Inference in Credal Networks: Branch-and-Bound Methods and the A/R+ Algorithm

José Carlos Ferreira da Rocha

Dept. de Informática, Universidade Estadual de Ponta Grossa
Ponta Grossa, PR - Brazil
jrocha@uepg.br

Fabio Gagliardi Cozman

Escola Politécnica, Universidade de São Paulo
Av. Prof. Mello Moraes 2231, Cidade Universitária
São Paulo, SP - Brazil
fgcozman@usp.br

ABSTRACT

A credal network is a graphical representation for a set of joint probability distributions. In this paper we discuss algorithms for exact and approximate inferences in credal networks. We propose a branch-and-bound framework for inference, and focus on inferences for polytree-shaped networks. We also propose a new algorithm, A/R+, for outer approximations in polytree-shaped credal networks.

International Journal of Approximate Reasoning 1994 11:1–158
© 1994 Elsevier Science Inc.
655 Avenue of the Americas, New York, NY 10010 0888-613X/94/$7.00

1. Introduction

A *credal network* provides a graphical representation for imprecision in probabilistic statements [10, 12, 19].[1] Such graphical models can be viewed as Bayesian networks with relaxed numerical parameters: each node in the graph represents a random variable, and each variable is associated with a set of probability measures — the "size" of the sets of probability measures encodes the imprecision in probability values. Such a model can be used to study robustness of probabilistic models, to investigate the behavior of groups of experts, or to represent incomplete or vague knowledge about probabilities [28].

An *inference* with a credal network is typically understood as the computation of upper and lower probabilities for each category of a *query* variable. This calculation, under the most commonly adopted semantics for credal networks (using *strong extensions*), is NP-hard even for polytree-shaped networks [15]. Exact and approximate algorithms have been proposed in the literature, but no algorithm can handle large credal networks exactly — even networks with a few nodes can present unsurmountable difficulties.

In this paper we propose new algorithms for inference in credal networks. The central idea is to use branch-and-bound techniques to search for upper and lower probability values. We also propose a new algorithm for outer approximations in polytree-shaped credal networks, which we call A/R+. This new algorithm modifies Tessem's A/R algorithm to produce significantly better approximations.

Sections 2 and 3 review elements of the theory of credal networks. Section 4 presents a branch-and-bound framework for inferences in credal networks. Sections 5 and 6 focus on polytree-shaped credal networks: Section 5 introduces the A/R+ algorithm and Section 6 describes a number of branch-and-bound techniques applied to polytree-shaped credal networks.

2. Credal sets and credal networks

A set of probability distributions for variable X, called a *credal set*, is denoted by $K(X)$ [23]. In this paper we assume that every variable is categorical and that every credal set is closed and convex with a finite number of vertices. A conditional credal set is a set of conditional distributions, obtained applying Bayes rule to each distribution in a credal set of joint distributions [28]. The credal sets $\{K(X|Y=y) : y \text{ is a category of } Y\}$

[1] See the overview paper on graphical models for imprecise probabilities in this issue.

are *separately specified* when there is no constraint on the conditional set $K(X|Y = y_1)$ that is based on the properties of $K(X|Y = y_2)$, for any $y_1 \neq y_2$ — that is, the conditional sets bear no relationship to each other. A collection of separately specified credal sets for X conditional on Y is denoted by $K(X|Y)$. Given a collection of marginal and conditional credal sets, an *extension* of these sets is a joint credal set with the given marginal and conditional credal sets.

Given a credal set $K(X)$ and a function $f(X)$, the *lower* and *upper* expectations of $f(X)$ are respectively $\underline{E}[f(X)] = \min_{p(X)\in K(X)} E_p[f(X)]$ and $\overline{E}[f(X)] = \max_{p(X)\in K(X)} E_p[f(X)]$ (here $E_p[f(X)]$ indicates standard expectation). *Lower* and *upper probabilities* are defined similarly.

A *credal network* is a directed acyclic graph where each node of the graph is associated with a variable X_i and with a collection of conditional credal sets $K(X_i|\mathrm{pa}(X_i))$, where $\mathrm{pa}(X_i)$ denotes the parents of X_i in the graph. Note that we have a conditional credal set for each category of $\mathrm{pa}(X_i)$. A root node is associated with a single marginal credal set.

We take that in a credal network every variable is independent of its nondescendants conditional on its parents. In this paper we adopt the concept of *strong independence*: two variables X and Y are strongly independent when every extreme point of $K(X, Y)$ satisfies standard stochastic independence of X and Y (that is, $p(X|Y) = p(X)$ and $p(Y|X) = p(Y)$) [12]. When necessary, we use *strong conditional independence*: X and Y are strongly independent conditional on Z when every extreme point of $K(X, Y|Z = z)$ satisfies standard stochastic independence conditional on every category z.

The *strong extension* of a credal network is the largest extension that satisfies the independence relations just discussed. That is, the strong extension is the convex hull of all joint distributions that satisfy the following Markov property: every variable is strongly independent of its nondescendants conditional on its parents [13]. Given a credal network with N variables, with local separately specified credal sets $K(X_i|\mathrm{pa}(X_i))$, the strong extension of the network is then the convex hull of the set of joint distributions

$$\left\{ p(\mathbf{X}) : p(\mathbf{X}) = \prod_{i=1}^{N} p(X_i|\mathrm{pa}(X_i)) \,, \right. \tag{1}$$

$$\left. p(X_i|\mathrm{pa}(X_i) = \rho_k) \text{ is a vertex of } K(X_i|\mathrm{pa}(X_i) = \rho_k) \right\}.$$

Note that a credal network can have several extensions [12]; in this paper we focus on strong extensions.

