Inference in Credal Networks using Multilinear Programming

Cassio P. de Campos
Pontificial Catholic University
São Paulo, SP - Brazil - cassio@pucsp.br

Fabio G. Cozman
Escola Politécnica, University of São Paulo
São Paulo, SP - Brazil - fgcozman@usp.br

Abstract. A credal network is a graphical tool for representation and manipulation of uncertainty, where probability values may be imprecise or indeterminate. A credal network associates a directed acyclic graph with a collection of sets of probability measures; in this context, inference is the computation of tight lower and upper bounds for conditional probabilities. In this paper we present new algorithms for inference in credal networks based on multilinear programming techniques. Experiments indicate that these new algorithms have better performance than existing ones, in the sense that they can produce more accurate results in larger networks.
Keywords: Uncertainty and reasoning, Sets of probability measures, Bayesian networks, Multilinear programming.

1 Introduction

This paper presents techniques for marginal inference in *credal networks*; the goal is to provide algorithms that can handle graphical models for precise and imprecise probabilistic assessments. Credal networks offer one such graphical representation, as they represent a set of joint probability measures through a directed acyclic graph and a collection of local sets of probability measures [8, 11, 20]. The structure of the graph indicates relations of independence between variables; the "size" of the sets of probabilities encodes the imprecision in the probability values. The introduction of imprecision in probability values (through sets of probabilities) allows one to study robustness of probabilistic models, to investigate the behavior of groups of experts, to represent incomplete or vague knowledge about probabilities [37], and to represent qualitative probabilistic networks. Overall, credal networks offer a flexible method for uncertainty handling, as it can deal with precise, imprecise, qualitative and comparative beliefs. Section 2 reviews basic properties of credal networks and their application in knowledge representation.

A marginal inference in the context of credal networks is a computation of upper/lower probability for some event. Existing algorithms for marginal inference cannot handle large networks; this paper aims at enlarging the class of networks that can be successfully processed. We present an exact inference algorithm based on multilinear programming (Sections 4 and 5), and show through experiments that the algorithm can process larger networks than

currently possible with existing algorithms. We also present an extension of the A/R+ approximate algorithm of Rocha et al [17]; we essentially extend the A/R+ algorithm using multilinear programming tools (Section 6). Approximations with this new algorithm, We show through experiments that approximations generated by the A/R++ algorithm are significantly better than the ones produced by the A/R+ algorithm. All experiments are discussed in Section 7. We should note that the algorithms discussed in this paper have been introduced in summary form elsewhere [14]; this paper presents a more detailed and complete discussion.

2 Credal sets and credal networks

A few preliminary definitions are important. A convex set of probability distributions is called a *credal set* [26]. A credal set for X is denoted by $K(X)$; we assume that every variable is categorical and that every credal set has 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 [37]. The sets $K(X|Y)$ 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 $y_2 \neq y_1$ — that is, the conditional sets bear no relationship to each other. In this paper we assume that local credal sets are always separately specified; justifications for this separability assumption can be found in [15]. Given a number of marginal and conditional credal sets, an *extension* of these sets is a joint credal set with the given marginal and conditional credal sets. In this paper we are exclusively concerned with the largest possible extension for any collection of marginal and conditional credal sets. Given a credal set $K(X)$ and an event A, the *upper* and *lower* probability of A are respectively $\overline{P}(A) = \max_{p(X) \in K(X)} P(A)$ and $\underline{P}(A) = \min_{p(X) \in K(X)} P(A)$.

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 value 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 nonparents given its parents. In this paper we adopt the concept of *strong independence*[1]: 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 $p(Y|X) = p(Y)$) [11]. Strong independence is the most commonly adopted concept of independence for credal sets, probably due to its obvious connection with standard stochastic independence.

The *strong extension* of a credal network is the largest joint credal set such that every variable is strongly independent of its nondescendants nonparents given its parents. The strong extension of a credal network is the joint credal set that contains every possible combination of vertices for all credal sets in the network [13]; that is, each vertex of a strong extension factorizes as follows:

$$p(X_1, \ldots, X_n) = \prod_i p(X_i|\mathrm{pa}(X_i)) \,. \tag{1}$$

[1] We note that other concepts of independence are found in the literature [9, 18].

