Evidence Propagation in Credal Networks: An Exact Algorithm Based on Separately Specified Sets of Probability

José Carlos F. da Rocha[1,2] and Fabio G. Cozman[1]

[1] Escola Politécnica, Universidade de São Paulo
Av. Prof. Mello Moraes, 2231
05508-900, São Paulo,SP, Brazil
fgcozman@usp.br
http://www.poli.usp.br/p/fabio.cozman.html
[2] Deinfo, Universidade Estadual de Ponta Grossa
84010.790 Ponta Grossa, PR, Brazil
jrocha@uepg.br
http://www.deinfo.uepg.br/jrocha

Abstract. Probabilistic models and graph-based independence languages have often been combined in artificial intelligence research. The Bayesian network formalism is probably the best example of this type of association. In this article we focus on graphical structures that associate graphs with sets of probability measures — the result is referred to as a *credal network*. We describe credal networks and review an algorithm for evidential reasoning that we have recently developed. The algorithm substantially simplifies the computation of upper and lower probabilities by exploiting an independence assumption (*strong independence*) and a representation based on separately specified sets of probability measures. The algorithm is particularly efficient when applied to polytree structures. We then discuss a strategy for approximate reasoning in multi-connected networks, based on conditioning.

1 Introduction

Graphical languages associated with probabilistic models have been commonly used in artificial intelligence to represent independence assumptions and to reason about uncertainty. In this context, Bayesian networks are the most popular formalism [17]. A Bayesian network is a directed acyclic graph where each node represents a random variable and each arc denotes a direct probabilistic dependency. Each node X contains a table, called a conditional probability table (CPT), that stores the conditional distribution $p(X|pa(X))$, where $pa(X)$ denotes the parents of X in the graph. A Bayesian network encodes a single joint distribution over all variables in the network [6]. Inference techniques such as the junction tree algorithm [15] and variable elimination [10] produce posterior marginal probabilities for any variable in the network.

What if an agent wishes to handle imprecision and uncertainty about the numerical parameters of CPTs? This can be useful when: i) the members of a group of experts involved in a knowledge engineering process intensely disagree about some CPT values; ii) the agent wants to represent and reason about beliefs that are incomplete or imprecise (for example, "the probability of rain is between 0.2 and 0.3"); iii) a number of experiments was made to estimate CPTs, but those experiments were not enough to obtain point estimates for every CPT entry; iv) the agent wants to perform sensitivity analysis to study how inferences are affected by variations in parameters. In some cases it may be possible or profitable to introduce hierarchical priors or second-order probabilities; here instead we focus on situations where sets of probabilities are used to model imprecision and uncertainty on probability values.

Credal networks associate graphical models with sets of probabilities [2, 8, 19]. The topology of a credal network can be used to represent several concepts of conditional independence; here we employ the concept of *strong independence*. Currently, algorithms for inference in credal networks with strong independence and general topology are very limited. Existing algorithms essentially search exhaustively over all joint probability distributions that can be generated from the sets of probability measures.

This article explores an strategy to implement credal network inference more efficiently than exhaustive search. The idea is first to encode credal networks using a representation that is directly connected to the way credal networks are usually specified in practice, and then to use this representation to reduce computations. We focus on sets of probability measures that are *separately specified*, as defined later. We have developed an algorithm, called *separable variable elimination* (SVE), that uses such sets to obtain computational gains. We have presented a thorough derivation of SVE elsewhere [18]. The objective of this paper is to present a detailed description of some inner operations of SVE and to discuss tests with random networks. We also investigate approximations that can be used in multi-connected networks.

The text is organized as follows. Section 2 introduces credal sets and the strong independence concept. Section 3 presents the credal network formalism. Section 4 describes the SVE algorithm and the propagation of evidence; experiments are described as well. Section 5 shows how SVE can be used to approximate reasoning in credal networks using conditioning operations. Finally, Section 6 indicates future efforts.