Figure 1 shows a small polytree-shaped credal network with separately specified credal sets. Variables X_1, X_2, X_4 and X_5 are binary and variable

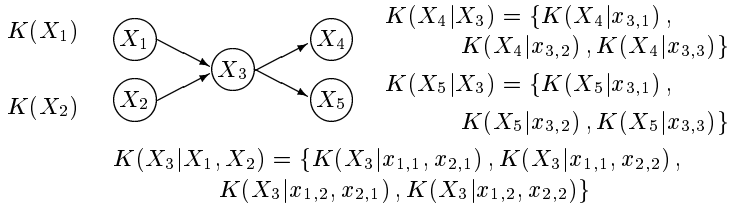

Figure 1. Credal network with separately specified credal sets.

X_3 has three categories. The symbol $x_{i,j}$ indicates the jth category of variable X_i.

3. Inferences and inference algorithms

A *marginal inference* in a credal network is the computation of lower/upper probabilities in an extension of the network. If X_q is a *query* variable and $\mathbf{X}_E$ represents a set of *observed* variables, then an inference is the computation of tight bounds for $p(X_q|\mathbf{X}_E)$ for one or more categories of X_q. The only known polynomial algorithm for strong extensions is the 2U algorithm, which processes polytrees with *binary* variables [19]. Other than this "pocket" of tractability, all other situations seem to offer tremendous computational challenges. In particular, inference is a NP-hard problem *even for polytrees* [15].

For inferences in strong extensions, the distributions that minimize or maximize $p(X_q|\mathbf{X}_E)$ are vertices of the extension [19]. As these vertices are combinations of vertices of local credal sets (Expression (1)), the problem we face is one of combinatorial optimization: we must find a vertex for each local credal set $K(X_i|\mathrm{pa}(X_i))$ so as to maximize/minimize $p(X_q|\mathbf{X}_E)$. Thus one can generate inferences by enumerating all combinations of vertices [5],[2] or possibly exploring additional structure in separately specified credal sets [9, 15]. In any case, enumeration algorithms can be understood as methods that perform probabilistic propagation (much in the style of variable elimination or junction tree algorithms) by exchanging *set-valued messages* during the propagation. Enumeration/set-propagation algorithms can be

[2]The enumeration method proposed by Cano, Cano and Moral [5] has been implemented in the JavaBayes system version 0.347 (a freely available inference engine distributed at http://www.pmr.poli.usp.br/ltd/Software/javabayes).

used in relatively small networks, and can be dwarfed even in seemingly trivial models.

We can also view an inference as a continuous optimization problem. Consider the computation of a posterior upper probability. The goal is to maximize the probability value

$$p(X_q = x_{q,k}|\mathbf{X}_E) = \frac{\sum_{X_1,\ldots,X_N \setminus \{X_q,\mathbf{X}_E\}} \prod_i p(X_i|\text{pa}(X_i))}{\sum_{X_1,\ldots,X_N \setminus \mathbf{X}_E} \prod_i p(X_i|\text{pa}(X_i))}, \qquad (2)$$

by finding a distribution $p(X_i|\text{pa}(X_i))$ in each local credal set $K(X_i|\text{pa}(X_i))$ (for each variable X_i and each category of $\text{pa}(X_i)$). Given our assumption of credal sets with finitely many vertices, the maximization is subject to linear constraints.

This maximization problem belongs to the field of *signomial* programming [2], as first observed by Andersen and Hooker [1]. Signomial programs are generally solved by dividing the feasible set ("branching" on various subsets) and obtaining outer approximations ("bounding" the objective function in each subset) [2, 18]. That is, signomial programming is solved by branch-and-bound procedures. The great advantage of signomial programming over more general optimization problems is that it is possible to obtain bounds for signomial programs using geometric programming — a well establish field that can be tackled efficiently through convex programming [2]. Note that strong extensions encode rather large signomial programs — the "degree of difficulty" of a geometric program depends on the number of polynomial terms in the program, and Expression (2) summarizes a large number of such terms.

Instead of resorting to enumeration techniques or signomial programming, a different approach for exact inference is to manipulate the values of the joint distribution $p(\mathbf{X})$ directly. That is, instead of optimizing Expression (2), one optimizes the linear fractional function

$$p(X_q = x_{q,k}|\mathbf{X}_E) = \frac{\sum_{X_1,\ldots,X_N \setminus \{X_q,\mathbf{X}_E\}} p(\mathbf{X})}{\sum_{X_1,\ldots,X_N \setminus \mathbf{X}_E} p(\mathbf{X})},$$

subject to *non-linear* constraints on $p(\mathbf{X})$. Note that the number of values of $p(\mathbf{X})$ is exponential on the number of variables in the network. Andersen and Hooker present a sophisticated algorithm to circumvent these difficulties, but their algorithm still requires signomial programming in auxiliary optimization problems [1].

Given the difficulties of exact inference, approximate inference algorithms have received considerable attention in the literature. We distinguish between *outer* and *inner* approximations; the former produce intervals that enclose the lower and upper probabilities, while the latter produce intervals that are enclosed by the correct probability interval. Tessem's A/R algorithm was the first one to produce tractable outer approximate inference

in polytree-shaped networks [26]. The A/R algorithm was later extended to general topologies by Ha et al [20]. Bounds have also been derived from approximate combination of credal sets [27] and from approximate representation of credal sets [9]. Inner approximations are usually produced with local optimization methods and can be found in [1, 6, 5, 11, 29]. Overviews of inference algorithms have been published by Cano and Moral [7, 8].

4. Branch-and-bound techniques for inference with strong extensions

In this section we discuss the use of branch-and-bound algorithms for inference with strong extensions. The idea is to view inference as a search: we must find vertices of local credal sets that maximize/minimize Expression (2). Even though we are inspired by signomial programming techniques, the idea here is to explore specific properties of strong extensions. To simplify the discussion, we focus only on the computation of upper probabilities; the computation of lower probabilities is analogous.

