ELSEVIER International Journal of Approximate Reasoning 29 (2002) 1–46

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Using probability trees to compute marginals with imprecise probabilities

Andrés Cano *, Serafín Moral

Department of Computer Science and Artificial Intelligence, E.T.S. Ingeniería Informática, University of Granada, Avda. de Andalucía 38., 18071 Granada, Spain

Received 1 May 2000; accepted 1 April 2001

Abstract

This paper presents an approximate algorithm to obtain a posteriori intervals of probability, when available information is also given with intervals. The algorithm uses probability trees as a means of representing and computing with the convex sets of probabilities associated to the intervals. © 2002 Elsevier Science Inc. All rights reserved.

Keywords: Dependence graphs; Interval probabilities; Propagation algorithms; Probability trees

1. Introduction

Bayesian networks are graphical structures which are used to represent joint probability distributions efficiently. The network structure encodes the independence relations among the variables. A variety of different tasks can be performed on Bayesian networks. One of the most common is the computation of posterior marginals given that the value of some of the variables is known. This task is called *probability propagation*. In this way, Bayesian networks are a powerful tool for building probabilistic Expert systems.

One of the main problems faced when building Bayesian networks is the introduction of a large number of initial exact probabilities. It can be very difficult for experts to give such a number of precise probabilities. Very often, an expert is

* Corresponding author.
E-mail addresses: acu@decsai.ugr.es (A. Cano), smc@decsai.ugr.es (S. Moral).

0888-613X/02/$ - see front matter © 2002 Elsevier Science Inc. All rights reserved.
PII: S0888-613X(01)00046-9

more comfortable giving an interval of probability rather than a precise probability. Even if we use a learning algorithm to obtain probabilities, we may only have small samples for certain configurations of variables in a distribution. Therefore, it may also be more appropriate to estimate some kind of imprecise probabilities in this case. For an expert, one of the most natural ways of giving imprecise probabilities is by means of an interval of probabilities.

In general, the use of imprecise probability models is useful in many situations. We can highlight the following situations [56]:

- When we have little information to evaluate probabilities [52,53,55].
- When available information is not specific enough. For example, when we take out balls from an urn with 10 balls, where five are red and five are white or black (but we do not know the exact number of each one) [21,31,42].
- In robust Bayesian inference, to model uncertainty about a prior distribution [3,22].
- To model the conflict between several sources of information [35,51].

There are various mathematical models for imprecise probability [52,56,57]: comparative probability orderings [25,26], possibility measures [23,61], fuzzy measures [28,47,58], belief functions [42,46], Choquet capacities [16,30], interval probabilities [6,59,60], coherent lower previsions [52,57], convex sets of probabilities [7,15,52,57], sets of desirable gambles [52,57]. Out of all these models, we think that convex sets of probabilities are the most suitable for calculating with and representing imprecise probabilities. We think that because there is a specific interpretation of numeric values [52,54], they are powerful enough to represent the result of basic operations within the model without having to make approximations that cause loss of information, as in interval probabilities [50]. Convex sets are a more general tool for representing unknown probabilities than intervals: there is always a convex set associated with a system of probabilistic intervals, but given a convex set there is not always a proper representation by using intervals. However, interval probabilities are the most natural way in which imprecise probabilities are present in practice. In this paper, therefore, we will assume that initial probability distributions are given with interval probabilities, but computations are carried out by considering their associated convex sets.

Some authors have considered the propagation of probabilistic intervals in graphical structures [1,24,48,49]. However in the procedures proposed, there is no guarantee that the calculated intervals are always the same as those obtained by using a global computation. 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. This is the approach followed by Cano et al. [15]. In this paper, they assume that there is a convex set of conditional probabilities for each configuration of parent variables in the dependence graph. They raise the problem of calculating imprecise probabilities as a problem of propagation with convex sets of probability

distributions. They give a model to compute such probabilities using local computations. However, working with convex sets may be very inefficient: if we have n variables and each variable, X_i, has a convex set with h_i extreme points as conditional information, the propagation of the convex set is of the order $O(K \prod_{i=1}^{n} h_i)$, where K is the complexity of carrying out a simple probabilistic propagation. This is so, because convex sets propagation is equivalent to the propagation of all the global probabilities that can be obtained by choosing an exact conditional probability in each of the convex sets.

Another solution to the problem of propagating the convex sets associated to the intervals, is by using an approximate algorithm using combinatorial optimization techniques such as simulated annealing [9,10], genetic algorithms [11], and gradient techniques [18,19].

Probability trees [13,41] can be used to represent probability potentials. In [13], the authors have used probability trees to propagate probabilities efficiently in Bayesian networks using a join tree when resources (memory and time) are limited. These algorithms always obtain a result, but when resources are low then results are approximated. Depending on the available time (or memory), the results will have a greater or smaller error.

In this paper, we propose the use of probability trees to represent the convex sets associated to the intervals. Probability trees are then used with a propagation algorithm to calculate an a posteriori convex set for a given variable of interest. From this a posteriori convex set, we can obtain probability intervals for each case of this variable.

This paper is divided into six sections. In Section 2 we describe the problem of the propagation of probabilities in Bayesian networks and how it can be solved by using the variable elimination algorithm [20,34,62]. Section 3 studies the use of probability trees to represent potentials compactly and how they can represent context specific independences; we also study how to build and operate with probability trees. In Section 4 we present basic notions about convex sets of probabilities and their relationships with probability intervals. Section 5 shows how to use probability trees in an approximate method of propagation with convex sets, a technique to eliminate non-extreme points in the a posteriori convex sets and a method to limit the error of the previous approximate method. Section 6 describes the experiments we have carried out in order to prove the approximate algorithm. Finally, Section 7 presents the conclusions and future lines of research.

2. Probability propagation in Bayesian networks

A *Bayesian network* is a directed acyclic graph where each node represents a random variable, and the topology of the graph shows the independence relations between variables (see Fig. 1), according to the d-separation criterion [36].

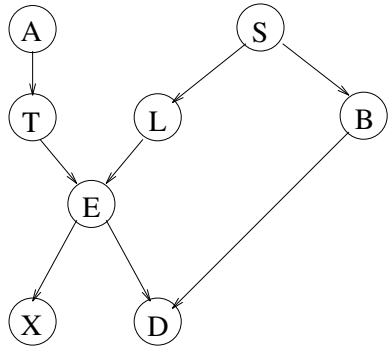


Fig. 1. A directed acyclic graph.

Let $X = \{X_1, \ldots, X_n\}$ be the set of variables in the network. Let us assume that each variable X_i takes values on a finite set U_i. For any set U, $|U|$ represents the number of elements it contains. If I is a set of indices, we will write X_I for the set $\{X_i | i \in I\}$. $N = \{1, \ldots, n\}$ will denote the set of indices of all the variables in the network; thus, $X_N = X$. The Cartesian product $\prod_{i \in I} U_i$ will be denoted by U_I. Given $x \in U_I$ and $J \subseteq I$, x_J will denote the element of U_J obtained from x dropping the coordinates which are not in J. Following Shenoy and Shafer's [43,44] general terminology, a mapping from a set U_I on [0,1] will be called a *valuation* defined on U_I. In the probabilistic case, valuations are known as *potentials*. Suppose V is the set of all our initial valuations (conditional distributions and observations), and $s(h)$ denotes the set of indices of variables for which h is defined. Given two valuations, h_1 and h_2, defined on U_I and U_J, then the combination of h_1 and h_2 will be a valuation $h_1 \otimes h_2$ defined on $U_{I \cup J}$ by means of pointwise multiplication:

$$h_1 \otimes h_2(u) = h_1(u^{\downarrow I}) h_2(u^{\downarrow J}), \tag{1}$$

where $u^{\downarrow I}$ is the element obtained from u by dropping the coordinates which are not in I.

If h is a valuation defined on U_I and $J \subseteq I$, then the marginalization of h in J, $h^{\downarrow J}$ is calculated by addition:

$$h^{\downarrow J}(u) = \sum_{v^{\downarrow J} = u} h(v). \tag{2}$$

If $F(i)$ are the parents of X_i in the graph, then we have a conditional probability represented by a valuation, p_i, defined on $U_{i \cup F(i)}$ and such that $p_i^{\downarrow F(i)} = h_0$, where h_0 is the identity mapping.

Given the independences encoded by the graph and a probability distribution p_i for each node conditioned on its parents, then there is a unique joint probability given by:

$$p(x) = \prod_{i \in N} p_i(x_i|x_{F(i)}) \quad \forall x \in U_N. \tag{3}$$

An *observation* is the knowledge about the exact value $X_i = e_i$ of a variable. The set of observations will be denoted by e, and called the *evidence set*. E will be the set of indices of the variables observed. Every observation, $X_i = e_i$, is represented by means of a valuation which is a Dirac function defined on U_i as $\delta_i(x_i; e_i) = 1$ if $e_i = x_i$, $x_i \in U_i$, and $\delta_i(x_i; e_i) = 0$ if $e_i \neq x_i$.

The aim of probability propagation is to calculate the a posteriori probability function $p(x'_k|e)$, for every $x'_k \in U_k$, where $k \in \{1, \ldots, n\} - E$.

If we have an evidence set e, then global conditional probability verifies that

$$p(x'_k|e) \propto \sum_{x_k = x'_k} \left(\prod_i p_i(x_i|x_{F(i)}) \prod_{e_i \in e} \delta_i(x_i; e_i) \right). \tag{4}$$

In fact, we have the following equality:

$$p(x'_k \cap e) = \sum_{x_k = x'_k} \left(\prod_i p_i(x_i|x_{F(i)}) \prod_{e_i \in e} \delta_i(x_i; e_i) \right). \tag{5}$$

The vector of values $(p(x'_k \cap e) \,|\, x'_k \in U_k)$ will be denoted as R_k.

Propagation algorithms calculate the a posteriori probability functions $p(x'_k|e)$ for each variable X_k by making a local computation. The variable elimination algorithm is one of the most popular algorithms to obtain a posteriori information using local computations. This algorithm was independently proposed by Shafer and Shenoy [43,44], Zhang and Poole [62] and Dechter [20]. The algorithm is as follows:

Algorithm 1 (*Variables elimination*).

Input: A variable X_k and a set of valuations V

Output: An a posteriori valuation R_k

1. Repeat the following step until all $j \neq k$ are deleted:
   Let $j \in \{1, \ldots, n\}$, $j \neq k$. Consider $J = \{h_i \in V : j \in s(h_i)\}$ and $L = s(\otimes J) - \{j\}$. Then V is transformed into
   $$V - J \cup \left\{ \left(\otimes_{h \in J} h \right)^{\downarrow L} \right\} \tag{6}$$
2. Then, $R_k = \otimes h | h \in V$.

3. Probability trees

A probability distribution has traditionally been represented as a table, using a vector of real numbers. This representation is exponential in the number of parameters (number of variables in the distribution).

Definition 1 (*Probability tree*). A *probability tree* $\mathscr{T}$ [5,8,13,27,32,37,41,45,62] for a set of variables X_I is a directed labeled tree, where each internal node represents a variable $X_i \in X_I$ and each leaf node represents a real number $r \in \mathbb{R}$. Each internal node will have as many outgoing arcs as possible values as the variable it represents has. We define the *size* of a tree by the number of leaves it has.

A probability tree can be used to represent a probability distribution, or in general any valuation for the set of variables X_I. For example, the probability tree in Fig. 2 represents the conditional distribution of such a figure.

A probability tree $\mathscr{T}$ on variables X_I represents the valuation $h : U_I \to \mathbb{R}$ if for each $x_I \in U_I$ the value $h(x_I)$ is the number stored in the leaf node that is obtained starting at the root node and selecting for each inner node labeled with X_i the child corresponding to coordinate x_i. The valuation represented by tree $\mathscr{T}$ will be denoted by $h_{\mathscr{T}}(x_I)$.

Probability trees are particularly useful when there are regularities in probability distributions. These regularities can be used to make a compact representation of distributions. Regularities in distributions generate *asymmetric independences* also known as *context-specific independences*.

Definition 2 (*Context-specific independence* [5]). Let X_K be a set of variables. A *context* on X_K is an assignment of one value to each variable in X_K, that is to say a configuration x_K of variables in X_K. Two contexts are *incompatible* if there is a variable that has been assigned different values in the contexts. Otherwise they are *compatible*.

Let X_I, X_J, X_L and X_K be four disjoint sets of variables. X_I and X_J are *independent given* X_L *in context* $X_K = x_K$, noted as $I_c(X_I; X_J | X_L; X_K = x_K)$, if

$$P(X_I | X_L, X_J, X_K = x_K) = P(X_I | X_L, X_K = x_K)$$

X1	X2	X3	P(X1 |X2 X3)
1	1	1	.3
1	1	2	.5
1	2	1	.4
1	2	2	.4
2	1	1	.7
2	1	2	.5
2	2	1	.6
2	2	2	.6

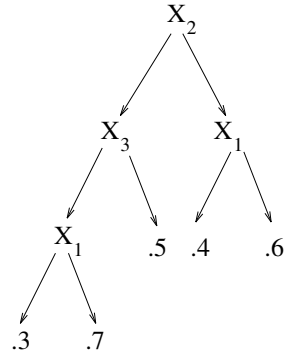


Fig. 2. A conditional probability distribution and an associated probability tree.

whenever $P(X_J, X_L, X_K = x_K) > 0$. When X_L is empty, it can be said that X_I and X_J are *independent in context* $X_K = x_K$.