2 Credal sets

A credal set, denoted by $\boldsymbol{Q}(X)$, is a closed convex set of probability measures associated with a random variable X. Usually, a credal set is represented by its extremes distributions $p_1(X)$, $p_2(X)$,...$p_n(X)$. This article deals only with closed convex sets of probability measures. Note that $p_i(X)$ are points in polytopes and indicate probability distributions. Given a credal set $\boldsymbol{Q}(X)$, a common problem is to compute bounds for probability of events. These bounds are the *lower* *probability*, $\underline{P}(X = x) = \min_{p(X)\in \boldsymbol{Q}(X)} P(X = x)$ and the *upper probability* $\overline{P}(X = x) = \max_{p(X)\in \boldsymbol{Q}(X)} P(X = x)$.

A conditional credal set $\boldsymbol{Q}(X|Y = y)$ contains conditional probability densities $p(X|Y = y)$. Conditioning is defined as elementwise application of Bayes rule over the joint credal set $\boldsymbol{Q}(X, Y)$. Note that given sets $\boldsymbol{Q}(X|Y = y)$ and $\boldsymbol{Q}(Y)$, there may be more than one set $\boldsymbol{Q}(X, Y)$ that produces $\boldsymbol{Q}(X|Y = y)$ and $\boldsymbol{Q}(Y)$. Any such set $\boldsymbol{Q}(X, Y)$ is called an *extension* of $\boldsymbol{Q}(X|Y = y)$ and $\boldsymbol{Q}(Y)$.

For the purposes of this paper, there are two ways to represent conditional credal sets for a variable X given variable Y. First, we may store a collection of CPTs (that is, a list of functions as $p_1(X|Y), \ldots, p_m(X|Y)$). Such a set is denoted by $L(X|Y)$ and is called an *extensive* conditional credal set. An alternative representation is to store a collection of credal sets, one for each value of Y (that is, a list of sets $\boldsymbol{Q}(X|Y = y)$). When a conditional credal set is represented as a list of credal sets, we denote it by $\boldsymbol{Q}(X|Y)$ and call it *separately specified.* As we show later, separately specified credal sets lead to significant computational savings. A discussion of terminology and theoretical properties of extensive and separately specified credal sets is provided in [18].

In this paper we adopt the concept of *strong independence* [7]. Two variables X and Y are strongly independents when every vertex $p(X, Y)$ of $\boldsymbol{Q}(X, Y)$ satisfies stochastic independence of X and Y.[3] We adopt the following direct generalization of strong independence: Variables X and Y are strongly independent given Z when the sets $\boldsymbol{Q}(X, Y|Z = z)$ have extreme points satisfying stochastic independence of X and Y for every value of Z.

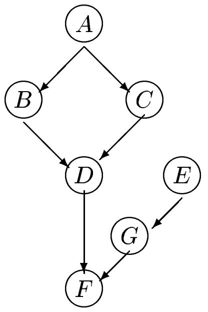
3 Credal networks

A credal network is a graphical structure that associates directed acyclic graphs with credal sets. Figure 1 depicts an example of credal network with separately specified sets.

Given the graphical structure and the credal sets in a credal network, we must decide how to combine the "local" credal sets into "joint" credal sets. In this paper we assume that separately specified credal sets are combined by "concatenating" the various distributions. That is, suppose that a separately specified credal set $\boldsymbol{Q}(X|Y)$ is given. We can produce an extensive credal set by taking every function $p(X|Y)$ that can be generated satisfying $p(X|Y = y) \in \boldsymbol{Q}(X|Y = y)$. An an example, consider variable F in Figure 1: There are four possible values for (E, G); the credal set produced by concatenating $\boldsymbol{Q}(F|E, G)$ contains 2^4 vertices. (Technical definitions of concatenation operators are given in [18].)

In this article, we adopt the convention that, in a credal network, every variable is strongly independent of its nondescendants nonparents given its parents — that is, we adopt a Markov condition on the network [17] and we use strong

[3] That is, given $\boldsymbol{Q}(X, Y)$, each vertex of this set satisfies $p(X|Y) = p(X)$ and $p(Y|X) = p(Y)$.



