7th International Symposium on Imprecise Probability: Theories and Applications, Innsbruck, Austria, 2011

Comparing Binary and Standard Probability Trees in Credal Networks Inference

Andrés Cano, Manuel Gómez-Olmedo, Andrés R. Masegosa, Serafín Moral
Department of Computer Science and Artificial Intelligence
University of Granada, Spain
{acu,mgomez,andrew,smc}@decsai.ugr.es

Abstract

This paper proposes the use of Binary Probability Trees in the propagation of credal networks. Standard and binary probability trees are suitable data structures for representing potentials because they allow to control the accuracy of inference algorithms by means of a threshold parameter. The choice of this threshold is a trade-off between accuracy and computing time. Binary trees enable the representation of finer-grained independences than probability trees. This leads to more efficient algorithms for credal networks with variables with more than two states. The paper shows experiments comparing binary and standard probability trees in order to demonstrate their performance.

Keywords. Bayesian and Credal networks, Inference algorithms, Imprecise probabilities, Variable elimination, Probability trees

1 Introduction

A Bayesian network (BN) is a probabilistic graphical model where precise assessments for the conditional probability mass functions of the variables in the network given the values of their parents are used. A credal network (CN) is also a graphical structure (a directed acyclic graph (DAG) [13]) which is similar to a BN [17], but now the conditional mass functions belong to convex sets of mass functions (*credal sets*).

There has been an increasing interest in propagation algorithms for CNs in the last years. Different algorithms have been proposed for propagation in CNs using standard probability trees (SPTs) [11, 10, 7]. In this paper we propose to apply binary probability trees (BPTs) [8] to propagate in CNs with the variable elimination (VE) algorihtm.

The remainder of this paper is organized as follows: In Section 2 we introduce Bayesian and credal networks, and the problem of inference on them. Section 3 explains the use of standard and binary probability trees to obtain compact representations of potentials and presents how they can be approximated by pruning them. Section 4 explains how to use the VE algorithm to propagate in CNs using BPTs. Section 5 provides details of the experimental work. Finally, Section 6 gives the conclusions.

2 Inference in Credal Networks

Bayesian and credal networks are based on a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ and a directed acyclic graph (DAG) $\mathcal{G}$, whose nodes are associated with the variables of $\mathbf{X}$. Let us assume that each variable X_i takes values on a finite set of states Ω_{X_i} (the domain of X_i). We shall use x_i to denote one of the values of X_i, $x_i \in \Omega_{X_i}$. If I is a set of indices, we shall write $\mathbf{X}_I$ for the set $\{X_i | i \in I\}$. The Cartesian product $\times_{i \in I} \Omega_{X_i}$ will be denoted by $\Omega_{\mathbf{X}_I}$. The elements of $\Omega_{\mathbf{X}_I}$ are called configurations of $\mathbf{X}_I$ (represented as $\mathbf{x}_I$). We use $|\Omega|$ to denote the cardinality of a set Ω. We denote by $\mathbf{x}_I^{\downarrow \mathbf{X}_J}$ the *projection* of the configuration $\mathbf{x}_I$ to the set of variables $\mathbf{X}_J$, $\mathbf{X}_J \subseteq \mathbf{X}_I$. We denote by Π_i the set of parents of X_i in $\mathcal{G}$ and $\pi_i \in \Omega_{\Pi_i}$ a configuration for the variables in Π_i. $P(X_i)$ is the mass function for X_i and $P(x_i)$ the probability that $X_i = x_i$. $P(X_i|\pi_i)$ denotes the probability mass function for X_i conditional on $\Pi_i = \pi_i$. A mapping from a set $\Omega_{\mathbf{X}_I}$ into $\mathbb{R}_0^+$ will be called a *potential* p for $\mathbf{X}_I$. The process of inference in probabilistic graphical models requires the definition of two operations on potentials: *combination* $p_1 \otimes p_2$ and *marginalization* $p^{\downarrow \mathbf{X}_J}$. If p_1 and p_2 are potentials for $\mathbf{X}_I$ and $\mathbf{X}_J$ respectively then $p_1 \otimes p_2$ is a potential for $\mathbf{X}_{I \cup J}$ that can be obtained by pointwise multiplication. If p is a potential for $\mathbf{X}_I$, and $J \subseteq I$ then $p^{\downarrow \mathbf{X}_J}$ is a potential for $\mathbf{X}_J$ that can be obtained by summing out all the variables not in X_J.

In a BN, each node labelled with a variable X_i has attached a conditional probability distribution

$P(X_i|\Pi_i)$, that defines a conditional mass function $P(X_i|\pi_i)$ for X_i given each $\pi_i \in \Omega_{\Pi_i}$. A BN determines the following joint probability distribution:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i) \quad \forall \mathbf{x} \in \Omega_{\mathbf{X}} \tag{1}$$

where x_i and π_i are the projections of $\mathbf{x}$ to X_i and Π_i respectively. Let be $\mathbf{E} \subset \mathbf{X}$ the set of observed variables and $\mathbf{e} \in \Omega_{\mathbf{E}}$ the instantiated value. Each observation, $X_i = e_i$, can be represented by means of a Dirac function defined as $\delta_{X_i}(x_i; e_i) = 1$ if $e_i = x_i$, $x_i \in \Omega_{X_i}$, and $\delta_{X_i}(x_i; e_i) = 0$ if $e_i \neq x_i$. An algorithm that computes the a posteriori distribution $P(x_q|\mathbf{e})$ for each $x_q \in \Omega_{X_q}$, $X_q \in \mathbf{X} \setminus \mathbf{E}$, ($X_q$ is a queried variable) by making local computations is called a *propagation algorithm*. This distribution verifies:

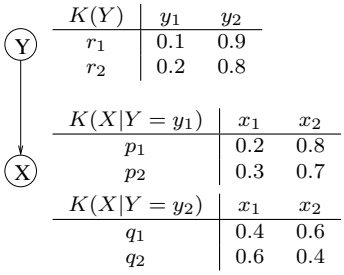
$$P(x_q|\mathbf{e}) \propto \sum_{\mathbf{X}_R} \prod_{X_i \in \mathbf{X}} P(x_i|\pi_i) \prod_{X_i \in \mathbf{E}} \delta_{X_i}(x_i; e_i) \tag{2}$$

where $\mathbf{X}_R = \mathbf{X} \setminus \{\{X_q\}, \mathbf{E}\}$. In fact, the previous formula is the expression for $P(x_q, \mathbf{e})$. $P(x_q|\mathbf{e})$ can be obtained from $P(x_q, \mathbf{e})$ by normalization.