As an example, in Fig. 2 we can see that $I_c(X_1; X_3|X_2 = 2)$ but it is not true that $I_c(X_1; X_3|X_2 = 1)$. Probability trees exploit these independences making the size of the representation more compact.

If $\mathscr{T}$ is a probability tree representing a valuation h on the set of variables X_I, the set of variables that label inner nodes in $\mathscr{T}$ will be denoted $\mathrm{Var}(\mathscr{T})$. This set must obey $\mathrm{Var}(\mathscr{T}) \subseteq X_I$.

Definition 3 (*Restricted tree*). Given a probability tree $\mathscr{T}$, representing a valuation h on the set of variables X_I, a set of variables $X_J \subseteq X_I$, and a configuration $x_J \in U_J$, $\mathscr{T}^{R(X_J=x_J)}$ denotes the restriction of $\mathscr{T}$ to the values of x_J of the variables in X_J, i.e. the tree obtained by substituting in $\mathscr{T}$ every node corresponding to variables X_k, $k \in J$ by subtrees $\mathscr{T}_k$ children of X_k corresponding to $X_k = x_k$.

For example, if $\mathscr{T}$ is the tree in Fig. 2, then $\mathscr{T}^{R(X_1=1)}$ is the probability tree shown in Fig. 3.

Each node of the tree, and in particular its leaves, is characterized by a configuration of values $X_J = x_J$, where $J \subseteq I$. The variables in X_J are the variables in the path from the root to the node and the values of the variables correspond to the branches we have to follow in order to arrive at the node.

3.1. Constructing probability trees

Cano and Moral [8] present a methodology to build a probability tree from a probability table. This methodology is used to represent conditional distribution of probabilities using trees, but it can be used to represent any

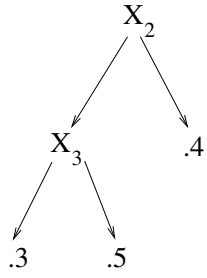


Fig. 3. A restricted probability tree $\mathscr{T}^{R(X_1=1)}$ being $\mathscr{T}$ the tree in Fig. 2.

valuation, in particular a convex set of probabilities. Cano and Moral [8] also propose a way of approximating potentials with limited size.

Given a potential ϕ over a set of variables X_I, the goal is to get an associated tree, $\mathscr{T}_\phi$, for ϕ. If the size of the tree is greater than a given maximum, we shall try to get the best approximation to ϕ by means of a tree $\mathscr{T}$ with a size lower than that maximum. The approximation should contain the same normalization factor, i.e. $\sum_{x\in U_I}\mathscr{T}(x)=\sum_{x\in U_I}\phi(x)$.

Let $p_{\mathscr{T}}$ and p_ϕ be the proportional probability distributions to $\mathscr{T}$ and ϕ, respectively, then the *distance* from the tree $\mathscr{T}$ to a potential ϕ is measured by Kullback–Leibler's cross-entropy [33] of $p_{\mathscr{T}}$ and p_ϕ:

$$D(\phi,\mathscr{T})=D(p_\phi,p_{\mathscr{T}})=+\sum_{x_I\in U_I}p_\phi(x_I)\log\frac{p_\phi(x_I)}{p_{\mathscr{T}}(x_I)}. \tag{7}$$

Proposition 1 [8]. *Let ϕ be a potential over a set of variables X_I, and $J\subseteq I$. If a tree $\mathscr{T}$ is such that every leaf $\mathscr{T}^{R(X_J=x)}$ contains the value $\sum_{y_J=x}\phi(y)/|U_{I-J}|$, then $\mathscr{T}$ minimizes the cross-entropy between any tree with the same normalization factor and the same structure* [1] *as $\mathscr{T}$ and ϕ.*

According to this proposition, given any structure $\mathscr{T}^*$, the best approximation to a potential ϕ with that structure is achieved by putting in each leaf the average of the values of ϕ for the configurations of the variables leading to that leaf. The problem is that, in general, the construction of an optimal structure $\mathscr{T}^*$ is a combinatorial problem, which is difficult to solve.

The Cano and Moral methodology [8] to build a probability tree is based on the methods for inducing classification rules from examples. One of these methods is Quinlan's ID3 algorithm [38], that builds a *decision tree* from a set of examples.

The process of constructing a tree consists of choosing, given a tree $\mathscr{T}$ with structure $\mathscr{T}^*$, the branch to be expanded and the variable to be placed in the new node. This selection is made in such a way that the distance to the potential is minimized. If a leaf node in $\mathscr{T}^*$ is defined by the values $X_J=x$, the structure obtained from $\mathscr{T}^*$ by expanding the leaf defined by $X_J=x$ with the variable X_k is denoted by $\mathscr{T}^*(X_J=x,X_k)$, and the corresponding tree by $\mathscr{T}(X_J=x,X_k)$. At each moment, both the branch and the variable are selected in order to minimize the distance to ϕ, i.e.

[1] We say that two trees $\mathscr{T}_1$ and $\mathscr{T}_2$ have the same structure, $\mathscr{T}^*$, if they contain the same inner nodes and differ, at most, in the numbers placed in the leaves.

$$D(\mathscr{T}(X_J = x, X_k), \phi) = \min\{D(\mathscr{T}(X_{J'} = x', X_{k'}), \phi)\}, \tag{8}$$

where $X_{J'} = x'$ is a leaf of $\mathscr{T}^*$ and $k' \in I - J'$.

The following proposition [8] shows an easy way of computing that minimum.

Proposition 2. *The pair* $(X_J = x, X_k)$ *minimizing expression* (8) *is the one which maximizes the measure of information*

$$\mathrm{Inf}(X_J = x, X_k|\phi) = S_{X_J=x} \cdot (\log|U_k| - \log S_{X_J=x}) - E[X_k|X_J = x], \tag{9}$$

where

$$S_{X_J=x} = \sum_{z_J=x} \phi(z),$$

$$E[X_k|X_J = x] = -\sum_{y\in U_k} \phi^{\downarrow k}(y|X_J = x) \log \phi^{\downarrow k}(y|X_J = x),$$

$$\phi^{\downarrow k}(y|X_J = x) = \sum_{\substack{z_J=x \\ z_J=y}} \phi(x_I).$$

The measure of information $\mathrm{Inf}(X_J = x, X_k|\phi)$ gives the distance from a tree $\mathscr{T}$ to a potential ϕ before and after expanding the branch $X_J = x$ with variable X_k. The proposition above means that we must select branches leading to configurations with high probability, and variables with small entropy. The reason is that by expanding in this way, we shall obtain leaves with different and important probability values. This is very intuitive, since we are interested in representing only different values; very similar values can be represented by just one, corresponding to their average.

With this, one procedure to construct an *exact* tree is to select nodes maximizing function Inf. The procedure would finish when, for every branch $X_J = x$, the values of ϕ are uniform, i.e. $\phi(y) = \phi(y')$ for all $y, y' \in U_I$ such that $y_J = y'_J = x$. The idea therefore is to include nodes until no new information is provided by adding new nodes.

Cano and Moral [8] propose different alternatives for constructing an *approximate* tree. One of the alternatives consists in adding nodes until an exact representation is obtained or a maximum number of nodes is reached.

Another option is to construct the entire tree and bound it afterwards. If $\mathscr{T}$ is such a tree, a *bounding* consists of selecting a node such that all its children are leaves and replacing it and its children by one node containing the average of the values of the leaf nodes being removed. We have two ways of performing a bounding:

1. Remove nodes while a maximum is exceeded. The selection of a node will be determined by that pair $(X_J = x, X_k)$ minimizing the measure $\mathrm{Inf}(X_J = x, X_k|\phi)$, i.e. the pair minimizing the increment of the distance to potential ϕ.

2. Remove nodes determined by pairs $(X_J = x, X_k)$ such that

$$\mathrm{Inf}(X_J = x, X_k|\phi) \leqslant \Delta, \tag{10}$$

where Δ is a threshold for the increment of the distance. The bounding would finish when there are no more pairs verifying condition (10).

3.2. Operations with probability trees

Propagation algorithms require three operations over potentials: *restriction*, *combination* and *marginalization*. In this section we describe the algorithms proposed by Cano and Moral [12] for performing these operations. Kozlov and Koller [32] and Salmerón et al. [13] also provide procedures to carry out the basic operations on trees, but they take the structure on one of the trees as a basis and start branching it according to the structure of the other tree. Here the structures of both trees will be mixed. This is more appropriate if we carry out an approximate computation.

The *Restriction* operation is trivial, and it has been already described in Definition 3. We shall therefore concentrate on *combination* and *marginalization*:

Given two trees $\mathscr{T}_1$ and $\mathscr{T}_2$ associated to potentials ϕ_1 and ϕ_2, respectively, the following algorithm computes a tree associated to $\phi = \phi_1 \cdot \phi_2$ (combination).

Algorithm 2 ($COMBINE(\mathscr{T}_1, \mathscr{T}_2)$).

1. Start with an empty list $\mathscr{L}_c$.
2. Create a tree node $\mathscr{T}_r$ initially with no label.
3. Select a variable $X_i \in \mathrm{Var}(\mathscr{T}_1) \cup \mathrm{Var}(\mathscr{T}_2)$.
4. Insert node $\{\{\mathscr{T}_1, \mathscr{T}_2\}, \mathscr{T}_r, X_i\}$ into list $\mathscr{L}_c$.
5. While $\mathscr{L}_c$ is not empty,
   (a) Remove a node $\{\{\mathscr{T}_i, \mathscr{T}_j\}, \mathscr{T}_r, X_k\}$ from $\mathscr{L}_c$ and make X_k the label of $\mathscr{T}_r$.
   (b) For each $x \in U_k$,
      i. Create a tree node $\mathscr{T}_h$ initially with no label and make it a child of $\mathscr{T}_r$.
      ii. Let L_i and L_j be the labels of the root nodes of $\mathscr{T}_i^{R(X_k=x)}$ and $\mathscr{T}_j^{R(X_k=x)}$.
      iii. If L_i and L_j are numbers, let $L = L_i \cdot L_j$. Make L the label of T_h.
      iv. Otherwise,
         A. Select a variable $X_l \in \mathrm{Var}(\mathscr{T}_i^{R(X_k=x)}) \cup \mathrm{Var}(\mathscr{T}_j^{R(X_k=x)})$.
         B. Insert node $\{\{\mathscr{T}_i^{R(X_k=x)}, \mathscr{T}_j^{R(X_k=x)}\}, \mathscr{T}_h, X_l\}$ into list $\mathscr{L}_c$.
6. Return $\mathscr{T}_r$.

We will denote the combination of trees by the symbol $\otimes$. With this, the algorithm above returns a tree $\mathscr{T}_r = \mathscr{T}_1 \otimes \mathscr{T}_2$.

Given a tree $\mathscr{T}$ associated to a potential ϕ defined over a set of variables X_I, the following algorithm computes a tree associated with $\phi^{\downarrow(I-\{i\})}$, with $i \in I$. That is to say, it removes variable X_i from $\mathscr{T}$.

Algorithm 3 ($MARGINALIZE(\mathscr{T}, X_i)$).
1. Let L be the label of the root node of $\mathscr{T}$.
2. If L is a number,
   (a) Create a node $\mathscr{T}_r$ with label $L \cdot |U_i|$.
3. Otherwise, let $X_k = L$.
   (a) If $X_k = X_i$, then $\mathscr{T}_r = ADD\_CHILD(\mathscr{T}, X_i)$.
   (b) Otherwise,
      i. Create a node $\mathscr{T}_r$ with label L.
      ii. For each $x \in U_k$
         A. Make $\mathscr{T}_h = MARGINALIZE(\mathscr{T}^{R(X_k=x)}, X_i)$ the next child of $\mathscr{T}_r$.
4. Return $\mathscr{T}_r$.

This algorithm uses procedure ADD_CHILD($\mathscr{T}, X_i$), which replaces node X_i, by the addition of the subtrees that are children of X_i. X_i must be the root of $\mathscr{T}$. The procedure is as follows:

Algorithm 4 ($ADD\_CHILD(\mathscr{T}, X_k)$).
1. Start with an empty list $\mathscr{L}_m$.
2. Create a tree node $\mathscr{T}_r$ initially without label.
3. Let $L_1, \ldots, L_j$ be the labels of the roots of $\mathscr{T}^{R(X_k=x_1)}, \ldots, \mathscr{T}^{R(X_k=x_j)}$, with $j = |U_k|$.
4. If $L_1, \ldots, L_j$ are numbers, make $L = L_1 + \cdots + L_j$ the label of $\mathscr{T}_r$.
5. Otherwise,
   (a) Select a variable $X_i \in \mathrm{Var}(\mathscr{T}^{R(X_k=x_1)}) \cup \cdots \cup \mathrm{Var}(\mathscr{T}^{R(X_k=x_j)})$.
   (b) Insert node $\{\{\mathscr{T}^{R(X_k=x_1)}, \ldots, \mathscr{T}^{R(X_k=x_j)}\}, \mathscr{T}_r, X_i\}$ into list $\mathscr{L}_m$.
   (c) While $\mathscr{L}_m$ is not empty,
      i. Remove a node $\{\{\mathscr{T}_1, \ldots, \mathscr{T}_h\}, \mathscr{T}_s, X_l\}$ from $\mathscr{L}_m$.
      ii. Make X_l the label of $\mathscr{T}_s$.
      iii. For each $x \in U_l$,
         A. Create a node $\mathscr{T}_t$ initially without label, and make it a child of $\mathscr{T}_s$.
         B. Let $L_1, \ldots, L_h$ be the labels of the roots of $\mathscr{T}_1^{R(X_l=x)}, \ldots, \mathscr{T}_h^{R(X_l=x)}$.
         C. If $L_1, \ldots, L_h$ are numbers, make $L = L_1 + \cdots + L_h$ the label of $\mathscr{T}_t$.
         D. Otherwise
            • Select a variable $X_m \in \mathrm{Var}(\mathscr{T}_1^{R(X_l=x)}) \cup \cdots \cup \mathrm{Var}(\mathscr{T}_h^{R(X_l=x)})$.
            • Add node $\{\{\mathscr{T}_1^{R(X_l=x)}, \ldots, \mathscr{T}_h^{R(X_l=x)}\}, \mathscr{T}_t, X_m\}$ to $\mathscr{L}_m$.
6. Return $\mathscr{T}_r$.