$P(a_0) \in [1/2, 3/5]$
$P(b_0|a_0) \in [1/2, 3/5]$
$P(b_0|a_1) \in [2/5, 1/2]$
$P(c_0|a_0) \in [1/5, 4/5]$
$P(c_0|a_1) \in [1/10, 1/2]$
$P(d_0|b_0, c_0) \in [1/10, 9/10]$
$P(d_0|b_0, c_1) \in [1/51/2]$
$P(d_0|b_1, c_0) \in [3/10, 1/2]$
$P(d_0|b_1, c_1) \in [1/10, 1/2]$
$P(e_0) \in [1/5, 11/20]$
$P(g_0|e_0) \in [1/51/2]$
$P(g_0|e_1) \in [1/7, 1/5]$
$P(f_0|d_0, g_0) \in [1/8, 9/10]$ $P(f_0|d_0, g_1) \in [3/4, 9/10]$
$P(f_0|d_1, g_0) \in [3/7, 3/4]$ $P(f_0|d_1, g_1) \in [1/10, 1/2]$

Fig. 1. Example credal network: all variables are binary, except C, which is ternary.

independence in this Markov condition. Given this assumption, the vertices of the joint credal set must satisfy stochastic independence [8]. This joint credal set is called the *strong extension* of the credal network, and is the most common graphical model associated with sets of probabilities [2]:

Definition 1 *The* strong extension *of a credal network is the convex hull of probability densities that satisfy the Markov condition on the network:*

$$\mathbf{Q}(\mathbf{X}) = \mathsf{CONVEXHULL}\begin{pmatrix} \prod_i p(X_i|pa(X_i)) \quad : \\ p(X_i|pa(X_i)) \in L(X_i|pa(X_i)) \end{pmatrix}. \tag{1}$$

We assume that $L(X_i|pa(X_i))$ is always given by concatenating $\boldsymbol{Q}(X_i|pa(X_i))$; we say the strong extension is *separately specified*. Given a credal network where every combination of variables has positive lower probability, any graphical d-separation relation in the credal network corresponds to a valid conditional independence relation in the strong extension network [8].

Several authors have proposed algorithms for inferences with strong extensions. Here *inference* means the computation of lower or upper probabilities (given evidence or not). There have been exact [8, 12] and approximate [3, 4, 14, 19] algorithms; a review of such algorithms is provided in [18].

The main difficulty with strong extensions is the potentially enormous number of vertices that must be explored to find probability bounds. Inference is a NP-complete problem with credal networks, even for networks that are based on polytrees [18]. Several techniques can be used to organize this search for bounds, as discussed in [18]. But not all vertices of the joint credal set $\boldsymbol{Q}(\mathbf{X})$ may affect the posterior set of interest $\boldsymbol{Q}(X_q|\mathbf{E})$ (where X_q is the variable of interest and $\mathbf{E}$ is the evidence in the network). Two factors can lead to removal of vertices: (1) some credal sets may not affect bounds due to d-separation, and (2) some vertices of $\boldsymbol{Q}(\mathbf{X})$ may lead to redundant points of $\boldsymbol{Q}(X_q|\mathbf{E})$. We assume that, before running any algorithm, d-separation relations are used to discard variables that do not affect computations; consequently, we focus here on the problem of eliminating redundant points of $\boldsymbol{Q}(X_q|\mathbf{E})$.

Suppose then that we have a separately specified strong extension; can we use the fact that sets are separately specified to simplify the detection/removal of redundant points? The answer is yes, and leads to the SVE algorithm, described in the next section. Correctness proofs are contained in [18]. It seems that SVE is the first exact algorithm that explores the separability of sets, even though the important work of Cano and Moral [5] has explicitly used separability in approximate algorithms and suggested the use of separability in exact algorithms.