CNs relax the precise probability assessments of BNs. In this work we suppose that the conditional mass functions of a CN are required to belong to a credal set defined as follows. A credal set for a variable X_i is a convex, closed set of probability distributions and shall be denoted by $K(X_i)$. We assume that every credal set has a finite number of extreme points (also called *vertices*), although it may contain an infinite number of mass functions. A credal set can be identified by enumerating its vertices.

An *extensive conditional credal set* [14] about X_i given the set of parent variables Π_i will be a closed, convex set $K(X_i|\Pi_i)$ of mappings $P : X_i \times \Pi_i \longrightarrow [0, 1]$, verifying $\sum_{x_i \in \Omega_{X_i}} P(x_i, \pi_\mathbf{i}) = 1$, $\forall \pi_i \in \Omega_{\Pi_i}$. Again, an extensive conditional credal set can be determined by its set of extreme points which we assume to be finite: $\text{Ext}[K(X_i|\Pi_i)] = \{P_1, \ldots, P_l\}$. In a CN each variable is associated with an extensive conditional credal set $K(X_i|\Pi_i)$. In this paper, we suppose that a *local* credal set $K(X_i|\Pi_i = \pi_i)$ is given for each π_i of Π_i. This is described by Rocha and Cozman [19] as *separately specified credal sets*. For example Fig. 1 shows a CN with two variables (X and Y). Conditional information for X is given by two separately specified credal sets ($K(X|Y = y_1)$ and $K(X|Y = y_2)$). From the separately specified credal sets, we obtain the extensive conditional credal set with:

$$K(X_i|\Pi_i) = \{P | P(x_i, \pi_i) \in K(X_i|\Pi_i = \pi_i), \quad \forall \pi_i \in \Omega_{\Pi_i}\} \tag{3}$$



$K(Y)$	y_1	y_2
r_1	0.1	0.9
r_2	0.2	0.8

$K(X\|Y = y_1)$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$K(X\|Y = y_2)$	x_1	x_2
q_1	0.4	0.6
q_2	0.6	0.4

Figure 1: A simple credal network

Table 1 shows the extensive conditional credal set $K(X|Y)$ obtained from the separately specified credal sets $K(X|Y = y_1)$ and $K(X|Y = y_2)$ of Figure 1.

$K(X\|Y)$	x_1, y_1	x_2, y_1	x_1, y_2	x_2, y_2
p_1, q_1	0.2	0.8	0.4	0.6
p_1, q_2	0.2	0.8	0.6	0.4
p_2, q_1	0.3	0.7	0.4	0.6
p_2, q_2	0.3	0.7	0.6	0.4

Table 1: An extensive conditional credal set

As in BNs, the topology $\mathcal{G}$, of a CN represents independence relations between variables using the d-separation criterion. The meaning of such independences depends on which concept of independence for credal sets is adopted. This paper uses the concept of *strong independence* [13, 12]. The *strong extension* $K(\mathbf{X})$ of a CN is the largest joint credal set such that every variable is strongly independent [13, 12] of its nondescendants nonparents given its parents. It is the joint credal set that contains every possible combination of vertices for all credal sets in the network, where the vertices are combined by multiplication as in Expression 1 [13]. That is, the strong extension $K(\mathbf{X})$ of the CN is the convex hull (CH) of the collection of joint mass functions that can be obtained with every possible combination of the vertices of the separately specified credal sets $K(X_i|\pi_i)$:

$$K(\mathbf{X}) = \text{CH}\{P(\mathbf{X}) : P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i), \quad \forall \mathbf{x} \in \Omega_{\mathbf{X}}, \forall \pi_i \in \Omega_{\Pi_i},\ P(X_i|\pi_i) \in K(X_i|\pi_i)\} \tag{4}$$

A CN can be regarded as a collection of BNs [1] where the topology is given by $\mathcal{G}$. The joint probability of each BN is defined by one of the vertices of $K(\mathbf{X})$. So, the CN defines the following collection of joint probabilities:

$$\mathbf{P}(\mathbf{X}) = \{P_k(\mathbf{X})\}_{k=1}^{n_v} \tag{5}$$

where n_v is the number of vertices in $Ext[K(\mathbf{X})]$.

This paper is dedicated to inference in the strong extension of a CN, in particular, to the computation of

tight bounds for the probability values of a queried variable X_q given a set of observed variables $\mathbf{E}$.

The *combination* of two credal sets is the convex hull of the set obtained by multiplying a mapping of the first credal set with a mapping of the second credal set (repeating the probabilistic combination for all pairs of vertices of the two credal sets). The *marginalization* of a credal set is defined by marginalizing each mapping of the credal set. A more detailed description of these operations can be found for example in [9]. With these operations, we can carry out the same propagation algorithms as in the probabilistic case.