In the algorithm above, we have used two lists, $\mathscr{L}_c$ for combination and $\mathscr{L}_m$ for addition of trees in marginalization. Each node $\{\{\mathscr{T}_1, \mathscr{T}_2\}, \mathscr{T}, X_i\}$ of $\mathscr{L}_c$, represents two subtrees that must be combined, storing the result in $\mathscr{T}$ and placing variable X_i as the root label. Analogously, each node $\{\{\mathscr{T}_1, \ldots, \mathscr{T}_j\}, \mathscr{T}, X_i\}$ of $\mathscr{L}_m$ contains j subtrees that must be added, storing the result in $\mathscr{T}$ and labeling the root node with variable X_i.

The operations described above are exact, i.e. they do not limit the size of the resulting tree. However, for exact computations a recursive algorithm is simpler and has a similar complexity. This way of organizing computations in which we keep a list of remaining computations that are scattered in different parts of the result tree, is more appropriate for approximate computations. Different procedures are obtained according to the selected variable to be placed as root of the subtrees combination or addition, and according to the way we extract elements from lists $\mathscr{L}_c$ or $\mathscr{L}_m$. The idea consists in placing the most informative variables at the top of the tree, or making the most important computations first. This is achieved if at every step the tree is expanded by the branches minimizing the distance to the resulting potential.

To determine the variable to be placed in the root of the result tree, all the variables of the trees we are operating with could be considered. But this is very time-consuming and in our experiments we have only evaluated the roots of each one of the combined trees. We have assumed that if these trees have been correctly built, then these are really the most informative variables in each one of them.

In the case of constructing a tree from a probability table, we have computed a measure of information for each possible variable X_k that can be put in each possible branch $X_J = x$. This information was calculated in an exact way, because we had the exact potential (the probability table). We do not now have the exact potential when we build the tree, the result of combining two trees or marginalizing a variable in a tree. The exact potential is just what we are computing. So, the best we can do is to take an estimation of the information value in the following way [8]:

1. *Combination*. Let ϕ_1 and ϕ_2 be the potentials to be combined. The information value of a variable X_k in a branch $X_J = x$ is measured as:

$$\mathrm{Inf}'(X_J = x, X_k|\phi_1 \cdot \phi_2) = \mathrm{Inf}\left(X_J = x, X_k|\phi_1^{R(X_J=x)^{\downarrow k}}\right) \times \mathrm{Inf}\left(X_J = x, X_k|\phi_2^{R(X_J=x)^{\downarrow k}}\right). \tag{11}$$

2. *Marginalization*. Let ϕ be the potential to be marginalized over $I - \{i\}$. The information value of a variable X_k in a branch $X_J = x$ is measured as:

$$\mathrm{Inf}'\left(X_J = x, X_k|\phi^{\downarrow I-\{i\}}\right) = \mathrm{Inf}\left(X_J = x, X_k|\phi^{R(X_J=x)^{\downarrow k}}\right). \tag{12}$$

According to this, we have two alternatives to obtain approximate results for a threshold of a given size:

- Once the threshold has been surpassed, each of the remaining branches (those determined by the elements of $\mathscr{L}_c$ or $\mathscr{L}_m$) are approximated by carrying out the corresponding operation in the average of all the values corresponding to that branch. For example, in the case of combination, in addition to the condition in step 5 of Algorithm 4, we must also check that

the maximum size has not been reached. When the maximum size is reached, list $\mathscr{L}_c$ contains several elements. Each one of these elements represents two trees that must be combined, putting the result at a leaf of the output tree. To do this, we approximate each one of these trees by its average value, and we put the product of such values in the corresponding leaf of the output tree. The case of marginalization is solved in the same way, but now product is replaced by addition.

- Generate the entire tree and bound it afterwards, in the same way as when constructing a new tree.

4. Basic notions about convex set of probability distributions

Let us assume that we have a population Ω and an n-dimensional variable $(X_1, X_2, \ldots, X_n)$ defined on Ω, and such that each X_i takes its values on a finite set U_i. In the previous section, we saw that the information about the variables in I was a conditional probability distribution of a node giving its parents in the graph. A piece of information relating the variables in I will now be a closed, convex set, H, of mappings:

$$p : U_I \to \mathbb{R}_0^+$$

with a finite set of extreme points. Every mapping is given by the vector of values $(p(u))_{u \in U_I}$. For example, Fig. 4 shows graphically, with the use of a system of triangular coordinates, a convex set for variable X with three possible values (u_1, u_2, u_3). This convex set has four extreme points (p_1, p_2, p_3, p_4).

If H is a convex set of mappings on U_I, with extreme points, $\mathrm{Ext}(H) = \{p_1, \ldots, p_k\}$, and $J \subseteq I$ then the *marginalization* of H to J is the convex set given by

$$H^{\downarrow J} = \mathrm{CH}\{p_1^{\downarrow J}, \ldots, p_k^{\downarrow J}\}, \tag{13}$$

where CH stands for the convex hull.

If H is a convex set of mappings in U_I, and H' is a convex set in U_J, with $\mathrm{Ext}(H) = \{p_1, \ldots, p_k\}$, $\mathrm{Ext}(H') = \{p'_1, \ldots, p'_l\}$. Then the *combination* of H and H' will be a convex set of mappings in $U_{I \cup J}$, $H \otimes H'$ given by

$$H \otimes H' = \mathrm{CH}\{p_1 \cdot p'_1, \ldots, p_1 \cdot p'_l, \ldots, p_k \cdot p'_1, \ldots, p_k \cdot p'_l\}. \tag{14}$$

If H is a convex set of mappings in U_I, and H' is a convex set in U_J, then $H \cap H'$ (*intersection*) is the convex set of mappings p defined on $U_{I \cup J}$ verifying that $p^{\downarrow I} \in H$ and $p^{\downarrow J} \in H'$.

If H is an a priori piece of information about (X, Y), that is to say a convex set of probability distributions on $U \times V$, then the *marginal* of H for X, H^X, is $H^{\downarrow U}$.

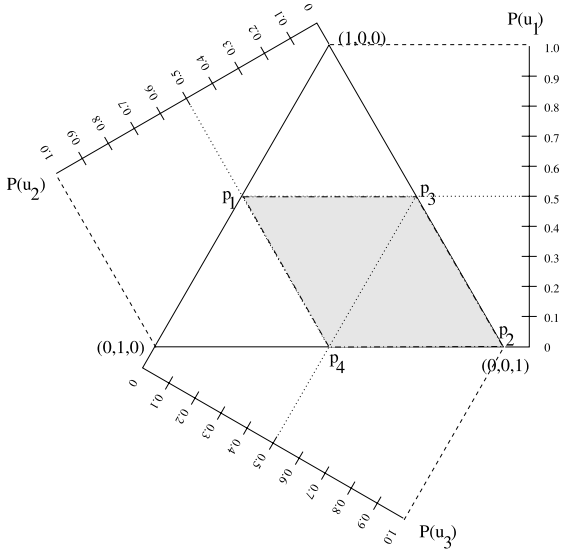


Fig. 4. An example of a convex set given by its extreme points $(p_1, \ldots, p_4)$.

The *conditional information* about Y, given X, will be a closed, convex set, $H^{Y|X}$, of mappings, $p : U \times V \to [0,1]$, verifying

$$\sum_{v \in V} p(u,v) = 1 \quad \forall u \in U$$

and with a finite set of extreme points, $\mathrm{Ext}(H^{Y|X}) = \{p_1, \ldots, p_l\}$.

Given H^X and $H^{Y|X}$ then $H = H^X \otimes H^{Y|X}$ is the *global information* about (X,Y).

Given $H = \mathrm{CH}\{p_1, \ldots, p_k\}$ for X. If we know X *belongs to* A, then the result of *conditioning* is $H|A = \mathrm{CH}\{p(\cdot|A) : p \in H, p(A) \neq 0\}$.

For convex sets we will use the so-called *strong conditional independence* as a notion of independence. See De Campos and Moral [7] and Couso et al. [17] for a detailed study of definitions of independence in convex sets. If $H^{X,Y,Z}$ is a global convex set of probabilities for (X,Y,Z), we say that X and Y are conditionally strong independent given Z if and only if, $H^{X,Y,Z} = H^{X,Z} \otimes H^{Y|Z}$ or $H^{X,Y,Z} = H^{Y,Z} \otimes H^{X|Z}$.

4.1. Propagation of convex sets of probabilities

The propagation of convex sets of probabilities is completely analogous to the propagation of probabilities. The procedures are the same. Here, we only describe the main differences and more details can be found in [15]. Valuations

are convex sets of possible probabilities, with a finite number of extreme points. A conditional valuation is a convex set of conditional probability distributions. 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 combining a mapping of the first convex set with a mapping of the second convex set (repeating the probabilistic combination for all the points of the two convex sets).

The marginalization of a convex set is defined by marginalizing each of the mappings of the convex set.

With these operations, we can carry out the same propagation algorithms as in the probabilistic case. The formulae are the same, but a valuation is now a convex set (represented by its extreme points) and the operations of combination and marginalization are the same as in the probabilistic case repeated for all the extreme points of the operated sets.

The result of the propagation for a variable, X_k, will be a convex set of mappings from U_k in [0,1]. For the sake of simplicity, let us assume that this variable has two values: x_k^1, x_k^2. The result of the propagation is a convex set on $\mathbb{R}^2$ of the form of Fig. 5 and that will be called R_k.

The points of this convex set, R_k, are obtained in the following way: if P is a global probability distribution, formed by selecting a fixed probability for each convex set, then associated to this probability, we shall obtain a point $(t_1, t_2) \in R_k$, where

$$\begin{aligned} t_1 &= P(x_k^1 \cap e), \\ t_2 &= P(x_k^2 \cap e), \end{aligned} \tag{15}$$

and e is the given evidence or family of observations.

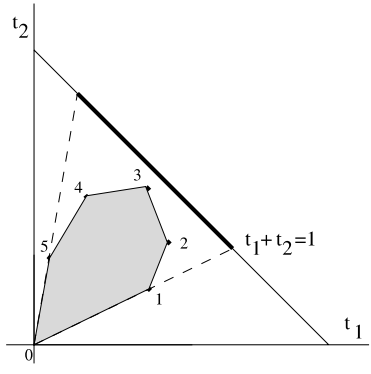


Fig. 5. Propagated convex set, R_k.

The projection of the point (t_1, t_2) on the line $t_1 + t_2 = 1$ is equivalent to dividing by $P(e)$ and gives rise to a normalized probability distribution: $P(x_k^i|e) = t_i/(t_1 + t_2),\ i = 1, 2$.

So, the final intervals $[a, b]$ associated to x_k^i can be calculated in the following way:

$$\begin{aligned} a &= \inf\{t_i/(t_1 + t_2) \,|\, (t_1, t_2) \in R_k\}, \\ b &= \sup\{t_i/(t_1 + t_2) \,|\, (t_1, t_2) \in R_k\}, \end{aligned} \tag{16}$$

that is to say, the interval given by the infimum and supremum of the conditional probabilities. This conditioning does not take likelihood information into account, but it is the most used in literature. In this paper, we try to calculate uncertainty intervals calculated according to this conditioning.

4.2. Convex sets and intervals

As we mentioned in Section 1, we have tried to solve the problem of propagation in dependence graphs where each initial conditional information is given by an interval of probability instead of an exact value of probability. The problem is that intervals of probability are a less general model than convex sets of probabilities [14,21]. The reason is that in the interval probabilities model we only know the bounds for events, and for convex sets we can specify bounds for every linear combination of the probabilities of the elementary events. However, we can obtain the convex sets associated to the initial informations (interval of probabilities), and then do computations using these convex sets and finally obtain the associated a posteriori intervals.

Intervals are a particular case of general restrictions. If we have a variable X taking values on a finite set U, then a convex set of probabilities, H, can be given by a set of linear restrictions, R. Each element in $r \in R$ is an inequality:

$$r \equiv \sum_{u \in U} \alpha_u p(u) \leqslant \beta. \tag{17}$$

For a convex set, H, we can use the representation given by a finite set of points including its extreme points or the one given by a finite set of restrictions defining it. In both cases, it is preferable for the representation to be minimal: i.e. the points are the extreme points and the set of restrictions is minimal. In this paper, we will use the representation given by a finite set of points because it is more appropriate for the operations (marginalization and combination) we need to do with convex sets.