Consider a generic maximization problem

$$\begin{aligned} (P) \quad & \max f(w) \\ & \text{s.t.} \qquad g(w) \leq 0, w \in \mathbf{W}, \end{aligned}$$

where $\mathbf{W} \subseteq \Re^n$, f and g are a real valued functions. In the first iteration of a branch-and-bound algorithm, the problem P is divided in sub-instances that are easier to solve or approximate than P. The partitioning is made so that the solution for P is present in one of the sub-instances [24]. Each sub-instance P_i is then analyzed. If P_i can be quickly solved, then P_i is said to be *trivial*. If P_i is trivial, then the maximum of f in P_i is computed; if this value is the highest value so far, it is retained as the current solution. If P_i is non-trivial, then it is evaluated with a relaxed algorithm that produces outer and inner bounds for P_i. These bounds are compared with the current solution and:

- if it can be proved that the solution space of P_i cannot contain the global maximum, then P_i is discarded;
- otherwise, P_i is said to be *critical* and is solved through branch-and-bound.

This process is repeated for each sub-instance while there is a promising alternative.

A *depth-first* branch-and-bound selects a promising sub-instance as soon as it is generated. The sub-instance is partitioned into new sub-instances.

A promising sub-instance is then selected, and partitioned. The procedure continues this way until a trivial problem is reached or until all sub-instances of a partition are found not to be promising. At those points the algorithm executes a backtracking step — that is, the algorithm returns to a previously evaluated sub-instance and selects a still unexplored sub-instance. The advantage of this scheme is the minimal memory consumption, because a few sub-instances are stored in memory at any given time.

A *best-first* branch-and-bound stores (potentially many) sub-instances in a *heap* as it goes along.[3] Sub-instances stored in the heap are ordered according to their outer bounds: the top element in the heap has the maximum outer bound. The algorithm always takes the first critical sub-instance from the heap to be processed and possibly partitioned. When a sub-instance is selected, it is evaluated; if it is critical, it is partitioned and their critical sub-problems are inserted in the heap. Thus the heap contains every sub-instance that was considered critical when it was generated. The nodes in the heap can be understood as the "frontier" of all sub-instances still unexplored. The advantage of best-first branch-and-bound is that it allows us to improve approximations; we can gradually refine outer and inner bounds by looking at all the nodes in the "frontier" (see Section 6.3). The disadvantage of best-first search is the potentially enormous cost in memory (necessary to store the heap).

Figures 2 and 3 present a convenient and informal summary of the depth-first and best first branch-and-bound techniques. These descriptions were adapted respectively from [25] and [3]. In both procedures the functions r and s are used as a pruning mechanism, and r is used to drive the search as well.

Any branch-and-bound algorithm relies on two decisions: how to produce bounds, and how to decompose a problem into sub-problems. We now consider the computation of an upper probability $\overline{p}(X_q = x_{q,k}|\mathbf{X}_E)$ for a given credal network $\mathcal{N}$. The discussion of bounding methods is postponed to the next section; here we deal with the decomposition strategy.

Suppose an inference is to be computed for credal network $\mathcal{N}$. The root of the search tree is then associated with $\mathcal{N}$, and every node of the search tree is associated with a credal network derived from $\mathcal{N}$. Consider then a node of the search tree associated with credal network $\mathcal{N}'$. Select one of the credal sets in $\mathcal{N}'$, say $K(X_i|\mathrm{pa}(X_i) = \rho)$, and suppose this credal set has v vertices. Denote these vertices by p_k, $k = 1 \ldots v$. Now create v nodes of the search tree and place them as children of the original node. Each one of these children nodes is associated with a "children" network $\mathcal{N}'_j$. Network

[3] A heap is a data structure where elements are kept sorted according to some priority scale. The elements with higher priority are in the "beginning" of the heap; the element with maximum priority is always the first one to be removed from the heap.

Procedure dfbb(P):

1. If P is trivial and its maximum $f(w)$ is such that $f(w) > f_{aux}$, then $f_{aux} \leftarrow f(w)$ and return.
2. Otherwise, decompose P into a list L of sub-instances P_i. For each P_i, if $r(P_i) \geq s(P_j)$ for all P_j in L, and if $r(\mathrm{P}_i) > f_{aux}$, then call recursively **dfbb(P_i)**.

Figure 2. Depth-first branch-and-bound: input is the problem P; algorithm uses a global variable f_{aux} (initialized with $-\infty$), and functions r and s that compute respectively outer and inner bounds. In the end, $\max f(w)$ is stored in f_{aux}.

$\mathcal{N}'_j$ inherits all credal sets from $\mathcal{N}'$ *except* the set $K(X_i|\mathrm{pa}(X_i) = \rho)$; the credal set $K(X_i|\mathrm{pa}(X_i) = \rho)$ is replaced by distribution p_j. This decomposition procedure is then applied recursively, following the branch-and-bound algorithm. Using this decomposition strategy, there is a gradual "thinning" of the local credal sets. Any leaf node of the search tree contains a Bayesian network, obtained by a particular selection of vertices in local credal sets. When a leaf node is reached, the computation of probabilities is "trivial": a variable elimination algorithm is used to perform inference in the Bayesian network at the leaf [14].

This decomposition strategy depends on the selection of credal sets for expansion. We always select the non-expanded credal set nearest to the queried variable, but we always keep the query variable to be processed last (a similar criterion is used by Draper and Hanks [17] to deal with partial evaluation of belief nets). We have tried several criteria for selecting credal sets to expand, and we found that the procedure just described is quite appropriate.

