Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
44 (2007) 261–280

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks

Andrés Cano *, Manuel Gómez, Serafín Moral, Joaquín Abellán

Department of Computer Science and Artificial Intelligence, E.T.S. Ingeniería Informática, University of Granada, 18071 Granada, Spain

Received 15 December 2005; received in revised form 30 June 2006; accepted 31 July 2006
Available online 29 September 2006

Abstract

This paper proposes two new algorithms for inference in credal networks. These algorithms enable probability intervals to be obtained for the states of a given query variable. The first algorithm is approximate and uses the hill-climbing technique in the Shenoy–Shafer architecture to propagate in join trees; the second is exact and is a modification of Rocha and Cozman's branch-and-bound algorithm, but applied to general directed acyclic graphs.

Keywords: Credal network; Probability intervals; Bayesian networks; Strong independence; Hill-climbing; Branch-and-bound algorithms

1. Introduction

A credal network is a graphical structure (a directed acyclic graph (DAG) [10]) which is similar to a Bayesian network [20], where imprecise probabilities are used to represent quantitative knowledge. Each node in the DAG represents a random event, and is associated with a set of convex sets of conditional probability distributions.

* Corresponding author. Tel.: +34 958 240803; fax: +34 958 243317.
E-mail addresses: acu@decsai.ugr.es (A. Cano), mgomez@decsai.ugr.es (M. Gómez), smc@decsai.ugr.es (S. Moral), jabellan@decsai.ugr.es (J. Abellán).

0888-613X/$ - see front matter
doi:10.1016/j.ijar.2006.07.020

There are several mathematical models for imprecise probabilities [28]. We consider convex sets of probabilities to be the most suitable for calculating and representing imprecise probabilities. They are powerful enough to represent the results of basic operations (combination and marginalization) within the model without having to make approximations. Such approximations on basic operations cause loss of information (as in interval probabilities).

In the following, we consider *inference* with credal networks as the problem of computing posterior upper and lower probabilities for each state of a specific *query* variable, given a set of observations. A very common hypothesis is that of separately specified credal sets [19,23] (there is no restriction between the set of probabilities associated to a variable given two different configurations of its parents). According to it, there is a different credal set for each one of the configurations of the parents [23]. This hypothesis will be accepted in this paper. Some authors have considered the propagation of probabilistic intervals directly in graphical structures [1,15,26,27]. In the proposed procedures, however, there is no guarantee that the calculated intervals are always the same as those obtained when a global computation using the associated convex sets of probabilities is used. In general, it can be said that calculated bounds are wider than exact ones. The problem is that exact bounds need a computation with the associated convex sets of probabilities. Following this approach, different approximate and exact methods have been proposed [6,4,3,9,23,21]. These assume that there is a convex set of conditional probabilities for each configuration of the parent variables in the dependence graph, and provide a model for obtaining exact bounds through the use of local computations. Working with convex sets, however, may be extremely inefficient: if we have n variables and each variable, X_i, has a convex set with l_i extreme points as the conditional information, the propagation algorithm has a complexity order of $O(K \cdot \prod_{i=1}^{n} l_i)$, where K is the complexity of carrying out a simple probabilistic propagation. This is so since the propagation of convex sets is equivalent to the propagation of all the global probabilities that can be obtained by choosing an exact conditional probability in each convex set.

In this paper, we shall propose a new approximate algorithm for inference in credal networks. This is a hill-climbing procedure which quickly obtains good inner approximations to the correct intervals. The algorithm is based on the Shenoy–Shafer architecture [25] for propagation in join trees. Rocha et al. [22] have presented another hill-climbing search, inspired by the Lukatskii–Shapot algorithm, for obtaining accurate inner approximations. We shall also propose a branch-and-bound procedure (also used in [21]) for inference in credal networks, where the initial solution is computed with our hill-climbing algorithm, reducing its running time. This approach is similar to the one in [22], but our algorithm will be applicable to general graph structures. Furthermore, our hill-climbing initial solution will reduce the total running time. de Campos and Cozman [13,14] also give a multilinear programming method which provides a very fast branch-and-bound procedure.

The rest of the paper is organized as follows: Section 2 describes the basics of probability propagation in Bayesian networks and the Shenoy–Shafer architecture; Section 3 presents basic notions about credal sets and credal networks; Section 4 introduces probability trees and briefly describes two algorithms based on variable elimination and probability trees for inference in credal networks; Section 5 details the proposed hill-climbing algorithm for inference in credal networks; Section 6 explains how to apply the proposed algorithm in Rocha and Cozman's branch-and-bound algorithm; Section 7 shows the experimental work; and finally Section 8 presents the conclusions.

2. Probability propagation in Bayesian networks

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of variables. Let us assume that each variable X_i takes values on a finite set Ω_{X_i} (the frame 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 $\{X_i | i \in I\}$. Let $N = \{1, \ldots, n\}$ be the set of all the indices. 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 $\mathbf{X}_I$ and will be written with $\mathbf{x}$ or $\mathbf{x}_I$. In Shenoy and Shafer's [25] terminology, a mapping from a set $\Omega_{\mathbf{X}_I}$ into $\mathbb{R}_0^+$ will be called a *valuation* h for $\mathbf{X}_I$. Shenoy and Shafer define two operations on valuations: *combination* $h_1 \otimes h_2$ (multiplication) and *marginalization* $h^{\downarrow J}$ (by summing out all the variables not in J).

A *Bayesian network* is a directed acyclic graph (see the left side of Fig. 1), where each node represents a random event X_i, and the topology of the graph shows the independence relations between variables according to the d-separation criterion [20]. Each node X_i also has a conditional probability distribution $p_i(X_i|\mathbf{Y})$ for that variable given its parents $\mathbf{Y}$. Following Shenoy and Shafer's terminology, these conditional distributions can be considered as valuations. A Bayesian network determines a unique joint probability distribution:

$$p(\mathbf{X} = \mathbf{x}) = \prod_{i \in N} p_i(x_i|\mathbf{y}_j) \quad \forall \mathbf{x} \in \Omega_{\mathbf{X}} \tag{1}$$

An *observation* is the knowledge of the exact value $X_i = x_i^j$ of a variable. The set of observed variables is denoted by $\mathbf{X}_E$; the configuration for these variables is denoted by $\mathbf{x}_E$ and it is called the *evidence set*. E will be the set of indices of the observed variables. Each observation, $X_i = x_i^j$, is represented by means of a valuation which is a Dirac function defined on Ω_{X_i} as $\delta_{X_i}(x_i; x_i^j) = 1$ if $x_i^j = x_i$, $x_i \in \Omega_{X_i}$, and $\delta_{X_i}(x_i; x_i^j) = 0$ if $x_i^j \neq x_i$.

The aim of probability propagation algorithms is to calculate the *a posteriori* probability function $p(x_q|\mathbf{x}_E)$, (for every $x_q \in \Omega_{X_q}$) by making local computations, where X_q is a given query variable. This distribution verifies:

$$p(x_q|\mathbf{x}_E) \propto \left(\prod_{X_i} p(x_i|\mathbf{y}_j) \prod_{x_i^j \in \mathbf{x}_E} \delta_{X_i}(x_i; x_i^j) \right)^{\downarrow X_q} \tag{2}$$

In fact, the previous formula is the expression for $p(x_q, \mathbf{x}_E)$. $p(x_q|\mathbf{x}_E)$ can be obtained from $p(x_q, \mathbf{x}_E)$ by normalization.

A known propagation algorithm can be constructed by transforming the DAG into a *join tree*. For example, the right-hand tree in Fig. 1 shows a possible join tree for the DAG on the left. There are several schemes [18,25,24,16] for propagating on join trees. We shall follow Shenoy and Shafer's scheme [25]. Every node in the join tree has a valuation Ψ_{C_i} attached, initially set to the identity mapping. There are two messages (valuations)

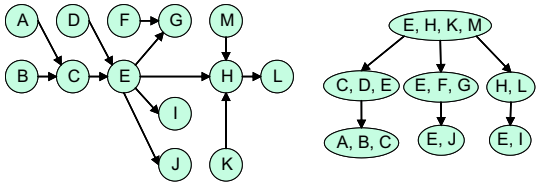

Fig. 1. A Bayesian network and its join tree.