3 Inference with strong extensions

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 values of X_q. For inferences in strong extensions, it is known that the distributions that minimize/maximize $p(X_q|\mathbf{X}_E)$ belong to the set of vertices of the extension [20].

An inference can be produced by combinatorial optimization, as we must find a vertex for each local credal set $K(X_i|\mathrm{pa}(X_i))$ so that Expression (1) leads to a maximum/minimum of $p(X_q|\mathbf{X}_E)$. In general, inference offers tremendous computational challenges — consider the following example, taken from Rocha et al [17]. Take a network with three nodes, $X \rightarrow Y \leftarrow Z$, where X, Y and Z have four values each, and where all credal sets have four vertices each. There are 4^{18} different joint distributions factorizing as Expression (1), where local distributions are vertices of local credal sets! Rocha et al [17] discuss branch-and-bound procedures that can handle situations such as this, but that still have difficulties in large multi-connected networks. The only known polynomial algorithm for strong extensions is the 2U algorithm, which processes polytrees with *binary* variables only [20]. Other exact inference algorithms based on enumeration examine all potential vertices of the strong extension to produce the required lower/upper values [5, 8, 11, 15]; these algorithms face serious difficulties in large networks.

A different way to look at the computation of inferences is to recognize that a lower/upper value for $p(X_q|\mathbf{X}_E)$ is obtained by minimization/maximization of a fraction containing polynomials in probability values. This is in fact the strategy discussed in Section 4; our results suggest that this is the most profitable strategy to take for exact inference with strong extensions.

In Section 6 we discuss approximate algorithms, and discuss the A/R++ algorithm for approximate inference with multilinear programming. We distinguish *outer* and *inner* approximations: the former produces intervals that enclose the correct probability interval between lower and upper probabilities, while the latter produces intervals that are enclosed by the correct probability interval. The A/R++ is an outer approximation scheme derived from the A/R+ algorithm of Rocha et al [17]. Other outer approximations can be found in [7, 22, 35]; inner approximations can be found in [1, 6, 5, 10, 17].

4 Inference as a multilinear programming problem

A marginal inference for a strong extension can be formulated as a multilinear programming problem. The goal is to minimize/maximize the expression

$$\sum_{X_i \setminus X_q} \prod_i p(X_i|\mathrm{pa}(X_i)) \tag{2}$$

subject to constraints on the local probabilities $p(X_i|\mathrm{pa}(X_i))$. In this problem we must deal with a large number of terms in the multilinear objective function (the number of terms is exponential on the size of the network). In this section we reformulate Expression (2) to transform it into a collection of smaller equalities.

To briefly illustrate the transformation, take a simple credal network with a "chain topology" $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$, where the variable $X \in \{A, B, C, D, E\}$ is ternary and

assumes the values x_0, x_1 and x_2. Suppose we want to evaluate the maximum possible value for the probability of $E = e_0$; this is obtained by solving:

$$\max \sum_{h,i,j,k \in \{0,1,2\}} p(e_0|d_h)\, p(d_h|c_i)\, p(c_i|b_j)\, p(b_j|a_k)\, p(a_k)$$

subject to linear constraints. We have a multilinear function with 81 nonlinear terms of degree four. Instead of dealing with this function, we can transform it into a problem with simpler multilinear functions of degree at most two, by grouping terms and introducing new variables. The result is a multilinear program with just 30 nonlinear terms:

$$\begin{aligned}
&\max \sum_{i \in \{0,1,2\}} p(e_0|d_i)\, p(d_i) \\
&\text{subject to} \\
&p(d_i) = \sum_{j \in \{0,1,2\}} p(d_i|c_j)\, p(c_j)\,, \text{ for } i = 0, 1, 2 \\
&p(c_i) = \sum_{j \in \{0,1,2\}} p(c_i|b_j)\, p(b_j)\,, \text{ for } i = 0, 1, 2 \\
&p(b_i) = \sum_{j \in \{0,1,2\}} p(b_i|a_j)\, p(a_j)\,, \text{ for } i = 0, 1, 2
\end{aligned}$$

plus the linear constraints. The transformation leads to less nonlinear terms than in the direct version given by Expression (1). Note also that the transformed problem contains terms of smaller degree than the original problem — this is important in multilinear programming, as it is difficult to handle problems with high degree.

