ otherwise. If $F = F_1 \cdot F_2 + (1 - F_1) \cdot F_3$ then $F(a) = F_1(a)F_2(a) + (1 - F_1(a))F_3(a)$. Finally, if $F = \mathsf{comb}\{F_1, \ldots, F_k|Y_1, \ldots, Y_m; \alpha\}$, then $F(a) = \mathsf{comb}(\mathcal{Q})$, where this latter comb is the corresponding combination function and $\mathcal{Q}$ is the multiset containing a number $F_i(a, b)$ for every $(a, b) \in \alpha^\mu$ (even if F_i does not depend on every coordinate). For example, $\mathsf{max}\{1, 2, 3|Y; Y = Y\}$ is interpreted as $\max\{1, 2, 3, 1, 2, 3\}$ if $|\mathcal{D}| = 2$. As another example, consider $\mathcal{D} = \{1, 2\}$; then $F(X, Y) = \mathsf{max}\{q_1, q_2|\emptyset; \neg(X = Y)\}$ is interpreted as $F(1, 1) = F(2, 2) = \mathsf{max}\{\} = 0$, and $F(1, 2) = F(2, 1) = \max\{q_1, q_2\}$.

Finally: Given a domain $\mathcal{D}$, an RBN with graph $G = (V, A)$ induces a probability distribution over interpretations $(\mathcal{D}, \mu)$ by

$$\mathbb{P}(\mu|\mathcal{D}) = \prod_{r \in V} \underbrace{\prod_{a \in r^\mu} F_r(a) \prod_{a \notin r^\mu} (1 - F_r(a))}_{\mathbb{P}(r^\mu | \{s^\mu : s \in \mathrm{pa}(r)\}, \mathcal{D})}\,,$$

where occurrences of $s \in \mathrm{pa}(r)$ in probability formula $F_r(a)$ are interpreted according to s^μ.

In essence, an RBN is a template model that for each domain $\mathcal{D}$ generates a Bayesian network where each node r is a multidimensional random variable taking values r^μ in the set of $2^{N^{|r|}}$ possible interpretations r^μ, where $N = |\mathcal{D}|$. Alternatively, we can associate for every relation symbol $r \in \mathcal{R}$ and tuple $a \in \mathcal{D}^{|r|}$ a binary *random variable* $r(a)$ that takes value 1 when $a \in r^\mu$ and 0 otherwise. An RBN also induces a joint distribution $\mathbb{P}(\{r(a) = \alpha_{r(a)}\}) = \mathbb{P}(\mu)$ over the associated random

variables, where $a \in r^{\mu}$ if $\alpha_{r(a)} = 1$ and $a \notin r^{\mu}$ if $\alpha_{r(a)} = 0$. Hence, we can refer to probabilities such as $\mathbb{P}(r(a) = 1 | s(b) = 0, t(a, c) = 1)$.

Our main focus in this paper is the following (unconditional) *marginal inference problem*: Given an RBN with graph (V, A), a domain $\mathcal{D}$ specified as a list of elements, and a variable $r(a)$ (with $r \in V$ and $a \in \mathcal{D}^{|r|}$), compute $\mathbb{P}(r(a) = 1)$. Note that a joint probability such as $\mathbb{P}(r_1(a_1) = \alpha_1, \ldots, r_n(a_n) = \alpha_n)$ can be computed as $\mathbb{P}(r(a) = 1)$ by defining $F_r = [1-]r_1(X_1) \cdot \cdots [1-]r_n(_n)$; and conditional probabilities can be obtained as the division of two such probabilities.

3. Computational Complexity Theory

We assume familiarity with complexity classes NP, PSPACE, EXP and related ones, and with the concept of oracle Turing machines (Papadimitriou, 1994).