In general, to obtain the extreme points of the convex set associated to a set of probability intervals we can use the following algorithm. In this algorithm it is assumed that $U = \{u_1, \ldots, u_n\}$. *Prob* is a list of the extreme probabilities found so far, and p is the current partial probability (this means that $\alpha(u_i) \leqslant p(u_i) \leqslant \beta(u_i)\ \forall i$, but not necessarily the restriction $\sum_i p(u_i) = 1$). *Expl* is

a set of already explored nodes, and λ is the amount of probability that has not yet been assigned: $1 - \sum_i p(u_i)$. Max($Expl$) returns the index with the higher value in the set of explored nodes.

The initialization steps are:

Algorithm 5. Initialization
1. $Prob \leftarrow \emptyset$
2. $Expl \leftarrow \emptyset$
3. $\lambda \leftarrow 1 - \sum_{i \leqslant n} \alpha(u_i)$
4. For $i = 1$ to n do
   $p(u_i) \leftarrow \alpha(u_i)$

Then we call Getprob($p, \lambda, Expl$) which calculates and appends the extreme probabilities to $Prob$.

Algorithm 6. [Getprob($p, \lambda, Expl$)]
1. For $i = 1$ to Max($Expl$) do
   If not belong($i, Expl$)
   Then If $\lambda < \beta(u_i) - \alpha(u_i)$
   Then
   $v \leftarrow p(u_i)$
   $p(u_i) \leftarrow p(u_i) + \lambda$
   Append($p, Prob$)
   $p(u_i) \leftarrow v$
2. For $i =$ Max($Expl$) $+ 1$ to n do
   If $\beta(u_i) - \alpha(u_i) > 0$
   Then
   If $\lambda \leqslant \beta(u_i) - \alpha(u_i)$
   Then
   $v \leftarrow p(u_i)$
   $p(u_i) \leftarrow p(u_i) + \lambda$
   Append($p, Prob$)
   $p(u_i) \leftarrow v$
   Else
   $v \leftarrow p(u_i)$
   $p(u_i) \leftarrow \beta(u_i)$
   Getprob($p, \lambda - \beta(u_i) + \alpha(u_i), Expl \cup \{i\}$)
   $p(u_i) \leftarrow v$;

This algorithm uses an implicit search tree where each node is a partial probability and a child node represents a refinement of its parent node by increasing one component $p(u_i)$. The leaf nodes are the extreme probabilities.

Example 1 (*Extreme points associated to a global information given with intervals*). Suppose we have a variable X taking values on the set $U = \{u_1, u_2, u_3, u_4\}$ and the intervals given by the following restrictions:

	u_1	u_2	u_3	u_4
α	0.2	0.15	0.4	0.1
β	0.25	0.3	0.45	0.15

then if we apply Algorithm 6 we will obtain the following possible extreme points:

	u_1	u_2	u_3	u_4
p_1	0.25	0.25	0.40	0.10
p_2	0.25	0.20	0.45	0.10
p_3	0.25	0.20	0.45	0.15
p_4	0.25	0.20	0.40	0.15
p_5	0.20	0.30	0.40	0.10
p_6	0.20	0.25	0.45	0.10
p_7	0.20	0.20	0.45	0.15
p_8	0.20	0.25	0.40	0.15

Suppose $X_I = \{X_1, \ldots, X_k\}$ is a set of random variables taking values on the finite set $U_I = U_1 \times \cdots \times U_k$ and Y a random variable taking values on a finite set V. Then, if we have a conditional distribution $P(Y|X_I)$ given with probability intervals, we must apply Algorithm 6 for each $u_I \in U_I$ to obtain the global convex set $H^{Y|X_I}$. That is to say, if Ext_{u_I} is the set of extreme points of the convex set associated to the distribution $P(Y|X = u_I)$ which we obtained with Algorithm 6, then the global convex set can be obtained with the following Cartesian product:

$$\mathrm{Ext}(H^{Y|X_I}) = \prod_{u_I \in U_I} \mathrm{Ext}_{u_I}. \tag{18}$$

Example 2 (*Extreme points associated to the conditional information given with intervals*). Let us suppose two random variables X and Y taking values on the sets $U\{u_1, u_2\}$ and $V\{v_1, v_2\}$, respectively. Suppose we have the conditional information $P(Y|X)$ given with the following set of probability intervals:

	u_1		u_2	
	v_1	v_2	v_1	v_2
α	0.2	0.7	0.4	0.4
β	0.3	0.8	0.6	0.6

If we apply Algorithm 6 for $X = u_1$, we obtain the convex set $H^{Y|X=u_1}$. This convex set has the following extreme points:

	v_1	v_2
p_1	0.2	0.8
p_2	0.3	0.7

Equally, for $X = u_2$ we obtain the convex set $H^{Y|X=u_2}$ with the following extreme points:

	v_1	v_2
q_1	0.4	0.6
q_2	0.6	0.4

With these two partial informations we can build the global information $H^{Y|X}$ using formula (18). We obtain the following extreme points:

	(u_1, v_1)	(u_1, v_2)	(u_2, v_1)	(u_2, v_2)
h_1	0.2	0.8	0.4	0.6
h_2	0.2	0.8	0.6	0.4
h_3	0.3	0.7	0.4	0.6
h_4	0.3	0.7	0.6	0.4

5. Using probability trees in the propagation of convex sets

In this section we will examine how probability trees can be used to make propagation of convex set more efficient. In order to do so, we can represent each initial conditional convex set $H^{Y|X_I}$ with a probability tree. We can then apply the same propagation algorithm described in Section 2 for the probabilistic case.

To use probability trees in the propagation of convex sets, we can transform the problem into an equivalent one. For each variable X_i, we originally give a valuation for this node conditioned to its parents. This valuation is a convex set of conditional probability distributions, $h_i = \{p_1, \ldots, p_l\}$. We then add a new node, T_i, with l cases $\{1, \ldots, l\}$. This node is made a parent of variable X_i in the dependence graph. On this node we consider that all the probability distributions are possible, that is to say, the valuation for this node is a convex set with l extreme points, each one degenerated in one of the possible cases of T_i. The probability of X_i given its parents is now a unique, determined probability distribution. If $F(X_i)$ are the original parents of X_i then the conditional probability of X_i given $F(X_i) \cup T_i$ is determined in the following way: given $T_i = k$ then the conditional probability of X_i given $F(X_i)$ is p_k. We can verify that the structure of the problem does not change with this transformation. The only

thing that has happened is that our lack of knowledge about the conditional probabilities is now explicit with the help of an additional node expressing all the possible conditional probability distributions. Nothing is known about this node. We suppose that the domain of T will be U_T. Moreover, if T takes n possible values, then we suppose that $U_T = \{1, \ldots, n\}$. The idea is to keep the different values of T as parameters in the problem.

Suppose we want to find a compact representation for the global conditional convex set $H^{Y|X_I}$, and we know which are the extreme points of each one of the convex sets $H^{Y|X_I=u_I}$.

In Example 2 the global information $H^{Y|X}$ is represented by four points of the global convex set. Each point is a vector with four real numbers. So, we need 16 real numbers to represent $H^{Y|X}$. A first approximation to use probability trees will be to put the transparent variable T at the root of the probability tree. This variable T will have as many cases as the points which our representation of $H^{Y|X}$ has (four points in Example 2). Each child of this node will be the probability tree that represents each one of the points of $H^{Y|X}$. In Fig. 6, we can see the probability tree that represents the extreme points of $H^{Y|X}$. In this probability tree an extreme point is found fixing T to one of its possible values.

A more compact representation can be obtained if we use a transparent variable T_{u_I} for each $u_I \in U_I$. That is to say, we associated a transparent variable for each configuration $u_I \in U_I$ of the set of variables X_I. T_{u_I} will have as many cases as the number of extreme points which $H^{Y|X_I=u_I}$ has. The reduction in the representation is obtained by taking asymmetric independences among these transparent variables into account. It is obvious that:

$$I_c(Y; T_{u_I} | X = u_J, u_I \neq u_J): \quad \forall u_I \in U_I. \tag{19}$$

A compact probability tree can then be built by putting the variables of X_I in the upper levels of the tree, and then the corresponding transparent variable. A leaf node of this provisional probability tree represents a configuration u_I of X_I. Next, for each leaf node, we put the transparent variable T_{u_I} associated to the

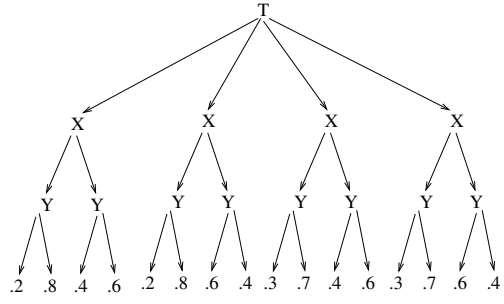


Fig. 6. Probability tree that represents $H^{Y|X}$ in Example 2.

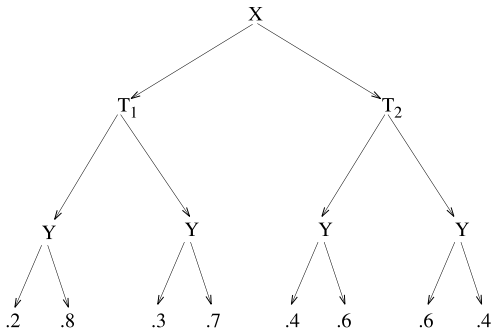


Fig. 7. A compact probability tree for $H^{Y|X}$ in Example 2.

configuration u_I that this leaf node represents. Finally, each T_{u_I} will have as many children as the number of extreme points which $H^{Y|X_I=u_I}$ has. Each one of these children will be the tree that represents one of the extreme points of $H^{Y|X_I=u_I}$. Fig. 7 shows the probability tree built so far, for convex set of Example 2. This new probability tree only needs eight real numbers to represent the same information as the tree of Fig. 6.

In this new tree, a point of the global convex set $H^{Y|X_I}$ can be found by fixing all transparent nodes T_{u_I} to one of its values.

5.1. Using probability trees to eliminate non-extreme points

In Section 4.1 we saw that the result of a propagation algorithm with convex sets of probabilities for a variable X_k is a convex set of mappings from U_k in $[0,1]$. This convex set was denoted by R_k. If we use probability trees to represent every convex set in the propagation algorithm, then R_k will also be represented by a probability tree $\mathscr{T}_k$. See Fig. 8 for an example of the probability tree $\mathscr{T}_k$ representing the result of a propagation algorithm for a variable X with three possible values $\{u_1, u_2, u_3\}$. Variables of $\mathscr{T}_k$ will be X_k and a set of transparent variables T_K.

Extreme points of R_k can be obtained from $\mathscr{T}_k$, fixing each one of the transparent variables (parameters) in $\mathscr{T}_k$ to a value. In this way, it is possible to obtain the same point more than once. Furthermore, some of these points will be non-extreme points because they can be obtained from a lineal combination of other points. For example, Table 1 shows the points obtained from the probability tree of Fig. 8. This table shows that point $(0.2, 0.4, 0.25)$ appears at the third and fourth row, and point $(0.3, 0.4, 0.25)$ appears at the seventh and eighth row. Repeated points appear because the tree can be express asymmetric independences that the table cannot express. For example, the probability tree of Fig. 8 shows that $I_c(X; T_3|T_2 = 2)$. Table 1 can reflect this independence, and

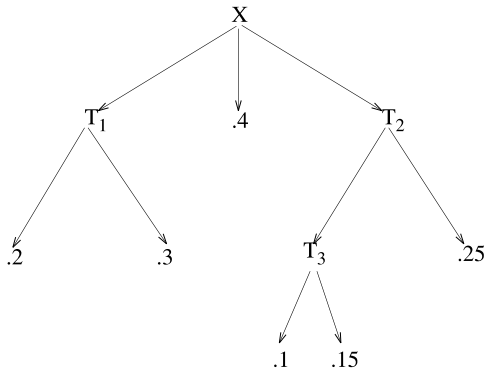


Fig. 8. Resulting probability tree for variable X.

Table 1
Points from the probability tree of Fig. 8

T_1	T_2	T_3	u_1	u_2	u_3
1	1	1	0.2	0.4	0.1
1	1	2	0.2	0.4	0.15
1	2	1	0.2	0.4	0.25
1	2	2	0.2	0.4	0.25
2	1	1	0.3	0.4	0.1
2	1	2	0.3	0.4	0.15
2	2	1	0.3	0.4	0.25
2	2	2	0.3	0.4	0.25

points $(0.2, 0.4, 0.25)$ and $(0.3, 0.4, 0.25)$ will appear twice. Table 1 contains two points $(0.2, 0.4, 0.15)$ and $(0.3, 0.4, 0.15)$ that are not extreme points because they can be obtained from a lineal combination of $(0.2, 0.4, 0.1)$ and $(0.2, 0.4, 0.25)$ for $(0.2, 0.4, 0.15)$, and from $(0.3, 0.4, 0.1)$ and $(0.3, 0.4, 0.25)$ for $(0.3, 0.4, 0.15)$.

We have designed an algorithm which solves the two previous problems (repeated points, and non-extreme points that are lineal combination of others). This algorithm can be decomposed in three different situations. Let us suppose we have a probability tree $\mathscr{T}_k$ representing the a posteriori convex set for variable X_k of a propagation algorithm.

In the first situation, suppose that X_k is at the root node of $\mathscr{T}_k$ (then the rest of the variables will be transparent). Furthermore, suppose that there are no common transparent variables among children trees of the root.