$M_{C_i \to C_j}$, $M_{C_j \to C_i}$ between every two adjacent nodes C_i and C_j. $M_{C_i \to C_j}$ is the message that C_i sends to C_j, and $M_{C_j \to C_i}$ is the message that C_j sends to C_i. Every conditional distribution p_i and every observation δ_i will be assigned to one node C_i which contains all its variables. Each node contains a (possibly empty) set of conditional distributions and Dirac functions, which must then be combined into a valuation, Ψ_{C_i}.

The propagation algorithm is performed by crossing the join tree from leaves to root and then from root to leaves, updating messages as follows:

$$M_{C_i \to C_j} = \left(\Psi_{C_i} \cdot \left(\prod_{C_k \in Ady(C_i, C_j)} M_{C_k \to C_i} \right) \right)^{\downarrow C_i \cap C_j} \tag{3}$$

where $Ady(C_i, C_j)$ is the set of adjacent nodes to C_i with the exception of C_j.

Once the propagation has been performed, the *a posteriori* probability distribution for X_q, $p(X_q|\mathbf{x}_E)$, can be calculated by looking for a node C_i containing X_q and normalizing the following expression:

$$p(X_q, \mathbf{x}_E) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M_{C_j \to C_i} \right) \right)^{\downarrow X_q} \tag{4}$$

where $Ady(C_i)$ is the set of adjacent nodes to C_i. The normalization factor is the probability of the given evidence that can be calculated from the valuation calculated in expression (4) using:

$$p(\mathbf{x}_E) = \sum_{x_q^i} p(x_q^i, \mathbf{x}_E) \tag{5}$$

In fact, we can compute the probability of the evidence, $p(\mathbf{x}_E)$, by choosing any node C_i, combining all the incoming messages with the valuation Ψ_{C_i}, and summing out all the variables in C_i.

3. Inference in credal networks

A credal set for a variable X_i is a convex, closed set of probability distributions and shall be denoted by H^{X_i}. We assume that every credal set has a finite number of extreme points. The extreme points of a convex set are also called *vertices*. A credal set can be identified by enumerating its vertices. A conditional credal set about X_i given a set of variables $\mathbf{Y}$ will be a closed, convex set $H^{X_i|\mathbf{Y}}$ of mappings $p : X_i \times \mathbf{Y} \to [0, 1]$, verifying $\sum_{x_i \in \Omega_{X_i}} p(x_i, \mathbf{y}_j) = 1, \forall \mathbf{y}_j \in \Omega_{\mathbf{Y}}$. Once again, we suppose a finite set of extreme points, $\mathrm{Ext}(H^{X_i|\mathbf{Y}}) = \{p_1, \ldots, p_l\}$. Some authors call *extensive* conditional credal set to this way of specifying a conditional credal set. See, for example, Ref. [11].

A credal network is a directed acyclic graph similar to a Bayesian network. Each node is also associated with a variable, but every variable is now associated with a credal set $H^{X_i|\mathbf{Y}}$, where $\mathbf{Y}$ are the parent variables of X_i in the graph. In this paper, we suppose that a *local* credal set $H^{X_i|\mathbf{Y}=\mathbf{y}_j}$ is given for each configuration $\mathbf{y}_j$ of $\mathbf{Y}$. This is described by Rocha and Cozman [23] as *separately specified credal sets*. For example Fig. 2 shows a credal network with two variables (X and Y). Conditional information for X is given by two separately specified credal sets ($H^{X|Y=y_1}$ and $H^{X|Y=y_2}$). From a separately specified credal set, we obtain

H^Y	y_1	y_2
r_1	0.1	0.9
r_2	0.2	0.8

$H^{X\|Y=y_1}$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$H^{X\|Y=y_2}$	x_1	x_2
q_1	0.40	0.6
q_2	0.60	0.4

Fig. 2. A simple credal network.

a conditional one (extensive conditional credal set) with $H^{X_i|\mathbf{Y}} = \{p|f_{\mathbf{y}_j}(x_i) = p(x_i, \mathbf{y}_j) \in H^{X_i|\mathbf{Y}=\mathbf{y}_j}, \forall \mathbf{y}_j \in \Omega_{\mathbf{Y}}\}$. Table 1 shows the extensive conditional credal set $H^{X|Y}$ obtained from the separately specified credal sets $H^{X|Y=y_1}$ and $H^{X|Y=y_2}$ of Fig. 2.

As in the case of Bayesian networks, the topology of a credal network 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* [10,8]. The *strong extension* of a credal network is the largest joint credal set such that every variable is strongly independent [10,8] of its non-descendants non-parents given its parents. The strong extension of a credal network is the joint credal set that contains every possible combination of vertices for all credal sets in the network, such that the vertices are combined by multiplication as in expression (1) [10].

An *inference* in an extension of a credal network is the computation of tight bounds for the probability values of a query variable X_q given a set of observed variables $\mathbf{X}_E$. This paper is dedicated to inference algorithms in the strong extension of a credal network. The propagation of credal sets is completely analogous to the propagation of probabilities; the procedures are the same. Here, we shall only describe the main differences and further details can be found in [6]. Here, valuations are convex sets of possible probabilities, with a finite number of vertices. A conditional valuation is a convex set of conditional probability distributions (extensively conditional credal set). An observation of a value for a variable will be represented in the same way as in the probabilistic case. The *combination* of two convex sets of mappings is the convex hull of the set obtained by multiplying a mapping of the first convex set with a mapping of the second convex set (repeating the probabilistic combination for all pairs of vertices of the two convex sets). The *marginalization* of a convex set is defined by marginalizing each mapping of the convex set. A more detailed

Table 1
An extensive conditional credal set

$H^{X\|Y}$	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

description of these operations can be found for example in [2]. With these operations, we can carry out the same propagation algorithms as in the probabilistic case.

The result of the propagation for a variable, X_q, will be a convex set of mappings from Ω_{X_q} in $[0,1]$, also called points (as Ω_{X_q} is finite). This convex set will be called R_q. If we assume that $\Omega_{X_q} = \{x_q^1, \ldots, x_q^m\}$, then the points of R_q are obtained in the following way: if p is a global probability distribution, formed by selecting a fixed probability for each convex set, then we shall obtain a point $(p_1, \ldots, p_m) \in R_q$ associated to this probability, where $p_i = p(x_q^i, \mathbf{x}_E)$, with $\mathbf{x}_E$ being the given evidence. If we normalize each point of R_q by computing $p_i(x_q^i|\mathbf{x}_E) = p_i / \sum_j p_j$, then we obtain the convex set of a posteriori conditional probability distributions for each $x_q^i \in \Omega_{X_q}$.

4. Probability trees

Probability trees [5] have been used as a flexible data structure that allows asymmetrical independences and exact or approximate representations of probability valuations (also called potentials) to be used.

A *probability tree* $\mathscr{T}$ is a directed labeled tree, where 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 associated with that node. The *size* of a tree $\mathscr{T}$, denoted by $size(\mathscr{T})$, is defined as its number of leaves.

A probability tree $\mathscr{T}$ on variables $\mathbf{X}_I = \{X_i | i \in I\}$ represents a valuation $h : \Omega_{\mathbf{X}_I} \to \mathbb{R}_0^+$ if for each $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$ the value $h(\mathbf{x}_I)$ is the number stored in the leaf node that is reached by starting from the root node and selecting the child corresponding to coordinate x_i for each internal node labeled X_i.