$K(\mathbf{X})$ can also be defined as the multiplication (combination) of all the (extensive) conditional credal sets $K(X_i|\Pi_i)$ in the credal network:

$$K(\mathbf{X}) = \prod_{i=1}^{n} K(X_i|\Pi_i) \tag{6}$$

The computation of the a posteriori credal set $K(X_q|\mathbf{E})$ for a queried variable X_q given some evidence $\mathbf{E}$ can be done in similar way as in Bayesian networks (expression 2) by calculating $K(X_q, \mathbf{E})$.

$$K(X_q, \mathbf{E}) = (K(\mathbf{X}) \prod_{X_i \in \mathbf{E}} \delta_{X_i}(x_i; e_i))^{\downarrow X_q} \tag{7}$$

The vertices in $K(X_q, \mathbf{E})$ are mappings from Ω_{X_q} in $[0, 1]$. $K(X_q|\mathbf{E})$ can be calculated by normalizing the vertices in $K(X_q, \mathbf{E})$. If $Ext[K(X_q, \mathbf{E})] = \{P_k(X_q)\}_{k=1}^{n_v}$ is the set of vertices of $K(X_q, \mathbf{E})$, then the computation of tight bounds for the a posteriori probabilities of X_q given the evidence $\mathbf{E}$ can be done with:

$$\begin{aligned} \underline{P}(x_q|\mathbf{e}) &= \min_{k=1,\ldots,n_v} \frac{P_k(x_q)}{\sum_{x_q} P_k(x_q)} \\ \overline{P}(x_q|\mathbf{e}) &= \max_{k=1,\ldots,n_v} \frac{P_k(x_q)}{\sum_{x_q} P_k(x_q)} \end{aligned} \tag{8}$$

Exact computation in CNs has a high complexity [5], much more than in BNs. It could be done by propagating in the n_v BNs defined by the CN.

3 Standard and Binary Trees

Probability trees [20] and binary probability trees [8] have been used as flexible data structures that enables the specification of *context-specific independences* (see [4]) and provides exact or approximate representations of probability potentials. SPTs and BPTs are usually a more compact representation of potentials than tables, because they allow inference algorithms to take advantage of context-specific independences. In previous works we have defined detailed algorithms [20, 8] for making the basic operations (*combination*, *marginalization* and *restriction*) on potentials, directly over SPTs and BPTs.

3.1 Probability Trees

A *standard probability tree* $\mathcal{T}$ is a directed labelled tree, in which each internal node represents a variable and each leaf represents a non-negative real number. Each internal node has one outgoing arc for each state of the variable that labels that node; each state labels one arc. The *size* of a tree $\mathcal{T}$, denoted by $size(\mathcal{T})$, is defined as its nodes count.

A subtree of $\mathcal{T}$ is a *terminal tree* if it contains only one node labelled with a variable name, and all the children are numbers (leaf nodes).

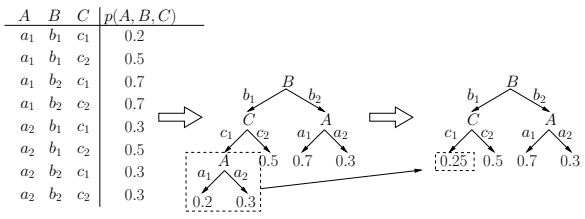


Figure 2: Potential p, its representation as a probability tree and its approximation after pruning

Figure 2 displays a potential p and its representation, using a SPT. This tree shows that the potential is independent of the value of A in the context $\{B = b_1, C = c_2\}$ (the value in the potential is 0.5 for $\{A = a_1, B = b_1, C = c_2\}$ and $\{A = a_2, B = b_1, C = c_2\}$). The tree contains the same information as the table, but only requires five values, while the table contains eight values. Furthermore, SPTs enable even more compact representations. This is achieved by pruning certain leaves, replacing them with the average value, as shown in the second tree shown in Fig. 2. The trade-off is a loss of accuracy.

3.2 Binary Probability Trees

A *binary probability tree* $\mathcal{BT}$ is similar to a SPT. It can also be defined as a directed labelled tree, where each internal node is labelled with a variable, and each leaf is labelled with a non-negative real number. But in this case, each internal node has always two outgoing arcs, and a variable can label several nodes in the path from the root to a leaf node. Another

difference is that, for an internal node labelled with X_i, the outgoing arcs can generally be labelled with more than one state of the domain of X_i, Ω_{X_i}. The size of a BPT (i.e., the number of nodes) is equal to twice the number of leaves minus one.

For example, Fig. 3 (ii) shows a BPT for the table in (i). In the figure, we use a superscript number at each node of the tree, in order to easily identify it. The domain of A, Ω_A, is $\{a_1, a_2, a_3\}$, and the domain of B, Ω_B, is $\{b_1, b_2, b_3\}$. This potential can also be represented with the SPT shown in Fig. 3 (iii). It can be seen that the BPT contains only five leaves, whereas the SPT contains seven. The SBT shown in Fig. 3 (iii) is able to capture a context-specific independence: *the potential does not depend on B when $A = a_1$*. The BPT in Fig. 3 (ii) captures the previous independence, but it is also able to capture other *fine-grained independences*. For example, *the potential does not depend on B when $A = a_2$ and $B \neq b_3$* ($B = b_1$ or $B = b_2$). This independence cannot be represented with the SPT of Fig. 3 (iii).

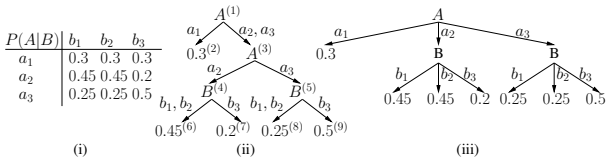


Figure 3: Potential $P(A|B)$ as table, BPT, and SPT

3.3 Constructing standard and binary trees

In [20] and [8] we have proposed a methodology for constructing a SPT $\mathcal{T}$ or a BPT $\mathcal{BT}$ from a potential p. These methods were inspired by the methods for inducing classification trees, such as Quinlan's ID3 algorithm [18], which builds a *decision tree* from a set of examples. But the measure used as the *splitting criterion* in SPTs and BPTs was specifically adapted to probabilities. Here we summarize the procedure for BPTs. For SPTs, a similar procedure is used.

Let p be a potential for a set of variables $\mathbf{X}_I$. It is generally possible to obtain several BPTs for p, depending on the order assigned to the variables of $\mathbf{X}_I$ in the internal nodes of the tree, and the distribution at each internal node of the available variable states over its outgoing arcs.

The process begins with a BPT $\mathcal{BT}_0$ with only one node (a leaf node) labelled with the average of the potential values: $L_t = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} p(\mathbf{x}_I)/|\Omega_{\mathbf{X}_I}|$.