4 Separable variable elimination

To understand the use of separability in the SVE algorithm, consider first the variable elimination algorithm [10, 9, 20]. This algorithm produces inferences for standard Bayesian networks. Variable elimination essentially builds a tree of *buckets* and passes messages from the leaves to the root of the tree. Each bucket is responsible for "eliminating" a variable X_j. Elimination occurs by selecting all functions that contain X_j and summing out X_j from the product of these functions. The result of this operation is a new function that replaces all functions in the bucket and is sent to some other bucket. This message is called *separator*. When message passing finishes, the root bucket contain the distribution of the queried variables. If we perform variable elimination with strong extensions, we must send messages that contains lists of vertices; that is, the separators become *sets of functions*.

A possible way to simplify inferences with credal sets would be to discard non-extreme vertices of separators; this idea has been suggested several times [14, 19]. Unfortunately, separators are commonly defined over high-dimensional spaces; even though convex hull and redundancy elimination algorithms are efficient in low dimensions [11], in high dimensions the complexity of convex hull algorithms is very high and the number of points actually discarded tends to be very low.

A reasonable strategy then is to always deal with separately specified credal sets, without ever turning them into the description $L(X_i|pa(X_i))$. This is the idea of the *separable variable elimination* algorithm, described next. Denote an arbitrary bucket by B. Any message sent by B is a set of functions $B(\mathbf{X}|\mathbf{Z})$, where $\mathbf{Z}$ indicates the conditioning variables.

Separable variable elimination algorithm (SVE)

- Run variable elimination, but keep messages $B(\mathbf{X}|\mathbf{Z})$ sent by buckets as separately specified sets with respect to $\mathbf{Z}$.
- Before a message $B(\mathbf{X}|\mathbf{Z})$ is sent by a bucket, run redundancy elimination *in each one of the sets* $B(\mathbf{X}|\mathbf{Z}=\mathbf{z})$ separately, for each value of $\mathbf{Z}$.

The central difference between SVE and standard variable elimination (with set-based separators) is that SVE operates on any bucket by taking one value of $\mathbf{Z}$ at a time. From an algorithmic point of view, this difference is apparent when performing products and sums inside buckets. These critical sections of the SVE algorithm are now described.

The SVE implementation of product and sum-out operations

- Input for bucket B:
  - The bucket variable X_B.
  - The separately specified credal sets $\boldsymbol{Q}(\mathbf{X}_j|\mathbf{Y}_j, \mathbf{Z}_j)$ received by B from other buckets and taken from the network CPTs. Note: Define $\mathbf{Z}_j$ so that any of its elements is a variable that is always in the conditioning side of any function where it appears. Define $\mathbf{Y}_j$ so that any of its elements are conditioning in some functions but conditioned in others.
- Output the separately specified separator $\mathbf{S} = \boldsymbol{Q}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Note: $\mathbf{X} = \cup\mathbf{X}_j \backslash X_B$, $\mathbf{Y} = \cup\mathbf{Y}_j \backslash X_B$ and $\mathbf{Z} = \cup\mathbf{Z}_j \backslash \{\mathbf{X} \cup \mathbf{Y}\}$.

For each instantiation $z \in \mathbf{Z}$:

1. collect every separately specified credal set where some variable conditioning matches with z or with X_B
2. collect every separately specified credal set where X_B is a conditioned variable
3. for each possible combination of functions collected in the previous two steps, construct a combined function v as follows:
   (a) concatenate every function associated to v; call the concatenated function v';
   (b) sum out X_B from v' to finally obtain the function v;
   (c) add v to $\mathbf{S}$ given z (v is an entry of $\mathbf{S}$ given z).
4. remove redundant elements of $\mathbf{S}$ given z running a convex hull algorithm.

We have noticed that, the fact that SVE keeps credal sets in separable form makes it more efficient (in terms of space, *even without redundancy elimination*) than algorithms that use extensive credal sets. Many networks can only be solved by SVE while failing with representations that directly use extensive credal sets.