We now present in detail the algorithm to transform a credal network inference into a multilinear programming problem.

Translation Algorithm:

1. Build an ordering of the N variables (nodes) of the network. The queried variable must be left as the last variable in the order (this ordering can be constructed in many ways. See for example [12, 32]). Use this order to number the variables, obtaining $X_N, X_{N-1}, \ldots, X_1$. The queried variable is indicated by X_0. Let V denote the set of variables X_i. Denote by p_{ij} the probability of the event x_{ij}, i.e., $p(X_i = x_{ij})$.

2. Place all network density functions in a pool. These functions can be written as $f(A|B)$, where $A \subseteq V$, $B \subseteq V$ and $A \cap B = \emptyset$.

3. For i from N to 0:

   - Create a data structure $Bucket_i$ containing all the density functions (from the pool) related to X_i, i.e. take all $f(A|B)$ such that $X_i \in A \cup B$. Name these functions as $f_1(A_1|B_1), f_2(A_2|B_2), \ldots, f_n(A_n|B_n)$.
   - For each density function $f_q(A_q|B_q)$ in $Bucket_i$, with q from 1 to n, do

– Suppose that $p_{a_{qu}|b_{qv}}$ is the probability of $A_q = a_{qu}$ given $B_q = b_{qv}$ (note that a_{qu} is the event representing a combination of all events of variables in A_q and b_{qv} has the same meaning for B_q). For each event b_{qv} of B_q, generate the function

$$\sum_u p_{a_{qu}|b_{qv}} = 1$$

In words, we sum over each combination of events of variables in the conditioned side A_q with the same (unchanged) combination of events of the conditioning side B_q. Then this process is repeated for each combination of the conditioning side. Note that $p_{a_{qu}|b_{qv}}$ are variables for the multilinear program we are building; their values are not fixed.

There are $|a_{qu}| \times |b_{qv}|$ functions generated here, where $|a_{qu}|$ is the number of events of A_q and $|b_{qv}|$ is the number of events of B_q. Write all these generated functions in the multilinear problem.

- Multiply all density functions in $Bucket_i$ and sum out X_i from the product, storing the resulting function in the pool. This operation is precisely $\sum_{X_i} \prod_{q=1}^{n} f_q(A_q|B_q)$, which will generate a new function, called here $f_0(A_0|B_0)$ for convinience. We know that $X_i \notin A_0 \cup B_0$. The representation of this process in the multilinear problem is as follows:

  – Suppose that $p_{a_{0u}|b_{0v}}$ is the probability of $A_0 = a_{0u}$ given $B_0 = b_{0v}$ (note that a_{0u} is the event representing a combination of all events of variables in A_0 and b_{0v} has the same meaning for B_0). For each event a_{0u} and each event b_{0v}, generate the function

$$\sum_{X_i} \left(\prod_{q=1}^{n} p_{a_{qu}|b_{qv}} \right) = p_{a_{0u}|b_{0v}}$$

where $p_{a_{qu}|b_{qv}}$ represents the probability of the event $a_{qu}|b_{qv}$ over the function $f_q(A_q|B_q)$. (Note that while writing these functions in the multilinear problem, it is necessary to verify whether the combination $A_q|B_q$, for any $q \in \{0, \ldots, n\}$, has appeared already. If so, we must use the same variable name in the multilinear problem for its probability; otherwise an unused variable name must be chosen for it.)

Exactly $|a_{0u}| \times |b_{0v}|$ functions are generated here. Add them to the multilinear problem. These functions show the relation between the variables that appear when eliminating X_i and the previously existing ones (before multiplying and summing out the bucket functions). Note that when processing $Bucket_0$ the process is sightly different as we do not want to eliminate X_0.