The *polynomial hierarchy* PH is the class of decision problems $\bigcup_k \Sigma_k^{\mathrm{P}}$ (over all integers k), where $\Sigma_k^{\mathrm{P}} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Sigma_1^{\mathrm{P}} = \mathrm{NP}$ and $\Sigma_0^{\mathrm{P}} = \mathrm{P}$. The complexity class #P contains the integer-valued functions computed by a counting Turing machine in polynomial time; a counting Turing machine is a standard non-deterministic Turing machine that prints in binary notation, on a separate tape, the number of accepting computations induced by the input (Valiant, 1979). The class $\#\Sigma_k^{\mathrm{P}}$ contains the integer-valued functions computed by a counting Turing machine with access to a Σ_k^{P} oracle. In particular, $\#\Sigma_0^{\mathrm{P}} = \#\mathrm{P}$. The *counting hierarchy* #PH is the union of all $\#\Sigma_k^{\mathrm{P}}$ (over all integers k). Toda and Watanabe (1992) have shown that a *deterministic* Turing machine equipped with a single call to an oracle #P can solve any problem in the counting hierarchy (i.e., $\#\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$). The class $\mathrm{FPSPACE}_{\mathrm{P}}$ contains the integer-valued functions computed by a standard non-deterministic Turing machine within polynomial space, and with a polynomially long output (Ladner, 1989). And $\#\mathrm{P}^{\mathrm{PSPACE}}$ is the class of integer-valued functions computed in polynomial time by a counting Turing machine with a PSPACE oracle. Ladner (1989) proved that $\mathrm{FPSPACE}_{\mathrm{P}} = \#\mathrm{P}^{\mathrm{PSPACE}}$.

The marginal inference problem with which we are concerned involves the computation of rational numbers in $[0, 1]$, and does not precisely fit into the counting hierarchy. To work with rational-valued functions, we resort to *weighted reductions* (Bulatov et al., 2012), which are parsimonious reductions (Karp reductions that preserve the number of accepting paths) scaled by a polynomial-time computable positive rational number. This accounts for the fact that (unconditional) probabilities are "normalized" integers. For $k \geq 0$, we say that a problem X is $\#\Sigma_k^{\mathrm{P}}$-hard if any problem in $\#\Sigma_k^{\mathrm{P}}$ can be reduced to X by a weighted reduction. If a problem is $\#\Sigma_k^{\mathrm{P}}$-hard and can be solved with a polynomial number of pre-processing steps followed by one call to a $\#\Sigma_k^{\mathrm{P}}$ oracle and a multiplication by a rational obtained with polynomial effort, then the problem is said to be $\#\Sigma_k^{\mathrm{P}}$-equivalent (as inspired by the work of de Campos et al. (2013)).

We also use the class #EXP, which contains functions computed by counting Turing machines in exponential time (this is not equal to the homonymous class defined by Valiant (1979)). Hardness and equivalence for this class are defined similarly to #P, with polynomial replaced by exponential. And we use FEXP, the functional variant of EXP.

4. Inferential Complexity of Relational Bayesian Networks

If the domain is a singleton, then the RBN is actually a Bayesian network. And in this case we know that inference is a #P-equivalent problem (Roth, 1996; de Campos et al., 2013).

We assume the domain $\mathcal{D}$ is given as a list describing N elements $c_1, \ldots, c_N$. An alternative would be to assume that the domain is specified by a single integer N, with the understanding that elements are $\{1, 2, \ldots, N\}$. In the latter case one might specify N in binary notation, and then the specification of the domain would contribute with $\mathcal{O}(\log N)$ symbols to the input; hence we would often need exponential effort to produce the output, as the output may need $\mathcal{O}(N)$ bits to be written. To see this, consider an RBN with a single unary relation r such that $F_r = \text{noisy-or}\{1/2|X; X = X\}$. Then $\mathbb{P}(r(1) = 0) = 1 - [1 - \prod_{i=1}^{N} 1/2] = 2^{-N}$. To avoid such a blow-up (that is not in itself related to RBNs), we prefer to assume an explicit specification of the domain.

The choice of the class of combination functions allowed in an RBN can significantly affect the complexity of inference. For instance, if the combination function is an arbitrary Turing machine (with no bounds on time and space) then inference becomes undecidable. So we need to constrain combination functions to make the study of inferential complexity more interesting. We start with the simplest case, where no combination function is available.

