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) G) whose nodes are associated with the variables of X. Let us assume that each variable X_i takes values on a finite set of states \Omega_{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 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_{X_I}. The elements of \Omega_{X_I} are called configurations of X_I (represented as x_I). We use |I| to denote the cardinality of a set I. We denote by X_I^{x_I} the projection of the configuration x_I to the set of variables X_J, X_J \subseteq X_I. We denote by \Pi_i the set of parents of X_i in G and \pi_i \in \Omega_{\Pi_i} a configuration for the variables in \Pi_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_{X_I} into \mathbb{R}_+ will be called a potential p for 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^{\Pi \setminus J} = p^{\Pi \setminus J}. If p_1 and p_2 are potentials for X_I and X_J respectively then p_1 \otimes p_2 is a potential for X_{I \cup J} that can be obtained by pointwise multiplication. If p is a potential for X_I, and J \subseteq I then p^{\Pi \setminus J} is a potential for X_J that can be obtained by summing all the variables not in X_J.

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

2 Inference in Credal Networks
Bayesian and credal networks are based on a set of random variables X = \{X_1, \ldots, X_n\} and a directed acyclic graph (DAG) G, whose nodes are associated with the variables of X. Let us assume that each variable X_i takes values on a finite set of states \Omega_{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 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_{X_I}. The elements of \Omega_{X_I} are called configurations of X_I (represented as x_I). We use |I| to denote the cardinality of a set I. We denote by X_I^{x_I} the projection of the configuration x_I to the set of variables X_J, X_J \subseteq X_I. We denote by \Pi_i the set of parents of X_i in G and \pi_i \in \Omega_{\Pi_i} a configuration for the variables in \Pi_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_{X_I} into \mathbb{R}_+ will be called a potential p for 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^{\Pi \setminus J} = p^{\Pi \setminus J}. If p_1 and p_2 are potentials for X_I and X_J respectively then p_1 \otimes p_2 is a potential for X_{I \cup J} that can be obtained by pointwise multiplication. If p is a potential for X_I, and J \subseteq I then p^{\Pi \setminus J} is a potential for X_J that can be obtained by summing all the variables not in X_J. 
In this work we suppose that the conditional mass function $P(X_i|\Pi_i)$, 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:

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

(1)

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

This distribution verifies:

$$P(x_q|e) \propto \sum_{X_R} \prod_{X_i \in X} P(x_i|\pi_i) \prod_{x_i \in E} \delta_X(x_i; e_i)$$

(2)

where $X_R = X \setminus \{X_q, E\}$. In fact, the previous formula is the expression for $P(x_q|e)$. $P(x_q|e)$ can be obtained from $P(x_q|e)$ 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 $\Pi_i$ will be a closed, convex set $K(X_i|\Pi_i)$ of mappings $P : X_i \times \Pi_i \rightarrow [0, 1]$, verifying $\sum_{x_i \in \Omega_{X_i}} P(x_i, \pi_i) = 1, \forall \pi_i \in \Omega_{\Pi_i}$. Again, an extensive conditional credal set can be determined by its set of extreme points which we assume to be finite: 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 $\pi_i$ of $\Pi_i$. 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 for two separately specified credal sets ($K(X|X = y_1)$ and $K(X|X = y_2)$). From the separately specified credal sets, we obtain the extensive conditional credal set with:

$$K(X_i|\Pi_i) = \{P|P(x_i, \pi_i) \in K(X_i|\Pi_i = \pi_i),$$

$$\forall \pi_i \in \Omega_{\Pi_i}\}$$

(3)

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 $K(X)$ of a CN is the largest joint credal set such that every variable is strongly independent [13, 12] of its non-descendants non-parents 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 $K(X)$ 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 $K(X_i|\pi_i)$:

$$K(X) = CH\{P(X) : P(x) = \prod_{i=1}^{n} P(x_i|\pi_i),$$

$$\forall x \in \Omega_X, \forall \pi_i \in \Omega_{\Pi_i}, P(X_i|\pi_i) \in K(X_i|\pi_i)\}$$

(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 $K(X)$. So, the CN defines the following collection of joint probabilities:

$$P(X) = \{P_h(X)\}_{h=1}^{n_v}$$

(5)

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

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 $E$.

