Approximate Algorithms for Credal Networks with Binary Variables

Jaime Shinsuke Ide [a,1], Fabio Gagliardi Cozman [b,2]

[a] *Department of Radiology - University of Pennsylvania*
3600 Market Street, Suite 370, Philadelphia, PA 19104-2644
jaime.ide@uphs.upenn.edu

[b] *Escola Politécnica - Universidade de São Paulo*
Av.Prof. Mello Moraes, 2231, São Paulo, SP - Brazil
fgcozman@usp.br

Abstract

This paper presents a family of algorithms for approximate inference in credal networks (that is, models based on directed acyclic graphs and set-valued probabilities) that contain only binary variables. Such networks can represent incomplete or vague beliefs, lack of data, and disagreements among experts; they can also encode models based on belief functions and possibilistic measures. All algorithms for approximate inference in this paper rely on exact inferences in credal networks based on polytrees with binary variables, as these inferences have polynomial complexity. We are inspired by approximate algorithms for Bayesian networks; thus the Loopy 2U algorithm resembles Loopy Belief Propagation, while the IPE and SV2U algorithms are respectively based on Localized Partial Evaluation and variational techniques.

Key words: Credal networks, loopy belief propagation, variational methods, 2U algorithm

1 Introduction

Consider a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, associated with general probabilistic assessments: for example, the probability of $\{X_1 = 1\}$ is larger than $1/2$, while the expected value of X_2 conditional on $\{X_3 = 0\}$ is smaller than

[1] This work was conducted while the first author was with Escola Politécnica, Universidade de São Paulo (Av. Prof. Mello Moraes, 2231, São Paulo, SP - Brazil).
[2] Corresponding author.

2. Such assessments may reflect incomplete or vague beliefs, or beliefs held by a group of disagreeing experts. In these circumstances, assessments characterize a *set* of probability distributions over $\mathbf{X}$. Suppose also that conditional independence relations over the variables are specified — we later discuss the exact definition of conditional independence for sets of probabilities; for now assume one has been given such relations. Assume these independence relations are specified by a directed acyclic graph where each node is a variable, and such that a variable and its nondescendants are conditionally independent given its parents. If one or more distributions can satisfy all assessments, then we call the set of assessments and independence relations a *credal network* [10,19,27]. Whenever a credal network represents a single distribution, we refer to it simply as a Bayesian network [51]. In fact, credal networks can be viewed as straightforward generalizations of the well-known Bayesian network model. The basic theory of sets of distributions, credal and Bayesian networks is reviewed in Section 2.

In this paper we produce algorithms for *approximate inference*; that is, algorithms that approximate *lower* and *upper* conditional probabilities for a variable given observations. Such algorithms are necessary in practice, as exact inference in credal networks is a complex problem, typically of higher complexity than exact inference in Bayesian networks [23]. The best existing exact algorithms operate by converting inference into an optimization problem [9,21,28]; currently they can produce inferences for medium-sized networks, provided the network topology is not dense. Even if future developments lead to extraordinary improvements in exact inference, it seems that approximate inference are unavoidable in applications.

Here we ask, can credal networks benefit from approximation techniques that have been very successful for Bayesian networks and that are based on polytrees? We answer this question positively. Most ideas in this paper can be applied to networks containing non-binary variables; however, their effectiveness depends on the existence of efficient algorithms for inference in auxiliary polytree-like networks. We propose algorithms for approximate inference that *exploit the surprising properties of polytree-like credal networks with binary variables*; specifically, the fact that in this case inference is polynomial, as shown by the *2U algorithm* [27].

We present three algorithms: [3]

(1) The *Loopy 2U* algorithm, presented in Section 3, extends the popular Loopy Belief Propagation [46] algorithm to credal networks with binary variables. Just as Loopy Belief Propagation modifies Pearl's Belief Propagation [51], Loopy 2U modifies the 2U algorithm, with excellent results

[3] These algorithms have been introduced in [38], [39], and [40].

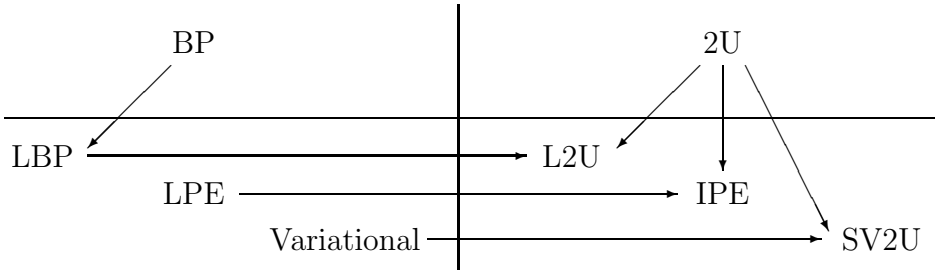


Fig. 1. Relationships amongst algorithms. Upper half displays existing exact algorithms (BP for Bayesian networks and 2U for credal networks with binary variables). The lower left cell displays existing approximate techniques for Bayesian networks: LBP, LPE, and variational methods. The lower right cell contains the contributions of this paper: all of them use the 2U algorithm and each one of them is inspired by an algorithm for Bayesian networks.

(fast response, outstanding accuracy).

(2) The *Iterated Partial Evaluation* algorithm, presented in Section 4, extends the Localized Partial Evaluation [25] algorithm by iterating through many instances of Localized Partial Evaluation (each instance corresponds to a loop cutset, and is run by the 2U algorithm). The Iterated Partial Evaluation algorithm produces lower and upper bounds on probabilities that surely enclose the tighest possible bounds.

(3) The *Structured Variational 2U*, presented in Section 5, uses a variational technique, often employed in large statistical models [56], to generate an approximating polytree-like credal network. When all variables are binary, this approximating credal network can be processed by the 2U algorithm.

Schematically, these algorithms can be organized as in Figure 1, where we use several abbreviations that are adopted throughout: L2U for Loopy 2U, LBP for Loopy Belief Propagation, BP for Belief Propagation, IPE for Iterated Partial Evaluation, LPE for Localized Partial Evaluation, SV2U for Structured Variational 2U. Similarly to their counterparts for Bayesian networks, not all algorithms guarantee convergence to proper bounds; we discuss this issue and investigate the practical behavior of the algorithms through experiments (Section 6). Overall, the Loopy 2U shows the best performance in terms of accuracy and running time, while the IPE is the only one with guarantees concerning accuracy, and the SV2U algorithm seems to be promising as a first step towards future treatment of continuous variables.

2 Background

A Bayesian network uses a directed acyclic graph to compactly encode a probability distribution [49,51]. The term "polytree" is often used to refer

to Bayesian networks whose underlying undirected graph is a tree, but in this paper we refer to such networks as polytree-like networks (because we have several types of networks, with differences that go beyond graph topology). In this paper the nodes of every graph are random variables; given a graph, the set of its nodes/variables is denoted by $\mathbf{X}$. In this paper all variables are categorical; in fact all variables are binary with values 0 and 1. If an edge leaves node X_i and reaches node X_j, then X_i is a *parent* of X_j. The set of parents of X_i is denoted by $\text{pa}\,(X_i)$.

A Bayesian network is endowed with a *Markov condition*: each node is independent of its nondescendants given its parents. Consequently, the distribution $p(\mathbf{X})$ factorizes as $\prod_{i=1}^{n} p(X_i|\text{pa}\,(X_i))$. Note that $p(X_i|\text{pa}\,(X_i))$ is the marginal of X_i whenever $\text{pa}\,(X_i)$ is empty. An inference in a Bayesian network is usually taken as the calculation of the distribution for a variable X_Q given a set $\mathbf{E}$ of assignments for variables $X_{\mathbf{E}}$ (this process is also referred to as *belief updating* [24]). For example, if $\mathbf{E} = \{X_2 = 0, X_5 = 1\}$, then $X_{\mathbf{E}} = \{X_2, X_5\}$. Thus an inference is:

$$p(X_Q|\mathbf{E}) = \frac{p(X_Q, \mathbf{E})}{p(\mathbf{E})} = \frac{\sum_{\mathbf{X}\setminus\{X_Q \cup X_{\mathbf{E}}\}} p(\mathbf{X})}{\sum_{\mathbf{X}\setminus X_{\mathbf{E}}} p(\mathbf{X})}. \tag{1}$$

In this expression, the summation is over the *values* of its subscripting variables, not over the variables themselves. Whenever a summation has subscripting variables, it runs over the values of the variables.

Inference in Bayesian network is an PP-complete problem [54]; however, there exist algorithms that work well in practical applications. Exact algorithms have explored Expression (1) to order operations efficiently [24,45], sometimes using auxiliary junction trees [15,42]. A few algorithms exploit conditioning operations [51,57] that reduce inference to manipulation of polytrees. These conditioning algorithms employ *loop cutsets*: a loop cutset is a set of edges that, once removed, leaves the underlying undirected graph as a tree [51]. For a network with n nodes and n_a arcs, we must remove $n_a - n + 1$ edges so as to obtain a loop cutset. There are also other inference algorithms that combine auxiliary operations and conditioning, without necessarily resorting to loop cutsets [20,52]. Finally, we note that Pearl's belief propagation (BP) is a polynomial algorithm for the special case of polytree-like Bayesian networks [51].

Given that large multi-connected networks pose serious difficulties for exact inference, approximate algorithms have received steady attention. Approximations are often based on Monte Carlo schemes [30,32,36], on structural or variational changes in networks [25,41,43], or in specialized techniques such as Loopy Belief Propagation [48,64]. We briefly review variational techniques in Section 5 as we use them in the SV2U algorithm.

Much as Bayesian networks offer an organized way to encode a single probability distribution, credal networks offer an organized way to encode a set of probability distributions. There are many different formalisms that can be expressed as or related to sets of probability distributions: belief functions and possibility measures [61], ordinal ranks and several types of qualitative probability [13,22,53]. There are also situations where probabilistic assessments are imprecise or vague, sometimes due to constraints in elicitation resources, sometimes due to properties of the representation. For example, consider probabilistic logics; that is, logics with probabilistic assessments over logical formulas [12,34,35,50]. In these logics it is almost impossible to guarantee that every set of formulas attaches a single probability number to each event; usually all that is guaranteed is that an event is associated with a probability interval. Another source of imprecision in probability values is lack of consensus, when several experts disagree on the probability of events or variables. As another source of imprecision, one may wish to abstract away details of a probabilistic model and let the modeling process stop at probability intervals [33].