4.1 RBNs without combination functions: plate models

Consider the case where a probability formula can be either a rational number, or an atom, or a convex combination of probability formulas. Then every logical variable in a probability formula is free. Thus, in our acyclic directed graph of relations no node can have an arity larger than any of its children.

As the domain is given as an explicit list of elements, we can always "ground" an RBN into a Bayesian network, and run inference there. However, if we use relations of unbounded arity, we may have to generate an exponentially large Bayesian network; hence one might suspect that inferences would require exponential effort. However:

Theorem 1 *Inference in* RBN*s without combination functions is* #P*-equivalent (even if the domain is specified solely by its size in binary notation).*

Proof Hardness follows from the fact that RBNs can represent any Bayesian network. Membership is obtained by adapting results on *plate models* (Cozman and Mauá, 2015b). A plate model is essentially an RBN with no combination function where logical variables are *typed*, and probabilities are specified by *template conditional probability tables* (Koller and Friedman, 2009, Chapter 6.3). A plate in a plate model groups all the relations that share a logical variable of a certain same type. An example of a plate model describing student performance is depicted in Figure 1 (plates are depicted as rectangles). In any plate model, the logical variables in a node are a subset of the logical variables in its children. Now each plate can be associated with a different domain; in the context of RBNs, all logical variables share the same domain. We can encode any RBN without combination function as a plate model by "solving" the probability formula so as to obtain a template conditional probability table. Since there are no combination functions, every RBN can be augmented with auxiliary relations so as to bound the number of relations in each formula (Jaeger, 2001). This augmented network, which induces the same distribution over the original relations and is obtained efficiently, can be reduced in polynomial time to a plate model by converting probability formulas into template conditional probability distributions (this is polynomial since the number of relations are now bounded). And inference in plate models is #P-equivalent (Cozman and Mauá, 2015b, Theorems 1 and 3). ■

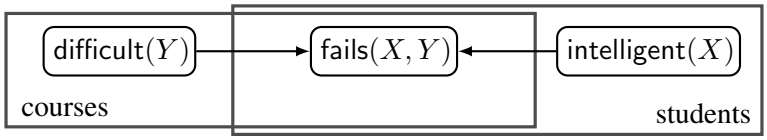

Figure 1: Plate model of student performance.

Theorem 1 can be understood as saying that inference in RBNs reduces to inference in Bayesian networks. Thus, even though such an RBN can represent an exponentially large Bayesian network, only a polynomial number of nodes is relevant for an inference.

4.2 RBNs with max combination functions

We now consider RBNs allowing only max combination functions. First, note that by combining max and convex combination we also allow min combination functions. Second, by using max (resp., min), we can encode existentially (resp., universally) quantified formulas such as $\varphi(X) := \exists Y[r(X,Y) \wedge \neg(X = Y)]$ by $F = \mathsf{max}\{r(X,Y)|Y; \neg(X = Y)\}$ (resp. $\varphi(X) := \forall Y[r(X,Y) \vee \neg(X = Y)]$ by $F = \mathsf{min}\{r(X,Y)|Y; \neg(X = Y)\}$). Thus, RBNs with max can encode *enhanced plates*, that is, plate models where a node can have logical variables that do not appear in some of its children (Cozman and Mauá, 2015b).

Enhanced plate models with binary variables and existential quantification as aggregation function have #EXP-equivalent inferential complexity, even if the domain is specified as an explicit list (Cozman and Mauá, 2015b). We can adapt that result to show that:

Theorem 2 *Inference is #EXP-equivalent when the only combination function is max.*

Proof Recall that an RBN specifies a Bayesian network whose variables take values in a exponentially large set. Membership follows since we can "guess" a configuration for every node and then compute the corresponding probability in exponential time in the input (computing a probability formula built out of max combination function takes at most exponential time). Adding up these probabilities and scaling them is then in #EXP. Hardness can be proved by showing that any enhanced plate model can be reduced to an equivalent RBN with max combination functions; inference in enhanced plate models with probabilities specified by function-free first-order formulas was shown to be #EXP-equivalent by Cozman and Mauá (2015b, Theorem 4). ■