4. Take each local credal set separately specified by $f(A|B)$ in the network and generate the linear functions defining it (if the credal sets are defined by their vertices, then it is necessary to transform them into linear inequalities [2]). Add all these functions to the multilinear problem. Again it is necessary to verify where the combination $A|B$ has appeared in the problem and then the same variable name chosen before for its probability must be re–used. The number of functions inserted by this step depends on the number of vertices in the credal sets and the types of the variables (binaries, ternaries, etc).

5. Now it is only necessary to define which probability p_{0j} of X_0 we wish to minimize / maximize.

Thus we obtain an optimization problem with multilinear and linear constraints for each node in the bucket tree generated during variable elimination [12]. Each one of these multilinear and linear constraints represents local information in the credal network; that is, constraints represent relations between neighbour nodes in the tree. The number of functions in this new multilinear programming problem is proportional to the parameters of the credal network. The transformation procedure can be quickly executed as its complexity is on the order of a Bayesian network inference.

Andersen and Hooker [1] treat a similar problem, but they work with binary variables and the constraints to represent independence relations generated there are obtained in a different manner (they use an extended description of possible *worlds*, which implies in extracting other independence relations from the network). The advantage here is the direct formulation, in the sense that generation of the multilinear problem follows exactly the same method used in inference over Bayesian networks. Due to this, it is easy to run specialized approximate algorithms over the problem (like the A/R++ described in section 6), obtaining tighter variable bounds which improve the multilinear solver performance.

5 Multilinear Programming

In this section we discuss the solution of multilinear programming problems such as the ones introduced in the previous section. Even though multilinear programming solutions for inference in credal networks have been mentioned before [1, 11, 38], it seems that no systematic study of this proposal has been conducted so far, and in particular there has not been any implementation of multilinear programming geared towards inference in credal networks. We thus give a relatively detailed account of multilinear programming and its application to strong extensions.

We look at multilinear programs such as:

$$MP: \qquad \{\max f_0(x) : x \in S \cap \Omega\}$$

where

$$\begin{aligned} S &= \{x \in \mathbb{R}^n : f_r(x) \geq \beta_r, \text{ for } r = 1, \cdots, R\} \\ \Omega &= \{x : 0 \leq l_j \leq x_j \leq u_j \leq 1, \text{ for } j = 1, \cdots, n\} \\ f_r(x) &= \sum_{t \in T_r} \alpha_{rt} \left[\prod_{j \in J_{rt}} x_j \right] \text{ for } r = 0, \cdots, R \end{aligned}$$

Here T_r is an index set defining the terms of f_r and α_{rt} is the real coefficient for the term t of the function f_r. Each set J_{rt} defines which variables appear in the term t of f_r. Let T be the set of all terms in the problem, ie, $T = \cup_r T_r$. It is clear that any multilinear problem can be stated in this way. Unlike geometric problems (where all α_{rt} have the same sign and exponents are arbitrary), MP problems are nonconvex and no known transformation can convexify them; they may display multiple local minima and nonconvex feasible regions.

Many *local* optimization approaches are applicable to MP: procedures based on condensation [3, 19], KKT conditions [31], linearization techniques [27], and general purpose nonlinear programming methods [29, 30, 25]. Rocha et al [17] presented an specialized solution

for credal sets based on Lukatskii and Shapot's work [27], with excellent speed and accuracy. A local optimization procedure typically finds local optima and therefore can generate only inner approximations to inferences. Existing *exact* algorithms are based on branch-and-bound techniques [28, 21, 33] or cutting plane methods [23, 36]. The branch-and-bound ideas presented by Maranas and Floudas [28], and Gochet and Smeers [21] are based on solving convex nonlinear subproblems, while Sherali and Tuncbilek's idea [33] is based on linear ones. The cutting plane methods proposed by Tuy [36] seem promising but have slow convergence when close to an optimum (although some ideas exist to overcome this problem).