Denote by $K(X)$ a set of distributions associated with variable X; such sets are referred to as *credal sets* [44]. A conditional credal set, that is, a set of conditional distributions, is denoted by $K(X|A)$, where A is the conditioning event. We denote by $K(X|Y)$ the collection of credal sets indexed by the values of variable Y (note that this is not the single set of functions $p(X|Y)$). Given a credal set $K(X)$, one can compute the *lower probability* $\underline{P}(A) = \min_{P \in K(X)} P(A)$ of event A. In words: the lower probability of event A is the tight lower bound for the probability of A. Similarly, the *upper probability* is $\overline{P}(A) = \max_{P \in K(X)} P(A)$. We assume that all credal sets are closed. To simplify the presentation, we also assume that all credal sets are convex. If a credal set is not convex, we can consider its convex hull for the purposes of this paper, as any lower/upper probability is attained at a vertex of the credal set [27].

Suppose a set of assessments, containing bounds on probability and possibly bounds on expectations, is specified. Consider for example a binary variable X and assessments $P(X = 0) \geq 1/2$ and $P(X = 1) \geq 2/3$. These assessments are inconsistent as no probability distribution can satisfy them; they are said to *incur sure loss* [60]. As another example, consider again binary X and assessments $P(X = 0) \geq 1/2$ and $P(X = 1) \leq 2/3$. These assessments *avoid sure loss*, as there is at least a probability distribution satisfying them [5]. However, the assessments are not as tight as possible, as $P(X = 1)$ must be smaller than $1/2$. If all assessments are tight, the set of assessments is *coherent*. For example, assessments $P(X = 0) \geq 1/2$ and $P(X = 1) \geq 1/3$ are coherent.

A set of assessments that avoids sure loss is usually satisfied by several sets of probability distributions. Each one of these sets is an *extension* of the assessments. We are always interested in the largest possible extension; for finite

domains, this largest extension is always well defined and called the *natural extension* of the assessments [60].

Consider then a directed acyclic graph where each node is associated with a variable X_i, and where the directed local Markov condition holds (that is, a node X_i is independent of its nondescendants given its parents). There are in fact several possible Markov conditions, as there are different concepts of independence for sets of probability distributions in the literature [14,18]. In this paper, "independence" of X and Y means that the vertices of $K(X,Y)$ factorize. That is, each distribution $p(X,Y)$ that is a vertex of the set $K(X,Y)$ satisfies $P(X,Y) = P(X)\,P(Y)$ for all values of X and Y (and likewise for conditional independence).

Suppose that each node X_i and each configuration ρ_{ik} of parents of X_i in a credal network is associated with a conditional credal set $K(X_i|\mathrm{pa}\,(X_i) = \rho_{ik})$. Suppose also that each set $K(X_i|\mathrm{pa}\,(X_i) = \rho_{ik})$ is specified separately from all others; that is, there are no constraints among distributions in these sets. The credal network is then said to be *separately specified*. The largest extension of this credal network that complies with the Markov condition in the previous paragraph is called the *strong extension* of the network [17]:

$$\left\{ \prod_{i=1}^{n} p(X_i|\mathrm{pa}\,(X_i)) : p(X_i|\mathrm{pa}\,(X_i)) \in K(X_i|\mathrm{pa}\,(X_i)) \right\}. \tag{2}$$

An inference in a credal network is usually taken as the calculation of a lower probability conditional on observations: it is necessary to minimize Expression (1) subject to constraints in (2). A similar formulation can be used to compute upper probabilities. The resulting optimization problems can be reduced to multilinear programming [21], and they can be solved in exact or approximate forms. Exact algorithms have either explored the exhaustive propagation of vertices of relevant credal sets [10,16], a process with high computational demands; or have explored more direct optimization methods [1,9,21,28]. Several approximate algorithms employ techniques such as local or genetic search and simulated annealing to produce bounds [6,7,16,65].

One of the first approximate algorithms for inference in credal networks is Tessem's propagation for polytree-like networks [59]. This algorithm mimics Pearl's BP, using only "local" operations (that is, summations and products in a node). While each local optimization can be solved exactly, their combined result produces approximate lower and upper probability bounds. Later, Zaffalon noticed that Pearl's BP could be modified and applied to polytree-like credal networks with binary variables so as to produce *exact* inference through local operations. The resulting algorithm, called 2U, is the only polynomial algorithm for inference in credal networks. As the 2U algorithm is the basis for all algorithms in this paper, it is presented in Appendix A and is assumed

known in the remainder of the paper — it is important to note that we use the notation in Appendix A without further explanation. In the 2U algorithm, interval-valued messages are propagated among nodes of a polytree-like network, much like Pearl's BP.

3 The L2U algorithm: Loopy belief propagation in credal networks

As indicated before, the L2U algorithm is a "loopy" version of the 2U algorithm, inspired by the Loopy Belief Propagation (LBP) algorithm that has been so successful in Bayesian networks [46,48].

The idea is simple. Consider a multi-connected credal network; that is, a network with cycles in the underlying graph. Take an ordering of the nodes, and initialize messages as in the 2U algorithm: that is, a root node X gets $\pi_X(x) = [\underline{P}(X = x), \overline{P}(X = x)]$, a barren node X gets $\Lambda_X = [1, 1]$ and an observed node X receives a dummy child X' that sends a message $\Lambda_{X',X} = 0$ if $\{X = 0\}$ and $\Lambda_{X',X} = \infty$ if $\{X = 1\}$ (Appendix A). All other messages are initialized with the interval $[1, 1]$. (If we are only interested in a particular variable X_Q, then it is possible to discard barren nodes, and several others, using d-separation [31]).

All nodes are then updated in the selected order. That is, messages are updated by running the formulas of the 2U algorithm. And the propagation does not stop after the nodes are exhausted; rather, a complete run over the network is a single *iteration* of L2U. The algorithm then keeps sending interval-valued messages. The process stops when messages converge or when a maximum number of iterations is executed.

A description of the L2U algorithm is presented in Figure 2. Lines 01 to 03 initialize various components of the algorithm. Lines 04 to 10 run the main loop, and line 11 produces the approximate bounds for $P(X_Q = x_Q|\mathbf{E})$. As in LBP, the nodes can be visited in any order; it has been empirically noted that the ordering of the nodes affects convergence of LBP [64], and we leave for future work an in-depth study of the relationship between orderings and convergence in L2U. It should also be noted that the algorithm updates all functions related to a node using the necessary messages from the *previous* iteration. This is also not required; messages produced in iteration $(t+1)$ may use other messages produced in the same iteration. In fact, in our implementation we use the most recent messages in the computation as the algorithm progresses, as we have concluded empirically that this strategy accelerates convergence and does not seem to affect accuracy.

Expressions (A.1) and (A.2) demand considerable computational effort. For

L2U: Loopy 2U

Input: credal network with binary variables;
evidence $\mathbf{E}$;
integer T (limit of iterations).

Output: approximate maximum/minimum values of $P(X_Q|\mathbf{E})$.

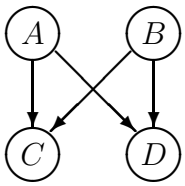
01. Initialize messages of root nodes, barren nodes, and observed nodes as in the 2U algorithm (Appendix A); these messages are marked with a superscript (0).
02. All messages not initialized in the previous step are initialized with the interval $[1, 1]$, and marked with a superscript (0).
03. Take $t \leftarrow 0$.
04. Repeat until convergence of messages or until $t > T$:
05. For each node X, compute:
06. a) $\pi_X^{(t+1)}(X)$ (Expressions (A.1) and A.2)) from $\pi_{U_i,X}^{(t)}(U_i)$;
07. b) $\Lambda_X^{(t+1)}$ (Expressions (A.3) and (A.4)) from $\Lambda_{Y_i,X}^{(t)}$;
08. c) $\pi_{X,Y_j}^{(t+1)}(X)$ (Expressions (A.7) and (A.8)) from $\pi_X^{(t)}(X)$ and $\Lambda_{Y_k,X}^{(t)}$;
09. d) $\Lambda_{X,U_i}^{(t+1)}$ (Expressions (A.15) and (A.16)) from $\Lambda_X^{(t)}$ and $\pi_{U_k,X}^{(t)}(U_k)$.
10. $t \leftarrow t + 1$.
11. Return $[\underline{P}^*(X_Q = x_Q|\mathbf{E}), \overline{P}^*(X_Q = x_Q|\mathbf{E})]$ (Expressions (A.5) and (A.6)) using the messages in the last iteration.

Fig. 2. The L2U algorithm.

each expression, we have a search among $2^{\#\mathrm{pa}(X_i)}$ numbers, where $\#\mathrm{pa}\,(X_i)$ indicates the number of parents of X_i; for each configuration, we must sum across $2^{\#\mathrm{pa}(X_i)}$ probability values. Therefore, if K is the largest number of parents for a node, and the algorithm stops after t^* iterations, the computational effort is $\mathcal{O}(t^* 4^K)$.

The most difficult issue with L2U is convergence. When all probability values are real-valued, L2U collapses to LBP; thus convergence of L2U includes the already difficult (and largely open) issue of convergence of LBP [47,58,62]. In fact, the convergence of L2U may depend on the convergence of LBP, for the following reason. As L2U iterates, it is possible that after some point the same extreme points of probability intervals are always selected in the local optimization problems (A.1), (A.2), (A.15) and (A.16). If that is the case, then L2U operates as propagation on two distinct Bayesian networks in parallel. We have observed this behavior in our tests: after some iterations L2U settles on two Bayesian networks that are then processed in a loopy scheme. We in fact conjecture that convergence of L2U will ultimately rely on the convergence of LBP for all Bayesian networks that are encoded in a credal network; an in-depth study of these matters is left for future work.