As the previous result shows, unbounded arity of relations can introduce exponential complexity in the presence of combination functions as simple as max. This motivate us to assume that:

Assumption 1 *The arity of relations is bounded.*

We then have that:

Theorem 3 *Inference is* FPSPACE$_P$*-complete, under Assumption 1, when the only combination function is max.*

Proof Membership: we devise a polynomial space algorithm for solving inference. Let k be a bound on the arity of the relations (Assumption 1); then an interpretation assigns a relation consisting of k values to each of the $n = |V|$ relations. This takes polynomial space. We show that probability formulas can be evaluated (for a fixed interpretation) in polynomial space by induction in the number of nested subformulas. If a formula has no combination functions, then only constant space is required to evaluate it; otherwise, assume that every subformula takes polynomial space. Convex combinations of such subformulas again take only polynomial space. So consider a probability formula $F = \mathsf{max}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$, where each F_i is by assumption computed using polynomial space. Evaluating F takes at most $\mathcal{O}(m \cdot k \cdot |\alpha| \cdot f)$ space (required to count the assignments of the logical variables and decide the maximum over $F_1, \ldots, F_k$ for each assignment), where $|\alpha|$ is the size of the constraint α, and f is an upper bound on space required to compute a subformula F_i. Thus, evaluating any probability formula (for a fixed interpretation) takes polynomial space. The probability of an interpretation can be computed in polynomial space by a multiplication of all the terms involved (we only need an accumulator to store intermediate values of the product computation and a counter over the relations $r \in V$ and the tuples $a \in \mathcal{D}^{|r|}$, all of which can be implemented in space $\mathcal{O}(|V| \cdot |r| \lg |\mathcal{D}|)$, which is a polynomial in the input size). The desired marginal probability can be computed in polynomial space by enumerating every possible interpretation and adding up the relevant parts (possibly followed by normalization).

Hardness: By reduction from #QBF, which is FPSPACE$_\text{P}$-complete (Bauland et al., 2010):
Input: A formula $\varphi(X_1, \ldots, X_n) = Q_1 Y_1 Q_2 Y_2 \ldots Q_m Y_m \psi(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$, where each Q_i is either $\exists$ or $\forall$, and ψ is a 3-CNF formula over variables $X_1, \ldots, X_n, Y_1, \ldots, Y_m$.
Output: The number of assignments to the variables $X_1, \ldots, X_n$ that satisfy φ.

Consider such a formula φ with k clauses. We denote by $Y_{i1}, \ldots, Y_{ir}$ the quantified variables appearing in the ith clause ($r \leq 3$ is a function of i). We will show how to solve this instance with a single inference to an RBN constructed in polynomial time (in the size of that instance), followed by a polynomial-time scaling. So, for each counted variable X_j, introduce a zero-arity relation x_j with no parents and with probability formula $F_{x_j} = 1/2$. For each clause, $i = 1, \ldots, k$, introduce a relation $c_i(Y_{i1}, \ldots, Y_{ir}, Z)$, and set the parents of c_i to be the corresponding relations x_j such that X_j appears in the ith clause. The variable Z is used to represent a true value ($Z = 1$). The corresponding probability formula encodes the ith clause: $F_{c_i}(Y_{i1}, \ldots, Y_{ir}, Z) = \mathsf{max}\{\mathsf{max}\{1|\emptyset; \alpha\}, \mathsf{max}\{[1-]x_{i1}, \ldots, [1-]x_{is}|\emptyset; \neg\alpha\}\}$, where we use $1 - x_{ij}$ iff X_{ij} appears negated in the clause. The constraint α is satisfied iff the assignment of $Y_{i1}, \ldots, Y_{ir}$ satisfies the clause. For example, for the clause $(\neg X_1 \vee X_2 \vee Y_2)$ we introduce a relation $c(Y, Z)$ and a corresponding probability formula $F_c(Y_2, Z) = \mathsf{max}\{\mathsf{max}\{1|\emptyset; Y_2 = Z\}, \mathsf{max}\{1-x_1, x_2|\emptyset; \neg(Y_2 = Z)\}\}$. Finally, introduce a relation $w(Z)$ whose probability formula is