5. Bounds and the A/R+ algorithm

The discussion so far has been completely general and can be applied to any credal network. In this section we concentrate on algorithms that produce outer and inner bounds for inferences with polytree-shaped credal networks (note that inference with polytree-shaped networks is already a NP-hard problem [15]). We again focus on upper probabilities; an outer bound for an upper probability is a number that is *larger* than the upper probability, while an inner bound for an upper probability is a number that is *smaller* than the upper probability.

Procedure bfbb(P):

1. $f_{max} \leftarrow -\infty$; initialize a heap OPEN with P.
2. Repeat:
   - **(a)** Remove an instance Q from OPEN (Q is thus the instance with the largest value of r in the heap);
   - **(b)** if Q is critical, decompose Q into a list L of sub-instances Q_i (sub-instance Q_i with feasible region $\mathbf{W}_i$), and for each Q_i:
     - i. If Q_i is trivial, calculate the maximum value of f in $\mathbf{W}_i$, denoted by f_i. If $f_i > f_{max}$ then $f_{max} \leftarrow f_i$.
     - ii. Otherwise, if $r(Q_i) \geq f_{max}$, insert Q_i in OPEN; if in addition $s(Q_i) > f_{max}$, then $f_{max} \leftarrow s(Q_i)$.
   - **(c)** If OPEN is empty, stop.

Figure 3. Best-first branch-and-bound: input is the problem P; algorithm uses functions r and s that compute respectively outer and inner bounds; the elements in OPEN heap are sorted with respect to their r bound. In the end, $\max f(w)$ is stored in f_{max}.

An inner bound can be produced by any method that maximizes Expression (2) up to a local maximum. Local optimization algorithms like gradient descent or the expectation-maximization algorithm can produce such inner bounds [11, 29]. It is also possible to obtain inner bounds with genetic programming, simulated annealing and similar methods [5, 6]. Generally these methods require tuning several parameters; we have implemented some of them and noticed that, while they produce reasonable solutions, they are far from easy to apply. We have thus developed a new algorithm for inner bounds using local search (reported elsewhere [16]). The idea is to fix all probabilities $p(X_i|\mathrm{pa}(X_i))$, except one — and then to find the maximizing value just for this "free" density, which can be done easily (the maximization problem has become a fractional linear one). Once the maximizing values for the "free" density are found, this density is fixed at its maximizing values, and a different density is "freed." Thus the algorithm follows a maximizing path where each movement changes only one density in the credal network, moving between vertices of a single local credal set at any given step. This algorithm produces excellent inner bounds (in many cases the exact upper probability is found) and runs rather quickly.[4]

Outer bounds for inference in polytree-shaped credal networks can be generated with Tessem's A/R algorithm [26]. The first assumption in

[4] The development of the algorithm for inner bounds was joint work with Cassio Polpo de Campos.

Tessem's algorithm is that every credal set is approximated by a collection of probability intervals. Such approximation is always possible (and always an outer approximation), as we can obtain the probability interval

$$\left[\min_{p(X)\in K(X)} p(x_j)\,,\ \max_{p(X)\in K(X)} p(x_j)\right] \tag{3}$$

for any category x_j of a variable X (and likewise for conditional probability values). Obviously the replacement of credal sets by probability intervals introduces potential inaccuracies into inferences.

Tessem's central idea was to generalize Pearl's belief propagation algorithm to accommodate probability intervals (in an approximate way). The functions λ and π used in belief propagation are still defined with identical purposes, but they are now interval-valued. Thus probability intervals for marginal probabilities $p(X)$ are computed from two interval-valued vectors $\pi(X)$ and $\lambda(X)$ that contain interval-valued versions of predictive and retrospective support for X. These vectors are computed from interval-valued "messages" that X receives from its parents and children. If Y and Z denote, respectively, a parent node and a child node of X, then the messages that X receives from those nodes are denoted by $\pi_X(Y)$ and $\lambda_Z(X)$.

The messages manipulated by the A/R algorithm are computed using interval arithmetic and two additional techniques called *annihilation* and *reinforcement* (thus the name A/R). To understand the mechanics of A/R, it is interesting to look at a particular operation, the computation of the interval-valued function π. Consider a function $\pi(X)$ to be computed at a node X with parents $Y_1, \ldots, Y_k$. Figure 4 shows the computation of the lower value $\pi_*(X)$ (the same operations can be adapted to compute the upper value $\pi^*(X)$).

The algorithm in Figure 4 initially combines the interval-valued messages that X receives from its parents into a "joint" interval-valued function β (step 1), using interval multiplication. Then the algorithm applies the *annihilation* operation (steps 3a and 3b) to determine a joint distribution $p(Y_1, \ldots, Y_k)$ that minimizes the sum of products in step 3c. Variable S' controls how much probability mass can be distributed during annihilation.

Tessem developed similar operations for computation of messages $\lambda_X(Y_i)$ and $\pi_{Z_i}(X)$ (where Z_i is a child of X). The A/R algorithm also employs direct interval multiplication to generate the function $\lambda(X)$. Finally, the algorithm uses annihilation or reinforcement operations to "normalize" the functions $\lambda_X(Y_i)$, $\pi_{Z_i}(X)$, and $\pi(X)\lambda(X)$ ("normalization" means simply computing bounds that account for the fact that probability distributions add up to one).