The probability tree in Fig. 8 verifies these conditions. In this situation, we can obtain a simpler probability tree $\mathscr{T}'_k$ which represents the same a posteriori convex set, with the following algorithm:

Algorithm 7 (*Find*1*SimplerTree*($\mathscr{T}_k$)).

Input: A probability tree $\mathscr{T}_k$ with target variable X_k at the root node and with no common transparent variables among children of the root.

Return: A simpler probability tree $\mathscr{T}'_k$ representing the same a posteriori information.

1. Put X_k at the root node of a new tree $\mathscr{T}'_k$.
2. For each u_k in U_k
   - Get min and max values of leaf nodes of $\mathscr{T}_k^{R(X_k=u_k)}$
   - If min and max are different:
     - Put a new transparent variable T_{u_k} as the child of the root node of $\mathscr{T}'_k$ corresponding to the branch $X_k = u_k$.
     - Put two leaf children to node T_{u_k} and label these children with min and max.
   - Otherwise, put a node labeled with min value as the child of the root node of $\mathscr{T}'_k$ corresponding to the branch $X_k = u_k$.
3. Return $\mathscr{T}'_k$

If we apply Algorithm 7 to the tree in Fig. 8 then we obtain the simpler tree in Fig. 9. From the tree in Fig. 9 we obtain the points in Table 2 instead of those in Table 1.

In a second situation, suppose X_k is also at the root node of $\mathscr{T}_k$, but we now have common transparent variables among children of the root node. In Fig. 10 we can see a probability tree verifying this situation.

Algorithm 8 (*Find*2*SimplerTrees*($\mathscr{T}_k, T_C$)).

Input: A probability tree $\mathscr{T}_k$ with target variable X_k at the root node and a set T_C of common transparent variables among children of the root.

Return: A set $Set_{\mathscr{T}'_k}$ of simpler probability trees $\mathscr{T}'_k$ representing the same a posteriori convex set for X_k.

1. Make $Set_{\mathscr{T}'_k}$ an empty set.
2. If T_C is not empty:
   - Take a transparent variable T from T_C.
   - For each value t_i of T:
     - Get $\mathscr{T}^{R(T=t_i)}$
     - Append trees returned by Find2SimplerTrees($\mathscr{T}^{R(T=t_i)}$,$T_C - T$) to the $Set_{\mathscr{T}'_k}$
3. Otherwise, append tree returned by Find1SimplerTree($\mathscr{T}_k$) to the $Set_{\mathscr{T}'_k}$
4. Return $Set_{\mathscr{T}'_k}$

For example, if we apply Algorithm 8 to the tree in Fig. 10 we will obtain the trees in Fig. 11. From these two trees, we obtain the points in Table 3.

In the third situation, variable X_k can appear at any node of $\mathscr{T}_k$. For example, the tree in Fig. 12 represents a general situation. The algorithm that

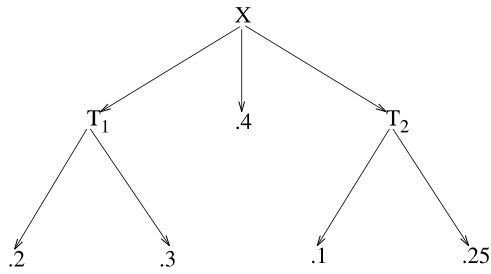


Fig. 9. Simpler tree associated to tree in Fig. 8.

Table 2
Points of the a posteriori convex set obtained with tree in Fig. 9

T_1	T_2	u_1	u_2	u_3
1	1	0.2	0.4	0.1
1	2	0.2	0.4	0.25
2	1	0.3	0.4	0.1
2	2	0.3	0.4	0.25

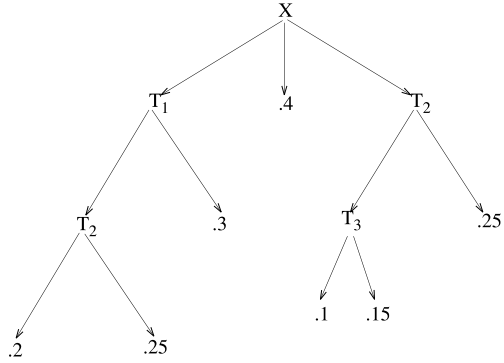


Fig. 10. A probability tree verifying conditions of the second situation.

solves this general situation does a depth traversal of $\mathscr{T}_k$ starting at the root node. The output of this algorithm is a set of equivalent, simpler probability trees. The traversal is stopped in a branch under two conditions. First, if a leaf node is visited, labeled with a real number r. This means that we have found a point $(r,\ldots,r)$ corresponding to a uniform distribution. In this case, we add a tree with only one node labeled with r to the set of output trees. Second, if variable X_k is found then we call to Algorithm 8 with the subtree in which X_k is the root obtaining a set of equivalent, simpler trees.

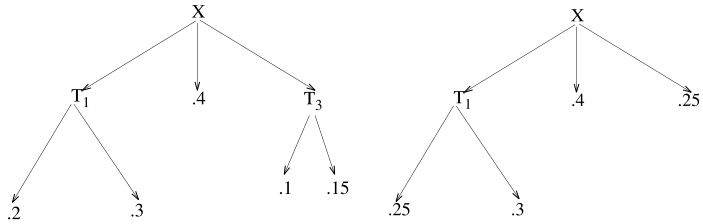


Fig. 11. Output trees from applying Algorithm 8 to tree in Fig. 10.

Table 3
Points of the a posteriori convex set obtained with trees in Fig. 11

T_1	T_3	u_1	u_2	u_3
1	1	0.2	0.4	0.1
1	2	0.2	0.4	0.15
2	1	0.3	0.4	0.1
2	2	0.3	0.4	0.15

T_1	u_1	u_2	u_3
1	0.25	0.4	0.25
2	0.3	0.4	0.25

The following algorithm solves the general situation:

Algorithm 9 (*Find3SimplerTrees*($\mathscr{T}_k$)).

Input: A probability tree $\mathscr{T}_k$ with a set of transparent variables and variable X_k.

Return: A set $Set_{\mathscr{T}'_k}$ of simpler probability trees $\mathscr{T}'_k$ representing the same a posteriori convex set for X_k.

1. Make $Set_{\mathscr{T}'_k}$ an empty set
2. If the root of $\mathscr{T}_k$ is labeled with a real number r append to $Set_{\mathscr{T}'_k}$ a tree with a node labeled with r
3. Otherwise, if the root of $\mathscr{T}_k$ is labeled with a transparent variable T
   - For each $t \in U_T$
     - Append trees returned by Find3SimplerTrees($\mathscr{T}_k^{R(T=t)}$) to $Set_{\mathscr{T}'_k}$
4. Otherwise, if the root of $\mathscr{T}_k$ is a target variable X_k
   - Search the set of common transparent variables T_C among children of the root node.
   - Append trees returned by Find2SimplerTrees($\mathscr{T}_k, T_C$) to $Set_{\mathscr{T}'_k}$
5. Return $Set_{\mathscr{T}'_k}$

If we apply Algorithm 9 to the tree in Fig. 12 we will obtain the points in Table 4.

Transformations of this section can be applied in intermediate steps of propagation. In these steps we have a set of valuations, each represented by a probability tree. When we have a probability tree $\mathscr{T}$ in which one of its subtrees has only transparent variables, then we can transform $\mathscr{T}$ into a new tree $\mathscr{T}'$ in which the subtree is replaced by a transparent node with two children

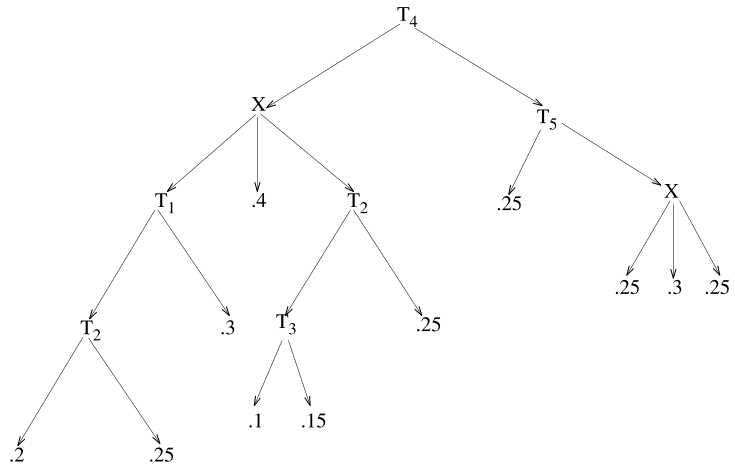


Fig. 12. A general a posteriori probability tree obtained as output of a propagation algorithm.

Table 4
Points of the a posteriori convex set in Fig. 12

u_1	u_2	u_3
0.2	0.4	0.1
0.2	0.4	0.15
0.3	0.4	0.1
0.3	0.4	0.15
0.25	0.4	0.25
0.3	0.4	0.25
0.25	0.25	0.25
0.25	0.3	0.25

labeled with the min and the max values of the old subtree. We have been able to prove in practice that this transformation in intermediate steps of propagation enormously reduces the amount of non-extreme points in the a posteriori convex set.

5.2. Getting and bounding the error of approximate intervals

In Section 3 we saw how to work with approximate probability trees in the probabilistic case, obtaining a posteriori approximate information. This allows us to solve more difficult problems in an approximate way than with the use of exact trees. In that section, we looked at how to use probability trees to propagate convex sets of probabilities. Approximated probability trees can also be used to propagate convex sets of probabilities. Approximations can be made when we build the tree from the original information, i.e. conditional information found in the dependence graph. Approximations can also be made

in operations with valuations: marginalization and combination. In both cases, approximations will be made using the same criterion as in the probabilistic case (Proposition 2). Moreover, Proposition 1 will be used to calculate the value that approximates a given subtree. In this section, we will see how to give a bound on the resulting errors when we use approximate probability trees in the propagation of convex sets of probabilities.

To bound the resulting errors, we used two new values at each leaf node in probability trees. If previously we only have one value $r \in \mathbb{R}$, we now add the values $r_{\min}, r_{\max} \in \mathbb{R}$ 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 equal. But when it is approximated then r will be between $r_{\min}$ and $r_{\max}$. Let us examine how to calculate $r_{\min}$ and $r_{\max}$ when we build a tree from original information, when we combine two trees and when we do a marginalization on a tree.

- *Construction of trees*. In this case, an approximation is made when several values in the true valuation are substituted by only one value r in the tree, according to Proposition 1. Let h be a valuation on a set of variables X_I. Suppose we are building a tree $\mathscr{T}$ for h and we are going to approximate a set of values of h with only one value r. This set of values is denoted by $h^{R(X_J=u_J)}$, representing the restriction of valuation h to configuration $X_J = u_J$, i.e. the values in h which are compatible with configuration $X_J = u_J$. Then, according to Proposition 1, r will be calculated as the average of $h^{R(X_J=u_J)}$. $r_{\min}$ and $r_{\max}$ will be calculated as:

$$\begin{aligned} r_{\min} &= \min_{u_J} h^{R(X_J=u_J)}, \\ r_{\max} &= \max_{u_J} h^{R(X_J=u_J)}. \end{aligned} \tag{20}$$

- *Combination of trees*. Suppose we are going to combine two trees $\mathscr{T}_1$ and $\mathscr{T}_2$. In Section 3 we studied two methods for making an approximated combination of these trees:
  - Generate entire resulting tree and prune it afterwards to the desired size. In this case, at each leaf node r, $r_{\min}$ and $r_{\max}$ will be calculated as in the construction of the tree.
  - Stop expanding the resulting tree at a given size. Suppose that we must put the result of the product of $\mathscr{T}_i$ and $\mathscr{T}_j$ at a leaf of the output tree, but we have no more space. Therefore, at that leaf node we put an approximate real number r, calculated as explained at the end of Section 3. $r_{\min}$ and $r_{\max}$ are calculated as:

$$\begin{aligned} r_{\min} &= \min r_{\min}(\mathscr{T}_i) \times \min r_{\min}(\mathscr{T}_j), \\ r_{\min} &= \max r_{\max}(\mathscr{T}_i) \times \max r_{\max}(\mathscr{T}_j), \end{aligned} \tag{21}$$

where $r_{\min}(\mathscr{T})$ and $r_{\max}(\mathscr{T})$ represent the set of $r_{\min}$ and $r_{\max}$ values of tree $\mathscr{T}$, respectively.

See Example 3 for how these approximations are made.

- *Marginalization in trees*. Again we have the same two possibilities. In both cases, the solution is completely analogous to combination, replacing the product by addition.

Example 3 (*Getting* $r_{\min}$ *and* $r_{\max}$ *in combination of trees*). Suppose that the output tree, resulting from combining two trees $\mathscr{T}_1$ and $\mathscr{T}_2$, has reached the maximum size. At this moment, the output tree is shown on the left of Fig. 13. After making the approximation, we obtain the output tree on the right-hand side of Fig. 13.

After applying a propagation algorithm we obtain a tree as in Fig. 14. From this tree we can obtain two kinds of intervals for each case of the target variable:

- Those obtained with the set of points extracted with the r values of leaf nodes, and using formula (16). These intervals will in general be approximated intervals. The set of points are extracted from the tree using the algorithms detailed in Section 5.1.
- Those obtained using the $r_{\min}$ and $r_{\max}$ values.