A probability tree is usually a more compact representation of a valuation than a table. This is illustrated in Fig. 3, which displays a valuation h and its representation using a probability tree. The tree contains the same information as the table, but using only five values instead of eight. Furthermore, trees enable even more compact representations to be obtained in exchange for loss of accuracy. This is achieved by pruning certain leaves and replacing them by the average value, as shown in the second tree in Fig. 3.

We say that part of a probability tree is a *terminal tree* if it contains only one node labeled with a variable, and all the children are numbers (leaf nodes).

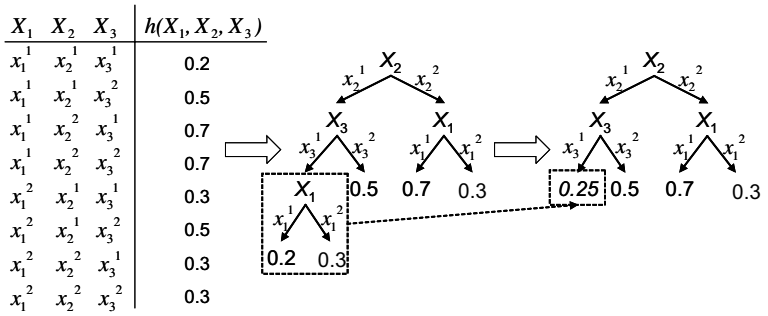

Fig. 3. A valuation h, its representation as a probability tree and its approximation after pruning various branches.

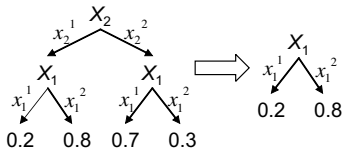

Fig. 4. Restriction of a tree to the value $X_2 = x_2^1$.

If $\mathscr{T}$ is a probability tree on $\mathbf{X}_I$ and $\mathbf{X}_J \subseteq \mathbf{X}_I$, we use $\mathscr{T}^{R(\mathbf{x}_J)}$ (probability tree restricted to the configuration $\mathbf{x}_J$) to denote the *restriction operation* which consists in returning the part of the tree which is consistent with the values of the configuration $\mathbf{x}_J \in \Omega_{\mathbf{X}_J}$. For example, in the left probability tree in Fig. 3, $\mathscr{T}^{R(X_2=x_2^1, X_3=x_3^1)}$ represents the terminal tree enclosed by the dashed line square. This operation is used to define combination and marginalization operations. It is also used for conditioning. Fig. 4 shows another example of the restriction operation. Right probability tree is the result of restricting left tree to $X_2 = x_2^1$.

The basic operations (*combination*, *marginalization*) over potentials can be carried out directly on probability trees (further details can be found in [5]). The combination of a probability tree comprising a single node labeled with a real number r, and another probability tree $\mathscr{T}$ is obtained by multiplying all the leaf nodes of $\mathscr{T}$ by r. The combination of two general probability trees $\mathscr{T}_1$ and $\mathscr{T}_2$ is obtained in the following way: for each leaf node l in $\mathscr{T}_1$, if $\mathbf{X}_J = \mathbf{x}_J$ is the configuration for the ancestor nodes of l, then l is replaced by the combination of node l and $\mathscr{T}_2^{R(\mathbf{X}_J=\mathbf{x}_J)}$. The *sum* of two probability trees is defined in the same way as the combination but by adding (rather than multiplying) real numbers. Marginalization is equivalent to deleting variables. A variable is deleted from a probability tree and is replaced by the sum of its children. Figs. 5–7 show examples of combination, marginalization and addition on probability trees., as described in the next algorithms.

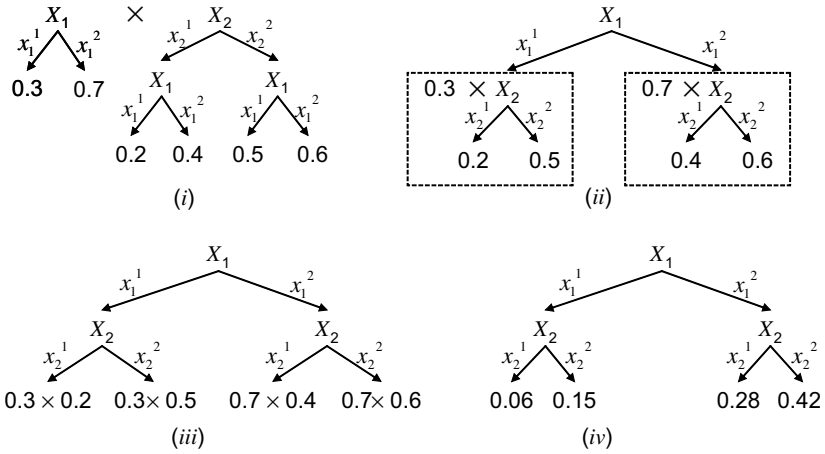

Fig. 5. Combination of two probability trees.

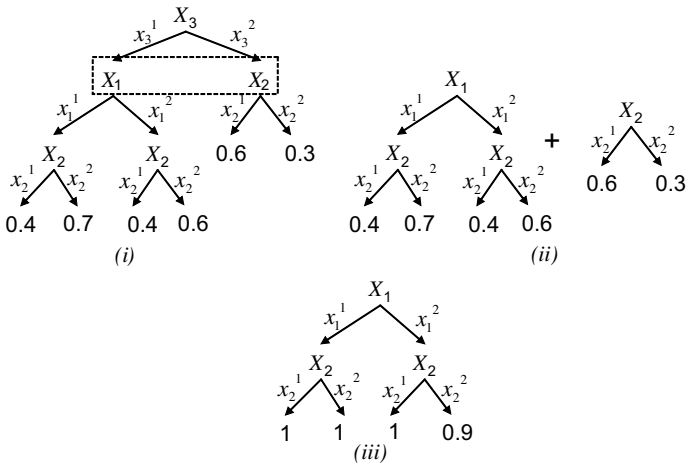

Fig. 6. Marginalizing out variable X_3.

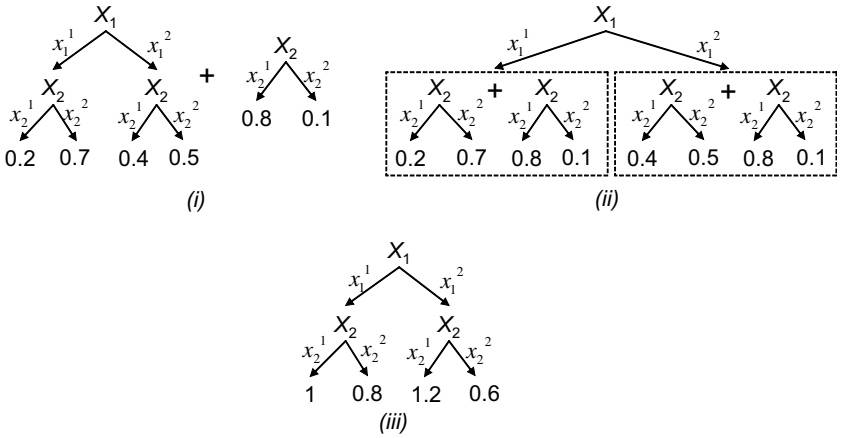

Fig. 7. Addition of two probability trees.

4.1. Propagating credal sets using probability trees

Different algorithms have been used for propagation in credal networks using probability trees [4,3]. For each X_i, we originally have a collection of m local credal sets $\{H^{X_i|\mathbf{Y}=\mathbf{y}_1}, \ldots, H^{X_i|\mathbf{Y}=\mathbf{y}_m}\}$, where m is the number of configurations of $\mathbf{Y}$. The problem is now transformed into an equivalent one by using a *transparent variable* $T_{\mathbf{y}_j}$ for each $\mathbf{y}_j \in \Omega_{\mathbf{Y}}$. $T_{\mathbf{y}_j}$ will have as many cases as the number of vertices in the local credal set $H^{X_i|\mathbf{Y}=\mathbf{y}_j}$. A vertex of the global credal set $H^{X_i|\mathbf{Y}}$ can be found by fixing all transparent variables $T_{\mathbf{y}_j}$ to one of its values. Let us use $\mathbf{T}$ to denote the set of all transparent variables in the credal network.