The A/R algorithm is clever, but it can be significantly improved as follows. Consider a new method to compute the interval-valued $\pi(X)$:

1. For each interval-valued message $\pi_X(Y_i)$ received by X, create a *credal*

Computing $\pi_*(X)$:

1. Construct the interval-valued function $\beta(Y_1, \ldots, Y_k)$ by interval-multiplication of the messages $\pi_X(Y_i)$ received by X (these messages are also interval-valued). Denote by $\beta_*(Y_1, \ldots, Y_k)$ and $\beta^*(Y_1, \ldots, Y_k)$ respectively the lower and upper values of $\beta(Y_1, \ldots, Y_k)$.
2. Initialize $S \leftarrow \sum_{Y_1,\ldots,Y_k} \beta_*(Y_1, \ldots, Y_k)$.
3. For each value x of X:
   - **(a)** Sort the lower probabilities $\underline{p}(X = x|Y_1, \ldots, Y_k)$ in increasing order. Denote by $I(z)$ the zth configuration of $(Y_1, \ldots, Y_k)$ in the ordering.
   - **(b)** Distribute probability mass consistently with $\beta(Y_1, \ldots, Y_k)$ as follows:
     i. Initialize $p(Y_1, \ldots, Y_k) \leftarrow \beta_*(Y_1, \ldots, Y_k)$ for all $(Y_1, \ldots, Y_k)$.
     ii. Initialize $c \leftarrow 1$, $S' \leftarrow S$.
     iii. While $S' < 1$:
       A. $p(I(c)) \leftarrow \min(\beta^*(I(c)), 1 - S' + \beta_*(I(c))$.
       B. $S' \leftarrow S' + \beta^*(I(c)) - \beta_*(I(c))$.
       C. $c \leftarrow c + 1$.
   - **(c)** $\pi(x) \leftarrow \sum_{Y_1,\ldots,Y_k} \underline{p}(x|Y_1, \ldots, Y_k)\, p(Y_1, \ldots, Y_k)$.

Figure 4. Computation of $\pi_*(X)$ in the A/R algorithm. The input is the set of lower probabilities $\underline{p}(X|\text{pa}(X))$ and the messages $\pi_X(Y_i)$ for $Y_i \in \text{pa}(X)$.

set $K_X(Y_i)$ that is the largest credal set with lower/upper probabilities represented by $\pi_X(Y_i)$. Such a credal set can be easily generated [4, 22].

2. Eliminate each parent Y_i by multiplying vertices of $K_X(Y_i)$ and vertices of $K(X|Y_1, \ldots, Y_k)$, and summing out the parents Y_i.
3. Use the resulting credal set $K(X)$ to produce probability intervals $\pi(X)$ through Expression (3).

Step 2 performs an exact combination of vertices obtained from received intervals and local distributions. This procedure operates with credal sets, then collapses potentially complex credal sets into probability intervals, locally using credal sets and propagating interval messages.

The operations just discussed can be extended to other interval-valued

messages used in belief propagation. For example, messages $\pi_{Z_i}(X)$ are computed using similar operations as for $\pi(X)$. We thus obtain the *A/R+* algorithm:

> *Run all the steps of the A/R algorithm, but whenever interval-valued messages must be multiplied, convert the messages into credal sets, operate with the credal sets locally, and convert the results back to probability intervals.*

The basic fact about A/R+ is that (proof in the Appendix):

THEOREM 1. *Interval-valued messages generated by the A/R+ algorithm are included or equal to interval-valued messages generated by the A/R algorithm, and include or are equal to probability intervals generated by (exact) set-valued message propagation.*

We note that the credal sets handled by A/R+ can still become unmanageably complex in some situations. When we must compute a message that requires an excessively complex credal set, we simply resort to the original approximation proposed by Tessem (we have a threshold indicating the maximum number of vertices the algorithm should handle explicitly).[5]

As the next section indicates, outer approximations generated by the A/R+ algorithm are usually much more accurate than the ones produced by A/R. The apparently mild difference between the algorithms leads to an order of magnitude improvement in the bounds.

6. Experiments

We have run a series of experiments with the branch-and-bound, A/R and A/R+ algorithms. Our purpose was to evaluate the accuracy and computational cost of these algorithms.

6.1. The A/R and A/R+ algorithms

To evaluate the performance of the A/R+ algorithm, particularly in comparison with the A/R algorithm, we ran two experiments.

The first experiment aimed at comparing the relative error of the A/R and A/R+ algorithms. Consider the computation of $\overline{p}(X_5 = x_{5,1})$ for three sets of networks with the graph given by Figure 5: a) the first group (30

[5] The A/R+ algorithm can be made even more flexible: we can postpone the reconversion of credal sets to interval-valued form until we reach a point where the number of vertices in the messages exceeds some limit. We have not tried this option so far.

networks) contained only variables with three categories and two vertices per local credal set; b) the second group (15 networks) contained only variables with three categories and three vertices per local credal set; c) the third group (15 networks) contained only variables with four categories and two vertices per local credal set.

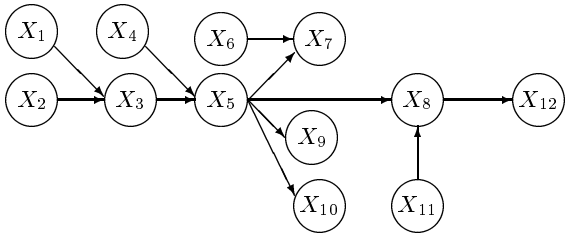

Figure 5. Polytree used in experiments.

The numerical parameters for each credal network were randomly generated with uniform distributions [21]. For every network in this test set, the upper probability $\overline{p}(X_5 = x_{5,1})$ was computed with an exact enumeration algorithm, in some cases running during a considerable amount of time. The relative error was then measured between the exact answer and the results obtained with the A/R and A/R+ algorithms. The mean error for the A/R algorithm was: 38% (first group), 15% (second group) and 27% (third group). The mean error for the A/R+ algorithm was: 4% (first group), 1% (second group) and 4% (third group). Even though the samples were rather small, these tests indicate the superior performance of the A/R+ algorithm (the main limiting factor in the number of samples and inferences in this experiment was the time required by the exact enumeration algorithm to run).