Let us now examine how to get a set of points from the $r_{\min}$ and $r_{\max}$ values of leaf nodes of an output probability tree. In order to do so, we can apply the same algorithms described in Section 5.1 for r values, but now in Algorithm 7 min and max are calculated as:

$$\begin{aligned} \min &= \min r_{\min}\left(\mathscr{T}_k^{R(X_k=u_k)}\right), \\ \max &= \max r_{\max}\left(\mathscr{T}_k^{R(X_k=u_k)}\right). \end{aligned} \tag{22}$$

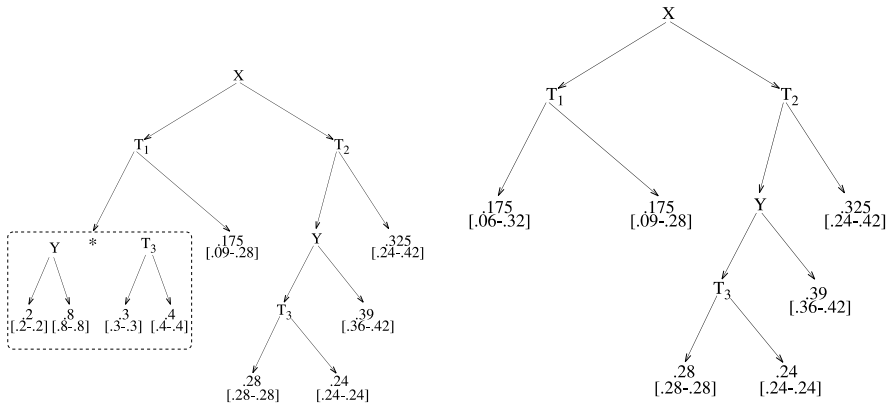


Fig. 13. Left tree represents an unfinished output tree resulting from combining two trees and right tree represents the same tree after making the approximations.

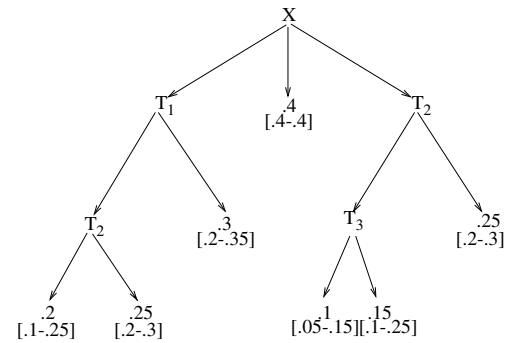


Fig. 14. An output tree obtained as result of a propagation algorithm using approximated operations.

The intervals obtained with $r_{\min}$ and $r_{\max}$ values will be wider than those obtained with the r values, and are useful to bound its error, that it approximated, and exact intervals will be contained in these intervals.

6. Experimental results

In order to study the behavior of our propagation algorithm using approximated probability trees to carry out the operations, we have done some experiments using four independence graphs: Boerlage92 [4], Boblo [39,40], Car Starts (a somewhat large network contributed by Sreekanth Nagarajan based on the automobile belief network that D. Heckerman et al. presented in [29]) and Alarm [2]. These graphs can be found in the literature for the probabilistic case, i.e. at each node we have a conditional probability distribution. Propagation on these networks is not very difficult from a probabilistic point of view.

We have transformed each probability p into a randomly chosen probability interval. This makes the problem of exact propagation very difficult to solve, since it amounts to making a tremendous number of probabilistic propagations: 9.007199×10^{15} in Boerlage92, 4.529848×10^{8} in Boblo, 5.242888×10^{5} in Car Starts and 1.713495×10^{93} in Alarm. To transform each probability p into an interval we use the following procedure: for each p we select a uniform number r from the interval $[0, \max\{p, 1-p, d\}]$ with $d \leqslant 1$ being a given threshold (we have used $d = 0.1$). Then p is transformed into the interval $[p-r, p+r]$. This way of selecting the interval ensures that $p - r \geqslant 0$. Moreover, when $p = 0.0$ or $p = 1.0$ we will obtain $[0.0, 0.0]$ or $[1.0, 1.0]$, respectively. We have used a variables elimination algorithm (Algorithm 1) applied to the case of a convex set of probabilities. In the experiments, combination and

marginalization are carried out using different maximum sizes for the output probability trees. Depending on the way we have performed these operations, we can distinguish two kinds of experiments:

- *PropWithTD*1. Combination and marginalization are carried out by constructing the output tree to a given threshold size.
- *PropWithTD*3. Combination and marginalization are carried out in an exact way, and are then pruned to a given threshold size.

Experiments have been run on an Intel Pentium II (400 MHz) computer with 384 MB of RAM and the Linux RedHat operating system with kernel 2.0.36. Algorithms have been implemented in C language.

Propagation algorithm gives us an output probability tree as in Fig. 14. From the output tree we have calculated two sets of points using algorithms described in Section 5.1:

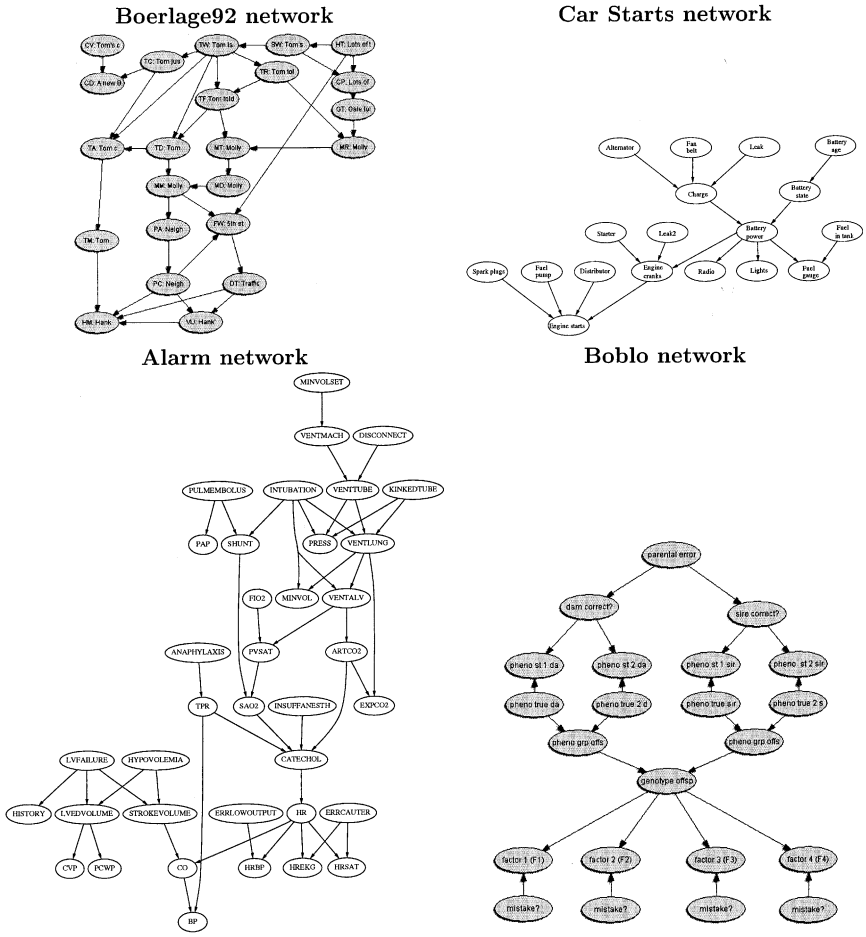
- Those obtained with the r values. From these points we can obtain an approximate interval $[l, u]$ for each case of the variable of interest using formula (16).
- Those obtained with the $r_{\min}$ and $r_{\max}$ values. From these points we again obtain an interval $[l_{\min}, u_{\max}]$ for each case of the variable of interest, that bounds the error in previous approximated intervals. These intervals are calculated again using formula (16).

Experiments have been carried out for different variables of interest and with some or none of the observations in the other variables (see Table 5). The following configurations are used:

- *Boerlage*92
  - *Boe*1: Variable of interest is "TD: Tom". No observations.
  - *Boe*2: Variable of interest is "MT: Molly". No observations.
  - *Boe*3: Variable of interest is "TD: Tom". Observation at "TR: Tom".
- *Boblo*
  - *Bob*1: Variable of interest is "factor 1 (F1)". No observations.
  - *Bob*2: Variable of interest is "factor 1 (F1)". Observations at variables "pheno st 1 dam", "pheno st 2 dam" and "pheno true sire".
- *Car Starts*
  - *Car*1: Variable of interest is "Engine starts". No observations.
  - *Car*2: Variable of interest is "Engine starts". Observations at variables "Alternator" and "Lights".
- *Alarm*
  - *Ala*1: Variable of interest is "BP". No observations.
  - *Ala*2: Variable of interest is "BP". Observations at variables "LVFAILURE", "PULMEMBOLUS" and "ARTCO2".

We have only calculated intervals $[l, u]$ and $[l_{\min}, u_{\max}]$ for the first case of one of the variables of the network (variable of interest). In each trial, we registered the computer time in seconds (t_s in tables) used by the algorithm.

Table 5
The four networks used in experiments



6.1. Results and discussion

It has been argued that in Bayesian networks, it is not necessary to worry about the correctness of initial probabilities as the final results are very insensitive to small modifications of numbers. Our experiments show that this is not true. We have cases in which the final probabilities are in the interval $[0.12, 0.22]$, interval $[0.90, 0.95]$, interval $[0.22, 0.53]$, or in the interval $[0.17, 0.34]$. Substituting the initial imprecise probabilities for one single value implies that the final a posteriori probabilities are numbers in these intervals.

There is no guarantee of the position of these numbers in the intervals, and it is possible that they are very close to one of the extremes. Therefore, computing the final a posteriori interval probabilities is necessary if we want our final decisions to take into account the imprecision in the values of probability.

Regarding the performance of the approximate algorithms in this paper, we have observed two different situations:

- There are situations in which the approximate interval and the interval $[l_{\min}, u_{\max}]$ are equal or very similar. In this case, we can be sure that this is the final a posteriori conditional interval. This happens in simple problems (Figs. 15 and 17) from a probabilistic point of view. However, the problem is not simple if we consider the number of different global probability distributions that should have been propagated for an exact solution of the problem (from 5.242888×10^5 in Car Starts to 1.713495×10^{93} in Alarm). This is due to the fact that we have added imprecision in all the values of probability. The problem would have been much simpler if almost all the probabilities had an exact value, even with more complex networks.
- In other situations the approximate interval is relatively small while the upper interval $[l_{\min}, u_{\max}]$ is very uninformative. In this case, we cannot be sure that the approximate interval really represents the real variation of the a posteriori probabilities and the method does not provide reliable information. These experiments also explain some comments about propagation of imprecise probabilities [48]: it has been said that the intervals tend to be $[0, 1]$. These intervals are only calculated given a lower and upper value for each value of a set of variables. In our system, this amounts to removing all the transparent variables and keeping only the minimum and maximum for the potentials. This is a very poor approximation which can be represented with small-sized trees (the size of a single probabilistic propagation). So, if we often obtain $[0, 1]$ intervals even with the stronger approximation procedures of this paper, it is very natural to get them with the usual weaker procedures.

Regarding the two basic procedures which we have applied in our approximation algorithms (PropWithTD1, which builds approximate trees until a given size while making computations, and PropWithTD3, which computes in an exact way and approximates afterwards), we have found that the latter provides better results in general but it needs more memory to carry out exact computations and in some complicated networks only PropWithTD1 has been applicable for this reason.

7. Conclusions

In this work we have adapted an algorithm that propagates with probability distributions to the case of the interval of probabilities. Intervals are transformed

into convex sets of probabilities, to carry out operations without losing information. Convex sets of probabilities are represented using probability trees with the help of transparent variables. We have presented some techniques that make propagation even more efficient exploiting asymmetric independences among transparent variables. We have also studied a technique to eliminate non-extreme points from the output of a propagation algorithm. This technique can also be applied in intermediate steps of propagation. Finally in our experimental work, we have proved our propagation algorithm using approximated operations with probability trees. This allows difficult problems to be solved in an approximate way. To bound the error committed in experiments we have used the method presented in Section 5.2. The experiments show that this method is promising, because we can control the accuracy of the a posteriori intervals with the size of the tree. Furthermore, intervals $[l_{\min}, u_{\max}]$ can inform us if we have obtained exact intervals for a given size of probability trees. If we have not obtained exact final intervals and we have more time, we can try with a greater limit for the tree.

The experiment in this paper should be interpreted as a first step in the computation of final a posteriori intervals. The algorithms could be improved in the following directions:

- Applying a reduction of the trees by removing convex combinations in intermediate steps of the algorithm and not only at the end when computing the final convex set.
- Applying global optimization procedures such as simulated annealing to maximize and minimize conditional probabilities for the different values of the transparent variables following the approach in [10].
- Adapting measures of information in the approximation of trees to the case of transparent variables. Now, the approximation procedure is based on considering that all the variables are of the same nature: probabilistic variables. However, transparent variables have a different behavior and approximations could follow different criteria.

Acknowledgements

This work has been supported by the CICYT under project TIC97-1135-C04-01. We are grateful for comments from Uffe Kjærulff and some other members of the Research Unit of Decision Support Systems at Aalborg University. This paper has also profited from the comments of the referees.

Appendix A

Tables 6–14 and Figs. 15–21 show the experimental results of our approximated algorithms.