To give an example of the SVE's gains in these operations, consider the network in Figure 1 and suppose each variable is ternary. Furthermore, assume that each separately specified credal set has three vertices. The order of variable elimination is E, G, A, C, D, E, F. Now take the bucket for variable G, denoted for B_G, and consider the operation *sum-out* on this bucket. In the calculation of the B_G's separator in SVE, the separately specified sets connected to $\boldsymbol{Q}(F|D, G)$ and to the separator from E's bucket, $\mathbf{S}(E)$, are present. The separator of B_G is $\mathbf{S}(G) = \boldsymbol{Q}(F|D)$ and it is composed of three separately specified credal sets each one with 3^4 vertices. The total number of distributions summing all sets is 3^5. In the standard variable elimination, this procedure processes 3^{13} functions. Additionally, this gain is propagated forward.

In our experiments, we have seen several examples in which the redundancy elimination step leads to a significant decrease in computational effort [18]. We have noticed that in general the reduction of vertices is not a dramatic factor in inference; moreover, the cost of redundancy elimination is not negligible and sometimes outweights its benefits.

Consider the following tests of SVE in the networks in Figures 1 (where we asked for the marginal of variable F) and 2 (where we asked for the marginal of variable L).[4] The first test is simply to run these networks with binary variables and with two vertices in each credal set. Both standard variable elimination (with separators containing sets of functions) and SVE obtained results in this case. Now, when we tried to deal only with ternary variables in the networks standard variable elimination failed. That is, standard variable elimination could not handle inference in these credal networks with ternary variables. But SVE dealt with both networks with ternary variables when credal sets had two vertices. When credal sets had three vertices, inferences with the network in Figure 1 failed, while inferences with the network in Figure 2 succeeded. Note that the network in Figure 2 has a strong extension with 2^{18} potential vertices; SVE succeeds basically because this network is a polytree and separators never have high dimensionality.

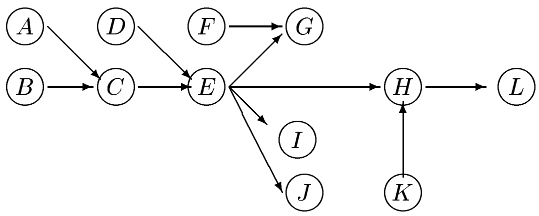


Fig. 2. An polytree credal net.

We remark that SVE can easily handle a body of evidence $\mathbf{E}$, just as standard variable elimination. After removing unnecessary variables by d-separation, SVE can handle messages from credal sets $\boldsymbol{Q}(X_i|pa(X_i))$ where any of X_i or $pa(X_i)$ belongs to $\mathbf{E}$.

[4] We run these networks with randomly generated credal sets (that is, the vertices of the sets are randomly generated conditional distributions). We have implemented SVE in Java, and have run the tests with Java San 1.4.0 interpreter with 72M of heap and 6M of stack, using a HP Pentium 4 with Windows 2000.

5 Approximated reasoning

Because SVE is well suited for networks that have a polytree structure, it would seem that inferences based in loop cutsets [17] could be a profitable way to handle multi-connected networks. We could consider conditioning operations as a general way to reduce the effort spent in inference.

The main idea here is to use loop cutset algorithms to transform complex credal networks, not necessarily multi-connected, in several simpler nets. In the context of credal networks, the use of cutsets produces only approximations, not exact inferences [12]. This happens because loop cutset produces several values for upper and lower probabilities (one for each combination of variables in cut set), and direct combination of these values is not equivalent to global lower and upper values. The resulting values are enclosing approximations (that is, the correct bounds are inside of approximated bounds).

We denote the loop cutset by C and the conditioned upper bounds, obtained by conditioning, as :

$$\overline{P}(X|C) = (\max_{p(X|C) \in \boldsymbol{Q}(X|C)} P(X = x_1|C), ..., \max_{p(X|C) \in \boldsymbol{Q}(X|C)} P(X = x_m|C)) \quad (2)$$

where $(x_0, ..., x_m)$ is an exhaustive enumeration of the X's states. To combine these bounds, we need to have probabilities for the variables in the cutset. If an inference is requested without evidence, it is easy to obtain the cutset probabilities when the cutset variables do not have parents.