We now discuss the steps of the L2U algorithm through a simple example. Con-



$P(a) \in [0.4, 0.5]$	$P(b) \in [0.6, 0.7]$
$P(c\|a, b) \in [0.7, 0.9]$	$P(d\|a, b) \in [0.1, 0.2]$
$P(c\|a, \neg b) \in [0.6, 0.8]$	$P(d\|a, \neg b) \in [0.3, 0.4]$
$P(c\|\neg a, b) \in [0.4, 0.6]$	$P(d\|\neg a, b) \in [0.5, 0.6]$
$P(c\|\neg a, \neg b) \in [0.2, 0.4]$	$P(d\|\neg a, \neg b) \in [0.7, 0.8]$

Fig. 3. Example of separately specified credal network with binary variables.

sider the credal network in Figure 3. In this example we adopt the convention that, for a variable X, the event $\{X = 0\}$ is denoted by $\neg x$ and the event $\{X = 1\}$ is denoted by x. Suppose then that $\mathbf{E} = \{C = 0, D = 1\} = \{\neg c, d\}$, and consider the calculation of $[\underline{P}(a|\mathbf{E}), \overline{P}(a|\mathbf{E})]$. Thus there is a Λ-message equal to 0 to node C, as $\{C = 0\} \in \mathbf{E}$; and there is a Λ-message equal to ∞ to node D, as $\{D = 1\} \in \mathbf{E}$. Suppose the nodes are visited in the sequence $\{B, D, A, C\}$ in each iteration of L2U. The algorithm then computes the following lower bounds (upper bounds have analogous expressions) as it propagates messages.

As B is a root node, $\underline{\pi}_B(b)$ is $\underline{P}(b)$. Also, $\underline{\Lambda}_B = 1$. The message sent to node D has $\underline{\pi}_{B,D}(b) = (1 - (1 - 1/\underline{\pi}_B(b))/\underline{\Lambda}_{C,B})^{-1}$ (Equation (A.7)), where $\underline{\Lambda}_{C,B} = 1$. Node D similarly processes messages; in particular, Expression (A.15) produces:

$$\underline{\Lambda}_{D,A} = \min_{f(b) \in \pi_{B,D}(b)} \left(\frac{\sum_B \underline{p}(d|a, B) \times f(B)}{\sum_B \overline{p}(d|\neg a, B) \times f(B)} \right),$$

and this message is sent to node A. As A is a root node, $\underline{\pi}(a) = \underline{P}(a)$. Node A processes its messages, and sends messages to C and D, as $\underline{\pi}_{A,C}(a) = (1 - (1 - 1/\underline{\pi}_A(a))/\underline{\Lambda}_{D,A})^{-1}$ and $\underline{\pi}_{A,D}(a) = (1 - (1 - 1/\underline{\pi}_A(a))/\underline{\Lambda}_{C,A})^{-1}$ (note that $\underline{\Lambda}_{C,A} = 1$ in this last expression). Node C processes the incoming messages and sends messages to A and B; in particular,

$$\underline{\Lambda}_{C,B} = \min_{f(a) \in \pi_{A,C}(a)} \left(\frac{1 - \sum_A \overline{p}(c|A, b) \times f(A)}{1 - \sum_A \underline{p}(c|A, \neg b) \times f(A)} \right).$$

All messages have been updated at this point; the first iteration has finished. The second iteration goes through all these calculations again, and so forth. A few messages are shown in Table 1. In this example messages reach convergence in 17 iterations; the resulting approximate inference is $[0.0318, 0.2764]$ (the exact solution is $[0.0362, 0.2577]$).

If we stop iterations at $t = 2$, then $\underline{P}(a|\mathbf{E}) = (1 - (1 - 1/\underline{\pi}_A(a))/(\underline{\Lambda}_{C,A} \times$

Table 1
Interval-valued messages propagated by the L2U algorithm for the credal network in Figure 3 ($t = 0, 1, 2$).

Interval-valued messages	$t = 0$	$t = 1$	$t = 2$
$\pi_{B,D}(b)$	[1.0,1.0]	[0.6000,0.7000]	[0.4037,0.7138]
$\Lambda_{D,A}$	[1.0,1.0]	[0.2424,0.4828]	[0.2392,0.5156]
$\pi_{A,C}(a)$	[1.0,1.0]	[0.1391,0.3256]	[0.1375,0.3402]
$\Lambda_{C,B}$	[1.0,1.0]	[0.4514,1.0693]	[0.4488,1.0733]
$\pi_{A,D}(a)$	[1.0,1.0]	[0.4000,0.5000]	[0.1182,0.4163]
$\Lambda_{D,B}$	[1.0,1.0]	[0.5000,0.8148]	[0.5264,0.8468]
$\pi_{B,C}(b)$	[1.0,1.0]	[0.4286,0.6553]	[0.4412,0.6640]
$\Lambda_{C,A}$	[1.0,1.0]	[0.2010,0.7132]	[0.2002,0.7140]

$\underline{\Lambda}_{D,A}))^{-1} = 0.0309$, and $\overline{P}(a|\mathbf{E}) = (1 - (1 - 1/\overline{\pi}_A(a))/(\overline{\Lambda}_{C,A} \times \overline{\Lambda}_{D,A}))^{-1} = 0.2691$. Note that convergence does not necessarily lead to more accurate bounds.

4 The IPE algorithm: Localized partial evaluation in credal networks

The IPE algorithm exploits the technique of *Localized Partial Evaluation (LPE)* developed by Draper and Hanks [25]. The idea here is to adapt LPE to our purposes and to iterate it over, so as to produce increasingly accurate bounds — thus the name Iterated Partial Evaluation. The most positive aspect of IPE is that the resulting bounds are guaranteed to enclose the exact inference (Theorem 1); the disadvantage of the algorithm is that our experiments indicate a loss of accuracy when compared to L2U and SV2U (Section 6).

The original LPE algorithm produces approximate inferences in Bayesian networks by "cutting" edges of the network and then sending *vacuous messages* through these *missing edges*. The vacuous messages are actually probability intervals, and the LPE algorithm then uses an approximate scheme to propagate these probability intervals. In principle the LPE algorithm can be directly applied to credal networks; just select the missing edges, introduce the interval-valued vacuous messages, and propagate all probability intervals together. In the case of binary networks this propagation can be efficient when the missing edges form a *loop cutset*: we can then employ the 2U algorithm to efficiently and exactly handle the vacuous messages. Figure 4 shows a multi-connected network (left) and the same network with missing edges removed so as to ob-

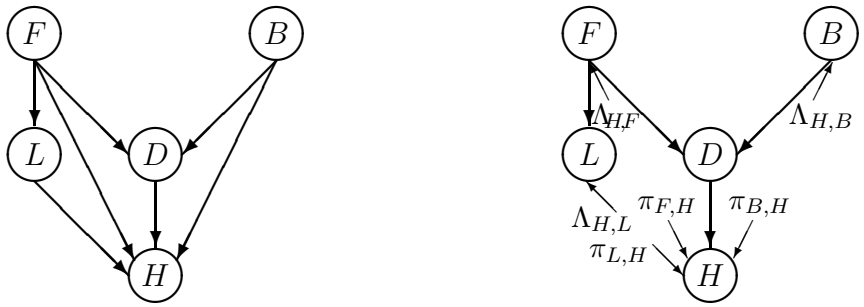


Fig. 4. Missing arcs in the IPE algorithm. Left: original multi-connected network. Right: polytree-like network with missing arcs and their respective vacuous messages, where $\pi_{F,H}$, $\pi_{L,H}$ and $\pi_{B,H}$ are equal to $[0,1]$ and $\Lambda_{H,F}$, $\Lambda_{H,L}$ and $\Lambda_{H,B}$ are equal to $[0,\infty)$.

tain a polytree (right). We emphasize: only in networks with binary variables we obtain an efficient and accurate method, due to the 2U algorithm.

Suppose then that missing edges do form a loop cutset, and that vacuous messages are propagated using the 2U algorithm, thus generating an interval I^* for $P(X_Q = x_Q|\mathbf{E})$. We now show that I^* in fact provides *outer bounds*; that is, $P(X_Q = x_Q|\mathbf{E}) \in I^*$ for every distribution in the strong extension of the credal network:

Theorem 1 *The IPE algorithm returns an outer bound; that is, an interval I^* such that $[\underline{P}(X_Q = x_Q|\mathbf{E}), \underline{P}(X_Q = x_Q|\mathbf{E})] \subseteq I^*$.*

Proof.[4] Only the extreme points of credal sets in the credal network must be inspected to find the lower and upper probabilities of interest [27]. Thus we have a finite number of Bayesian networks that must be inspected; take a loop cutset and for each one of these networks, propagate probability intervals. In our setting, simply run the 2U algorithm as we only have binary variables. We obtain an interval for each Bayesian network; now we use a key result by Draper, who proves that for a particular Bayesian network the produced interval encloses the exact (real-valued) inference for that network [26]. If we run the 2U algorithm directly on the credal network with vacuous messages, the result will certainly include the approximate intervals for each one of the Bayesian networks just mentioned, and by Draper's result, the exact inference for each Bayesian network — and thus the 2U algorithm will produce an interval that encloses the exact probability interval of interest in the original

[4] A reviewer suggested the following interesting proof: for each edge $X \to Y$ that is cut, consider edges $X \to Y'$ and $X' \to Y$ with new variables X' and Y'; a single iteration of IPE then performs the "conservative inference" where X' and Y' are missing *not* at random [3,66]. As each IPE iteration is correct, the intersection of these results is correct.

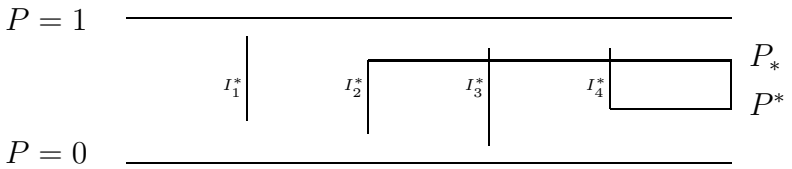


Fig. 5. Intersection of approximate intervals in IPE, to produce outer bounds P_* and P^*.

credal network. This is true for any loop cutset, so if we have a collection of loop cutsets C_t, every interval I_t^* encloses the exact interval, and the intersection $\cap_t I_t^*$ encloses the exact interval as well. QED

Hence it is natural to consider the following procedure. Select a loop cutset C_1 and produce an approximation I_1^* as described; then select another loop cutset C_2 and produce an approximation I_2^*; repeat this for a sequence of loop cutsets. Each loop cutset C_t leads to an interval I_t^* that contains the exact probability interval of interest, thus we can always combine the sequence of approximations by taking their intersection. Figure 5 illustrates this argument (intervals are not related to Figure 4).