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.

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

$$K(X) = \prod_{i=1}^{n} K(X_i|I_i) \quad (6)$$

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

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

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

$$P(x_q|e) = \min_{k=1,...,n} \frac{P_k(x_q)}{\sum_{x_q} P_k(x_q)}$$

$$P(x_q|e) = \max_{k=1,...,n} \frac{P_k(x_q)}{\sum_{x_q} P_k(x_q)} \quad (8)$$

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 $T$ 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 $T$, denoted by $size(T)$, is defined as its nodes count.

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

$$\begin{array}{c|c|c|c}
A & B & C & p(A, B, C) \\
\hline
a_1 & b_1 & c_1 & 0.2 \\
a_2 & b_1 & c_2 & 0.5 \\
a_1 & b_2 & c_1 & 0.7 \\
a_2 & b_2 & c_2 & 0.7 \\
a_1 & b_3 & c_1 & 0.3 \\
a_2 & b_3 & c_2 & 0.3 \\
a_2 & b_2 & c_1 & 0.3 \\
a_2 & b_2 & c_2 & 0.3 \\
\end{array}$$

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 $BT$ 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
The process begins with a BPT at each internal node of the available variable states \( X \). 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 \( A, \Omega_A \), is \( \{a_1, a_2, a_3\} \), and the domain of \( B, \Omega_B \), 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, \( \text{Fig. 3 (i)} \) shows a BPT for the table \( \text{inducing classification trees, such as Quinlan's ID3 algorithm} \) [18], which builds a decision tree [20] and [8] we proposed as a potential for a set of variables \( \Omega \). When \( A \) is satisfied. At each step, a new BPT is obtained from the previous one, \( BT_j \). The greedy step requires the choice of a splitting criterion. It consists of expanding one of the leaf nodes \( t \) in \( BT_j \) 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 \( \Omega_X \), \( \Omega_X^b \subseteq \Omega_X^a \), is the set of available states of \( X \) at node \( t \). It is also necessary to distribute the set of available states \( \Omega_X^b \) of the chosen candidate variable \( X_i \) into two subsets, \( \Omega_X^b \) and \( \Omega_X^a \), 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 \( BT_j \) is expanded using variable \( B \). The set of available states of \( B \) at node \( t \), \( \Omega_B^t = \{b_1, b_2, b_3\} \) was partitioned into the sets \( \Omega_B^t = \{b_1\} \) and \( \Omega_B^t = \{b_2, b_3\} \). After applying this process, we say that the leaf node \( t \) has been expanded with variable \( X_i \) and the sets of states \( \Omega_X^a \) and \( \Omega_X^b \).

\[
\begin{align*}
&D(p, BT) = \sum_{x_I \in \Omega_X} p(x_I) \log \frac{p(x_I)}{BT(x_I)}
\end{align*}
\]

Kullback-Leibler’s divergence is always positive or zero. It is equal to zero if \( BT \) 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 represent 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.

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

Figures 3 and 4: Expansion of the terminal tree \( t \) with \( B \).
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:

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

$(10)$

where $\mathcal{T}_j(t, X_i)$ is the pruned tree obtained with the previous procedure is the tree that minimizes the information gain obtained after expanding node $t$.

It is immediate to see that $I(t, X_i, \Omega_{X_i}^l, \Omega_{X_i}^r) \geq 0$. By maximizing $I(t, X_i, \Omega_{X_i}^l, \Omega_{X_i}^r)$ 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 $\Delta$. 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 $\Pi_i$. The problem is transformed into an equivalent one by using a transparent variable $T_{\pi_i}$ for each configuration of the parents of $X_i$ ( $\pi_i \in \Omega_{\Pi_i}$). $T_{\pi_i}$ 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 $T_{\pi_i}$ 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 propor-
tional 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 val-
ues. 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, redu-
ing the space necessary with respect to the table
representation.