To compare the amplitude of probability intervals computed by the A/R and A/R+ algorithms, we run a few additional tests with the graph in Figure 5. We generated five groups of credal networks, each one containing one hundred networks. Groups were characterized by the number of categories per variable and number of vertices per local credal set. Table 1 summarizes the results of inferences on the event $\{X_{12} = x_{12,1}\}$. While this experiment does not reveal by itself the accuracy of A/R+, note the decrease in interval length (remember that the correct intervals are enclosed by the intervals generated by A/R+).

To avoid the possibility that differences between A/R+ and A/R are just being magnified by the particular graph in Figure 5, we considered a second experiment with a set of randomly generated polytrees (using the generator described by Ide and Cozman [21]). In all inferences we could try, we verified the same pattern of reduction in the relative error. We generated

Table 1. Comparison of interval lengths computed with the A/R and A/R+ algorithms.

# categories	# vertices	A/R mean interval length	A/R+ mean interval length
03	02	0.47	0.39
03	03	0.61	0.56
03	04	0.67	0.63
04	02	0.45	0.36
04	03	0.52	0.47

9 different polytrees, each with 20 variables; for each polytree, we generated 5 sets of local credal sets. For each node of this set of networks, we run the A/R and the A/R+ algorithms, and in some of them we compared the approximations with the exact inferences (obtained with a branch-and-bound algorithm). The mean length of the intervals generated by the A/R algorithm are significantly larger than the mean length for the A/R+ algorithm.

6.2. Depth-first branch-and-bound

We have tested the depth-first version of the branch-and-bound algorithm for computation of exact inferences. We ran experiments with networks containing variables with tree and four states. Each configuration was again tested using several randomly generated credal nets [21]. The main goal was to evaluate the reduction in computational costs for inference, compared to exact enumeration algorithms.

First we took the polytree in Figure 5. Results for queries on variable X_5 (with depth-first branch-and-bound) are reported in Table 2. The table shows results when the branch-and-bound algorithm uses A/R and A/R+ as bounding methods. The first four columns summarize the characteristics of each group of networks and inferences; the remaining four columns compare the performance of branch-and-bound using A/R and A/R+ as bounding methods.

We observe that the size of the search tree explored by branch-and-bound is usually a tiny fraction of the potential number of vertices of the strong extension. Note the enormous difference between potential vertices of the strong extension and actually expanded vertices. We can also see that A/R+ is superior to A/R.

As another example of the efficiency of the algorithm, take the computation of $\overline{p}(X_8 = x_{8,1})$ in the graph of Figure 5, with variables with three categories, and with a random collection of credal sets, where each credal

Table 2. Exact inference on $x_{5,1}$ in the network of Figure 3 with depth-first branch-and-bound. Columns indicate the number of networks tested (#s), the number of categories per variable (#c), the number of vertices in each credal set (#v), the total number of potential vertices for the strong extension (#p), the time spent in inferences (t) and the number of expanded nodes in the search tree (#n).

#s	#c	#v	#p	t A/R secs (mean)	#n AR (mean)	t A/R+ secs (mean)	#n A/R+ (mean)
35	3	2	2^{21}	5.4	3356.8	0.97	365.7
10	3	3	3^{21}	395	254559.3	17.44	7271.2
10	4	2	2^{35}	1511	527756.8	584	37143.7

set has three vertices. In this case, depth-first branch-and-bound obtained the exact solution after examining just 4634 vertices of the strong extension — note that the strong extension potentially contains 3^{50} vertices. The relative error between the exact result and the inner bound, and the exact result against A/R+ are 0.002 and 0.015 respectively.

We have observed such behavior in many experiments on randomly generated networks. We have observed that polytrees with up to 10 variables can be usually handled without problems.

6.3. Best-first branch-and-bound

We have also investigated the performance of best-first branch-and-bound methods for inference. Our tests indicated that the amount of memory required by these algorithms is too large for exact inference — that is, the size of the OPEN heap grows too quickly. However, it is still possible to use best-first branch-and-bound for *approximate* outer intervals. The basic idea is to run the algorithm up to a prescribed memory limit; upon termination, the OPEN heap is examined, and the element with the maximum bound is selected and returned. Note that the top of the heap has the maximum value of the outer bound r amongst all elements in the frontier — its r value is thus guaranteed to be an outer bound for the original problem (it may even contain the exact upper probability, in which case the whole frontier displays the same r value). As more sub-problems are generated and stored in the OPEN heap, the outer bound for the top of the heap can either stay the same or decrease, as sub-problems have feasible regions that are contained in their generating problem.

We have conducted tests where the algorithm in Figure 3 terminates after a prescribed number of search nodes. Experimental results are summarized

by Table 3. The objective was to measure accuracy after evaluating at most 25,000 nodes of the search tree (in some cases the exact result was obtained before that). We used the graph in Figure 3 and produced several credal networks by generating random local credal sets. The upper probabilities $\overline{p}(X_5 = x_{5,1})$, $\overline{p}(X_7 = x_{7,1})$ and $\overline{p}(X_8 = x_{8,1})$ were computed for each one of the resulting networks. Table 3 shows the characteristics of the generated networks and the "quality" of approximations; here we present the difference between the answer generated by best-first branch-and-bound and the inner bound generated by local search (discussed at the beginning of Section 5). We also show the difference between the A/R+ outer bound and the inner bound, for comparison. Similar results were obtained in tests with randomly generated graphs.

Table 3. Mean difference between outer and inner approximations for $\overline{p}(X_5 = x_{5,1})$, $\overline{p}(X_7 = x_{7,1})$ and $\overline{p}(X_8 = x_{8,1})$. The branch-and-bound procedure terminates after evaluating 25,000 nodes of search tree. Columns indicate the inference, the number of networks tested, the number of categories per variable, the number of vertices in each credal set, the total number of potential vertices for the strong extension, and the relationship between outer and inner bounds, with best-first branch-and-bound and A/R+ algorithms. The symbol A/R+* denotes the difference between A/R+ approximations and inner bounds, while the symbol bfbb* indicates the difference between best-first branch-and-bound approximations and inner bounds.