The basic computations in the IPE algorithm are depicted in Figure 6. Basically, lines 02 to 05 execute an adapted LPE algorithm, and line 07 returns the intersection of approximate intervals. Lines 02 and 03 produce a polytree by selecting a loop cutset. The 2U algorithm is run in line 05 using vacuous messages. The original LPE algorithm uses intervals $[0, 1]$ for all vacuous messages; here we can use the same strategy for the π-messages but not for the Λ-messages. The later messages represent ratios of probability values, so a vacuous Λ-message is the open interval $[0, \infty)$. The messages flowing from missing edges need not be updated during propagation.

The complexity of IPE algorithm is of same order of 2U algorithm. For T iterations, the complexity is $\mathcal{O}(T4^K)$ where K is the maximum number of parents of a node. For every network there is clearly a limit on T, that is, a maximum number of different loop cutsets that can be generated. Even medium sized networks admit so many loop cutsets that in practice the cutsets are not exhausted. A detailed analysis of the trade-off between the number of visited cutsets and accuracy is left for future work.

Consider again the credal network depicted in Figure 3, and the calculation of $P(a|\mathbf{E})$ where $\mathbf{E} = \{\neg c, d\}$ (we again use x to denote the event $\{X = 1\}$ and similarly for $\neg x$). Remove the edge from B to C and introduce the corresponding vacuous messages: $\Lambda_{C,B} = [0, \infty)$, $\pi_{B,C}(b) = [0, 1]$. Node C receives a Λ-message equal to zero, while node D receives a Λ-message equal to infinity, due to the evidence $\mathbf{E}$. We now run the 2U algorithm; again we only report the lower bounds as upper bounds have similar expressions.

IPE: Iterated Partial Evaluation

Input: credal network with binary variables;
evidence $\mathbf{E}$;
integer T (limit of iterations).

Output: approximate maximum/minimum values of $P(X_Q|\mathbf{E})$.

01. Take $t \leftarrow 0$ and repeat while $t \leq T$:
02. a) Select a cutset C_t.
03. b) Produce a network $\mathcal{B}_t$ by removing the edges in C_t.
04. c) Insert vacuous messages: for a missing edge from X to Y, send
$\pi_{X,Y}(x) = [0, 1]$ to Y, and
$\Lambda_{Y,X} = [0, \infty)$ to X.
05. d) Run the 2U algorithm in the network $\mathcal{B}_t$, producing the interval
$I_t^* = [\underline{P}_t(X_Q = x_Q|\mathbf{E}), \overline{P}_t(X_Q = x_Q|\mathbf{E})]$.
06. e) $t \leftarrow t + 1$.
07. Return $I^* = \cap_t I_t^*$.

Fig. 6. The IPE algorithm.

First, node B sends a message to D, where $\underline{\pi}_{B,D} = (1-(1-1/\underline{\pi}_B(b))/\underline{\Lambda}_{C,B})^{-1} = 0$ using the appropriate conventions. Then D sends a message to A; as $\underline{\Lambda}_D > 1$, we have:

$$\underline{\Lambda}_{D,A} = \min_{f(b)\in\pi_{B,D}(b)} \left(\frac{\sum_B \underline{p}(d|a, B) \times f(B)}{\sum_B \overline{p}(d|\neg a, B) \times f(B)} \right) = 0.1667.$$

Node C also sends a message to A; as $\overline{\Lambda}_C < 1$, we have:

$$\underline{\Lambda}_{C,A} = \min_{f(B)\in\pi_{B,C}(b)} \left(\frac{1 - \sum_B \overline{p}(c|a, B) \times f(B)}{1 - \sum_B \underline{p}(c|\neg a, B) \times f(B)} \right) = 0.1667.$$

By similar computations we obtain $\overline{\Lambda}_{C,A} = 0.5714$ and $\overline{\Lambda}_{D,A} = 0.75$. Hence, $I_1^* = [\underline{P}_1(a|\mathbf{E}), \overline{P}_1(a|\mathbf{E})] = [0.0182, 0.2999]$. The exact interval is $[0.0362, 0.2577]$, clearly contained in I_1^*. This procedure can be repeated for each loop cutset; in this network we only have four possible cutsets. The intersection of the four resulting intervals is returned by the IPE algorithm.

5 The SV2U algorithm: Structured variational methods in credal networks

There are several "variational" methods for approximate inference in Bayesian networks, Markov random fields and similar models [41,55,56,63]. Typically, a variational method selects a family of distributions with desirable properties,

and approximates a distribution P by some distribution Q in the family; one seeks to minimize the distance between P and Q without actually performing inferences with P.

In this section we explore the following idea. Given a credal network with binary variables, we search for the best polytree-like network with binary variables that approximates the original network. Then we process the approximating network with the 2U algorithm. The search for polytree-like approximations mimics the usual variational techniques, but we resort to additional approximations to reduce computational complexity.

5.1 Structured variational methods

We start by briefly reviewing some basic concepts. Suppose we have a Bayesian network $\mathcal{B}$ associated with a joint distribution $P(\mathbf{X})$, where $\mathbf{X}$ represents the set of variables in the network. Suppose variables $X_{\mathbf{E}}$ are observed (that is, the event $\mathbf{E}$ is observed), and define $\mathbf{Y} = \mathbf{X} \backslash \mathbf{X}_{\mathbf{E}}$. We assume that $\mathbf{X}$ and $\mathbf{Y}$ are so ordered that: (i) variables in $X_{\mathbf{E}}$ are the last elements of $\mathbf{X}$; (ii) variables in $\mathbf{Y}$ are in the same order as in $\mathbf{X}$, so that Y_i is the same variable as X_i. For instance, if $\mathbf{X} = \{X_1, X_2, X_3\}$ and $X_{\mathbf{E}} = \{X_3\}$, then $\mathbf{Y} = \{X_1, X_2\}$, so that Y_1 is exactly X_1.

We now want to approximate $P(\mathbf{Y}|\mathbf{E})$ by a distribution $Q(\mathbf{Y})$. We take the Kullback-Leibler (KL) divergence as a "distance" between $P(\mathbf{Y}|\mathbf{E})$ and $Q(\mathbf{Y})$; that is, $KL(Q(\mathbf{Y})||P(\mathbf{Y}|\mathbf{E})) = \sum_{\mathbf{Y}} Q(\mathbf{Y}) \ln Q(\mathbf{Y})/P(\mathbf{Y}|\mathbf{E})$ (note that the Kullback-Leibler divergence is not a true metric).

The goal is to find a good approximation $Q(\mathbf{Y})$ to $P(\mathbf{Y}|\mathbf{E})$ by minimizing $KL(Q||P)$. The approximate distribution $Q(\mathbf{Y})$ should also be easier to handle than $P(\mathbf{Y}|\mathbf{E})$; in a *structured variational method*, one assumes that $Q(\mathbf{Y})$ factorizes as $\prod_i Q_i$, where each Q_i denotes a function of a small number of variables. We restrict attention to approximations that can be represented by Bayesian networks; thus we assume that $Q(\mathbf{Y})$ factorizes as $\prod_{Y_i \in \mathbf{Y}} Q_i(Y_i|\mathrm{pa}\,(Y_i)')$. Note that $\mathrm{pa}\,(Y_i)'$ refers to the parents of Y_i in the approximating distribution, not the original distribution. To simplify the notation, we use P_i and Q_i instead of the more complete forms $P(Y_i|\mathrm{pa}\,(Y_i))$ and $Q_i(Y_i|\mathrm{pa}\,(Y_i)')$.

Consider then the iterative minimization of $KL(Q(\mathbf{Y})||P(\mathbf{Y}|\mathbf{E}))$ by minimizing one component Q_i at a time. That is, we fix all components Q_j for $j \neq i$ and modify Q_i so as to minimize $KL(Q(\mathbf{Y})||P(\mathbf{Y}|\mathbf{E}))$ locally. We then cycle over variables in $\mathbf{Y}$, and keep repeating this procedure until the Kullback-Leibler divergence reaches a stationary point.

Denote by G_i the set containing i and indexes of the children of Y_i in the original network $\mathcal{B}$. Likewise, denote by C_i the set containing the indexes of the children of Y_i in the approximate network $\mathcal{B}'$. It can be shown that once we fix all components Q_j for $j \neq i$, the Kullback-Leibler divergence is minimized with respect to Q_i by taking [63, page 104]:

$$Q_i^*(Y_i|\mathrm{pa}\,(Y_i)') = \lambda_i \exp\left[\sum_{\mathbf{Y}\setminus Y_i}\left(Q_{-i}(\mathbf{Y})\left(\sum_{k'\in G_i}\ln P_{k'} - \sum_{k''\in C_i}\ln Q_{k''}\right)\right)\right], \quad (3)$$

where $Q_{-i}(\mathbf{Y}) = \prod_{j\neq i} Q_j$ and λ_i is a constant such that $\sum_{Y_i} Q_i^*(y_i|\mathrm{pa}\,(Y_i)') = 1$. Note that inner summations run over indexes of variables, not over values of variables. We now observe that many variables are summed out for each term in Expression (3), and consequently:

$$Q_i^*(Y_i|\mathrm{pa}\,(Y_i)') = \lambda_i \exp\left(\left(\sum_{k'\in G_i} M'_{i,k'}\right) - \left(\sum_{k''\in C_i} M''_{i,k''}\right)\right), \quad (4)$$

where

$$M'_{i,k'} = \sum_{\{Y_{k'},\mathrm{pa}(Y_{k'})\}\setminus Y_i}\left(\left(\prod_{Y_{l'}\in\{Y_{k'},\mathrm{pa}(Y_{k'})\}} Q_{l'}\right)\ln P_{k'}\right),$$

$$M''_{i,k''} = \sum_{\{Y_{k''},\mathrm{pa}(Y_{k''})'\}\setminus Y_i}\left(\left(\prod_{Y_{l''}\in\{Y_{k''},\mathrm{pa}(Y_{k''})'\}} Q_{l''}\right)\ln Q_{k''}\right).$$

Note that summations in Expression (4) go over sets of indexes, while summations in the expressions of $M'_{i,k'}$ and $M''_{i,k''}$ go over values of variables; products in the expressions of $M'_{i,k'}$ and $M''_{i,k''}$ go over the variables themselves.