![Figure 5: A binary probability tree for \( K(X|Y) \)](image)

Algorithm 1: Variable Elimination

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

**Output:** \( \tilde{P}(x_q|e) \) and \( \tilde{P}(x_q|e) \) for each \( x_q \in \Omega_{X_q}, \ Y_q \in X \setminus E \)

1. Get the set \( S_B T \) of binary trees, building each binary tree
   \( B T_i \) from the credal sets \( K(X_i|\pi_i), \forall \pi_i \in \Omega_{X_i} \), (as in Figure 5)
2. Transform each \( B T_i \) into \( B T_{e(i)} \) (restrict to evidence)
3. Reorder variables and split sets in every \( B T_i \)
4. Prune each \( B T_i \) with the \( \Delta \) threshold
5. foreach \( Y \in X \setminus (E \cup \{X_q\}) \) do
6.  Let \( S_Y = \{B T_i|Y \in s(B T_i)\} \)
7.  Calculate \( B T_{prod} = \prod_{B T_i \in S_Y} B T_i \)
8.  Calculate \( B T_{sum} = B T_{prod}\setminus Y \)
9.  Reorder variables and split sets in \( B T_{sum} \)
10. Prune \( B T_{sum} \) using the \( \Delta \) threshold
11. \( S_B T = (S_B T \setminus S_Y) \cup B T_{sum} \)
12. Calculate \( B T_q = \prod_{B T_i \in S_B T} B T_i \)
13. Get \( \tilde{P}(x_q|e) \) and \( \tilde{P}(x_q|e) \) by normalizing the vertices in \( B T_q \)

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 \( \{K(X_i|\pi_i), \forall \pi_i \in \Omega_{X_i}\} \). 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 \( \Delta \) 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 per-
formed 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 marginaliza-
tion (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 re-
spectively. The pruning operation can select any vari-
able (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). Fi-
ally the upper and lower bounds for the probability
of the queried variable can be obtained by normaliz-
ing each one of the vertices in \( B T_q \) (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 algo-
rithm. When a \( \Delta = 0 \) threshold is used, no variable
will be pruned unless there are context specific inde-
dependences in the potentials. In the worst case, using
\( \Delta = 0 \), 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
\( \Delta = 0 \), 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 \( \Delta \)
greater than 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 con-
ditional credal sets for each variable \( X_i \) and each con-
figuration 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 $\Omega_{\Pi}$, we associated a given number of vertices in the credal set $K(X_i|\pi_i)$. 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 does 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 ($|E|$), the number of vertices per credal set (nvpe), the percentage of configurations (per) of $\Omega_{\Pi}$ that will contain nvpe vertices, and the potential size of the strong extension ($n_v$) of the CN. In the calculus of $n_v$ we suppose that the barren nodes for the given query have been removed from the network.

| Ex | Var        | Network | $|E|$ | nvpe | per      | $n_v$ |
|----|------------|---------|------|------|----------|-------|
| 1  | Venttube   | Alarm   | 0    | 3    | 90       | 364294|
| 2  | Expo2      | Alarm   | 0    | 3    | 17       | 177147|
| 3  | RiskAv      | Insurance | 0    | 3    | 70       | 177147|
| 4  | DrivHist   | Insurance | 0    | 3    | 31.5     | 177147|
| 5  | Venttube   | Alarm   | 6    | 3    | 12.25    | 364294|
| 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:

$$
\sqrt{\frac{\sum_{\omega_q \in \Omega_X} ((P^*(x_q|e) - P(x_q|e))^2 + (P^*(x_q|e) - \overline{P}(x_q|e))^2)}{2 \cdot |\Omega_X|}}
$$

(12)

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

These parameters (mean square error, maximum size and time) are measured running the Algorithm 1 with several values for the $\Delta$ threshold using SPTs and BPTs. We have used values for $\Delta$ 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 $\Delta$ cause large errors but require lower computing time and smaller trees. Small values of $\Delta$ 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 RiskAv (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
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 $\Delta$ used for pruning the trees. The choice of $\Delta$ 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.
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