Probability trees enable a conditional credal set $H^{X|\mathbf{Y}}$ to be represented efficiently when we start with m local credal sets $\{H^{X_i|\mathbf{Y}=\mathbf{y}_1}, \ldots, H^{X_i|\mathbf{Y}=\mathbf{y}_m}\}$ 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. 8, we can see one example where a probability tree represents the global information $H^{X|Y}$ associated to the two credal sets $H^{X|Y=y_1}$ and $H^{X|Y=y_2}$. The figure also shows the global conditional credal set $H^{X|Y}$ by means of a table. In the probability tree in Fig. 8, we obtain the extreme points by fixing T_{y_1} and T_{y_2} to one of its values. For example, if the probability tree is restricted to $T_{y_1} = t^1_{y_1}$ and $T_{y_2} = t^2_{y_2}$, we obtain a new probability tree that gives us the extreme point r_2. The tree avoids repetition of probability values, reducing the space necessary with respect to the table representation.

The simpler approximate algorithm for propagating credal sets using probability trees is based on variable elimination [4]. In this algorithm, all the variables should be removed (by marginalization) except the query variable and the transparent variables, which are used to compute the upper and lower probability bounds. The algorithm applies a pruning procedure in order to reduce the size of the probability trees that represent initial conditional distributions, and the potentials obtained after combination or marginalization in the propagation procedure. This allows to maintain probability trees in a reasonable size. The pruning is an iterative method that selects a terminal tree and replaces it with the average of its leaf nodes. A terminal tree can be selected for pruning if Kullback–Leibler's cross-entropy [17] between the potential before pruning and the potential after pruning is below a given threshold σ. The pruning operation can select any variable in the probability tree including a transparent variable. The method is applied until there is no terminal tree with a distance smaller than a given threshold σ. The greater the parameter σ, the smaller the probability tree obtained. Further information about the measure used to select the terminal tree for pruning can be found in Proposition 2 in [4].

When we apply the variable elimination algorithm described above to calculate probability intervals for a given variable X_q, then, in general, the correct intervals enclose the

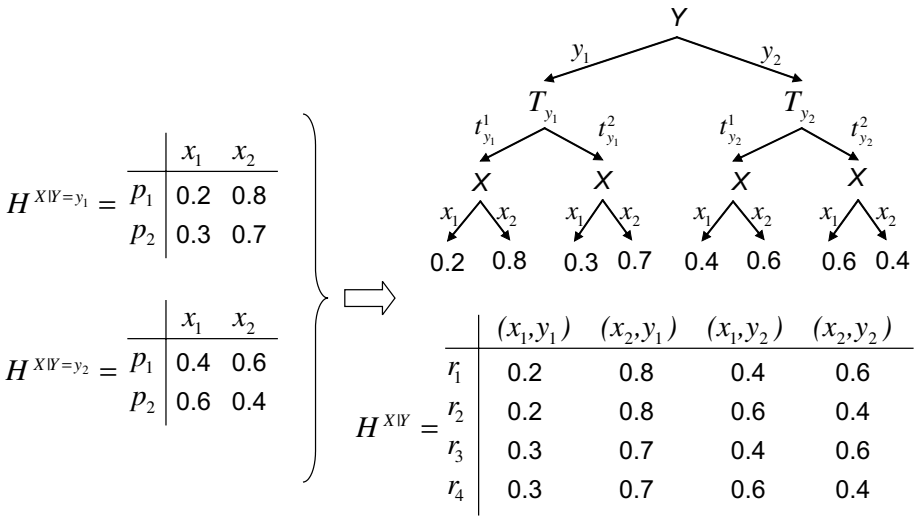

Fig. 8. A probability tree for $H^{X|Y}$.

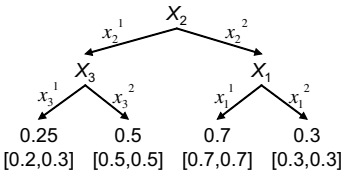

Fig. 9. Probability tree with $r_{\min}$ and $r_{\max}$ values.

approximate ones (*inner* approximation). In [4], a method is also proposed for obtaining *outer* approximations. The proposal consists in using two new values at each leaf node in probability trees. If previously we only had one value $r \in \mathbb{R}_0^+$, we now add the values $r_{\min}$, $r_{\max} \in \mathbb{R}_0^+$ at each leaf node. These values inform us about the interval in which the true value can oscillate. When a branch of the tree has not been approximated, then $r_{\min}$, $r_{\max}$ and r will be the same; when it is approximated, however, then r will be between $r_{\min}$ and $r_{\max}$. For example, Fig. 9 shows the probability tree in Fig. 3, supposing that the only approximation is the one shown in this figure. In [4], details can be found about the calculation of the $r_{\min}$ and $r_{\max}$ values for the probability trees resulting from combination and marginalization operations. Once the variable elimination algorithm has finished, we obtain a probability tree with the variable of interest X_q and some transparent variables. The leaf nodes contain $r_{\min}$, $r_{\max}$ values. These values enable *outer* bounds to be obtained (see [4] for further details).

5. Hill-climbing algorithm

The objective of the proposed algorithm is to obtain the upper or lower bound for $p(x_q^i|\mathbf{x}_E)$ (posterior probability of a case x_q^i of a given query variable X_q). This can be solved by selecting the configuration of transparent variables (a configuration $\mathbf{t}_s$ of transparent variables determines a global probability distribution $p_{\mathbf{t}_s}$) which results in a minimum value for $p(x_q^i|\mathbf{x}_E)$ (and the configuration for the maximum). Let us suppose that our problem consists in finding the configuration $\mathbf{t}_s$ which results in the upper probability for $p(X_q = x_q^i|\mathbf{x}_E)$:

$$\max_{\mathbf{t}_s} p_{\mathbf{t}_s}(X_q = x_q^i|\mathbf{x}_E) \tag{6}$$

The proposed algorithm is a hill-climbing algorithm based on the Shenoy–Shafer propagation algorithm for Bayesian networks. A run of the algorithm is directed to compute the lower or upper probability for a single state x_q^i of a given query variable X_q. It will obtain inner bounds for $p(X_q = x_q^i)$ in the strong extension of a given credal network.

The algorithm begins with the construction of a join tree from the credal network (see Section 2), like the one shown in Fig. 1. Now, a double message system is necessary. For each pair of connected nodes, C_i and C_j, there are two messages going from C_i to C_j, $M^1_{C_i \to C_j}$ and $M^2_{C_i \to C_j}$, and two messages going from C_j to C_i, $M^1_{C_j \to C_i}$ and $M^2_{C_j \to C_i}$ (see Fig. 10). Messages $M^1_{C_i \to C_j}$ will be calculated as usual, according to formula (3). Messages $M^2_{C_i \to C_j}$ will be also calculated as usual but we assume that the observation $X_q = x_q^i$ is added to the evidence set $\mathbf{x}_E$.

After constructing the join tree, each conditional credal set $H^{X_i|\mathbf{Y}}$ is assigned to one node as we explained for Bayesian networks in Section 2. Each conditional credal set $H^{X_i|\mathbf{Y}}$ is

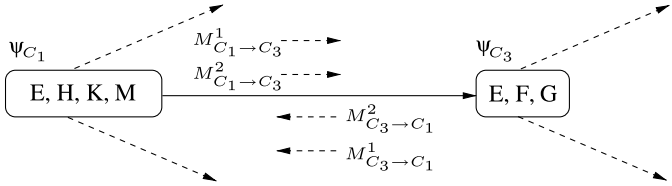

Fig. 10. Double messages between two nodes of the join tree.

represented by means of a probability tree (see Section 4 and Fig. 8). For observations, we follow a different procedure from the one explained in Section 2. If we have an observation $X_i = x_i^j$, then we should combine its valuation δ_{X_i} with a valuation Ψ_{C_i} containing X_i. We achieve the same result with the operation of *restriction* in probability trees; that is, for each observation $X_i = x_i^j$, we restrict Ψ_{C_i} to $X_i = x_i^j$ if node C_i contains X_i. In this way, valuations are significantly reduced in size, making posterior operations (combination and marginalization) more efficient. The valuation Ψ_{C_i} is then calculated for every node C_i by combining all the valuations assigned to node C_i and restricting the combination of all its assigned valuations to the evidence set $\mathbf{x}_E$. The resulting Ψ_{C_i} is saved on a copy valuation $\Psi^c_{C_i}$.