$$F_w(Z) = \mathsf{opt}_1\{\cdots \mathsf{opt}_m\{c_1(Y_{11}, \ldots, Y_{1r}, Z) \times \cdots \times c_k(Y_{k1}, \ldots, Y_{kr}, Z)|Y_m\} \cdots |Y_1\}\,,$$

where opt_j is max if $Q_j = \exists$ and opt_j is min if $Q_j = \forall$, and $Y_{i1}, \ldots, Y_{ir}, Z$ is the set of logical variables in atom c_i. Then, the probability of $\{w(1) = 1\}$ equals the number of assignments to $X_1, \ldots, X_n$ for which all clauses are satisfied for every assignment of the quantified variables. That is, $\mathbb{P}(w(1) = 1) = \sum_{x_1,\ldots,x_n} (1/2)^n \mathrm{opt}_{Y_1} \cdots \mathrm{opt}_{Y_m} \prod_{i=1}^k F_{c_i}(Y_{i1}, \ldots, Y_{ir}, 1)$, where the first sum is over the assignments $x_i \in \{0, 1\}$, and each F_{c_i} is evaluate under those assignments. To answer the #QBF problem, return $2^n \mathbb{P}(w(1) = 1)$. ■

We define the nesting level of a probability formula as follows:

Definition 4 *A probability formula $F = q$ or $F = r(X_1, \ldots, X_n)$ has nesting level zero. A probability formula $F = F_1F_2 + (1 - F_2)F_3$ has nesting level equal to the highest nesting level of F_1, F_2, F_3. Finally, a probability formula $F = \mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_M; \alpha\}$ has nesting level equal to the highest nesting level over all probability formulas F_i in it, plus one.*

Nesting combination functions can provide additional computational power, but an unbounded number of nestings does not seem necessary for modeling realistic domains. Thus, one might assume that:

Assumption 2 *The nesting level of any probability formula is bounded by a constant $k \geq 0$.*

Limiting the nesting level brings inference to the counting hierarchy:

Theorem 5 *Inference is $\#\Sigma_k^{\mathrm{P}}$-equivalent, under Assumptions 1 and 2, when the only combination function is* max.

Proof Membership: we can solve inference by a non-deterministic Turing machine that "guesses" an interpretation μ and performs binary search to find the value of each $F_r(a)$ with polynomially many calls to oracles Σ_k^{P}. To see this, consider the computation of $F(a)$ where F is a formula with nesting level 1, that is, F is a polynomial on the value of combination expressions (of nesting level 0). Each combination expression can be evaluated (at a) by a binary search that calls an NP oracle to decide whether the maximum value exceeds a certain threshold. By induction hypothesis, we can use the same argument to solve a formula $F_r(a)$ of nesting level $\ell < k$; this is a polynomial on combination expressions of nesting level $< \ell$, each being solved with polynomially many calls to a $\Sigma_{\ell-1}^{\mathrm{P}}$ oracle. Finally, the probability $\mathbb{P}(\mu)$ is computed polynomially as the product of $F_r(a)$ or $1 - F_r(a)$.

Hardness: By reduction from $\#\Pi_k$SAT, which is complete for $\#\Sigma_k^{\mathrm{P}}$ (Durand et al., 2005):
Input: A formula $\varphi(X_1, \ldots, X_n) = \forall Y_1 \exists Y_2 \ldots Y_k \psi(X_1, \ldots, X_n, Y_1, \ldots, Y_k)$, where each Y_i is a tuple of variables, and ψ is a 3-CNF formula over variables $X_1, \ldots, X_n, Y_1, \ldots, Y_k$.
Output: The number of assignments to the variables $X_1, \ldots, X_n$ that satisfy φ.