Consider an example. If we take the network in Figure 1 with binary variables and two points for separately specified credal set, we obtain the exact upper bounds $\overline{P}(F) = (0.4953; 0.7767)$. Then, if we set $C = \{A\}$, the conditioned bounds are $\overline{P}(F|A) = (0.4964; 0.7777)$. By combining these bounds with $\boldsymbol{Q}(A)$ we obtain an approximation that matches with exact values up to the fourth decimal digit.

It may be difficult to combine conditioned bounds in more complicated topologies. The key point to apply that technique depends on following question: is it easy to obtain the loop cutset's credal set? If the answer is no, an approach is to use the conditioned bounds as an approximation. That is, instead of trying to combine them for each instantiation, we can obtain the upper values for each state of cutset variables, and then simply take the largest and smallest values respectively. The resulting values are enclosing approximations, too. In previous example (Figure 1) the approximated bounds for F are $\overline{P^*}(F) = (0.4964; 0.7777)$.

Consider a few other examples. We compute a query about the variable L in the network of the Figure 2, with ternary variables and 4 points by credal set. That query could not be computed by SVE. But, using the loop cutset in D, K, and this scheme, we obtain the next upper bounds $\overline{P^*}(L) = (0.54; 0.524; 0.459)$. Figure 3 brings two more examples, one with binary variables (left) and one with ternary variables (right). In both networks every variable is associated with credal sets that have two vertices, except that in the last network the variables E, G, C and F are associated with a single distribution. In the left network, we selected the cutset $\{B, C\}$ and asked for the marginal distribution of N. The exact results are: $\underline{P}(N) = (0.4784; 0.2577)$ and $\overline{P}(N) = (0.7422; 0.52159)$; the approximated results are: $\underline{P^*}(N) = (0.4779; 0.2569)$ and $\overline{P^*}(N) = (0.7430; 0.5220)$. In the right network we computed the lower and upper values for variable C, for each instantiation of the cutset variables $\{A, B\}$; we took just the maxima and minima to illustrate the application of Expression (2). The exact results are: $\underline{P}(C) = (0.4011; 0.4080; 0.2699)$ and $\overline{P}(C) = (0.3398; 0.3705; 0.2185)$; the approximated results are: $\underline{P^*}(C) = (0.5410; 0.4551; 0.3201)$[5] and $\overline{P^*}(C) = (0.3173; 0.3223; 0.1366)$ (showing the lower quality of this approximation).

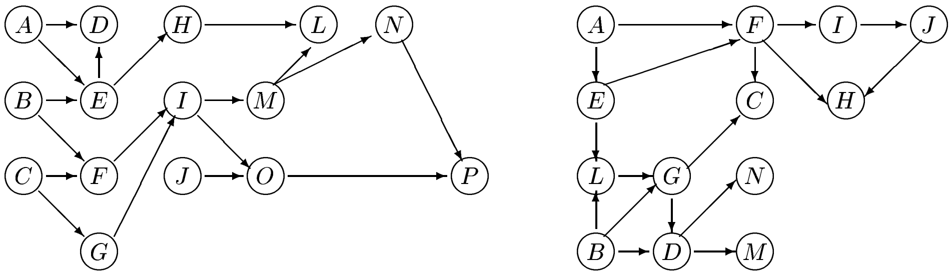


Fig. 3. Examples: credal network with binary variables (left) and with ternary variables (right).

6 Concluding remarks

In this paper we have described SVE, an algorithm for inferences in credal networks. Our goal was describe implementation and test aspects of SVE, so as to complement the theoretical derivation of SVE presented in [18]. We have also discussed a different dimension of SVE, by exploring its use in connection with loop cutset algorithms. We have proposed a scheme for generating approximations through conditioning operations — namely, by taking maxima and minima across the values of the conditioned variables.