At a given time, the algorithm has associated a configuration $\mathbf{t}$ for the transparent variables. The configuration $\mathbf{t}$ will be modified in the hill-climbing step in order to optimize the conditional probability $p_{\mathbf{t}}(X_q = x_q^i|\mathbf{x}_E)$. A random initial configuration $\mathbf{t}_0$ is selected for the set of transparent variables $\mathbf{T}$. This initial configuration of $\mathbf{T}$ is appended to the evidence set $\mathbf{x}_E$. Each probability tree Ψ_{C_i} is then restricted according to this new set of observations. For example, in Fig. 8, if the initial configuration for $\mathbf{T}$ contains $T_{y_1} = t^1_{y_1}$ and $T_{y_2} = t^2_{y_2}$, then the valuation (probability tree) Ψ_{C_i} containing $H^{X_i|\mathbf{Y}}$ will be restricted and the tree in Fig. 11 shall be obtained.

The valuations Ψ_{C_i} no longer contain transparent variables because all of them are observed and disappear from the probability trees after restricting to the observed values. In this join tree, a propagation from root to leaves and then from leaves to root would allow the probability distribution $p_{\mathbf{t}_0}(X_q|\mathbf{x}_E)$ to be obtained for the vertex of the a posteriori credal set $H^{X_q|\mathbf{x}_E}$ corresponding to $\mathbf{T} = \mathbf{t}_0$, by looking for a node C_i containing the variable X_q and using expression (4); however, we follow a different procedure. The algorithm makes an *initial propagation* (using the double message system) sending messages from leaf nodes towards the root. Messages are calculated as we explained above.

After the initial propagation, we begin the hill-climbing step by crossing the join tree from root to leaves and vice versa a given number of times, or until the algorithm does not improve the best solution found so far. Every time we visit a node C_i, we locally maximize $p(X_q = x_q^i|\mathbf{x}_E)$ by choosing a state for each transparent variable in C_i. Transparent

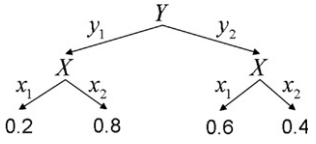

Fig. 11. Restricted tree for $H^{X|Y}$ using $T_{y_1} = t^1_{y_1}$ and $T_{y_2} = t^2_{y_2}$.

variables are improved one by one. The process can require the set of transparent variables of C_i to be treated several times. We leave the current node when there is no further transparent variable to improve.

In the crossing of the join tree, let us suppose that we are now visiting a node C_i and that $\mathbf{t}$ is the current configuration for the transparent variables. Let us also assume that $T_{\mathbf{y}_j}$ is the transparent variable which will be improved now (one of the transparent variables included in node C_i). The hill-climbing algorithm must locally select the best value for the variable $T_{\mathbf{y}_j}$. At this moment all transparent variables are fixed to a value. This determines a point of the joint credal set. If we discard the observation for $T_{\mathbf{y}_j}$ we obtain $|\Omega_{T_{\mathbf{y}_j}}|$ points of the joint credal set. The idea is to assign to $T_{\mathbf{y}_j}$ the case that maximizes $p(X_q = x_q^i|\mathbf{x}_E)$. In order to do so, we need to calculate the probability of the evidence $p(\mathbf{x}_E)$ and the probability $p(x_q^i, \mathbf{x}_E)$ for each value of $T_{\mathbf{y}_j}$. With $p(\mathbf{x}_E)$ and $p(x_q^i, \mathbf{x}_E)$, we can compute $p(x_q^i|\mathbf{x}_E)$, the value we want to maximize. The vector of values $p(\mathbf{x}_E)$ is obtained as follows: in the first message system, we discard the previous observation of variable $T_{\mathbf{y}_j}$ in node C_i, and then for each $t_{y_j}^i \in \Omega_{y_j}$ we add $T_{\mathbf{y}_j} = t_{y_j}^i$ to the set of observations, computing $p(\mathbf{x}_E)$ for the resulting configuration by the procedure indicated at the end of Section 2. The vector of values $p(x_q^i, \mathbf{x}_E)$ is calculated in a similar way with the second message system.

In the previous computation, we can discard the previous observation for $T_{\mathbf{y}_j}$ using the following procedure: a new Ψ_{C_i} is calculated restricting the saved valuation $\Psi_{C_i}^c$ to the current configuration of transparent variables $\mathbf{t}$, but removing the observation for $T_{\mathbf{y}_j}$ in such a configuration. In this way, the only transparent variable in Ψ_{C_i} is $T_{\mathbf{y}_j}$. A new configuration $\mathbf{t}$ will be obtained by selecting in $T_{\mathbf{y}_j}$ the state that maximizes $p(x_q^i|\mathbf{x}_E)$. The process continues by improving the next transparent variable in C_i, and so on until modification of any transparent variable does not improve $p(x_q^i|\mathbf{x}_E)$. In this case, a new node is visited.

Algorithm 1 shows the main steps of the hill climbing procedure.

Algorithm 1

Input: A credal network $\mathcal{N}_0$
Output: The value of $\overline{p} = \max p(x_q^i|\mathbf{x}_E)$
Represent each conditional credal set $H^{X_i|\mathbf{Y}}$ using a probability tree $\mathcal{T}$
Build a *join tree* from the credal network
Associate each conditional credal set $H^{X_i|\mathbf{Y}}$ (a probability tree) to one node C_i of the join tree
for each *available probability tree* $\mathcal{T}$ **do**
  Incorporate evidence $\mathbf{x}_E$ by using the *restriction* operation on $\mathcal{T}$: $\mathcal{T}^{R(\mathbf{x}_E)}$
end
for each C_i *in the join tree* **do**
  Calculate Ψ_{C_i} by combining all the probability trees associated to node C_i.
end
Choose a random initial configuration $\mathbf{t}_0$ for the set of transparent variables $\mathbf{T}$
Incorporate evidence $\mathbf{t}_0$ to all the probability trees in the join tree using *restriction* operation
Carry out an *initial propagation* in the join tree using a double system of messages

- Messages $M^1_{C_i \to C_j}$ are used to propagate the given evidence $\mathbf{x}_E$ and the current configuration for **T:** they allow to compute $p(\mathbf{x}_E)$

- Messages $M^2_{C_i \to C_j}$ propagate $\mathbf{x}_E$, the current configuration for $\mathbf{T}$ and $X_q = x^i_q$: they allow to compute $p(x^i_q, \mathbf{x}_E)$

for each step $= 1$ to m **do**
Traverse the join tree from root to leaves and vice versa
for each visited node C_i and transparent variable T_j in Ψ_{C_i} **do**
- Calculate $p(\mathbf{x}_E)$ and $p(x^i_q, \mathbf{x}_E)$ for each $t^i_j \in \Omega_{T_j}$
  (1) Discard the current observation for T_j.
  (2) Calculate values for $p(\mathbf{x}_E)$ and $p(x^i_q, \mathbf{x}_E)$ for each $t_j \in \Omega_{T_j}$:

$$p(\mathbf{x}_E, T_j) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M^1_{C_j \to C_i} \right) \right)^{\downarrow T_j}$$

$$p(x^i_q, \mathbf{x}_E, T_j) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M^2_{C_j \to C_i} \right) \right)^{\downarrow T_j}$$

- From the vectors $p(\mathbf{x}_E, T_j)$ and $p(x^i_q, \mathbf{x}_E, T_j)$ calculate a new vector:

$$p(x^i_q | \mathbf{x}_E, T_j) = \frac{p(x^i_q, \mathbf{x}_E, T_j)}{p(\mathbf{x}_E, T_j)}$$

- Choose the case $t^i_j \in \Omega_{T_j}$ giving the maximum in $p(x^i_q | \mathbf{x}_E, T_j)$
- $T_j = t^i_j$ is appended to the configuration of transparent variables $\mathbf{T}$.

end
end
Take $\overline{p}$ as the biggest $p(x^i_q | \mathbf{x}_E, T_j)$ found in the m steps.