The branch-and-bound algorithm described in Sherali and Tuncbilek [33] is based on a Reformulation-Linearization (RL) technique. The central idea is to substitute each product of variables $\prod_{j \in J_{rt}} x_j$ with a new artificial variable $X_{J_{rt}}$, for all terms $t \in T$, obtaining a linearized LP sub-problem. The solution of the LP problem gives an upper bound to the solution of the corresponding MP problem (since the LP problem is a relaxation of the MP problem). Just making this substitution does not lead us to a globally convergent method. Thus some additional restrictions should be incorporated into the LP problem. To achieve a global optimization method, we iterate over the variables by branching their intervals whenever necessary until each $X_{J_{rt}}$ is close enough to $\prod_{j \in J_{rt}} x_j$, solving LP sub-problems over each branching node. To guarantee the convergence of the method, some additional artificial functions must be included in the linear sub-problems. We call them "artificial" because they are redundant for the original problem but not redundant for the linearized version. These new artificial functions are created through products of original constraints and/or factors $(x_j - l_j)$ and $(u_j - x_j)$ (called bound-factors by Sherali and Tuncbilek) provided that the degree of the new functions do not exceed the maxdegree $\delta = \max_{r,t} |J_{rt}|$ of the original problem (because this would increase the complexity of the overall procedure). As we can write each original constraint as $f_r - \beta_r \geq 0$ (if needed, equalities are written using two inequalities and reverse inequalities are multiplied by -1) and we know that $(x_j - l_j) \geq 0$ and $(u_j - x_j) \geq 0$, multiplying these functions lead us to new redundant functions of the same type. To illustrate the idea, suppose we have a multilinear term $x_1 x_2$ in the original problem. Then we create the artificial functions (where $X_{12} = x_1 x_2$):

$$\begin{array}{lccccc}
(x_1 - l_1)(x_2 - l_2) = & +X_{12} & -l_2 x_1 & -l_1 x_2 & +l_1 l_2 & \geq 0 \\
(x_1 - l_1)(u_2 - x_2) = & -X_{12} & +u_2 x_1 & +l_1 x_2 & -l_1 u_2 & \geq 0 \\
(u_1 - x_1)(x_2 - l_2) = & -X_{12} & +l_2 x_1 & +u_1 x_2 & -u_1 l_2 & \geq 0 \\
(u_1 - x_1)(u_2 - x_2) = & +X_{12} & -u_2 x_1 & -u_1 x_2 & +u_1 u_2 & \geq 0
\end{array}$$

Although these functions are redundant in the original problem (since the bounds $l_j \leq x_j \leq u_j$ should be already ensured in the MP problem), their linearized versions supply "barriers" to the X_{12} variable.

The approximation given by the linearization process improves as we increase the number of artificial functions, but we have to limit this number — otherwise the linear sub-problems grow too large. To ensure convergence it is enough to produce every possible product of bound-factors, without using any original functions for creating the artificial ones.

The method is slow when δ is large, because the number of artificial functions needed to guarantee the convergence is exponential on δ. Sherali and Tuncbilek [33] suggested $\sum_{k=0}^{\delta} \binom{m+k-1}{k} \binom{m+(\delta-k)-1}{\delta-k}$ artificial functions, where m is the number of variables in the MP problem (note that the number of variables in the MP problem for solving a credal network problem is much greater than the number of nodes in the network; see tables 1 and 2). Sherali

and Tuncbilek's method generates products of bound-factors for all possible combinations of variables up to terms of δ degree. Fortunately δ is not large due to the transformation from the credal network to a multilinear problem (previous section). Furthermore, we found out that creating new artificial functions only when the new terms that appear on them are already terms of the multilinear problem, and constructing functions that are products of original constraints by bound-factors (until a maximum degree of δ is reached) significantly increases the performance of the algorithm.

The algorithm proceeds by bounding inferences and branching on the range of certain variables. To choose the branching variable, the algorithm looks for the greatest difference between the artificial variables and the products that they represent. Branching the range of a variable may worsen the solution of the corresponding LP sub-problems. Every time a LP solution is MP-feasible, we verify whether it is the best solution known so far and update the best known solution; the node is then discarded, because no optimal solution can be found from it.

The algorithm can use an inner bound to prune branches that cannot lead to the optimum. In our implementation, we employ the local search algorithm presented by Rocha et al [17] as the initial best solution for the MP. The global best known solution at each step imposes a branching limit to the branch tree; if the LP solution of a node is worse than the best known so far, then this node is discarded. Note that such an initial solution allows the algorithm to "bracket" the result of the inference, and to stop at any time with an enclosing interval — hence the algorithm can be easily used as an approximation method.