Experiments show that SVE is an significant advance when compared to existing exhaustive methods. The improvement obtained by SVE is derived from its reliance on separately specified credal sets. We are now investigating efficient heuristic methods to reduce the cost of convex hull operations through divide and conquer algorithms.

Acknowledgements

We thank CAPES and CNPq for partially supporting the first and second authors respectively. Thanks also to Marsha Duro from HP Labs, Edson Nery from

[5] To generate approximations for lower probabilities, substitute min for max in Expression (2).

HP Brasil and Instituto de Pesquisas Eldorado, for providing equipment that was used in tests.

References

1. Campos, L. M. de, Moral, S.: Independence concepts for convex sets of probabilities. Proc. of the XI Conf. on Uncertainty in Artificial Intelligence, Montreal, Canada (1995) 108–115.
2. Cano, J., Delgado, M., Moral, S.: An axiomatic framework for propagating uncertainty in directed acyclic networks. International Journal of Approximate Reasoning, (1993) 8:253-280.
3. Cano, A., Cano, J.E., Moral, S.: Convex sets of probabilities propagation by simulated annealing. IPMU'94, Paris France (1994) 978–983
4. Cano, A., Moral, S.: A genetic algorithm to approxiamte convex sets of probabilities. IPMU'96, Vol. 2, (1999) 859–864
5. Cano, A., Moral, S.: Using probabilities trees to compute marginals with imprecise probabilities. TR-DECSAI-00-02-14, University of Granada, (2000)
6. Charniak, E.: Bayesian Networks without tears. AI Magazine, Vol.12. **4** (1994) 50–63.
7. Couso, I. Moral, S., Walley, P.: Examples of independence for imprecise probabilities. 1st ISIPTA, Ghent Belgium (1999).
8. Cozman, F.G.: Credal networks. Artificial Intelligence (2000) 1–35.
9. Cozman, F.G.: Generalizing variable elimination in Bayesian networks. Workshop on Probabilistic Reasoning in Artificial Intelligence - IBERAMIA/SBIA, Atibaia, Brazil (2000).
10. Dechter, R.: Bucket elimination: A unifying framework for probabilistic inference. Proc. of the XII Conf. on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA (1996) 211–219.
11. Edelsbrunner, H.: Algorithms in combinatorial geometry. Springer-Verlag (1987).
12. Fagiuoli, E., Zaffalon, M.: 2U: an exact interval propagaion algorithm for polytrees with binary variables. AI, Elsevier, **106** (1998) 77–107.
13. Giron, F.J., Rios, S.: Quasi-Bayesian behavior: A more realistic approach to decision making? in: Bernado, J.M., DeGroot, J.H., Lindley, J.H. Smith, A.F.M. (eds.) Bayesian Statistics, University Press, Valencia, (1980) 17–38.
14. Ha, V.A. *et al*: Geometric foundations for interval-based probabilities. Annals of Mathematics and Artificial Intelligence, Vol.24, **1-4** (1998) 1–21
15. Jensen, F.V., Olensen,K.G., Andersen, S.K.: An algebra of Bayesian belief universes for Knowledge-Based Systems. Networks, Vol. 20. (1990) 637–659.
16. Neapolitan, R.E.: Probabilistic reasoning in expert systems. Theory and algorithms. John Willey and Sons, New York, (1990).
17. Pearl, J.: Probabilistic reasoning in intelligent systems. Networks of Plausible Inference. Morgan Kaufmann Publishers, San Mateo CA (1988).
18. Rocha, J. C. F. da, Cozman, F. G.: Inference with Separately Specified Sets of Probabilities in Credal Networks, Proc. of the Conf. on Uncertainty in Artificial Intelligence, Edmonton, Canada, (2002).
19. Tessem, B.: Interval Probability propagation. Int. Journal of Approximated Reasonin **7**, (1995) 95–120.
20. Zhang, N.L., Poole, D. Explointing causal independence in Bayesian networks inference. Journal of Artificial Intelligence Research, **5** (1996) 301-328.