A greedy step is then applied successively until we obtain an exact BPT, or until a given *stop criterion* is satisfied. At each step, a new $\mathcal{BT}_{j+1}$ is obtained from the previous one, $\mathcal{BT}_j$. The greedy step requires the choice of a *splitting criterion*. It consists of expanding one of the leaf nodes t in $\mathcal{BT}_j$ with a terminal tree (with t rooting the terminal tree, and two new nodes t_l and t_r as children of t). Node t will be labelled with one of the *candidate variables*. Suppose $\Omega^t_{X_i}$, $\Omega^t_{X_i} \subseteq \Omega_{X_i}$, is the set of availabe states of X_i at node t. It is also necessary to distribute the set of available states $\Omega^t_{X_i}$ of the chosen candidate variable X_i into two subsets, $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$, to label the two outgoing arcs (left and right) of t. This process is illustrated in Fig. 4, where the terminal node t in tree $\mathcal{BT}_j$ is expanded using variable B. The set of available states of B at node t, $\Omega^t_B = \{b_1, b_2, b_3\}$ was partitioned into the sets $\Omega^{t_l}_B = \{b_1\}$ and $\Omega^{t_r}_B = \{b_2, b_3\}$. After applying this process, we say that the leaf node t has been expanded with variable X_i and the sets of states $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$.

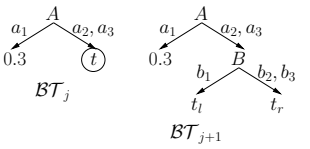


Figure 4: Expansion of the terminal tree t with B

The choice of the splitting criterion requires a distance to measure the goodness of the approximation of a BPT $\mathcal{BT}$ for a given potential p. If we denote by $\overline{\mathcal{BT}}$ and $\overline{p}$ the probability distributions (normalized potentials) proportional to $\mathcal{BT}$ and p, respectively, then the *distance* from a BPT $\mathcal{BT}$ to a potential p is measured using the Kullback-Leibler divergence [16]:

$$D(p, \mathcal{BT}) = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \overline{p}(\mathbf{x}_I) \log \frac{\overline{p}(\mathbf{x}_I)}{\overline{\mathcal{BT}}(\mathbf{x}_I)} \qquad (9)$$

Kullback-Leibler's divergence is always positive or zero. It is equal to zero if $\mathcal{BT}$ provides an exact representation of the potential p. It is a standard divergence used in information theory to measure the difference between two probability distributions. Here we use it to measure differences between potentials that are not really probability distributions (they represents conditional credal sets containing transparent variables), but experiments show that its use is a good heuristic procedure applied when reordering the variables of a tree or when pruning leaf nodes.

In [8] we proposed as *splitting criterion* to choose the partition that maximizes the *information gain* obtained for the current BPT $\mathcal{BT}_j$ after performing the mentioned expansion on leaf node t. For SPTs the information gain is calculated with:

$$I(t, X_i) = D(p, \mathcal{T}_j) - D(p, \mathcal{T}_j(t, X_i)) \quad (10)$$

where $\mathcal{T}_j(t, X_i)$ is the SPT $\mathcal{T}_j$ after expanding node t with the variable X_i.

For BPTs the information gain obtained after expanding node t is calculated with:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = D(p, \mathcal{BT}_j) - D(p, \mathcal{BT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})) \quad (11)$$

where $\mathcal{BT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})$ is $\mathcal{BT}_j$ after expanding node t with variable X_i and a partition of its available states $\Omega_{X_i}^t$ into sets $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$.

It is immediate to see that $I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \geq 0$. By maximizing $I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})$ in the current greedy step, we manage to minimize Kullback-Leibler's distance to potential p in that step.

The information gain (expressions 10 and 11) obtained by expanding node t, can be efficiently calculated in SPTs and BPTs (see Proposition 1 in [20, 8]).

The methodology explained in this section for building a SPT or BPT can also be used to reorder the variables (or the split sets) of a SPT or BPT resulting from an operation of combination or marginalization. This enables us to move the most informative variables to the upper levels of the tree. So, if a pruning operation is applied, only the less informative variables will be removed. The process to reorder a BPT $\mathcal{BT}$ is the same as the one for building a BPT from a potential p. Here, p is the potential that $\mathcal{BT}$ represents. So, we can build a new BPT applying the same procedure explained in this section.

3.4 Pruning standard and binary trees

During the inference process it is possible that some trees have a large size, making it impossible to obtain any result with the available memory of our computer. Pruning of SPTs [20] was proposed as a way to control the size of trees during the propagation process. This operation has also been extended to BPTs [8]. In this way, we can obtain a result from an inference algorithm although it will be approximate. Basically, a *pruning* in a SPT or BPT consists of replacing a terminal tree by the average of values that it represents. For example, if we wish to prune the terminal tree rooted by node (4) in the BPT of Fig. 3 (ii), we must replace it by $(0.45 + 0.45 + 0.2)/3$. In [6] we demonstrated that the pruned tree obtained with the previous procedure is the tree that minimizes the Kullback-Leibler divergence between the exact potential and all the trees with the same structure as that pruned tree.

In [20, 8] it is proposed to repeat the pruning process until the tree contains no terminal tree which *information loss* is under a given threshold Δ. The information loss is also calculated with the difference of the Kullback-Leibler's distances, before and after pruning (expressions 10 and 11). The goal of the pruning of a tree involves detecting leaves that can be replaced by one value without a big increment in Kullback-Leibler's divergence of the potential represented by that tree, before and after pruning.

Again, the information loss can be locally computed at node t in the current SPT or BPT.

4 Propagating credal sets using binary probability trees

The simpler approximate algorithm for propagating credal sets using SPTs is based on the *Variable Elimination* algorithm [11]. VE is one of the most popular algorithms for computing *a posteriori* information in probabilistic graphical models using local computations. It was independently proposed by Shafer and Shenoy [21], Zhang and Poole [22] and Dechter [15]. The input of this algorithm is a set of potentials and a queried variable. It iteratively eliminates variables from the set of potentials by using combination and marginalization until only the queried variable remains in the set of potentials.