6 A/R++

The performance of the RL-based algorithm presented in the previous section is greatly enhanced if approximate ranges for some or all variables are known [33]. One way to obtain approximate ranges for the variables is to use the A/R+ algorithm [17]. The A/R+ algorithm (and the original A/R proposed by Tessem [35]) produces local approximations for the messages sent between nodes during inference. A node sends/receives approximate probability intervals from/to its parents and children; these approximations are quickly computed and transmitted. Note that the A/R and the A/R+ algorithms were designed to work only for polytree-shaped networks; to deal with non-polytree networks, we apply the same procedure on the variable elimination tree instead of the original network (which can be multi-connected).

The important point here is that messages in the A/R+ algorithm (and in its generalization for multi-connected networks) are obtained by local optimization procedures; in fact, the local optimization problems are multilinear programs themselves. Thus it is possible to solve small multilinear problems "inside" the A/R+ algorithm, shrinking the intervals whenever branching is performed.

To process multi-connected networks, the A/R+ algorithm is executed over the elimination tree in the same manner as done by the translation algorithm described in section 4. The nonlinear functions generated in step 4 of that algorithm represent the probabilities of the elimination tree variable. They can be minimized/maximized locally by multilinear programming.

The information passed by the A/R+ algorithm is a collection of upper and lower probabilities for the values of each variable. Once we have multilinear programming, it is possible to pass even more information: we can perform several local optimizations, obtaining upper

Test set	Network topology	# nodes	# vertices by credal set
A	dense binary	10	2
B	binary Alarm	37	2
C	dense ternary	10	3
D	ternary Alarm	37	3
E	dense quaternary	10	4

Table 1: The test sets

and lower probabilities for several events, and transmit this extended information as in the A/R+ algorithm. We call the resulting algorithm A/R++. In short, the idea of A/R++ is to evaluate not just intervals for values of the variables at each node, but the intervals of other events defined in the node. We pass all these intervals (including those of the A/R+) to the node's neighbourhood; as in the A/R+ algorithm, only local information is processed. In our implementation we chose to evaluate $\min / \max\ \alpha + \beta$, for every combination of atomic values α and β at each node, although any other linear function based on the local variables could be used. These new bounds are not redundant since all we know without them is that $\min\ (\alpha+\beta) \geq \min \alpha + \min \beta$ (similarly for maximization). Note that the intervals computed by A/R++ are always better (more precise) than or equal the intervals of the A/R+. The new information that should be propagated does not pose a computational difficulty, since it can be represented by linear functions and is suitable for the RL-based algorithm described in the previous section.

On top of the improvement obtained by the A/R++ over the initial intervals of the MP problem, it is possible to run it for range reduction at every step of the branch-and-bound method. Sherali and Adams [34] suggested that such a technique could be used to reduce the ranges of variable intervals; we have implemented the range reduction using the A/R++ at every new node created by the branching procedure, which tends to reduce the number of branches required to reach the optimum.

7 Computational results

To illustrate the behavior of our methods, we present two sets of experiments. First we deal with test sets containing either a dense ten-variable network topology (randomly generated using the BNGenerator software [24]) or the topology of the Alarm network [4]. We then discuss a few different examples, including some very large ones.

Tables 1 and 2 show results for five different types of networks. Each row in Table 2 averages results for ten random generated multi-connected credal networks. All the tests are maximizations of probabilities and were done by transforming inferences in multi-connected credal networks into multilinear programming problems. The chosen inferences represent the most challenging inference for each network. We processed networks with different variables (binary, ternary and quaternary), and different sizes of credal sets per node of the network. Table 2 shows the size of the multilinear and linearized problems; note that the problem size grows substantially with the number of vertices by credal set and the number of variables of the original network. Computational difficulties for solving the linear sub-problems were met in the last two lines (in the last line most sub-problems, which have more than thirty thousand