6. Branch-and-bound algorithm

Rocha and Cozman [21] have given an exact algorithm for inference on credal networks with a polytree structure which is based on branch-and-bound optimization search algorithms. The authors distinguish between *outer* and *inner* approximate inference algorithms. The former are produced when the correct interval between lower and upper probabilities is enclosed in the approximate interval; the latter approximations are produced when the correct interval encloses the approximate one. The algorithm presented in previous section produces inner approximations.

In the Rocha and Cozman algorithm, given a query variable X_q and a credal network $\mathcal{N}$, a single run of the branch-and-bound algorithm computes the lower or upper *a posteriori* probability for a single state x^i_q of X_q, as in the hill-climbing algorithm explained in Section 5, but now the exact solution is obtained. The algorithm to find the lower or upper probability for a single state x^i_q of X_q uses a credal network $\mathcal{N}_0$ as the input, obtained from $\mathcal{N}$ by discarding variables that are not used to compute the inference by d-separation [12]. Let us suppose that we are interested in looking for $\max p(x^i_q | \mathbf{x}_E)$. The branch-and-bound technique requires a procedure to obtain a bound r (overestimation)

for the solution: r must produce an estimation $r(\mathcal{N})$ of $\max p(x_q^i|\mathbf{x}_E)$, verifying $r(\mathcal{N}) \geqslant \max p(x_q^i|\mathbf{x}_E)$. Rocha and Cozman use Tessem's A/R algorithm [26], an algorithm that produces outer bounds rather quickly. The A/R algorithm focuses on polytrees. Algorithm 2 shows the main steps of the branch-and-bound procedure.

Algorithm 2

Input: A credal network $\mathcal{N}_0$
Output: The value of $\overline{p} = \max p(x_q^i|\mathbf{x}_E)$
Initialize $\hat{p}$ with $-\infty$
if *the credal net $\mathcal{N}_0$ contains a single vertex* **then**
  Update $\hat{p}$ with $p(x_q^i|\mathbf{x}_E)$ if $p(x_q^i|\mathbf{x}_E) > \hat{p}$
end
else
  Using the k possibilities of one of the credal sets in $\mathcal{N}_0$, obtain a list of k credal networks $\{\mathcal{N}_{01}, \ldots, \mathcal{N}_{0k}\}$ from $\mathcal{N}_0$
  for each $\mathcal{N}_{0h}$ **do**
    if $r(\mathcal{N}_{0h}) > \hat{p}$ **then**
      Call recursively depth-first branch-and-bound over $\mathcal{N}_{0h}$
    end
  end
end
Take the last $\hat{p}$ as $\overline{p}$.

The algorithm can be explained as a search in a tree where the root node contains $\mathcal{N}_0$. The root node is divided into several simpler credal networks $\{\mathcal{N}_{01}, \ldots, \mathcal{N}_{0k}\}$. Each of these networks is obtained by taking one of the transparent variables $T_{\mathbf{y}_j}$ in $\mathcal{N}_0$ and producing as many networks as the number of states in $T_{\mathbf{y}_j}$, by fixing $T_{\mathbf{y}_j}$ to its different states (that is, we select one credal set in $\mathcal{N}_0$, and we produce as many networks as vertices in that credal set). Resulting networks are inserted as children of the root node in the search tree. The decomposition procedure is applied recursively. At each step, a transparent variable is expanded. In this way, a leaf node contains a Bayesian network, obtained by a particular selection of states in all the transparent variables (that is, a selection of vertices in all credal sets in the credal network). Rocha and Cozman apply the variable elimination algorithm to perform inference in the Bayesian network defined by the leaf.

We introduce the following modifications in our version of the branch-and-bound algorithm:

- We use the hill-climbing algorithm in Section 5 to initialize $\hat{p}$. This is a better solution because it is a fast algorithm and initialization with a good underestimation of $\max p(x_q^i|\mathbf{x}_E)$ can save a lot of search work, and therefore a lot of computation time in the branch-and-bound algorithm.
- We also use the variable elimination algorithm for inference in leaf nodes (Bayesian networks) but now we use the probability tree version [4] without using approximations. The use of probability trees in this stage does not greatly affect efficiency and it is specially good in later steps of the algorithm.

- The algorithm for overestimating the solution (computing $r(\mathcal{N}_0)$) in inner nodes of the search tree is also based on the variable elimination algorithm with probability trees. However, we now use the version that makes use of the $r_{\min}$ and $r_{\max}$ values explained in Section 4. In the algorithm, the probability trees corresponding with the initial conditional information are pruned using the method described in Section 4. Such a method is also used to prune the probability trees resulting from a combination or marginalization operation. This enables the probability trees to be maintained in a reasonable size, making propagation possible when the problem is hard. Another advantage of this algorithm is that it can be used in general Bayesian networks and not only in polytrees as this is a generic outer approximation algorithm.

7. Experiments

We have analyzed several kind of networks to better check the performance of the two proposed algorithms. The basic structure of the networks is the one used in [21] (see Fig. 1), but three different variants are considered, varying the number of states for its variables. All the tests are done using two vertices (extreme points) for each conditional credal set $H^{X_i|\mathbf{Y}=\mathbf{y}_j}$. The following combinations are considered: two states for every variable (BN2s); three states for all variables except for variables A, B and C (two states) (BNMixedRed); and three states for all variables except for variable C (two states) (BNMixed). We obtain 30 credal networks for each of the three variants. The numerical values for the extreme points are randomly generated. This random generation is made in the following way. We take a Bayesian network and transform it into a credal network by generating a given number l of random vertices for each conditional distribution $p(X_i|\mathbf{Y}=\mathbf{y})$. The generation of a random vertex from $p(X_i|\mathbf{Y}=\mathbf{y})$ is made by producing a random uniform number $r \in [-1.0, 1.0]$ for each $x_i \in \Omega_{X_i}$, and adding r to $p(x_i|\mathbf{Y}=\mathbf{y})$. If the new $p(x_i|\mathbf{Y}=\mathbf{y}) < 0.0$, then we change its sign. Finally, the values $p(X_i|\mathbf{Y}=\mathbf{y})$ are normalized to obtain the new vertex. The experiments use E as the query variable with no evidence. The potential number of vertices of the strong extension for the three kind of networks is 2^{11}, 2^{13} and 2^{18} (BN2s, BNMixedRed, and BNMixed, respectively).

We have made several tests to check critical issues:

- For the branch-and-bound algorithm: *maximum size of the probability trees* required during the search, the *size of the search tree*, and *computation time*. The maximum size of the probability trees is the size of the biggest tree used in the computations. These parameters are measured running the algorithm with several values for the threshold σ (used when pruning the probability trees). We have proved 64 values for σ: 32 ranging from 0.0 to 0.000002, 21 from 0.000002 to 0.00005 and 11 covering the values from 0.00005 to 0.01. It should be noted that this algorithm always obtains exact computations for all the values of σ. A small value for σ requires a low number of visited nodes in the search space, but it needs large probability trees. A large σ, however, requires a high number of visited nodes but smaller probability trees.
- For the hill-climbing algorithm: *computation time* and *root mean square error* for $p(x_q^i|\mathbf{x}_E)$.