The reduction is very similar to the one used to prove Theorem 3, except that we have to be careful not to include unnecessary nestings of combination formulas. So consider a formula φ with m clauses. For each variable X_j, introduce a zero-arity relation x_j with no parents and with probability formula $F_{x_j} = 1/2$. As before, we denote by $Y_{i1}, \ldots, Y_{ir}$ the quantified variables in the ith clause. For each clause, $i = 1, \ldots, m$, introduce relations $c_i(Y_{i1}, \ldots, Y_{ir}, Z)$, $d_i(Y_{i1}, \ldots, Y_{ir})$ and $e_i(Y_{i1}, \ldots, Y_{ir}, Z)$. The parents of c_i are d_i and e_i; d_i has no parents; the parents of e_i are the relations x_j corresponding to counted variables in the ith clause. Each relation d_i encodes whether the assignment of $Y_{i1}, \ldots, Y_{ir}$ satisfies the respective clause, while the relation e_i encodes whether the clause is satisfied by some assignment of the counted variables (and setting the quantified variables so that their values do not affect satisfiability). Let $X_{i1}, \ldots, X_{is}$ be the counted variables in the ith clause. Specify the corresponding probability formulas: $F_{d_i}(Y_{i1}, \ldots, Y_{ir}, Z) = \mathsf{max}\{1|\alpha\}$, $F_{e_i}(Y_{i1}, \ldots, Y_{ir}, Z) = \{[1-]x_{i1}, \ldots, [1-]x_{is}\}$ and $F_{c_i}(Y_{i1}, \ldots, Y_{ir}, Z) = \mathsf{max}\{d_i(Y_{i1}, \ldots, Y_{ir}, Z), e_i(Y_{i1}, \ldots, Y_{ir}, Z)|\emptyset\}$, where we use $1 - x_{ij}$ iff X_{ij} appears negated in the clause. The constraint α is satisfied iff the assignment of $Y_{i1}, \ldots, Y_{ir}$ satisfies the clause. For example, for the clause $(\neg X_1 \vee X_2 \vee Y_2)$ we introduce relations $c(Y, Z)$, $d(Y, Z)$, $e(Y, Z)$, and specify $F_d(Y, Z) = \max\{1|\emptyset; Y = Z\}$, $F_e(Y, Z) = \max\{1 - x_1, x_2|\emptyset; \neg(Y = Z)\}$, $F_c(Y, Z) = \max\{d(Y, Z), e(Y, Z)\}$. Finally, introduce a relation $w(Z)$ whose probability formula is $F_w(Z) = \mathsf{max}_1\{\cdots \mathsf{opt}_k\{c_1(Y_{11}, \ldots, Y_{1r}, Z) \times$

$\cdots \times c_m(Y_{m1}, \ldots, Y_{mr}, Z)|Y_k\} \cdots |Y_1\}$, where opt_j is max if j is odd and opt_j is min otherwise, and Y_j is the tuple variables referred to by the jth quantifier in the input. To solve $\#\Pi_k$SAT return $2^n\mathbb{P}(w(1) = 1)$. ■

Note that for nesting level $k = 0$, the proof above reduces a #SAT problem (complete for #P) and provides an alternative proof that inference is #P-equivalent in RBNs without combination function. For a nesting level $k = 1$, Theorem 5 shows that inference is $\#\Sigma_1$-equivalent (that is, #NP-equivalent), which suggests that the use of combination functions (even as simple as max) adds computational power to the model.

4.3 RBNs with polynomial combinations functions

We now consider RBNs with arbitrary combination functions given as part of the input. First, note that one might take an arbitrarily complex combination function to begin with, and then the complexity of inference would be entirely washed over by the complexity of evaluating combination functions (as already noted in Section 4). To prevent such situations, we might adopt:

Assumption 3 *Any combination function $\mathsf{comb}\{q_1, \ldots, q_k\}$ is polynomial-time computable in the size of a reasonable encoding of its arguments $q_1, \ldots, q_k$.*