Test set	RL-based # vars	RL-based # funcs	Linearized # vars	Linearized # funcs	A/R++ error	RL-based error	# RL nodes
A	105	172	665	3996	2.8684%	0.0484%	301
B	363	576	1395	6876	5.5706%	1.076%	765
C	412	576	5920	40181	10.4304%	0.3290%	1
D	1657	2214	13780	70612	22.3293%	2.5954%	3
E	1145	1474	30073	213376	13.4146%	0.6071%	1

Table 2: Average size of the multilinear problems and their corresponding linearized versions. The errors induced by intervals containing the optimum value (average of the best value found divided by the farthest value that possibly is the optimum for each test), and the average evaluated nodes in the branch-and-bound tree.

Network # nodes	Multilinear # variables	Multilinear # functions	RL-based solution	Error
8	53	65	$[0.6376, 0.6376]$	0.0%
13	223	351	$[0.2590, 0.2595]$	0.2%
37	2042	777	$[0.5605, 0.5645]$	0.7%
84	441	781	$[0.7803, 0.7811]$	0.1%
96	437	772	$[0.6794, 0.6885]$	1.3%
126	1117	1751	$[0.8531, 0.9985]$	14.5%
126	8211	9291	$[0.9199, 1.0000]$	8.0%

Table 3: Tests with random networks.

variables and two hundred thousand restrictions, could not be solved).

Moreover, Table 2 shows the average size of the error induced by the intervals containing the optimum value generated by the A/R++ algorithm and by the branch-and-bound (RL-based) with the linearization technique (which internally uses the A/R++). To illustrate the effort required by the RL procedure, the last column shows the number of evaluated nodes in the branch-and-bound tree. Note that in the test set D only eighty percent of the networks could be solved and in the test set E less than fifty percent could be solved. All tests were performed on a Pentium IV 1.7Ghz with 1GB of RAM, with a time-limit of ten minutes for the test sets A, B and C, and one hour for the test sets D and E.

Table 3 shows the solutions produced for test cases from eight to one hundred and twenty six variables, with random generated topologies. As far as we know, no other existing algorithm can handle networks with this size. All tests were executed with a time-limit of five minutes each (in the last two lines the limit imposed was 30 minutes). Since each global solution lies inside the interval shown, the error was evaluated dividing the interval's medium value by the interval's minimum value. If we restrict our attention just for polytrees, then the method can handle even larger networks with smaller error.

8 Conclusion

We discussed in this paper a new idea for inference over credal networks. The main contribution is the use of multilinear programming techniques for inference. Although many authors have suggested the multilinear programming as a possible approach to inference, as far as we

know no deeper investigation or implementation have been conducted before.

Results produced in our experiments seem promising and surpass existing algorithms for inference with credal networks. Other multilinear programming techniques could certainly be investigated in future work, perhaps combining some multilinear programming techniques (like cutting plane methods and local search procedures) to achieve better performance and accurancy.

Multilinear programming techniques may fail for very large networks; in this case approximate inference seems the natural solution. As shown by the A/R++ algorithm, multilinear programming can be quite helpful for approximation procedures as well.

Acknowledgements

We thank Jaime S. Ide and José Carlos F. da Rocha for generating the random networks used in experiments.

This work was (partially) developed in collaboration with HP Brazil R&D. The work has also been partially supported by CNPq (through grant 3000183/98-4).

References

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

[2] D. Avis, lrs: A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm, *Polytopes - Combinatorics and Computation*, G. Kalai & G. Ziegler eds., Birkhauser-Verlag, DMV Seminar Band 29, pp. 177-198, 2000.

[3] M. Avriel, R. Dembo and U. Passy, Solution of generalized geometric programs, *International Journal for Numerical Methods in Engineering*, 9 (149), 1975.

[4] I. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *2nd European Conf. on Artificial Intelligence in Medicine*, pages 247–256, 1989.

[5] A. Cano, J. E. Cano, and S. Moral. Convex sets of probabilities propagation by simulated annealing. *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 4–8, Paris, 1994.

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

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

[8] 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.

[9] I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5:165–181, 2000.