Table 2
Root mean square error for $\min p(x_q^i|\mathbf{x}_E)$ and $\max p(x_q^i|\mathbf{x}_E)$ using the hill-climbing algorithm

	BN2s	BNMixedRed	BNMixed
$\min p(x_q^1|\mathbf{x}_E)$	1.4137E−4	7.342E−35	6.519E−4
$\max p(x_q^1|\mathbf{x}_E)$	1.2516E−4	1.989E−4	6.02E−4
$\min p(x_q^2|\mathbf{x}_E)$	1.3611E−4	1.56E−33	5.972E−4
$\max p(x_q^2|\mathbf{x}_E)$	1.5902E−4	1.944E−5	6.237E−4
$\min p(x_q^3|\mathbf{x}_E)$	–	7.288E−5	4.982E−4
$\max p(x_q^3|\mathbf{x}_E)$	–	6.740E−35	4.721E−4

The two algorithms are implemented in Java language (j2sdk 1.5) within the Elvira system [7]. The experiments have been run on a Pentium IV 3400 MHz computer, with 2 GBytes of Ram memory, and the Linux Fedora Core 3 operating system. In the experiments, the hill-climbing algorithm makes only three crosses in the join tree (the initial propagation from leaves towards the root, and then a crossing from the root towards the leaves and vice versa).

7.1. Hill-climbing algorithm performance

The performance when only the hill climbing algorithm from Section 5 is applied is shown in Table 2. This table shows the *root mean square error* of $\max p(x_q^i|\mathbf{x}_E)$ and $\min p(x_q^i|\mathbf{x}_E)$ for all the states of the query variable. In this case, the average running time for the three kinds of networks is 0.023, 0.2118 and 4.59 s (BN2s, BNMixedRed, and BNMixed, respectively). It can be observed that the hill-climbing algorithm produces a very good approximation in a short time.

7.2. Branch-and-bound algorithm performance

Using the branch-and-bound algorithm with the hill-climbing algorithm at the initialization step, we obtain the value of the different parameters in Figs. 12–14. The diagrams show the average values resulting from the 30 runs for each threshold σ and for each kind of network. A logarithmic scale (base 10) is used as the differences in the parameters for the different kinds of networks are too large. Fig. 12 shows the relation between the maximum size of the probability trees with respect to the threshold σ used for pruning. It is clear that the greater σ is, the smaller the probability trees obtained. The savings when pruning is used should also be observed, even with a very small threshold σ ($\sigma = 2\text{E}{-6}$).

Fig. 13 shows the number of visited nodes in the search tree of the branch-and-bound algorithm with respect to the threshold σ. In this case, the greater σ is, the greater the search required, as the approximation is poorer which implies that less branches are pruned and thus the solution to the problem is reached after a deeper search. In any case, the size of the search tree by the branch-and-bound is a small fraction of the potential number of vertices of the strong extension.

Fig. 14 shows the computation time with respect to the threshold σ. Time is quickly decreased until σ reaches 0.001, and then it is slowly increased. For $\sigma = 0$ (no pruning), we have the smallest search space, but the algorithm for overestimating $p(x_q^i|\mathbf{x}_E)$ (variable elimination with $r_{\min}$ and $r_{\max}$ values) uses large probability trees, requiring a lot of time.

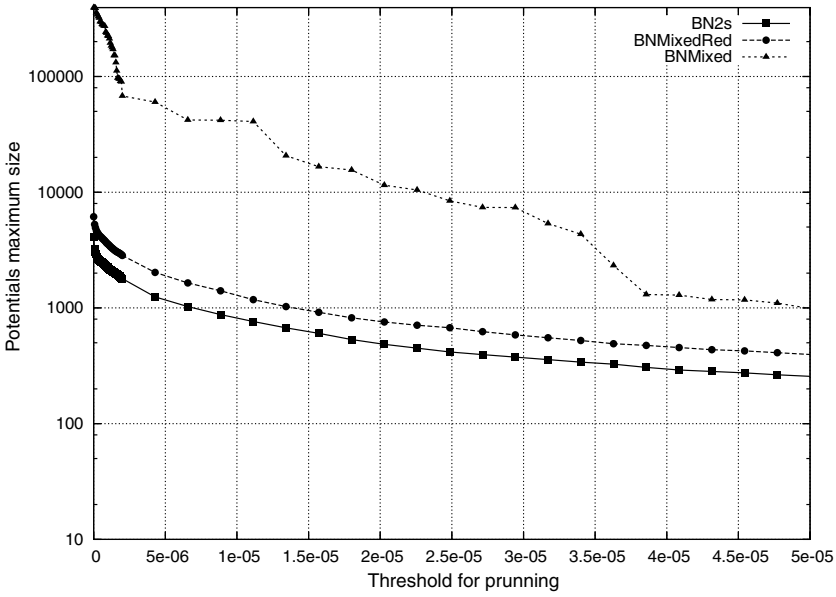

Fig. 12. Prob. tree maximum size – threshold σ.

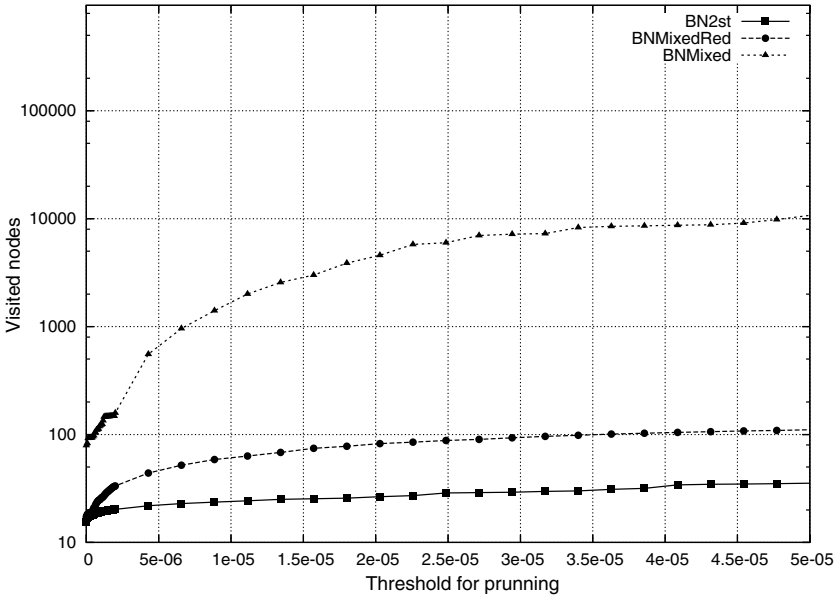

Fig. 13. Nodes visited during the search.

When σ is small (close to 5E−6), the search space is increased because the algorithm for overestimating $p(x_q^i|\mathbf{x}_E)$ produces worse bounds than before, but the running time is compensated with a decrease in the time employed by that algorithm (the probability trees are now smaller). When σ is large, the search space continues increasing but the reduction in

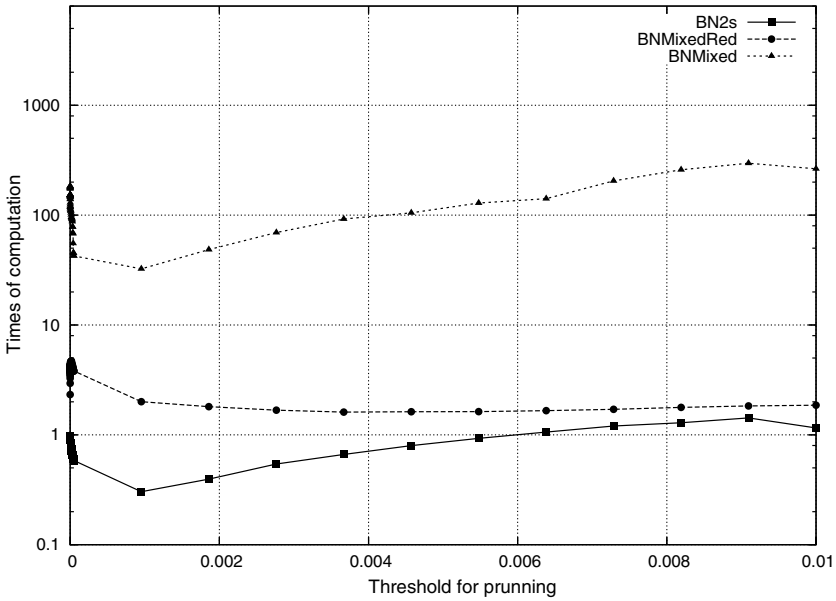

Fig. 14. Computation times.