In this paper, we propose to use also the VE algorithm to propagate in CNs, but using BPTs (see Algorithm 1) to represent the credal sets $K(X_i|\Pi_i)$. In CNs, all the variables should be removed (by marginalization) except the queried variable and the transparent variables (see bellow for an explanation of transparent variables). Here, the set of potentials is the set $\{K(X_i|\Pi_i)\}$ of extensive conditional credal sets in the CN.

For each X_i, we originally have a collection of m separately specified credal sets $\{K(X_i|\pi_1), \ldots, K(X_i|\pi_m)\}$, where m is the number of configurations of Π_i. The problem is transformed into an equivalent one by using a *transparent variable* T_{π_i} for each configuration of the parents of X_i ($\pi_i \in \Omega_{\Pi_i}$). T_{π_i} will have as many cases as the number of vertices in the separately specified credal set $K(X_i|\pi_i)$. Each vertex of the extensive conditional credal set $K(X_i|\Pi_i)$ can be obtained by fixing each transparent variable T_{π_i} to one of its values. This transformation is equivalent in size to the one proposed by Antonucci et al. in [1], although

that one requires modifications in the graph of the CN.

SBTs and BPTs enable an extensive conditional credal set $K(X_i|\Pi_i)$ to be represented efficiently when it comes from m separately specified credal sets $\{K(X_i|\pi_1), \ldots, K(X_i|\pi_m)\}$ and with a single data structure (the necessary space for the tree is proportional to the sum of the necessary spaces for the m local trees). In Fig. 5, we can see one example where a BPT represents the extensive conditional credal set $K(X|Y)$ associated to the two separately specified credal sets $K(X|Y = y_1)$ and $K(X|Y = y_2)$. In the BPT in Fig. 5, we can obtain the extreme points of $K(X|Y)$ by fixing T_{y_1} and T_{y_2} to each one of its values. For example, if the BPT is restricted to $T_{y_1} = t^1_{y_1}$ and $T_{y_2} = t^2_{y_2}$, we obtain a new BPT that gives us the extreme point of $K(X|Y)$ associated to p_1 and q_2. The tree avoids repetition of probability values, reducing the space necessary with respect to the table representation.

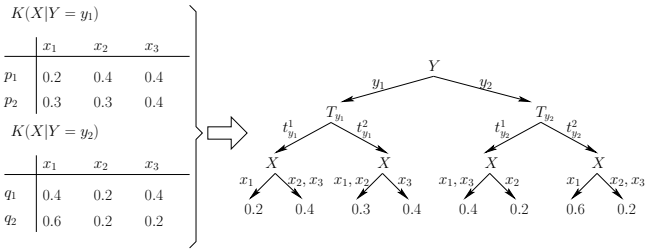


Figure 5: A binary probability tree for $K(X|Y)$

Algorithm 1: Variable Elimination

Input : $\mathbf{K} = \{K(X_i|\pi_i) : i = 1, \ldots, n\}$ the set of separately specified credal sets in the CN; $\mathbf{e}$ the set of observed values, $\mathbf{e} \in \Omega_{\mathbf{E}}$; a variable of interest X_q, $X_q \in \mathbf{X} \setminus \mathbf{E}$; and Δ the threshold for pruning

Output: $\underline{P}(x_q|\mathbf{e})$ and $\overline{P}(x_q|\mathbf{e})$ for each $x_q \in \Omega_{X_q}$, $X_q \in \mathbf{X} \setminus \mathbf{E}$

1 Get the set $S_{\mathcal{BT}}$ of binary trees, building each binary tree $\mathcal{BT}_i$ from the credal sets $K(X_i|\pi_i), \forall \pi_i \in \Omega_{\Pi_i}$ (as in Figure 5)
2 Transform each $\mathcal{BT}_i$ into $\mathcal{BT}_i^{R(\mathbf{e})}$ (restrict to evidence)
3 Reorder variables and split sets in every $\mathcal{BT}_i$
4 Prune each $\mathcal{BT}_i$ with the Δ threshold
5 **foreach** $Y \in \mathbf{X} \setminus (\mathbf{E} \cup \{X_q\})$ **do**
6 Let $\mathbf{S}_Y = \{\mathcal{BT}_i | Y \in s(\mathcal{BT}_i)\}$
7 Calculate $\mathcal{BT}_{prod} = \prod_{\mathcal{BT}_i \in \mathbf{S}_Y} \mathcal{BT}_i$
8 Calculate $\mathcal{BT}_{sum} = \mathcal{BT}_{prod}^{\downarrow s(\mathcal{BT}_{prod}) \setminus Y}$
9 Reorder variables and split sets in $\mathcal{BT}_{sum}$
10 Prune $\mathcal{BT}_{sum}$ using the Δ threshold
11 $S_{\mathcal{BT}} = \{(S_{\mathcal{BT}} \setminus \mathbf{S}_Y\}) \cup \mathcal{BT}_{sum}$
12 Calculate $\mathcal{BT}_q = \prod_{\mathcal{BT}_i \in S_{\mathcal{BT}}} \mathcal{BT}_i$
13 Get $\underline{P}(x_q|\mathbf{e})$ and $\overline{P}(x_q|\mathbf{e})$ by normalizing the vertices in $\mathcal{BT}_q$