We have reached an updating scheme that depends only on "local" features of the original network (that is, on the variables in the Markov blanket of Y_i). For a network $\mathcal{B}$, we can produce several structured variational approximations by selecting different factorizations for $Q(\mathbf{Y})$. A particularly popular factorization is the *complete* one, in which Q_i depends only on Y_i; this is often called the *mean field* approximation and is attractive for its simplicity, even as it is not always very accurate [41,55]. Then C_i is empty, and

$$Q_i^*(Y_i) = \lambda_i \exp\left(\sum_{k\in G_i}\sum_{\{Y_k,\mathrm{pa}(Y_k)\}\setminus Y_i}\ln P_k \prod_{Y_l\in\{Y_k,\mathrm{pa}(Y_k)\}} Q_l\right). \quad (5)$$

Suppose we have a credal network $\mathcal{B}$ and we must find approximate bounds for $P(X_Q = x_Q|\mathbf{E})$. We wish to construct a structured variational approximation; to do so, we must select a factorization for $Q(\mathbf{Y})$. To clarify the issues involved, we start with the mean field approximation, where Q is represented by a Bayesian network without edges. We have to go over the variables and, for each one of them, update Q_i according to Expression (5). The "exact" way to apply Expression (5) would be to compute it for each possible vertex of the local credal sets. But this would produce a list of distributions for Q_i^*, and this list would have to be combined with the various lists of Q_j^* for $j \neq i$ in the next iteration of the method. That is: while in a Bayesian network the variational techniques require only manipulation of local factors, in a credal network we must keep track of which products of Q_i^* are possible from iteration to iteration. The number of possible combinations becomes unmanageable as iterations are executed.

We propose the following updating scheme. Instead of applying Expression (5) to every possible combination of vertices of local credal sets, we simply compute the upper and lower bounds for $Q_i^*(Y_i)$. For example, the lower bound is

$$\underline{Q}_i^*(Y_i) = \min_{P_k, Q_l} \lambda_i \exp\left(\sum_{k \in G_i} \sum_{\{Y_k, \mathrm{pa}(Y_k)\} \setminus Y_i} \ln P_k \prod_{Y_l \in \{Y_k, \mathrm{pa}(Y_k)\}} Q_l \right),$$

where the minimization is over the relevant local credal sets in the original network (that is, P_k) and in the approximating network (that is, Q_l). Note that $\sum_{Y_i} Q_i(Y_i) = 1$, thus it is only necessary to compute upper and lower bounds of Q_i for one value of Y_i. Such bounds can be computed using local information only, as they depend on the bounds for local credal sets and other Q_j^*. This interval-valued updating introduces approximations beyond those induced by the particular structure of the Q_i; in particular, we do not have guarantees of convergence to a local minimum of the Kullback-Leibler divergence (in standard variational methods it is usually the case that a global minimum of the Kullback-Leibler is attained). However, note that in our setting we cannot expect convergence to a single minimum, as we are dealing with sets of distributions and this may introduce a partial order over approximating distributions. Moreover, the validity of variational methods lies in their practical success, not in the fact that they minimize a "distance" that is not even symmetric; thus we have investigated the validity of these approximations empirically (Section 6), particularly for structured approximations using polytrees (as the naive mean field approximation turned out not to be accurate in our preliminary experiments [38]).

SV2U: Structured Variational 2U

Input: credal network $\mathcal{B}$ with binary variables;
evidence $\mathbf{E}$;
integer T (limit of iterations).

Output: approximate maximum/minimum values of $P(X_Q|\mathbf{E})$.

01. Select a loop cutset and generate a polytree from $\mathcal{B}$.
02. Build the sets G_i and C_i for all variables Y_i, where $\mathbf{Y} = \mathbf{X} \backslash \mathbf{X_E}$.
03. For each Y_i:
04. If an incoming edge to Y_i has been removed by the cutset,
05. then mark Y_i and take Q_i as uniform for each combination of $\text{pa}\,(Y_i)'$ (that is, take $Q_i(Y_i|\text{pa}\,(Y_i)') = [0.5, 0.5]$);
06. otherwise, leave Y_i unmarked and take $Q_i = P_i$.
07. Repeat until convergence of distributions Q_i (or until number of iterations reaches T):
08. For every marked Y_i, and for every configuration of $\text{pa}\,(Y_i)'$:
09. Update Q_i using the probability interval (Expressions (6) and (7)) $[\underline{Q}^*(Y_i|\text{pa}\,(Y_i)'), \overline{Q}^*(Y_i|\text{pa}\,(Y_i)']$.
10. Run the 2U algorithm in the approximating network (a polytree-like network with distributions Q_i) using the 2U algorithm, thus producing approximate bounds on $P(X_Q = x_Q|\mathbf{E})$.

Fig. 7. The SV2U algorithm.

The resulting algorithm is presented in Figure 7. Given a credal network $\mathcal{B}$ with binary variables, the algorithm first constructs an approximating network $\mathcal{B}'$ that is based on a polytree (lines 01-09) and then runs the 2U algorithm on $\mathcal{B}'$. The approximating network $\mathcal{B}'$ is built in several steps. First, a loop cutset for $\mathcal{B}$ is selected and applied (line 01); then distributions Q_i are initialized (lines 02-06). The loop in lines 03-06 makes sure that a node Y that is not affected by the cutset is also untouched by the variational approximation. Lines 07-09 are responsible for the variational approximation, by iterating the lower and upper bounds of Q_i. That is, by iterating

$$\underline{Q}_i^*(Y_i|\text{pa}\,(Y_i)') = \min \lambda_i \exp\left(\left(\sum_{k' \in G_i} M'_{i,k'}\right) - \left(\sum_{k'' \in C_i} M''_{i,k''}\right)\right), \tag{6}$$

$$\overline{Q}_i^*(Y_i|\text{pa}\,(Y_i)') = \max \lambda_i \exp\left(\left(\sum_{k' \in G_i} M'_{i,k'}\right) - \left(\sum_{k'' \in C_i} M''_{i,k''}\right)\right), \tag{7}$$

where the minimization/maximization is over the values of distributions P_k and Q_l.

The computational effort demanded by the SV2U algorithm depends basically on the size of the Markov blankets in a network. Expressions (6) and (7) require

the examination of $2^{\#G_i}$ configurations (where $\#G_i$ is the size of the Markov blanket of Y_i), and for each configuration a summation over $2^{\#G_i}$ is calculated.

An example may help clarify the details of the SV2U algorithm. Consider the *Pyramid* network depicted in Figure 8 [48]. This network has 28 binary variables. We associate each variable with a randomly generated credal set (that is, with probability intervals). Suppose there is no evidence ($\mathbf{E}$ is empty). A loop cutset is formed by the edges (1,6), (2,6), (2,8), (3,8), (3,10), (4,10) and (4,12). In the resulting polytree-like network we only have to update local credal sets for variables X_6, X_8, X_{10} and X_{12}. Consider the updating of variable X_6: we have $G_6 = \{6, 14, 15, 16\}$ and $C_6 = \{14, 15, 16\}$. Because $Q_i(X_i|\mathrm{pa}\,(X_i)) = P(X_i|\mathrm{pa}\,(X_i))$ for $j = 1, 2, 14, 15, 16$, Expression (4) yields for X_6:

$$\begin{aligned} Q_6^*(x_6) &= \lambda_6 \exp\left(\sum_{X_1,X_2} Q_1(X_1)Q_2(X_2)\ln P(x_6|X_1,X_2)\right) \\ &= \lambda_6 \exp\left(\sum_{X_1,X_2} P(X_1)\,P(X_2)\ln P(x_6|X_1,X_2)\right), \end{aligned}$$

and this expression must be minimized/maximized to produce $\underline{Q}_6$ and $\overline{Q}_6$. Analogously, minimum and maximum values of other approximated local credal sets are derived from:

$$\begin{aligned} Q_8^*(x_8) &= \lambda_8 \exp\left(\sum_{X_2,X_3} P(X_2)\,P(X_3)\ln P(x_8|X_2,X_3)\right), \\ Q_{10}^*(x_{10}) &= \lambda_{10} \exp\left(\sum_{X_3,X_4} P(X_3)\,P(X_4)\ln P(x_{10}|X_3,X_4)\right), \\ Q_{12}^*(x_{12}) &= \lambda_{12} \exp\left(\sum_{X_4} P(X_4)\ln P(x_{12}|X_4)\right). \end{aligned}$$

One iteration already produces the variational approximations, with probability intervals $[0.099, 0.346]$ for $Q_6(1)$, $[0.203, 0.664]$ for $Q_8(1)$, $[0.278, 0.753]$ for $Q_{10}(1)$ and $[0.532, 0.810]$ for $Q_{12}(1)$. The 2U algorithm can now be used to produce approximate inferences.

6 Experiments

Empirical analysis is a necessary companion to the algorithms presented so far. In fact, even their versions for Bayesian networks have relatively scant conver-

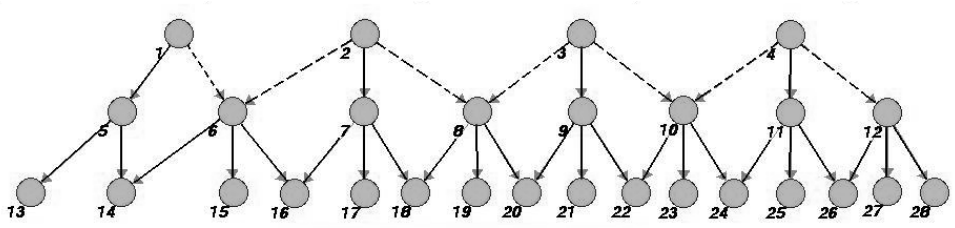


Fig. 8. The *Pyramid* network; dashed arcs belong to the selected loop cutset.

gence and accuracy guarantees; thus a complete understanding of their value must include experiments with simulated and real networks. In this section we report on experiments we have conducted with the L2U, IPE and SV2U algorithms. We report on experiments with randomly generated networks (Section 6.1) and with well-known networks (Section 6.2). When designing these experiments we had to take a few facts into account. First, the generation of ground truth for experiments with credal networks is not a trivial matter. Current exact algorithms can handle networks of up to forty nodes [9,21,28], so we cannot have ground truth for large networks. Moreover, existing approximate algorithms do not have clear guarantees on accuracy, and there are no standard implementations available for them.

We first run tests with small and large artificial networks generated according to several parameters; among those, the *density* of the connections in the network was given attention — density is defined as the ratio between the number of edges and the number of nodes [26]. Then we run experiments with the well-known networks *Pyramid* and *Alarm*.