The max combination function clearly satisfies this assumption. Thus, given the previous results, inference under Assumption 3 can still produce problems with complexity ranging from #P to FPSPACE_p, depending on the maximum nesting level allowed. This assumption, however, is not strong enough to prevent exponential behavior *even* under Assumptions 1 and 2, solely due to idiosyncrasies of combination functions. To see this, consider the probability formula

$$F(X_1, \ldots, X_n, Z) = \mathsf{comb}\{G(X_1, Z) + 2^{-1}G(X_2, Z) + \cdots + 2^{-n}G(X_n, Z)|Y_1, \ldots, Y_n\}\,, \quad (1)$$

where $G(X, Z) = \mathsf{max}\{1|\emptyset; X = Z\}$. To evaluate such a function, one may face an exponentially large multiset $\{q_1, \ldots, q_{2^n}\}$; if it is indeed necessary to go through all these elements, then the overall effort is exponential even if the function itself is polynomial on the size of the multiset. We have that:

Theorem 6 *Inference is* FEXP*-complete under Assumptions 1 and 3.*

Proof Membership: There are polynomially many groundings for all relations in the RBN. Thus there are exponentially many truth assignments for these groundings; go over each one of them, computing (and adding) the probabilities produced by evaluating polynomially many combination functions (each evaluation with effort at most exponential).

Hardness: Consider a combination function $F(X_1, \ldots, X_n, Z)$ as in Expression (1). Select some FEXP-complete problem, and assume that comb encodes this problem (that is, it gets a string $x_1x_2\cdots x_n$ as input, and runs an unavoidably exponentially long computation and outputs an integer z). Note that such a combination function is allowed by Assumption 3, because comb can select any of the 2^n replicas in its multiset, each encoding a binary number with n bits, and spend exponential time $O(2^{\text{poly}(n)})$, which is polynomial in the encoding of the multiset (that takes $O(2^n \cdot n)$ bits). Now specify the domain as $\mathcal{D} = \{0, 1\}$, and consider the free variables $X_1, \ldots, X_n$ of this formula

as specifying the input to the FEXP-complete problem encoded by comb; the variable Z specifies a bit set. Computing $F(x_1, \ldots, x_n, 1)$ then solves the FEXP-complete problem, and requires only polynomial-sized input and output. ■

An alternative to Assumption 3 would be to adopt:

Assumption 4 *Any probability formula* $\mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$ *is polynomial-time computable in the size of a reasonable encoding of its description.*

One way to satisfy this assumption is to adopt Assumption 1 and in addition to assume that the number of bound logical variables is bounded; the latter is essentially the same as assuming a bound on the *quantifier depth* as defined by Jaeger (2001). Indeed, if the bound on the number of bound logical variables is say M, then there are at most $|\mathcal{D}|^M$ tuples to test with the equality constraint; for each one of them, recursively obtain the values to be used in the combination function, and then compute the latter with polynomial effort.

In any case, if Assumption 4 is imposed, it severely limits the computational power to that of a Bayesian network. Indeed, we obtain:

Theorem 7 *Inference is #*P*-equivalent under Assumptions 1 and 4.*

Proof Hardness: An RBN without combination functions and a domain with a single element suffice to specify any Bayesian network, hence #P-hardness obtains. Membership: Given Assumption 1, there is a polynomial number of groundings; once a truth assignment is guessed nondeterministically for all these groundings, the computation of their joint probability can be done in polynomial time given Assumption 4. Hence by adding up the result of all these truth assignments (in fact, a suitably scaled result), we obtain the desired inference. ■

These results can be interpreted as follows. If we want to use arbitrary combination functions, and we only accept polynomial-time functions, then the problem becomes exponentially hard and fails to reveal many interesting cases. If on the other hand we take arbitrary functions but assume their interpretations to be polynomial, then RBNs are essentially a syntactic sugar for representing large Bayesian networks, with no additional power. It remains an open question to constrain RBNs so that arbitrary combination functions can be used, while still leading to interesting complexity results.

5. Conclusion

We have examined the complexity of inference in relational Bayesian networks. We argued that a great deal of complexity is added by the combination functions alone, which led us to analyze the complexity in terms of the type of combination functions allowed. We first considered networks with no combination function, and showed that (from a complexity viewpoint) they are essentially as powerful as Bayesian networks. We then analyzed the complexity when we only allow maximization as combination function. In this case, complexity results were much more interesting, covering the entire counting hierarchy up to FPSPACE_P and to #EXP. The distinction of complexity classes was produced by limiting the arity of relations and the nestings of combination functions. We then

discussed the use of arbitrary combination functions. We showed that enforcing polynomial combination functions can still lead to exponential inference problems. On the other hand, constraining probability formulas to be polynomial in their encoding brings inference to the same complexity as Bayesian networks, and simplifies the discussion. We left as future work the development of criteria that allow arbitrary combination functions and yet enables interesting complexity results.