[10] F. G. Cozman. Robustness analysis of Bayesian networks with local convex sets of distributions. *XIII Conference on Uncertainty in Artificial Intelligence*, pages 108–115, San Francisco, California, 1997. Morgan Kaufmann.

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

[12] F. G. Cozman. Generalizing variable elimination in Bayesian networks. *Workshop on Probabilistic Reasoning in Artificial Intelligence*, pages 27–32, São Paulo, Brazil 2000.

[13] F. G. Cozman. Separation properties of sets of probabilities. *XVI Conference on Uncertainty in Artificial Intelligence*, pages 107-115, San Francisco, 2000. Morgan Kaufmann.

[14] F. G. Cozman, C. P. de Campos, J. S. Ide, J. C. F. da Rocha. Propositional and Relational Bayesian Networks Associated with Imprecise and Qualitative Probabilistic Assessments, *Conference on Uncertainty in Artificial Intelligence*, to appear.

[15] J. C. F. da Rocha and F. G. Cozman. Inference with separately specified sets of probabilities in credal networks. *XVIII Conference on Uncertainty in Artificial Intelligence*, pages 430–437, San Francisco, 2002. Morgan Kaufmann.

[16] J. C. F. da Rocha and F. G. Cozman. Inference in credal networks with branch-and-bound algorithms. *Third International Symposium on Imprecise Probability and Their Applications*, 2003.

[17] J. C. F. da Rocha, F. G. Cozman and C. P. de Campos, Inference in polytrees with sets of probabilities, *Conference on Uncertainty in Artificial Intelligence*, pp.217–224, Morgan Kaufmann, 2003.

[18] L. de Campos and S. Moral. Independence concepts for convex sets of probabilities. *XI Conference on Uncertainty in Artificial Intelligence*, pages 108–115, San Francisco, 1995. Morgan Kaufmann.

[19] R. J. Duffin, Linearizing geometric problems, *SIAM Review*, 12(211), 1970.

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

[21] W. Gochet and Y. Smeers, A branch-and-bound method for reversed geometric programming, *Operations Research*, 27(5): 983–996, 1979.

[22] V. A. Ha. Geometric foundations for interval-based probabilities. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):1–21, 1998.

[23] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, Springer–Verlag, 1995.

[24] J. S. Ide and F. G. Cozman. Random generation of Bayesian networks. *Brazilian Symposium on Artificial Intelligence*, pages 366–375, 2002.

[25] P. C. Haarhoff and J. D. Buys, A new method for the optimization of a nonlinear function subject to nonlinear constraints, *Comp. J.*, 13 (178), 1970.

[26] I. Levi. *The Enterprise of Knowledge*. MIT Press, Cambridge, Massachusetts, 1980.

[27] A. M. Lukatskii and D. V. Shapot. Problems in multilinear programming. *Computational Mathematics and Mathematical Physics*, 41(5):638-648, 2000.

[28] C. D. Maranas and C. A. Floudas, Global optimization in generalized geometric programming, *Computers and Chemical Engineering*, 21(4):351–370, 1997.

[29] P. M. Pardalos and J. B. Rosen, Constrained Global Optimization: Algorithms and Applications, *Lecture Notes in Computer Science*, 268, Springer–Verlag, 1987.

[30] M. Ratner, L. S. Lasdon and A. Jain, Solving geometric problems using GRG: Results and Comparisons, *JOTA*, 26, (253), 1978.

[31] M. J. Rijckaert and X. M. Martens, Comparison of generalized geometric programming algorithms, *JOTA*, 26 (205), 1978.

[32] G. Shafer, *Probabilistic Expert Systems*. CBMS-NSF regional conference series in applied mathematics 67, SIAM, 1996.

[33] H. D. Sherali and C. H. Tuncbilek, A global optimization algorithm for polynomial programming problems using a Reformulation-Linearization technique, *Journal of Global Optimization*, 2, pages 101–112, 1992.

[34] H. D. Sherali and W. P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic Publishers, 1999.

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

[36] H. Tuy, *Convex Analysis and Global Optimization*, Nonconvex Optimization and Its Applications, 22, Kluwer Academic Publishers, 1998.

[37] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

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