Inference	# net.	# cat.	# vert.	# potential vertices	A/R+* (mean)	bfbb* (mean)
$X_5 = x_{5,1}$	16	03	03	3^{21}	0.006	0.00008
$X_5 = x_{5,1}$	10	03	04	2^{42}	0.024	0.011
$X_5 = x_{5,1}$	10	04	03	3^{35}	0.035	0.025
$X_7 = x_{7,1}$	16	03	03	3^{32}	0.016	0.011
$X_7 = x_{7,1}$	10	03	04	2^{64}	0.021	0.012
$X_7 = x_{7,1}$	10	04	03	3^{35}	0.022	0.017
$X_8 = x_{8,1}$	16	03	03	3^{50}	0.017	0.007
$X_8 = x_{8,1}$	10	03	04	2^{100}	0.014	0.009
$X_8 = x_{8,1}$	04	04	03	3^{101}	0.035	0.029

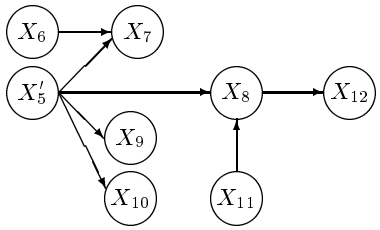

Figure 6. Transformed credal network.

7. Inference with network fragments

If the credal network $\mathcal{N}$ is large, it may not be possible to run the branch-and-bound algorithms to optimality. In this section we propose strategies to handle such problems. The basic idea is to divide the credal network in parts and to run branch-and-bound in these sub-networks, in some suitable order. We illustrate this idea through an example.

Consider the network in Figure 5, with variables with three categories and two vertices per credal set. Suppose that we want to compute exact lower and upper probabilities for variable X_7 and that our space and time constraints allow us to perform an exact inference just for X_5, but not for X_7. We then run branch-and-bound and obtain lower and upper probabilities for X_5. For example, in a particular instance of the network shown in Figure 5, we obtained $p(X_5 = x_{5,1}) \in [0.199; 0.587]$, $p(X_5 = x_{5,2}) \in [0.084; 0.375]$, and $p(X_5 = x_{5,3}) \in [0.212; 0.604]$. We can easily generate the largest credal set that is consistent with these intervals. We obtain $K(X_5)$ defined by the vertices

$$\{(0.413; 0.375; 0.212), (0.312; 0.084; 0.604), (0.587; 0.084; 0.329),$$

$$(0.199; 0.197; 0.604), (0.587; 0.201; 0.212), (0.199; 0.375; 0.426)\}.$$

Now we can remove X_5 and its ascendants from the network, and replace X_5 by a new node X'_5 that has the marginal credal set of X_5 as its marginal credal set. The transformed network is displayed in Figure 6. We then run exact branch-and-bound inference for X_7, obtaining intervals $p(X_7 = x_{7,1}) \in [0.091; 0.447]$, $p(X_7 = x_{7,2}) \in [0.157; 0.564]$, and $p(X_7 = x_{7,3}) \in [0.208; 0.591]$. Incidentally, we computed the same inferences with the exhaustive algorithm in the JavaBayes system and obtained the same values.

If inferences in the transformed credal network are still unfeasible, we can run an approximate inference algorithm in the transformed credal network. Consider running Tessem's algorithm in the network in Figure 6. We obtain

the intervals $p(X_7 = x_{7,1}) \in [0.053, 0.502]$, $p(X_7 = x_{7,2}) \in [0.116, 0.663]$, and $p(X_7 = x_{7,3}) \in [0.128, 0.644]$. We note that Tessem's algorithm alone in the complete example network produced the intervals $p(X_7 = x_{7,1}) \in [0.040, 0.524]$, $p(X_7 = x_{7,2}) \in [0.106, 0.698]$, and $p(X_7 = x_{7,3}) \in [0.097, 0.667]$.

8. Conclusion

This paper has proposed a collection of simple ideas that advance the state of affairs concerning inferences in credal networks. Perhaps the following perspective is useful. As far as exact inference with strong extensions is concerned, our branch-and-bound methods go considerably beyond what can be done with existing enumeration techniques. However, they can handle relatively small networks, and they should be most effective as tools for evaluating other (approximate) algorithms. It seems that general medium and large credal networks will hardly admit exact inference, and approximate algorithms are likely to be important in those situations. Thus one should have fast and accurate approximate methods, and one should have ways to validate the accuracy of these approximate methods.

In this perspective, it is possible that the A/R+ algorithm will be the contribution with most practical significance, while the branch-and-bound approach will serve as a validation tool for other algorithms. In fact, the branch-and-bound strategy is best viewed as a family of solutions for inference in strong extensions. Depth-first and best-first techniques can be used in different scenarios, as they require different levels of resources and have different characteristics. Outer approximations are certainly "safer" than inner ones, being able to produce both approximations can give valuable information about inferences.

Note that, even though we have restricted our experiments to polytree-shaped networks, multi-connected credal networks can be handled by bounding techniques such as the algorithms of Ha et al [20] and Cano et al [9].

We also would like to emphasize the idea that a network can be processed in pieces, as discussed in the previous section. Such a strategy seems to be appropriate for large networks, possibly using different levels of accuracy in each one of the partial inferences.

Acknowledgements

This work has received generous support from HP Labs; we thank Marsha Duro from HP Labs for establishing this support and Edson Nery from

HP Brazil for managing it. The work has also been supported by CNPq (through grant 3000183/98-4) and by CAPES.