Table 6
Intervals and time using PropWithTD1 (a) and PropWithTD3 (b) with configuration Boe1 in the *Boerlage92* network

	10 000	20 000	30 000	40 000	50 000	65 000	70 000	100 000	200 000
(a)									
l	0.131587	0.130855	0.130740	0.130740	0.130740	0.130733	0.124713	0.124487	0.124486
u	0.212634	0.213963	0.213963	0.213973	0.213973	0.213973	0.224924	0.225327	0.225327
$l_{\min}$	0.114457	0.114457	0.114457	0.114457	0.114457	0.114457	0.124486	0.124486	0.124486
$u_{\max}$	0.243323	0.243323	0.243323	0.243323	0.243323	0.243323	0.225327	0.225327	0.225327
t_s	6.5	13.5	29.4	34	34.5	35	37	61	62
(b)									
	5000	10 000	20 000	30 000	40 000	50 000	100 000		
l	0.128374	0.124875	0.124724	0.124671	0.124620	0.124509	0.124487		
u	0.232749	0.224244	0.224336	0.224809	0.225223	0.225312	0.225327		
$l_{\min}$	0.110878	0.111654	0.114568	0.114653	0.124486	0.124486	0.124486		
$u_{\max}$	0.247272	0.246851	0.242907	0.242698	0.225327	0.225327	0.225.327		
t_s	69.7	153.7	192.7	273.55	366.7	565.6	878.9		

Table 7
Intervals and time using PropWithTD1 (a) and PropWithTD3 (b) with configuration Boe2 in the *Boerlage92* network

	10 000	20 000	30 000	40 000	50 000	60 000		
(a)								
l	0.105220	0.102918	0.102548	0.101518	0.099977	0.099885		
u	0.203419	0.203787	0.204783	0.202381	0.203109	0.200739		
$l_{\min}$	0.042534	0.042997	0.042997	0.042997	0.043098	0.043432		
$u_{\max}$	0.528020	0.527453	0.527453	0.527453	0.527453	0.577453		
t_s	185	405	818	1497	2793	10529		
(b)								
	5000	10 000	20 000	30 000	40 000	50 000	60 000	100 000
l	0.100187	0.295483	0.107630	0.412786	0.301805	0.286207	0.230603	0.447639
u	0.511125	0.500000	0.516692	0.522224	0.514695	0.514841	0.514103	0.500005
$l_{\min}$	0.024886	0.020750	0.010792	0.049314		0.029298	0.034785	0.032752
$u_{\max}$	0.975113	0.979249	0.989207	0.950685		0.970701	0.965214	0.967247
t_s	228.6	390.7	1990.6	3180.1	7138.8	7839.7	6447	25571

Table 8
Intervals and time using PropWithTD1 (a) and PropWithTD3 (b) with configuration Boe3 in the *Boerlage92* network

	1000	2500	5000	7500	10 000	12 500	15 000	17 500	20 000	40 000
(a)										
l	0.272973	0.270832	0.251653	0.251653	0.251358	0.222883	0.222883	0.222883	0.222883	0.222867
u	0.489364	0.489273	0.489545	0.489807	0.495341	0.522341	0.529459	0.531509	0.531531	0.531531
$l_{\min}$	0.190970	0.193818	0.194207	0.194207	0.194207	0.209124	0.222867	0.222867	0.222867	0.222867
$u_{\max}$	0.549067	0.544407	0.542507	0.542507	0.542507	0.542507	0.542507	0.531531	0.531531	0.531531
t_s	1	2.5	3.1	4	5	6.1	7.4	8.3	9.2	12.4
(b)										
	1000	2500	5000	7500	10 000	12 500	17 500	22 500	32 500	35 000
l	0.477410	0.259620	0.223022	0.222883	0.222883	0.222883	0.222883	0.222883	0.222871	0.222867
u	0.553631	0.553631	0.553042	0.531183	0.531418	0.531418	0.531531	0.531531	0.531531	0.531531
$l_{\min}$	0.121570	0.116982	0.118544	0.132799	0.133030	0.133030	0.139251	0.148725	0.153454	0.222867
$u_{\max}$	0.878429	0.883220	0.881455	0.870560	0.867066	0.812689	0.810565	0.808564	0.803545	0.531531
t_s	1	5.5	20	29	30	30	30	31	63	65

Table 9
Intervals and time using PropWithTD1 with configuration Bob1 in the *Boblo* network

	20 000	30 000	40 000	50 000	60 000
l	0.5	0.5	0.5	0.5	0.5
u	0.5	0.5	0.5	0.5	0.5
$l_{\min}$	0.000402	0.000402	0.000333	0.000333	0.000333
$u_{\max}$	0.999597	0.999597	0.999666	0.999666	0.999666
t_s		4423.4	7092.4	8627	10294

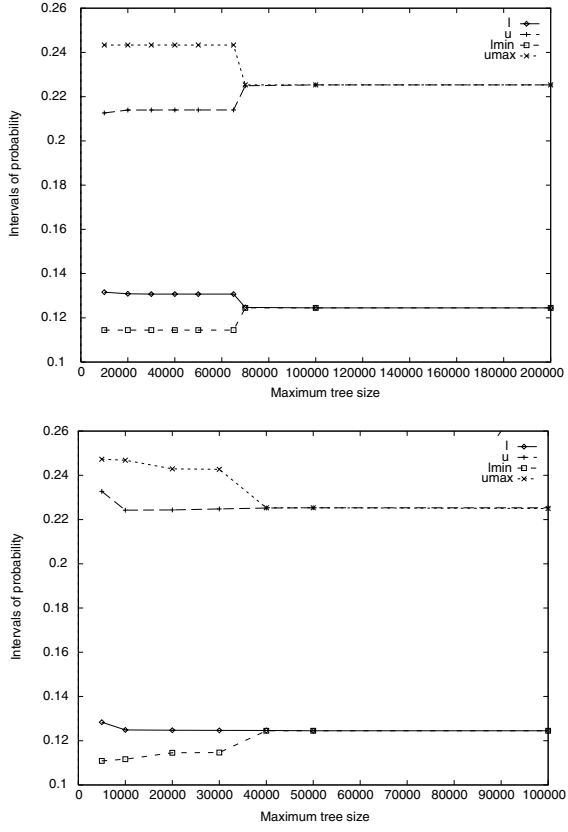


Fig. 15. Intervals with PropWithTD1 and PropWithTD3 algorithms with configuration Boe1 in the *Boerlage92* network (Table 6).

Table 10
Intervals and time using PropWithTD1 algorithm with configuration Bob2 in the *Boblo* network

	20 000	30 000	40 000	50 000	60 000
l	0.5	0.429848	0.429848	0.475508	0.479056
u	0.550398	0.552048	0.552048	0.552055	0.543777
$l_{\min}$	0.0	0.0	0.0	0.0	0.0
$u_{\max}$	1.0	1.0	1.0	1.0	1.0
t_s	1218	2436	4956	9650	14179

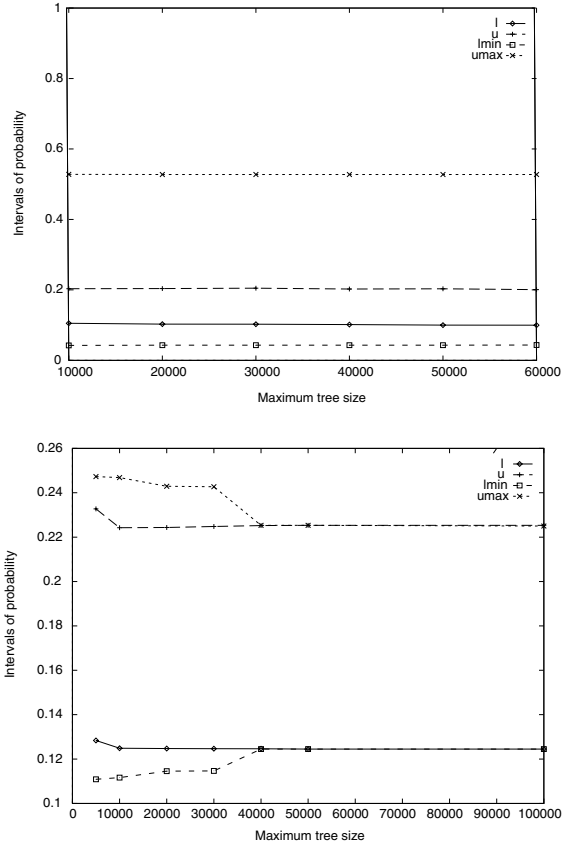


Fig. 16. Intervals with PropWithTD1 and PropWithTD3 algorithms with configuration Boe2 in the *Boerlage92* network (Table 7).

Table 11
Intervals and time using PropWithTD1 (a) and PropWithTD3 (b) algorithms with configuration Car1 in the *Car Starts* network

	20 000	30 000	40 000	50 000	60 000
(a)					
l	0.901337	0.901329	0.901327	0.901327	0.901327
u	0.957187	0.957197	0.957197	0.957197	0.957197
$l_{\min}$	0.901322	0.901323	0.901323	0.901323	0.901323
$u_{\max}$	0.958928	0.958928	0.958928	0.958928	0.958928
t_s	764	1301	1494	1799	2048
(b)					
l	0.496910	0.496910	0.496910	0.496910	0.496910
u	0.957758	0.957758	0.957765	0.957765	0.957765
$l_{\min}$	0.041273	0.041273	0.041273	0.041273	0.041273
$u_{\max}$	0.958727	0.958726	0.958726	0.958726	0.958726
t_s	530	700	1177	1088	1332

Table 12
Intervals and time using PropWithTD1 (a) and PropWithTD3 (b) algorithms with configuration Car2 in the *Car Starts* network

	20 000	30 000	40 000	50 000	60 000
(a)					
l	0.177708	0.177707	0.176318	0.176318	0.176318
u	0.343760	0.343762	0.343762	0.343762	0.343762
$l_{\min}$	0.174571	0.174571	0.174571	0.174571	0.174571
$u_{\max}$	0.348653	0.348653	0.348653	0.348653	0.348653
t_s	1044	1469	1892	2341	2785
(b)					
l	0.177744	0.176601	0.176586	0.175635	0.176718
u	0.345960	0.345960	0.346113	0.348650	0.348656
$l_{\min}$	0.174024	0.1730291	0.173029	0.173029	0.174024
$u_{\max}$	0.349661	0.349516	0.349661	0.351259	0.351259
t_s	667	787	875	1094	1807

Table 13
Intervals and time using PropWithTD1 algorithm with configuration ala1 in the *Alarm* network

	20 000	30 000	40 000	50 000	60 000	100 000
l	0.331890	0.332938	0.333333	0.330522	0.330522	0.330522
u	0.336223	0.333333	0.333333	0.335667	0.335667	0.335667
$l_{\min}$	0.0	0.0	0.0	0.0	0.0	0.0
$u_{\max}$	1.0	1.0	1.0	1.0	1.0	1.0
t_s	3260	6009	13584	22434	30440	103281

Table 14
Intervals and time using PropWithTD1 algorithm with configuration ala2 in the *Alarm* network

	20 000	30 000	40 000	50 000	60 000	100 000
l	0.329612	0.328413	0.328363	0.328381	0.328381	0.328675
u	0.333956	0.333877	0.333877	0.333876	0.333876	0.333834
$l_{\min}$	0.0	0.0	0.0	0.0	0.0	0.0
$u_{\max}$	1.0	1.0	1.0	1.0	1.0	1.0
t_s	3377	6447	11588	19844	29699	78024

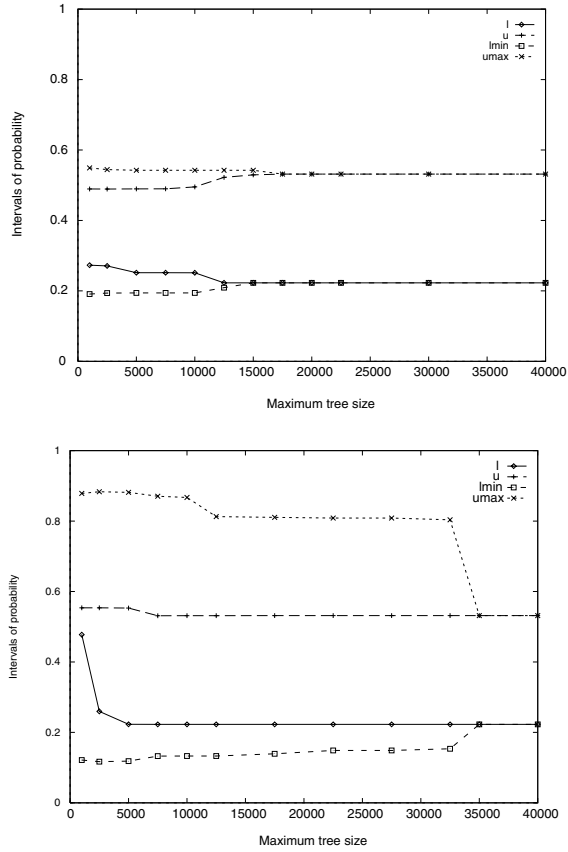


Fig. 17. Intervals with PropWithTD1 and PropWithTD3 algorithms with configuration Boe3 in the *Boerlage92* network (Table 8).