Experiments were conducted using implementations of 2U, L2U, IPE and SV2U in the Java language (version 1.4), in a PC Intel Pentium 1.7GHz with 480MB RAM. All algorithms in this paper, plus the 2U algorithm, were implemented by the first author in a freely available package called *2UBayes* (http://www.pmr.poli.usp.br/ltd/Software/2UBayes/2UBayes.html). User interfaces and input/output facilities were adapted from the source code of the *JavaBayes* system, a freely available package for inference with Bayesian networks (http://www.pmr.poli.usp.br/ltd/Software/JavaBayes). The graphical user interface is presented in Figure 9. The code declares two real-valued quantities to be equal if their difference is smaller than 10^{-12}; this is used to check convergence.

We compared approximations with exact inferences whenever we could generate the latter, using one of the best algorithms for exact inference in credal networks (we used the LR-based algorithm by Campos and Cozman [21]). We waited up to 30 minutes for an exact inference before declaring it to be unfeasible. The quality of approximations $(\underline{P}^*, \overline{P}^*)$ was measured by the *Mean*

Square Error (MSE) between exact and approximate results [11,29]:

$$\sqrt{(1/2N)\sum_{X}\Big((\underline{P}^{*}(x|\mathbf{E})-\underline{P}(x|\mathbf{E}))^{2}+(\overline{P}^{*}(x|\mathbf{E})-\overline{P}(x|\mathbf{E}))^{2}\Big)}, \tag{8}$$

where the summation runs over all configurations of *unobserved* variables (that is, variables not in $\mathbf{X_E}$), and N is the number of such configurations. We also present later the *Maximum Absolute Error (MAE)*, defined as the largest difference between an approximate inference p and the correct value p^*; that is, $\max|p-p^*|$ (the maximization is over all inferences in a particular credal network). The MAE is not as meaningful as the MSE, as it only displays the absolutely worst result in a large sequence of approximations; however, it is useful later to suggest the relative advantages of the L2U algorithm over the IPE and SV2U algorithms. Clearly it would be desirable to investigate other performance measures such as relative entropy between exact and approximate credal sets, but this often leads to more complex calculations than the inference itself.

6.1 Randomly generated credal networks

We started with tests in small networks, so that approximations could be compared with exact inferences. We generated sets of ten networks with ten nodes each, and varying densities. Here we report on results for networks with densities 1.2 and 1.6; similar results were obtained for density 1.4. These networks were generated with a modified version of the *BNGenerator* package (http://www.pmr.poli.usp.br/ltd/Software/BNGenerator) [37]; this modified version produces random directed acyclic graphs according to various parameters, and then produces random probability intervals for the local credal sets. In all tests (here and in the next section) the IPE algorithm was run with 100 randomly generated cutsets. For each one of these small networks, approximate intervals are computed for the marginal probability of each variable (no evidence was used). The MSE and clock time for inferences are presented in Tables 2 and 3; the results for networks with density 1.4 are omitted as they are similar. Note in particular that one of the networks led to huge processing effort with the SV2U algorithm, possibly as its specific structure led to many combinations between local results.

As we have remarked, the MAE criterion is perhaps too pessimistic as it only captures the worst performance of algorithms in large numbers of runs. But even then, it is interesting to look at MAE values to note the impressive performance of the L2U algorithm; Table 4 shows MAE values that correspond to runs in Table 2. The total average is 0.034436 for L2U, 0.2343 for IPE, and 0.1610 for SV2U. Even more impressive is the fact that for networks with

Table 2
Results (MSE and time) with simulated credal networks: 10 binary variables and 12 edges (density 1.2). Time in seconds.

Credal	L2U		IPE		SV2U	
Network	MSE	Time	MSE	Time	MSE	Time
1	0.007131	0.172	0.08257	8.000	0.01951	0.125
2	0.000054	0.156	0.007918	7.875	0.02733	0.110
3	0.000406	0.156	0.01678	7.860	0.05405	0.141
4	0.001198	0.156	0.1021	7.828	0.017856	0.985
5	0.006121	0.203	0.1100	8.203	0.03783	0.766
6	0.02856	0.250	0.09553	8.156	0.05262	1.172
7	0.004878	0.078	0.02247	8.234	0.03250	1.843
8	0.01816	0.172	0.1031	8.172	0.04208	0.078
9	0.01117	0.093	0.07652	8.266	0.04141	0.625
10	0.01278	0.172	0.05820	8.235	0.1057	1.500

density 1.6 (that is, corresponding to Table 3), the average MAE for the L2U algorithm remains essentially the same, while it grows significantly for the other two algorithms: 0.03444 for L2U, 0.4436 for IPE, and 0.2595 for SV2U.

We also performed experiments with the L2U algorithm in large and very dense credal networks. Unfortunately in this case we cannot compare approximations with exact inferences, thus these experiments are meant to verify convergence and time spent in calculations. Results were rather promising. For example, in credal networks with 50 binary variables and 150 edges (thus, with density 3), we obtained convergence in about a dozen iterations, taking a few minutes of computer time.

6.2 Networks in the literature

Experiments were also run using the structure of the *Pyramid* and *Alarm* networks, mimicking the tests of LBP by Murphy et al. [48]. The *Pyramid* network, depicted in Figure 8, has binary variables and a regular structure that often appears in image processing. The *Alarm* network is a classic tool for medical diagnosis. As some of the variables in the original *Alarm* network are not binary, we modified those nodes so that every variable is binary. We generated probability intervals for several realizations of these networks, running inference (using the L2U, IPE and SV2U algorithms) for all nodes and computing the MSE for each one of them. Again, we run tests without

Table 3
Results (MSE and time) with simulated credal networks: 10 binary variables and 16 edges (density 1.6). Time in seconds.

Credal	L2U		IPE		SV2U	
Network	MSE	Time	MSE	Time	MSE	Time
1	0.01785	0.094	0.2237	8.297	0.04440	0.172
2	0.006300	0.094	0.2087	8.375	0.2203	17.359
3	0.004125	0.157	0.1092	8.219	0.06616	0.828
4	0.02343	0.203	0.1491	8.203	0.05151	2.953
5	0.01650	0.109	0.1620	8.360	0.1027	812.360
6	0.005526	0.188	0.1509	8.406	0.1371	1.281
7	0.002232	0.188	0.2252	8.468	0.02425	0.219
8	0.01416	0.172	0.1123	8.437	0.1211	1.812
9	0.003502	0.172	0.1465	8.328	0.05479	1.281
10	0.01141	0.172	0.1238	8.406	0.04220	0.250

Table 4
Results (MAE) with simulated credal networks: 10 binary variables and 12 edges (density 1.2).

Network	L2U	IPE	SV2U
1	0.02818	0.3362	0.05579
2	0.0002394	0.03056	0.08760
3	0.001424	0.05780	0.2360
4	0.004276	0.4304	0.2369
5	0.02158	0.3110	0.1331
6	0.1271	0.3219	0.2033
7	0.02084	0.06030	0.08795
8	0.05503	0.4096	0.1169
9	0.03932	0.1729	0.1140
10	0.04637	0.2122	0.3382

evidence.

On average, the L2U algorithm converges in just 4 iterations for the *Pyramid* network, and in 9 iterations for the "binarized" *Alarm* network; approximate inference takes a few milliseconds, and the MSE is about 0.013 for both net-

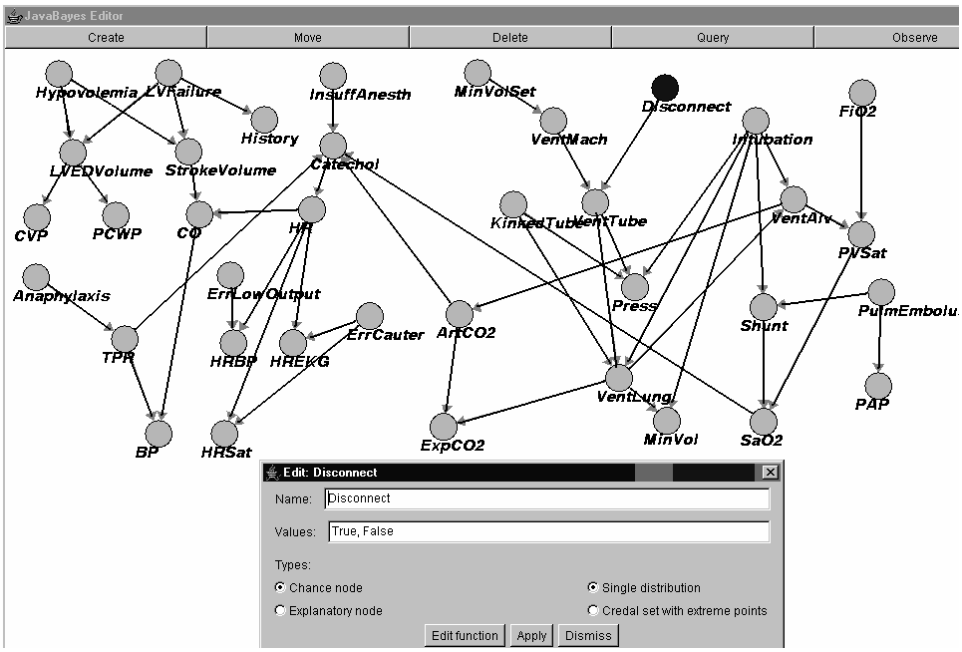


Fig. 9. The "binarized" *Alarm* network (density 1.24) in the JavaBayes user interface.

works. Results for the L2U algorithm are presented in Figure 10 (the figure summarizes all inferences in a single instantiation of the networks). Similar results are presented in Figure 11 for the IPE algorithm; approximations are clearly less accurate (again, all inferences in a single instantiation of the networks). In fact, the MSE is 0.05 for the *Pyramid* network and 0.072 for the "binarized" *Alarm* network, using 100 iterations (both networks are always processed in less than 10 seconds). We could improve accuracy by increasing the number of iterations; we have empirically noted that 100 iterations is a good trade-off between accuracy and computational effort. Figure 12 shows results for the SV2U algorithm (again for a single instantiation of the networks) — MSE is 0.02 for the *Pyramid* network (in 0.078 seconds) and 0.029 for the "binarized" *Alarm* network (in 0.422 seconds).