We thank Jaime Ide for generating the random networks that were used in our tests, and Cassio Polpo de Campos for leading the development and implementation of inner bounds.

Appendix A. Proof of Theorem 1

We first show that the approximate intervals calculated by the A/R+ algorithm are contained in those computed by the A/R algorithm. Then we show that A/R+ produces outer approximations.

Part I: Both A/R and A/R+ maximize/minimize the same quantities, at every step of the propagation scheme. In each step of message propagation, the A/R+ algorithm enforces a constraint that A/R does not (the constraint that messages received by a node X from its parents represent a normalized quantity). To illustrate this fact, note that step 1 in Figure 4 uses direct interval multiplication and does not constrain functions defined by β to add to one. Thus the feasible set in each optimization is smaller for A/R+ than it is for A/R, and consequently the bounds computed by A/R+ for the predictive support and the $\lambda_Z(X)$ messages are tighter than those computed by A/R. As the computation of lower and upper probabilities for any node X is obtained by manipulating these messages recursively, results produced by A/R+ are equal to or tighter than approximations by A/R at every message propagation.

Part II: Each operation transforming credal sets into intervals (Expression (3)) produces outer bounds, because it enlarges the feasible set in each optimization problem generated during propagation. When intervals are locally transformed into credal sets, the optimization problems manipulating these credal sets produce the tighest possible bounds; however they start from larger feasible sets and consequently contain the exact set-valued messages. Thus at any node X we obtain bounds that contain the correct upper and lower probabilities.

References

1. K. A. Andersen and J. N. Hooker. Bayesian logic. *Decision Support Systems*, 11:191–210, 1994.

2. M. Avriel. *Advances in Geometric Programming.* Plenum Press, New York, 1980.

3. D.P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewoods, 1987.

4. L. M Campos, J. F. Huete, and S. Moral. Uncertainty management using probability intervals. *Fifth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 431–436, 1994.

5. A. Cano, J. E. Cano, and S. Moral. Convex sets of probabilities propagation by simulated annealing. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 4–8, Paris, France, July 1994.

6. A. Cano and S. Moral. A genetic algorithm to approximate convex sets of probabilities. *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2:859–864, 1996.

7. A. Cano and S. Moral. A review of propagation algorithms for imprecise probabilities. In G. de Cooman, F. G. Cozman, S. Moral, and P. Walley, editors, *First International Symposium on Imprecise Probabilities and Their Applications*, pages 51–60, Ghent, Belgium, 1999. Imprecise Probabilities Project, Universiteit Ghent.

8. A. Cano and S. Moral. Algorithms for imprecise probabilities, In *Handbook of Defeasible Reasoning and Uncertainty Management Systems (vol. 5)*, pages 369–420. Kluwer Academic Publishers, 2000.

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

10. J. Cano, M. Delgado, and S. Moral. An axiomatic framework for propagating uncertainty in directed acyclic networks. *International Journal of Approximate Reasoning*, 8:253–280, 1993.

11. F. G. Cozman. Robustness analysis of Bayesian networks with local convex sets of distributions. In D. Geiger and P. Shenoy, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 108–115, San Francisco, 1997. Morgan Kaufmann.

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

13. F. G. Cozman. Separation properties of sets of probabilities. In C. Boutilier and M. Goldszmidt, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 107–115, San Francisco, July 2000. Morgan Kaufmann.

14. F. G. Cozman. Generalizing variable elimination in Bayesian networks. In L. N. de Barros, R. Marcondes, F. G. Cozman, and A. H. R. Costa, editors, *Workshop on Probabilistic Reasoning in Artificial Intelligence*, pages 27–32, São Paulo, Brazil, 2000. Editora Tec Art.

15. J. C. F. da Rocha and F. G. Cozman. Inference with separately specified sets of probabilities in credal networks. In A. Darwiche and N. Friedman, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 430–437, San Francisco, California, 2002. Morgan Kaufmann.

16. J. C. F. da Rocha, F. G. Cozman and C. P. de Campos. Inference in polytrees with sets of probabilities. In *Conference on Uncertainty in Artificial Intelligence*, pages 217–224, San Francisco, California, 2003. Morgan Kaufmann.

17. D. Draper and S. Hanks. Localized partial evaluation of belief networks. *Conference on Uncertainty in Artificial Intelligence*, 1994.

18. R. J. Duffin and E. L. Peterson. Geometric programming with signomials. *Journal of Optimization Theory and Applications*, 11(1):3–35, 1973.

19. E. Fagiuoli and M. Zaffalon. 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 106(1):77–107, 1998.

20. V. A. Ha, A. Doan, V. H. Vu, and P. Haddawy. Geometric foundations for interval-based probabilities. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):1–21, 1998.

21. J. S. Ide and F. G. Cozman. Random generation of Bayesian networks. In *Brazilian Symposium on Artificial Intelligence (SBIA)*, pages 366–375, Porto de Galinhas, Pernambuco, Brasil, 2002.

22. M. Lavine. Sensitivity in Bayesian statistics, the prior and the likelihood. *Journal of the American Statistical Association*, 86(414):396–399, June 1991.

23. I. Levi. *The Enterprise of Knowledge*. MIT Press, Cambridge, Massachusetts, 1980.

24. C. Papadimitriou and I. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

25. B.R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. Willey, New York, 2000.

26. B. Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, 7:95–120, 1992.

27. H. Thone, U. Guntzer, and W Kieβling. Increased robustness of Bayesian networks through probability intervals. *International Journal of Approximate Reasoning*, 17:37-76, 1997.

28. P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

29. M. Zaffalon. *Inferenze e Decisioni in Condizioni di Incertezza con Modelli Grafici Orientati*. PhD thesis, Università di Milano, February 1997.