Another interesting avenue to pursue is the case of combination functions that can count. This is the case, for example, of the arithmetic mean function. Such functions can in principle simulate a counting Turing machine, so we expect inferential complexity to require a #P oracle. This seems to be significantly different from the results we obtained.

Yet another possible direction for the future is to investigate the complexity with respect to graphical features of the model. For instance, inference in bounded-treewidth Bayesian networks is polynomial (Koller and Friedman, 2009). Our results suggest that even when the grounding of the relational network produces a polynomial-sized graph, inference can still remain intractable.

Following Jaeger (2000) and Beame et al. (2015), one could entertain other dimensions of complexity: for instance, one could analyze inference with respect to the *query complexity* (when the relational model is fixed and query is a joint assignment to random variables that varies), or with respect to the *domain complexity* (when model and query are fixed and the domain size varies).

Acknowledgments

The first author received financial support from the São Paulo Research Foundation (FAPESP) grant #2016/01055-1. The second author is partially supported by the CNPq grant # 308433/2014-9 (PQ).

References

M. Bauland, E. Böhler, N. Creignou, S. Reith, H. Schnoor, and H. Vollmer. The complexity of problems for quantified constraints. *Theory Computing Systems*, 47:454–490, 2010.

P. Beame, G. Van den Broeck, E. Gribkoff, and D. Suciu. Symmetric weighted first-order model counting. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 313–328, 2015.

A. Bulatov, M. Dyer, L. Goldberg, M. Jalsenius, M. Jerrum, and D. Richerby. The complexity of weighted and unweighted #CSP. *Journal of Computer and System Sciences*, 78:681–688, 2012.

F. Cozman and D. Mauá. Bayesian networks specified using propositional and relational constructs: Combined, data, and domain complexity. In *Proc. of the AAAI Conf. on AI*, 2015a.

F. Cozman and D. Mauá. The complexity of plate probabilistic models. In *Proc. of the Conf. on Scalable Uncertainty Management (SUM)*, pages 36–49, 2015b. LNCS 9310.

C. de Campos, G. Stamoulis, and D. Weyland. A structured view on weighted counting with relations to quantum computation and applications. Technical report, 2013. Electronic Colloquium on Computational Complexity.

L. de Raedt. *Logical and Relational Learning*. Springer-Verlag, 2008.

L. de Raedt, P. Frasconi, K. Kersting, and S. Muggleton. *Probabilistic Inductive Logic Programming: Theory and Applications*. Springer-Verlag, 2008.

A. Durand, M. Hermann, and P. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science*, 340(3):496–513, 2005.

H. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

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

W. Gilks, A. Thomas, and D. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178, 1993.

M. Jaeger. Relational Bayesian networks. In *Proc. of the 13th Conf. of Uncertainty in AI (UAI)*, pages 266–273, 1997.

M. Jaeger. On the complexity of inference about probabilistic relational models. *Artificial Intelligence*, 117(2):297–308, 2000.

M. Jaeger. Complex probabilistic modeling with recursive relational Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1):179–220, 2001.

D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT press, 2009.

D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proc. of the 13th Conf. on Uncertainty in AI (UAI)*, pages 302–313, 1997.

D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. of the 15th National Conf. on AI (AAAI)*, pages 580–587, 1998.

R. Ladner. Polynomial space counting problems. *SIAM Journal of Computing*, 18(6):1087–1097, 1989.

D. Lunn, C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter. *The BUGS Book: A Practical Introduction to Bayesian Analysis*. CRC Press/Chapman and Hall, 2012.

C. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing, 1994.

D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1–2):273–302, 1996.

S. Toda and O. Watanabe. Polynomial-time 1-Turing reductions from #PH to #P. *Theoretical Computer Science*, 100:205–221, 1992.

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