6.3 Summary

The experiments discussed so far are summarized in Table 5. The L2U algorithm definitively produces the best results (smallest MSE and processing times; the algorithm always converged in all our tests). Note also that L2U's performance seems not to be much affected by the density of the network. The drawback of L2U is the lack of theoretical guarantees concerning convergence

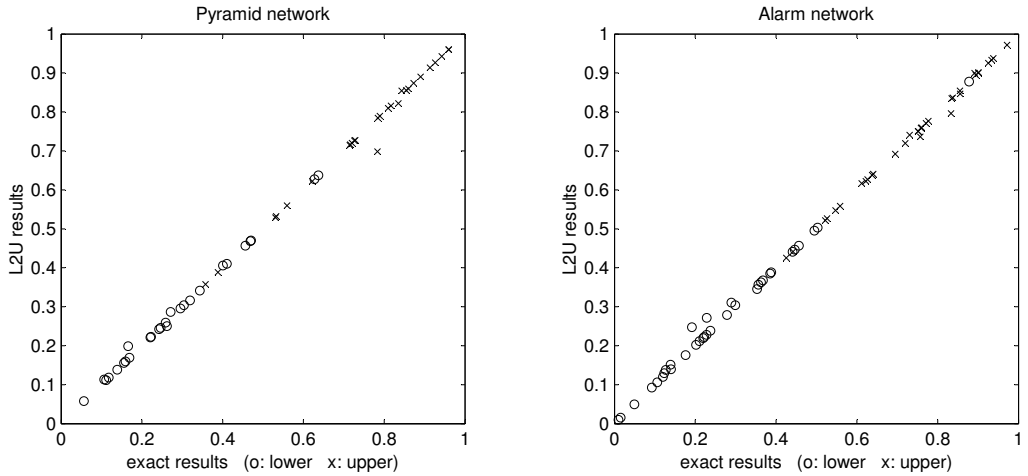


Fig. 10. Correlation between exact and approximate interval extreme values produced by the L2U algorithm for the *Pyramid* network (left) and the "binarized" *Alarm* network (right).

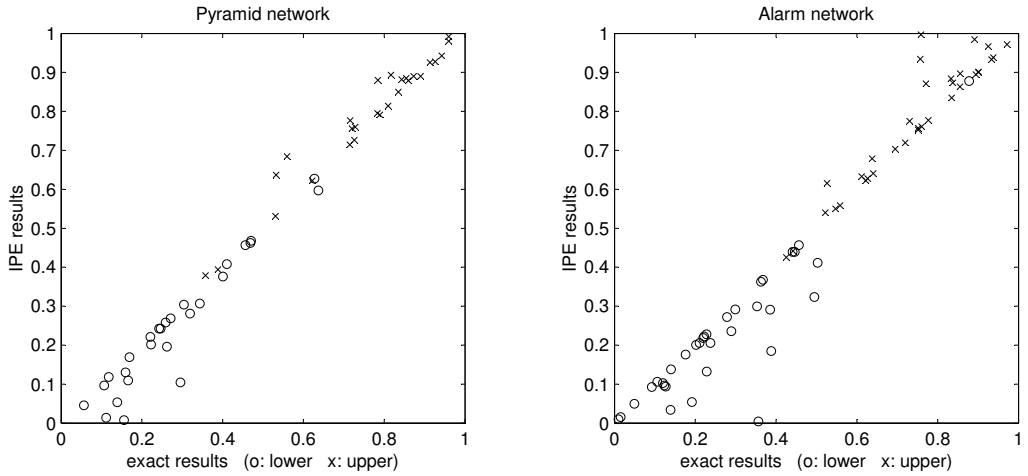


Fig. 11. Correlation between exact and approximate interval extreme values produced by the IPE algorithm for the *Pyramid* network (left) and the "binarized" *Alarm* network (right).

and accuracy. Overall, the algorithm follows the pattern of the LBP algorithm in the literature: excellent empirical results despite few guarantees.

The IPE algorithm offers a different combination: it produces outer bounds, but its accuracy is not spectacular, and processing time is relatively high. The SV2U algorithm offers intermediate accuracy, but large processing times. The reason for this is the following. Both L2U and IPE depend polynomially on the size of the network, and exponentially on the number of parents; however L2U is faster because it requires less iterations. It is always possible that in a particular run the IPE algorithm will hit the best cutsets right on; however in our tests we have seen that many random cutsets have to be generated before we have reasonable accuracy. The SV2U algorithm instead depends exponen-

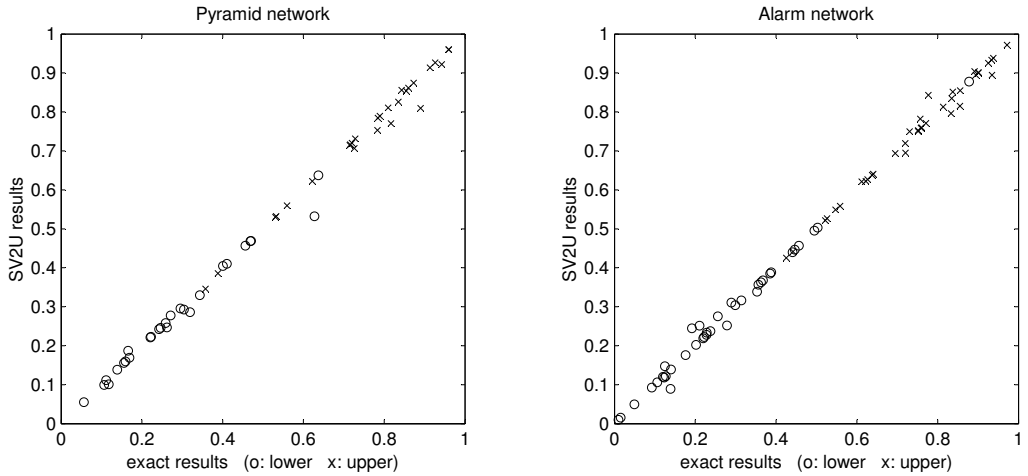


Fig. 12. Correlation between exact and approximate interval extreme values produced by the SV2U algorithm for the *Pyramid* network (left) and the "binarized" *Alarm* network (right).

Table 5
Average MSE and processing time (in seconds) for experiments.

Algorithm	L2U	IPE	SV2U
Artificial networks	0.009	0.068	0.048
(density 1.2)	0.2 sec.	8.0 sec.	0.7 sec.
Artificial networks	0.012	0.189	0.077
(density 1.4)	0.2 sec.	8.3 sec.	25 sec.
Artificial networks	0.011	0.161	0.086
(density 1.6)	0.2 sec.	8.3 sec.	83 sec.
Pyramid network	0.013	0.05	0.02
	0.13 sec.	5.6 sec.	0.08 sec.
"Binarized" Alarm network	0.013	0.072	0.029
	0.16 sec.	7.2 sec.	0.42 sec.

tially on the number of variables in the Markov blanket, and this quantity grows quite fast as density increases. We clearly observe this phenomenon in Table 5. An intriguing aside is that, contrary to L2U, both IPE and SV2U display high variability in performance as density increases.

7 Conclusion

In this work we have produced new algorithms for approximate inference in credal networks, by taking advantage of the 2U algorithm. We have investigated analogues of algorithms that are successful in dealing with Bayesian networks; thus the L2U algorithm mimics LBP, the IPE algorithm extends LPE, and the SV2U algorithm adapts insights from standard structured variational methods. These algorithms can in principle be applied to credal networks with general categorical variables. However, approximations will then require considerable computational effort, because inference in polytree-like credal networks is NP-hard in general [23]. One solution is to "binarize" a network before applying the algorithms; that is, to transform each non-binary variable into a set of binary variables [2].

Each algorithm has strengths and weaknesses. The L2U algorithm is the clear winner for credal networks with binary variables regarding both accuracy and processing time; in fact, this algorithm is possibly the most important contribution of this paper. The IPE algorithm is relatively slow and not very accurate, but it has theoretical guarantees that may make it useful as a component of branch-and-bound algorithms [9,28]; it is thus to be added to a few existing algorithms that produce guaranteed bounds with varying degrees of effort [8,21,59]. The SV2U algorithm offers intermediate accuracy and faces difficulties handling dense networks. Perhaps the most valuable aspect of the SV2U algorithm is that it suggests how variational techniques can be applied to credal networks. Such techniques may be the only effective way to deal with continuous variables in credal networks, a topic that has received scant, if any, attention.

In fact, there are several loosely connected "variational techniques" in the literature, and a natural sequel to the present work would be to explore these techniques. One might seek a better way to minimize the "interval-valued" Kullback-Leibler divergence. Or one might propose a more appropriate distance for interval-probability, for example inspired by Bethe and Kikuchi distances [64]. In fact, we note that Loopy Belief Propagation can be viewed as the iterative minimization of the Bethe energy function, and consequently the L2U algorithm can be framed as an interval-valued version of this variational technique. Apart from such extensions, the most pressing body of work that we leave for the future is the study of convergence in the L2U and the SV2U algorithms.

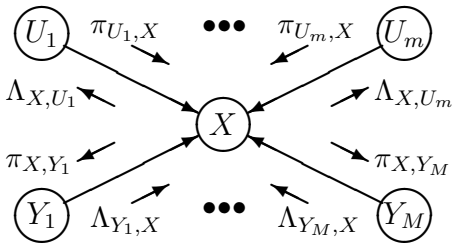


Fig. A.1. Messages propagated in the 2U algorithm [27]. Every node X in a network receives messages $\pi_{U_i,X}$ from its parents and messages $\Lambda_{Y_j,X}$ from its children. Messages are used to update π_X and Λ_X. Then node X sends messages π_{X,U_i} to its parents and Λ_{X,Y_j} to its children.

A The 2U algorithm

The 2U algorithm modifies Pearl's belief propagation (BP) in such a way that inferences are exact for polytree-like credal networks with binary variables [27]. As all variables are binary, the (convex hull of) a conditional credal set $K(X|U=u)$ is completely specified by a coherent probability interval $[\underline{P}(x|U=u)\,,\overline{P}(x|U=u)]$ (for x equal to 0 or to 1).

Messages propagated in the 2U algorithm are depicted in Figure A.1 for a generic node X with parents $U=\{U_1,\ldots,U_m\}$ and children $Y=\{Y_1,\ldots,Y_M\}$. Every message is *interval-valued*. A π-message is an interval-valued function of the variable in its first subscript (for example, both π_Y and $\pi_{Y,X}$ are functions of Y). Thus for each message, say $\pi_{Y,X}$, we have the tight lower bound $\underline{\pi}_{Y,X}(x)$ and the tight upper bound $\overline{\pi}_{Y,X}(x)$. A Λ-message is a single interval, also completely characterized by a tight lower and a tight upper bound.