In our version of the VE algorithm (Algorithm 1), each conditional credal set $K(X_i|\Pi_i)$ is represented with a BPT as in Fig. 5 from the set of separately specified credal sets $\{K(X_i|\pi_i), \forall \pi_i \in \Omega_{\Pi_i}\}$. This is done in step 1 of the algorithm. The evidence (if available) is incorporated with restriction operations (step 2). Then each tree is reordered (step 3) so that the most informative variables appear in the upper levels of the tree, using the procedure described in Section 3.3. Step 4 consists of pruning the trees using a given Δ threshold in order to reduce their sizes as much as possible. The loop (step 5) deletes a variable (non-transparent) in each iteration. The combination of trees containing the variable to be removed is performed in step 7. This operation is made directly over trees (see [8]). The resulting tree is marginalized to discard the variable to be removed using marginalization (step 8). Again, this operation is made directly over the tree (see [8]). Steps 9 and 10 reorder the variables of the tree (see Section 3.3) and prune it respectively. The pruning operation can select any variable (normal or transparent one) in the tree. Finally, the resulting trees (all of them will be defined only on the queried variable and on transparent variables) are combined to produce a single tree (step 12). Finally the upper and lower bounds for the probability of the queried variable can be obtained by normalizing each one of the vertices in $\mathcal{BT}_q$ (step 13) using expression 8. The pruning reduces the complexity of posterior operations. The more transparent variables are pruned the less vertices appears in the final credal set obtained with the BPT in step 12 of the algorithm. When a $\Delta = 0.0$ threshold is used, no variable will be pruned unless there are context specific independences in the potentials. In the worst case, using $\Delta = 0.0$, the BPT obtained in step 12 corresponds to a credal set with n_v possible vertices.

With respect to the complexity of Algorithm 1, using $\Delta = 0.0$, if the potentials do not contain any context specific independence, no pruning will be done, and so inference is equivalent to make n_v propagations in a BN. This is the worst case. Using values of Δ greater than 0.0 we can reduce the size of potentials and so computing times. A theoretical evaluation of the computational complexity is out of the scope of this paper.

5 Experiments

In order to compare the performance of SPTs and BPTs we have used two classical BNs (*Alarm* [2] and *Insurance* [3]). The number of states for the variables in these networks is maintained as in their original specifications. These networks contain variables with more than two states. For each model, we obtained a CN by randomly generating separately specified conditional credal sets for each variable X_i and each configuration of the parents of X_i. The number of vertices

at each $K(X_i|\pi_i)$ is selected as follows: For a given percentage of the configurations in Ω_{Π_i} we associated a given number of vertices in the credal sets $K(X_i|\pi_i)$. For the rest of configurations we used only one vertex. This allows us to control the potential size of the strong extension of the CN, so that exact inference is not too difficult to be done in our computers, in order to allow the comparison of the error of approximate inference with respect to the exact one. The process to randomly select the probabilities for the vertices at each separately specified credal set $K(X_i|\pi_i)$ is as follows. When only one vertex must be used we take the probability values in the original BN. When several vertices are used we take as basis the probability distribution in the original BN ($P(X_i|\pi_i)$). If a value equal to 0.0 is found for a given configuration of $P(X_i|\pi_i)$, it will be kept for that configuration. If a value equal to v, $v > 0.0$, is found, we select a new uniform random value in the interval $[-v, v]$ (negative values are converted into positive). The resulting vertex is then normalized. This procedure do not produce too much context specific independences in the resulting potentials, but we must take into account that these kind of independences are present in our representation of extensive conditional credal sets by means of trees. For example, in Fig. 5 the potential do not depend on T_{y_2} when $Y = y_1$.

Several experiments have been done using different variables for each network. In some cases we have considered that some of the variables of the network are observed. In Table 2 we show for each experiment (Ex), the chosen variable (Var), the name of the network, the number of observed variables ($|\mathbf{E}|$), the number of vertices per credal set (nvpc), the percentage of configurations (per) of Ω_{Π_i} that will contain nvpc vertices, and the potential size of the strong extension (n_v) of the CN. In the calculus of nv we suppose that the barren nodes for the given query have been removed from the network.

Ex	Var	Network	$\|\mathbf{E}\|$	nvpc	per	n_v
1	Venttube	Alarm	0	3	90	354294
2	Expco2	Alarm	0	3	17	177147
3	RiskAversion	Insurance	0	3	70	177147
4	DrivHist	Insurance	0	3	31.5	177147
5	Venttube	Alarm	6	3	12.25	354294
6	DrivHist	Insurance	9	3	12	944784

Table 2: Experiments we have done

We have measured the maximum required size of SPTs and BPTs during the propagation (biggest tree used in the computations), the *mean square error* for the a posteriori bounds of the queried variable and the running time used by the propagation algorithm. The mean square error for a queried variable X_q is measured using the following expression:

$$\sqrt{\frac{\sum_{x_q \in \Omega_{X_q}} ((\underline{P}^*(x_q|\mathbf{e}) - \underline{P}(x_q|\mathbf{e}))^2 + (\overline{P}^*(x_q|\mathbf{e}) - \overline{P}(x_q|\mathbf{e}))^2)}{2 \cdot |\Omega_{X_q}|}} \tag{12}$$

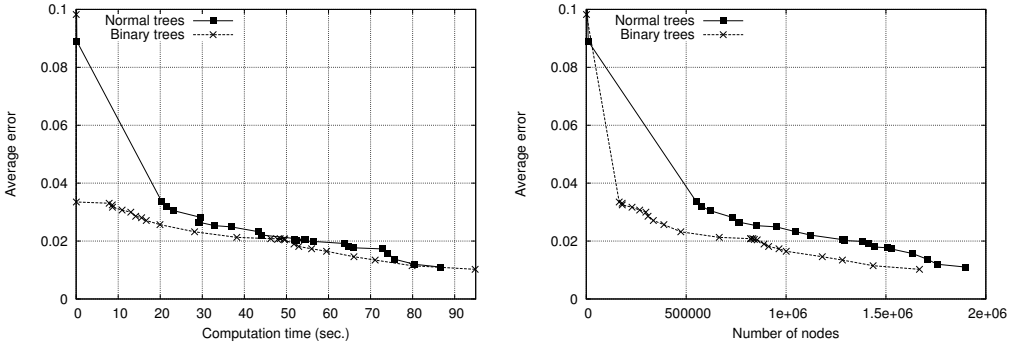
where $\underline{P}^*(x_q|\mathbf{e}, \overline{P}^*(x_q|\mathbf{e})$ are the approximate lower and upper bounds and $\underline{P}(x_q|\mathbf{e}, \overline{P}(x_q|\mathbf{e})$ the exact ones.