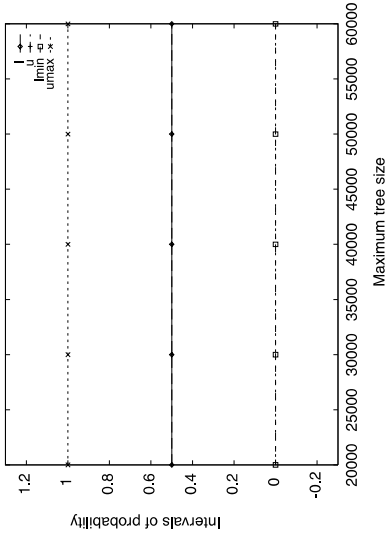


Fig. 18. Intervals with PropWithTD1 algorithm with configuration Bob1 in the *Boblo* network (Table 9).

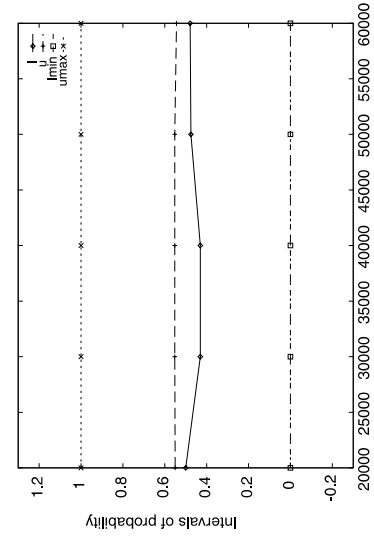


Fig. 19. Intervals with PropWithTD1 algorithm with configuration Bob2 in the *Boblo* network (Table 10).

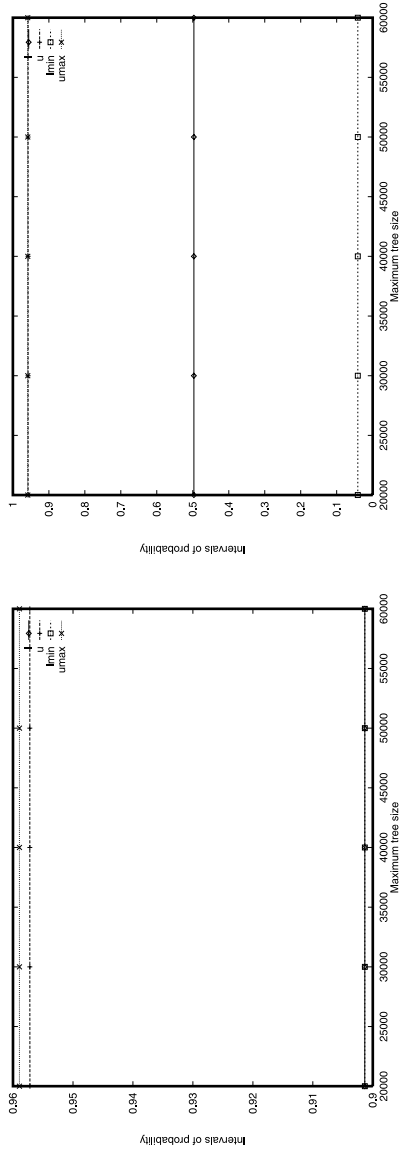


Fig. 20. Intervals with PropWithTD1 and PropWithTD3 algorithms with configuration Car1 in the *Car Starts* network (Table 11).

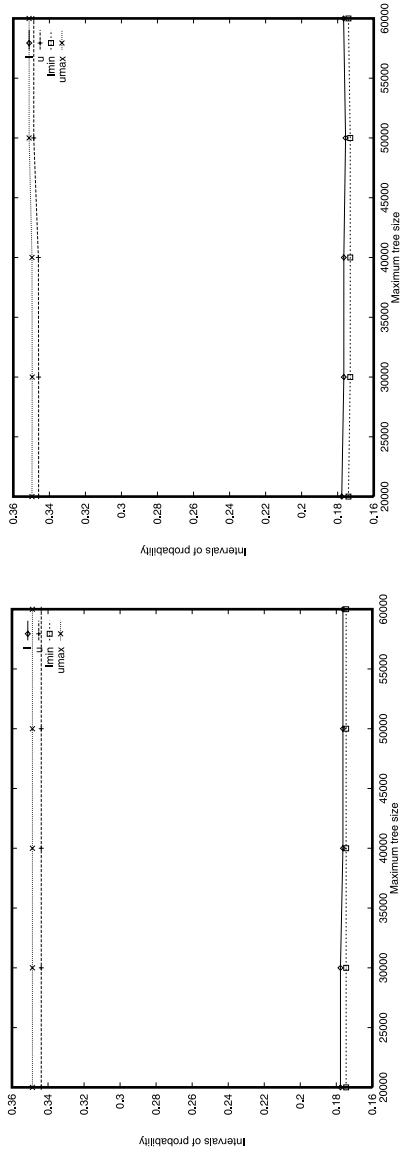


Fig. 21. Intervals with PropWithTD1 and PropWithTD3 algorithms with configuration Car2 in the *Car Starts* network (Table 12).

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 Seventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1991, pp. 26–34.
[2] I. Beinlich, G. Suermondt, R. Chavez, G. Cooper, The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks, in: Second European Conference on AI and Medicine, Springer, Berlin, 1989.
[3] J.O. Berger, An overview of robust Bayesian analysis (with discussion), Test 3 (1994) 5–124.
[4] B. Boerlage, Link strength in Bayesian networks, Ph.D. thesis, Department of Computer Science, University of British Columbia, Canada, 1992.
[5] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96), Portland, Oregon, 1996, pp. 115–123.
[6] L.M. de Campos, J.F. Huete, S. Moral, Probability intervals: a tool for uncertain reasoning, Int. J. Uncertainty, Fuzziness Knowledge-based Syst. 2 (1994) 167–196.
[7] L.M. de Campos, S. Moral, Independence concepts for convex sets of probabilities, in: Ph. Besnard, S. Hanks (Eds.), Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1995, pp. 108–115.
[8] L.M. de Campos, S. Moral, Removing partial inconsistency in valuation based systems, Int. J. Intell. Syst. 12 (1997) 629–653.
[9] A. Cano, J.E. Cano, S. Moral, Simulation algorithms for convex sets of probabilities, Technical Report TR-93-2-xx, DECSAI, Universidad de Granada, 1993.
[10] A. Cano, J.E. Cano, S. Moral, Convex sets of probabilities propagation by simulated annealing, in: Proceedings of the Fifth International Conference IPMU'94, Paris, 1994, pp. 4–8.
[11] A. Cano, S. Moral, A genetic algorithm to approximate convex sets of probabilities, in: Proceedings of Information Processing and Management of Uncertainty in Knowledge-based Systems Conference (IPMU'96), vol. 2, 1996, pp. 859–864.
[12] A. Cano, S. Moral, Propagación exacta y aproximada mediante árboles de probabilidad en redes causales, in: Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial, Málaga, 1997, pp. 635–644.
[13] A. Cano, S. Moral, A. Salmerón, Penniless propagation in join trees, Int. J. Intell. Syst. 15 (11) (2000) 1027–1059.
[14] J.E. Cano, S. Moral, J.F. Verdegay-López, Combination of upper and lower probabilities, in: B. D'Ambrosio, Ph. Smets, P.P. Bonissone (Eds.), Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1991, pp. 61–68.
[15] 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, Amsterdam, 1993, pp. 15–26.
[16] A. Chateauneuf, J.-Y. Jaffray, Some characterizations of lower probabilities and other monotone capacities through the use of Möbius inversion, Math. Soc. Sci. 17 (1989) 263–283.
[17] 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.
[18] F. Cozman, Robust analysis of Bayesian networks with finitely generated convex sets of distributions, Technical Report CMU-RI-TR96-41, Carnegie Mellon University, 1996.
[19] F. 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.

[20] R. Dechter, Bucklet elimination: a unifying framework for probabilistic inference, in: E. Horvitz, F.V. Jennsen (Eds.), Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, 1996, pp. 211–219.
[21] A.P. Dempster, Upper and lower probabilities induced by a multivalued mapping, Ann. Math. Stat. 38 (1967) 325–339.
[22] L. DeRobertis, J.A. Hartigan, Bayesian inference using intervals of measures, Ann. Stat. 14 (1986) 461–468.
[23] D. Dubois, H. Prade, Possibility Theory, Plenum Press, New York, 1988.
[24] 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.
[25] T.L. Fine, Theories of Probability, Academic Press, New York, 1973.
[26] T.L. Fine, An argument for comparative probability, in: R.E. Butts, J. Hintikka (Eds.), Basic Problems in Probability and Linguistics, D. Reidel, Dordrecht, MA, 1977, pp. 105–119.
[27] N. Friedman, M. Goldszmidt, Learning Bayesian networks with local structure, in: Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96), Portland, Oregon, 1996, pp. 252–262.
[28] M. Grabisch, H.T. Nguyen, E.A. Walker, Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference, Kluwer Academic Publishers, Dordrecht, MA, 1995.
[29] D. Heckerman, J. Breese, K. Rommelse, Troubleshooting under uncertainty, Technical Report msr-tr-94-07, Microsoft Research, Advanced Technology Division, Microsoft Corporation, 1994.
[30] P.J. Huber, Robust Statistics, Wiley, New York, 1981.
[31] G.J. Klir, T.A. Folger, Fuzzy Sets, Uncertainty, and Information, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[32] D. Kozlov, D. Koller, Nonuniform dynamic discretization in hybrid networks, in: D. Geiger, P.P. Shenoy (Eds.), Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1997, pp. 302–313.
[33] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1951) 76–86.
[34] Z. Li, B. D'Ambrosio, Efficient inference in Bayes networks as a combinatorial optimization problem, Int. J. Approx. Reason. 11 (1994) 55–81.
[35] S. Moral, J. del Sagrado, Aggregation of imprecise probabilities, in: B. Bouchon-Meunier (Ed.), Aggregation and Fusion of Imperfect Information, Physica-Verlag, Heidelberg, 1997, pp. 162–168.
[36] J. Pearl, Probabilistic Reasoning with Intelligent Systems, Morgan Kaufmann, San Mateo, 1988.
[37] D. Poole, Probabilistic partial evaluation: exploiting rule structure in probabilistic inference, in: Proceedings of the 15th IJCAI Conference (IJCAI'97), Nagoya, Japan, 1997, pp. 1284–1291.
[38] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–105.
[39] L.K. Rasmussen, Bayesian network for blood typing and parentage verification of cattle, Technical Report Dina research report no. 38, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1995.
[40] L.K. Rasmussen, Boblo: an expert system based on Bayesian networks to blood group determination of cattle, Technical Report Research report 16, Research Center Foulum, Denmark, PB 23, 8830 Tjele, Denmark, 1995.
[41] A. Salmerón, A. Cano, S. Moral, Importance sampling in Bayesian networks using probability trees, Comput. Stat. Data Anal. 34 (2000) 387–413.
[42] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ, 1976.

[43] P.P. Shenoy, G. Shafer, Axioms for probability and belief-function propagation, in: R.D. Shachter, et al. (Eds.), Uncertainty in Artificial Intelligence, vol. 4, North-Holland, Amsterdam, 1990, pp. 169–198.
[44] P.P. Shenoy, A valuation-based language for expert systems, Int. J. Approx. Reason. 3 (1989) 383–411.
[45] S.E. Shimony Jr., E. Santos, Exploiting case-based independence for approximating marginal probabilities, Int. J. Approx. Reason. 14 (1996) 25–54.
[46] Ph. Smets, Belief functions, in: Ph. Smets, E.H. Mandani, D. Dubois, H. Prade (Eds.), Non-Standard Logics for Automated Reasoning, Academic Press, London, 1988.
[47] M. Sugeno, Theory of fuzzy integrals and its applications, Ph.D. thesis, Tokyo Institute of Technology, Tokyo, 1974.
[48] B. Tessen, Interval probability propagation, Int. J. Approx. Reason. 7 (1992) 95–120.
[49] H. Thöne, U. Güntzer, W. Kießling, Towards precision of probabilistic bounds propagation, in: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, 1992, pp. 315–322.
[50] J.F. Verdegay-López, Representaicón y Combinación de la Información con Incertidumbre mediante Convexos de Probabilidades, Ph.D. thesis, Universidad de Granada, 1997.
[51] P. Walley, The elicitation and aggregation of beliefs, Technical report, University of Warwick, 1982.
[52] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman & Hall, London, 1991.
[53] P. Walley, Inferences from multinomial data: learning about a bag of marbles (with discussion), J. Roy. Stat. Soc., Series B 58 (1996) 3–57.
[54] P. Walley, Measures of uncertainty in expert systems, Artif. Intell. 83 (1996) 1–58.
[55] P. Walley, A bounded derivative model for prior ignorance about a real-valued parameter, Scand. J. Stat. 24 (1997) 463–483.
[56] P. Walley, General introduction to imprecise probabilities http://ensmain.rug.ac.be/~ipp/documentation/introduction/introduction.html, 1997/98.
[57] P. Walley, Towards a unified theory of imprecise probability, Int. J. Approx. Reason. 24 (2–3) (2000) 125–148.
[58] Z. Wang, G.J. Klir, Fuzzy Measure Theory, Plenum Press, New York, 1992.
[59] K. Weichselberger, The theory of interval-probability as a unifying concept for uncertainty, Int. J. Approx. Reason. 24 (2–3) (2000) 149–170.
[60] P.M. Williams, Indeterminate probabilities, in: M. Przelecki, K. Szaniawski, R. Wojcicki (Eds.), Formal Methods in the Methodology of Empirical Sciences, D. Reidel, Dordrecht, MA, 1976, pp. 229–249.
[61] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems 1 (1978) 3–28.
[62] N.L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, Int. J. Intell. Res. 5 (1996) 301–328.