Table 3
Average increase without using hill-climbing in initialization of $\hat{p}$

	BN2s	BNMixedRed	BNMixed
Max. pot. size (%)	11.25	9.31	10.42
Visited nodes (%)	83.66	82.93	12.84
Comput. time (%)	157.14	161.68	112.87

the size of the probability trees no longer compensates for that increase in the search space. The best performance is obtained when σ is around 5E−6.

In order to gain further insight into the performance of the hill-climbing algorithm in combination with the branch-and-bound algorithm, we have repeated the previous experiments, but now without using the hill-climbing algorithm to initialize $\hat{p}$ (the initial bound are $+\infty$ for $\min p(x_q^i|\mathbf{x}_E)$, and $-\infty$ for $\max p(x_q^i|\mathbf{x}_E)$). With this change, the performance of the branch-and-bound search is not so good. Table 3 includes the average percentages of increases for the maximum size of potentials, number of visited nodes, and computation times.

While the maximum size of potentials is not very different, the computation time is greatly increased; a worse initial bound requires a deeper search in order to reach the final solution. This also explains the increase in the number of visited nodes.

8. Conclusions

In this paper, we have presented two new algorithms for inference in credal networks. They are used to obtain probability intervals for the states of a query variable given a set

of observed variables $\mathbf{X}_E$. The first is an approximate algorithm that provides *inner* bounds close to the correct ones in a very short time (see Table 2). The second is based on Rocha and Cozman's branch-and-bound algorithm. This makes use of the proposed hill-climbing algorithm to initialize the starting point $\hat{p}$. This is a good decision as Table 3 demonstrates. The branch-and-bound algorithm uses the variable elimination algorithm with the $r_{\min}$ and $r_{\max}$ values to overestimate the intervals in the inner nodes of the search. In this way, it can be applied to any credal network structure (not only to polytrees). The variable elimination algorithm is controlled by a parameter σ. When the variable elimination requires more memory than that available in our computer, we apply large values of σ in order to reduce the size of the probability trees which enable propagation, although there will be an increase in the search space. In this way, the best performance of the branch-and-bound algorithm is reached with a trade-off between the parameter σ and the number of visited nodes.

In the future, we would like to prove the two algorithms in credal networks with a more complex structure. We would also like to prove alternative strategies in the hill-climbing algorithm (for example, increasing the number of iterations in the join tree). Another point is to study the effect of the order in which the transparent variables are expanded in the search tree on the running time and number of visited nodes.

Acknowledgement

This work has been supported by *Dirección General de Investigación, Ministerio de Educación y Ciencia (Spain)* under project TIN2004-06204-C03-02.

References

[1] S. Amarger, D. Dubois, H. Prade, Constraint propagation with imprecise conditional probabilities, in: B.D'. Ambrosio, Ph. Smets, P.P. Bonissone (Eds.), Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 1991, pp. 26–34.
[2] A. Cano, 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.
[3] A. Cano, S. Moral, Computing probability intervals with simulated annealing and probability trees, Journal of Applied Non-Classical Logics 12 (2) (2002) 151–171.
[4] A. Cano, S. Moral, Using probabilities trees to compute marginals with imprecise probabilities, International Journal of Approximate Reasoning 29 (2002) 1–46.
[5] A. Cano, S. Moral, A. Salmerón, Penniless propagation in join trees, International Journal of Intelligent Systems 15 (11) (2000) 1027–1059.
[6] J.E. Cano, S. Moral, J.F. Verdegay-López, Propagation of convex sets of probabilities in directed acyclic networks, in: B. Bouchon-Meunier et al. (Eds.), Uncertainty in Intelligent Systems, Elsevier, 1993, pp. 15–26.
[7] Elvira Consortium, Elvira: an environment for creating and using probabilistic graphical models, in: J.A. Gámez, A. Salmerón (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models, 2002, pp. 222–230.
[8] I. Couso, S. Moral, P. Walley, Examples of independence for imprecise probabilities, in: Proceedings of the First International Symposium on Imprecise Probabilities and their Applications (ISIPTA'99), 1999.
[9] F.G. Cozman, Robustness analysis of Bayesian networks with local convex sets of distributions, in: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, San Mateo, 1997.
[10] F.G. Cozman, Credal networks, Artificial Intelligence 120 (2000) 199–233.
[11] F.G. Cozman, Graphical models for imprecise probabilities, International Journal of Approximate Reasoning 39 (2005) 167–184.

[12] F.G. Cozman, Irrelevance and independence relations in quasi-Bayesian networks, in: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), Morgan & Kaufman Publishers, San Francisco, CA, 1998, pp. 89–96.
[13] C.P. de Campos, F.G. Cozman, Inference in credal networks using multilinear programming, in: Proceedings of the Second Starting AI Researcher Symposium, 2004, pp. 50–61.
[14] C.P. de Campos, F.G. Cozman, Computing lower and upper expectations under epistemic independence, in: Proceedings of the Forth International Symposium on Imprecise Probabilities and their Applications (ISIPTA'05), Pittsburgh, PA, USA, 2005, pp. 78–87.
[15] K.W. Fertig, J.S. Breese, Interval influence diagrams, in: M. Henrion, R.D. Shacter, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 5, North-Holland, Amsterdam, 1990, pp. 149–161.
[16] J. Kohlas, Information Algebras: Generic Structures for InferenceDiscrete Mathematics and Theoretical Computer Science, Springer-Verlag, 2003.
[17] S. Kullback, R.A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1951) 76–86.
[18] S.L. Lauritzen, D.J. Spiegelhalter, Local computation with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society, Ser. B 50 (1988) 157–224.
[19] S. Moral, A. Cano, Strong conditional independence for credal sets, Annals of Mathematics and Artificial Intelligence 35 (2002) 295–321.
[20] J. Pearl, Probabilistic Reasoning with Intelligent Systems, Morgan & Kaufman, San Mateo, 1988.
[21] J.C.F. Rocha, F.G. Cozman, Inference in credal networks with branch-and-bound algorithms, in: Proceedings of the Third International Symposium on Imprecise Probabilities and their Applications (ISIPTA'03), 2003, pp. 482–495.
[22] J.C.F. Rocha, F.G. Cozman, C.P. de Campos, Inference in polytrees with sets of probabilities, in: Christopher Meek, Uffe Kjærulff (Eds.), Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 2003, pp. 217–224.
[23] J.C.F. Rocha, F.G. Cozman, Inference with separately specified sets of probabilities in credal networks, in: A. Darwiche, N. Friedman (Eds.), Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 2002.
[24] P.P. Shenoy, Binary join trees, in: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96), Portland, Oregon, 1996, pp. 492–499.
[25] P.P. Shenoy, G. Shafer, Axioms for probability and belief-function propagation, in: Shachter et al. (Eds.), Uncertainty in Artificial Intelligence, vol. 4, North-Holland, 1990, pp. 169–198.
[26] B. Tessen, Interval probability propagation, International Journal of Approximate Reasoning 7 (1992) 95–120.
[27] H. Thöne, U. Güntzer, W. Kießling, Towards precision of probabilistic bounds propagation, in: Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence, 1992, pp. 315–322.
[28] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.