These parameters (mean square error, maximum size and time) are measured running the Algorithm 1 with several values for the Δ threshold using SPTs and BPTs. We have used values for Δ in the interval $[10^{-7}, 10^{-2}]$. Each experiment was run ten times. Each time, we began randomly generating the probabilities for each credal set. So, average of mean square error, maximum size and time (in seconds) are calculated and reported in figures 6 to 11 for the different experiments. For each experiment, we show the average mean square error versus largest tree size required in the two versions of the propagation algorithm (using SPTs and BPTs) and the average mean square error versus average time required in the two versions of the propagation algorithm (using SPTs and BPTs).

As expected with both kind of trees, high values of Δ cause large errors but require lower computing time and smaller trees. Small values of Δ give small errors but require a high computing time and large trees.

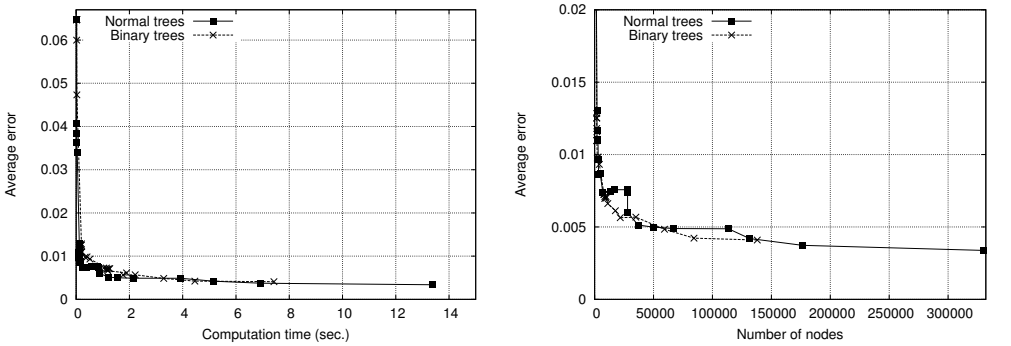
The figures allow to compare propagation with SPTs and BPTs for each experiment. In some cases, we can see a noticeable reduction in the size and required time using BPTs with respect to SBTs: that is, the same level of error can be achieved with BPTs, but with a very important reduction in size and time. This is the case of Experiment 1 for VENTTUBE variable (4 states) in Alarm network (Fig. 6), Experiment 3 for RiskAversion (4 states) in Insurance network (Fig. 8), Experiment 5 for VENTTUBE variable in Alarm network using 6 observed variables (Fig. 10). There are also cases where the performance of SPTs and BPTs is quite similar. For example, see Experiment 2 for EXPCO2 variable (4 states) in Alarm network (Fig. 7) or Experiment 6 for DrivHist in Insurance network using 9 observed variables (Fig. 11).

We have also tried to propagate using tables for representing the extensive conditional credal sets, like the one in Table 1, but our computer run out of memory in all the experiments in about 18 minutes. This is because a table does not allow to capture the context specific independences for transparent variables, and so the size of potentials increases quicker for tables in the propagation process, even if we do not use pruning in trees. We have also compared the maximum tree size and computing time. Obviously computing time increases when bigger trees are used (figures are



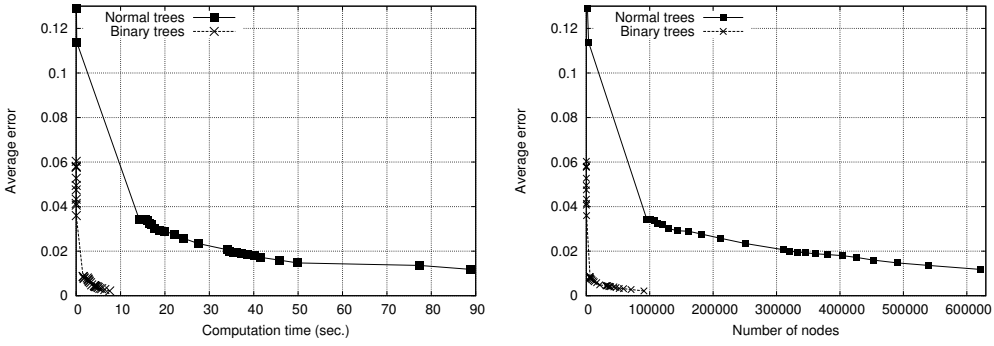
(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 6: Inference for VENTTUBE in Alarm network (no evidence)



(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 7: Inference for EXPCO2 in Alarm network (no evidence)



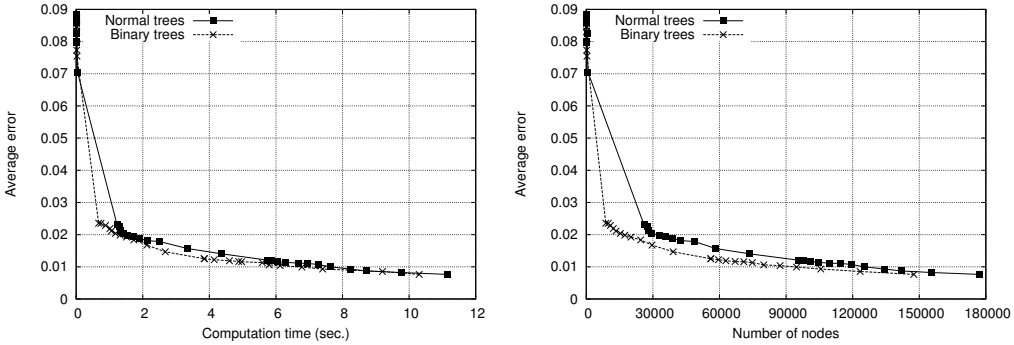
(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 8: Inference for RiskAversion in Insurance network (no evidence)

not includes because of the space).

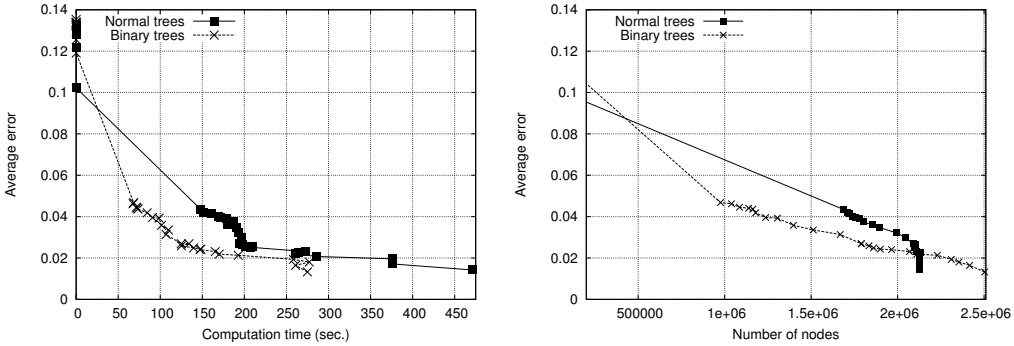
6 Conclusions

In this paper we have proposed the use of BPTs to propagate in CNs. BPTs and SPTs make possible to control the accuracy of the propagation by means of a given threshold Δ used for pruning the trees. The choice of Δ is a trade-off between accuracy and computing time. The experiments show that BPTs offer better performance than SPTs in some cases, and similar one in other cases. So, we think that BPTs is a better representation for the potentials of a CN.



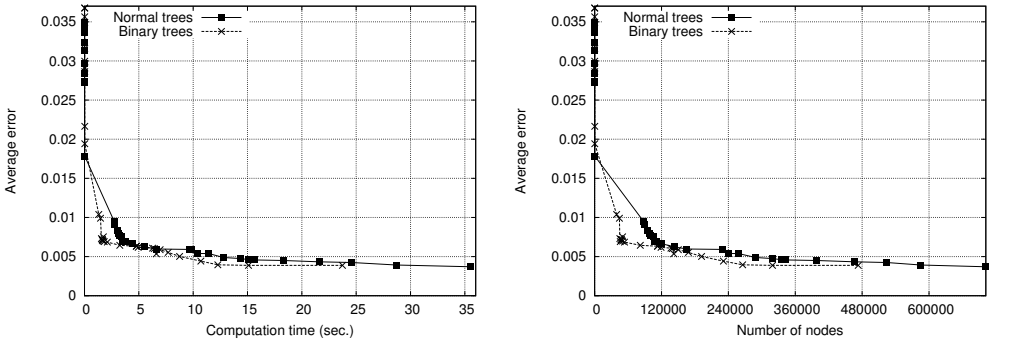
(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 9: Inference for DrivHist in Insurance network (no evidence)



(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 10: Inference for VENTTUBE in Alarm network (evidence in 6 variables)



(a) MSE versus computing time (b) MSE versus maximum tree size

Figure 11: Inference for DrivHist in Insurance network (evidence in 9 variables)

In the future we intend to perform more exhaustive experiments so we can characterize the situations where BPTs will be better than SPTs. In this way we will check the complete list of unobserved variables in these networks and in other classical BNs. We will also analyze the impact of the number of vertices in the conditional credal sets in the performance of BPTs with respect to SPTs.

Acknowledgements

This work has been supported by the Spanish Ministry of Science and Innovation, under projects TIN2007-67418-C03-03, TIN2010-20900-C04-01 and

by the European Regional Development Fund (FEDER). We are also very grateful to the anonymous reviewers for their valuable comments and suggestions.

References

[1] A. Antonucci, Y. Sun, C. P. de Campos, and M. Zaffalon. Generalized loopy 2u: A new algorithm for approximate inference in credal networks. *Int. J. Approx. Reasoning*, 51(5):474–484, 2010.

[2] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Second European Conference on AI and Medicine*, Berlin, 1989. Springer-Verlag.

[3] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29, 1997.

[4] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 115–123, Portland, Oregon, 1996.

[5] C. P. De Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1313–1318, 2005.

[6] A. Cano. *Propagación aproximada de intervalos de probabilidad en grafos de dependencias*. PhD thesis, Universidad de Granada, Dpt. Computer Sciences and Artificial Intelligence, June 1999.

[7] A. Cano, M. Gómez, S. Moral, and J. Abellán. Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks. *International Journal of Approximate Reasoning*, 44:261–280, 2007.

[8] A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52:49–62, 2011.

[9] A. Cano and S. Moral. A review of propagation algorithms for imprecise probabilities. In *Proceedings of the First International Symposium on Imprecise Probabilities and their Applications (ISIPTA'99)*, Ghent, 1999.

[10] A. Cano and S. Moral. Computing probability intervals with simulated annealing and probability trees. *Journal of Applied Non-Classical Logics*, 12(2):151–171, 2002.

[11] A. Cano and S. Moral. Using probabilities trees to compute marginals with imprecise probabilities. *International Journal of Approximate Reasoning*, 29:1–46, 2002.

[12] I. Couso, S. Moral, and P. Walley. Examples of independence for imprecise probabilities. In *Proceedings of the First International Symposium on Imprecise Probabilities and Their Applications (ISIPTA'99)*, 1999.

[13] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

[14] F. G. Cozman. Graphical models for imprecise probabilities. *International Journal of Approximate Reasoning*, 39:167–184, 2005.

[15] R. Dechter. Bucklet elimination: A unifying framework for probabilistic inference. In E. Horvitz and F.V. Jennsen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[16] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86, 1951.

[17] J. Pearl. *Probabilistic Reasoning with Intelligent Systems*. Morgan & Kaufman, San Mateo, 1988.

[18] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–105, 1986.

[19] J.C. F. Rocha and F.G. Cozman. Inference with separately specified sets of probabilities in credal networks. In A. Darwiche and N. Friedman, editors, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2002.

[20] A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413, 2000.

[21] P. P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In Shachter et al., editors, *Uncertainty in Artificial Intelligence, 4*, pages 169–198. North-Holland, 1990.

[22] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *International Journal of Intelligent Research*, 5:301–328, 1996.