When a node X receives all messages $\pi_{U_i,X}$ and $\Lambda_{Y_i,X}$, the node updates its "internal" functions π_X and Λ_X as follows:

$$\underline{\pi}_X(x)=\min\left(\sum_U \underline{p}(x|U)\prod_{i=1}^m f_i(U_i) : f_i(u_i)\in\pi_{U_i,X}(u_i), \sum_{U_i} f_i(u_i)=1\right), \quad \text{(A.1)}$$

$$\overline{\pi}_X(x)=\max\left(\sum_U \overline{p}(x|U)\prod_{i=1}^m f_i(U_i) : f_i(u_i)\in\pi_{U_i,X}(u_i), \sum_{U_i} f_i(u_i)=1\right), \quad \text{(A.2)}$$

$$\underline{\Lambda}_X=\prod_{j=1}^M \underline{\Lambda}_{Y_j,X}, \quad \text{(A.3)}$$

$$\overline{\Lambda}_X = \prod_{j=1}^{M} \overline{\Lambda}_{Y_j,X}, \tag{A.4}$$

where Expressions (A.1) and (A.2) require optimization across messages $\pi_{U_i,X}$, and f_i refers to auxiliary real-valued functions. Solutions to these optimization problems are always found at the extremes of $\pi_{U_i,X}$ [27]; consequently solutions can be found by visiting the 2^m possible configurations of U.

It can be shown that all π-messages encode bounds on the probability of X given all evidence in polytrees "above" node X. Likewise, Λ-messages encode bounds on the ratio between the probability of evidence "below" X given $\{X = 1\}$ over the probability of the same evidence given $\{X = 0\}$. Once π_X and Λ_X are computed, we obtain:

$$\underline{P}(X = x|\mathbf{E}) = \left(1 - \left(1 - 1/\underline{\pi}_X(x)\right)/\underline{\Lambda}_X\right)^{-1}, \tag{A.5}$$

$$\overline{P}(X = x|\mathbf{E}) = \left(1 - \left(1 - 1/\overline{\pi}_X(x)\right)/\overline{\Lambda}_X\right)^{-1}. \tag{A.6}$$

Node X can also send messages to its children:

$$\underline{\pi}_{X,Y_j}(x) = \left(1 - \left(1 - 1/\underline{\pi}_X(x)\right) / \left(\prod_{k\neq j} \underline{\Lambda}_{Y_k,X}\right)\right)^{-1}, \tag{A.7}$$

$$\overline{\pi}_{X,Y_j}(x) = \left(1 - \left(1 - 1/\overline{\pi}_X(x)\right) / \left(\prod_{k\neq j} \overline{\Lambda}_{Y_k,X}\right)\right)^{-1}. \tag{A.8}$$

Messages from X to its parents use several auxiliary functions. The message to parent U_i uses auxiliary functions $\underline{g}_i$, $\overline{g}_i$, g'_i, g''_i, $\underline{h}_i$, $\overline{h}_i$, These auxiliary functions are of the form $k_i(\gamma, F)$, where γ is a real number and F is a *set of functions*. During the computation of the message to U_i the set of functions is $\{f_j(U_j)\}_{j=1,\ldots,m;j\neq i}$; that is, there is a function for every parent of X except U_i. Each function $f_j(U_j)$ is completely specified by two real numbers as every variable is binary; it will be clear later that each function f_j must satisfy $\sum_{U_j} f_j(u_j) = 1$, and consequently each function f_j is in fact defined by a single number. To simplify the notation, we denote these sets of functions by $\{f_j\}_{j\neq i}$, to emphasize the fact that function f_i is absent. We also simplify the notation by not indexing explicitly the auxiliary functions by X.

We have:

$$\underline{g}_i(\Lambda, \{f_j\}_{j\neq i}) = \begin{cases} g'_i(\Lambda, \{f_j\}_{j\neq i}) \text{ if } \Lambda \leq 1, \\ g''_i(\Lambda, \{f_j\}_{j\neq i}) \text{ if } \Lambda > 1, \end{cases} \tag{A.9}$$

$$\overline{g}_i(\Lambda, \{f_j\}_{j\neq i}) = \begin{cases} g_i''(\Lambda, \{f_j\}_{j\neq i}) \text{ if } \Lambda \leq 1, \\ g_i'(\Lambda, \{f_j\}_{j\neq i}) \text{ if } \Lambda > 1, \end{cases}, \tag{A.10}$$

where

$$g_i'(\Lambda, \{f_j\}_{j\neq i}) = \frac{(\Lambda-1)\overline{h}_i(1, \{f_j\}_{j\neq i})+1}{(\Lambda-1)\underline{h}_i(0, \{f_j\}_{j\neq i})+1}, \tag{A.11}$$

$$g_i''(\Lambda, \{f_j\}_{j\neq i}) = \frac{(\Lambda-1)\underline{h}_i(1, \{f_j\}_{j\neq i})+1}{(\Lambda-1)\overline{h}_i(0, \{f_j\}_{j\neq i})+1}, \tag{A.12}$$

and finally

$$\underline{h}_i(u_i, \{f_j\}_{j\neq i}) = \sum_{\{U_1,\ldots,U_m\}\setminus U_i} \underline{p}(X=1|U\setminus U_i, U_i=u_i) \prod_{k\neq i} f_k(U_k), \tag{A.13}$$

$$\overline{h}_i(u_i, \{f_j\}_{j\neq i}) = \sum_{\{U_1,\ldots,U_m\}\setminus U_i} \overline{p}(X=1|U\setminus U_i, U_i=u_i) \prod_{k\neq i} f_k(U_k). \tag{A.14}$$

With these definitions in place, node X can produce messages to its parents by local optimization:

$$\begin{aligned} \underline{\Lambda}_{X,U_i} = \min & \underline{g}_i(\Lambda, \{f_j\}_{j\neq i}) \\ & \text{subject to } \Lambda \in \{\underline{\Lambda}_X, \overline{\Lambda}_X\}, \\ & f_j(U_j) \in \pi_{U_j,X}(U_j), \sum_{U_j} f_j(U_j) = 1, \end{aligned} \tag{A.15}$$

$$\begin{aligned} \overline{\Lambda}_{X,U_i} = \min & \overline{g}_i(\Lambda, \{f_j\}_{j\neq i}) \\ & \text{subject to } \Lambda \in \{\underline{\Lambda}_X, \overline{\Lambda}_X\}, \\ & f_j(U_j) \in \pi_{U_j,X}(U_j), \sum_{U_j} f_j(U_j) = 1. \end{aligned} \tag{A.16}$$

Solution of these optimization problems are always found at the extremes of the constraints [27]; consequently solutions can be found by visiting the 2^m extreme points.

The algorithm propagates messages as in Pearl's BP. A root node X is initialized with $\pi_X(x) = [\underline{P}(X=x), \overline{P}(X=x)]$; a barren node X is initialized with $\Lambda_X = [1,1]$. Finally, a node X that is observed (belongs to $X_{\mathbf{E}}$) is processed as follows. A dummy node X' is created and X' sends to X a message $\Lambda_{X',X}$ that is equal to 0 if $\{X=0\} \in \mathbf{E}$ and is equal to ∞ if $\{X=1\} \in \mathbf{E}$. For this to be consistent, it is necessary to propagate messages with value ∞; in some cases it is also necessary to handle messages that apparently require division by zero. As discussed by Fagiuoli and Zaffalon, the algorithm handles all cases correctly provided that: (i) whenever $1/\infty$ is met, it is replaced by

0; (ii) whenever $1/0$ is met, it is replaced by ∞; (iii) whenever Λ is ∞ in Expression (A.9), $\underline{g}_i(\infty, \{f_j\}_{j\neq i}) = \underline{h}_i(1, \{f_j\}_{j\neq i})/\overline{h}_i(0, \{f_j\}_{j\neq i})$; (iv) whenever Λ is ∞ in Expression (A.10), $\overline{g}_i(\infty, \{f_j\}_{j\neq i}) = \overline{h}_i(1, \{f_j\}_{j\neq i})/\underline{h}_i(0, \{f_j\}_{j\neq i})$.

Acknowledgments

The first author was supported by FAPESP (grant 02/0898-2). The second author was partially supported by CNPq (grant 3000183/98-4). The work received substantial support from FAPESP (grant 04/09568-0) and from HP Brazil R&D.

We thank Cassio Polpo de Campos for substantial help in producing exact inferences for our experiments, and the reviewers for valuable suggestions.

References

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

[2] A. Antonucci, M. Zaffalon, J. Ide, and F. G. Cozman. Binarization algorithms for approximate updating in credal nets. In *Third European Starting AI Researcher Symposium (STAIRS'06)*, pages 120–131. IOS Press, 2006.

[3] A. Antonucci, M. Zaffalon. Equivalence between Bayesian and credal nets on an updating problem. In J. Lawry, E. Miranda, A. Bugarin, S. Li, M. A. Gil, P. Grzegorzewski, O. Hryniewicz, editors, *Soft Methods for Integrated Uncertainty Modelling*, pages 223-230. Springer, 2006.

[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. *Second European Conference on Artificial Intelligence in Medicine*, pages 247–256, 1989.

[5] V. Biazzo and A. Gilio. A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments. *International Journal of Approximate Reasoning*, 24(2-3):251–272, 2000.

[6] 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, *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 4–8, Paris, France, July 1994.

[7] 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.

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

[9] A. Cano, M. Gómez, and S. Moral. Application of a hill-climbing algorithm to exact and approximate inference in credal networks. In *Fourth International Symposium on Imprecise Probabilities and their Applications*, pages 88–97, 2005.

[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] J. Cheng and M. Druzdzel. Computational investigation of low-discrepancy sequences in simulation algorithms for Bayesian networks. In C. Boutilier and M. Goldszmidt, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 72–81, San Francisco, California, 2000. Morgan Kaufmann Publishers.

[12] G. Coletti and R. Scozzafava. *Probabilistic Logic in a Coherent Setting*. Trends in logic, 15. Kluwer, Dordrecht, 2002.

[13] G. Coletti. Coherent numerical and ordinal probabilistic assessments. *IEEE Transactions on Systems, Man and Cybernetics*, 24(12):1747–1753, 1994.

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

[15] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, 1999.

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

[17] 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.

[18] F. G. Cozman. Constructing sets of probability measures through Kuznetsov's independence condition. In *International Symposium on Imprecise Probabilities and Their Applications*, pages 104–111, Ithaca, New York, 2001.

[19] F. G. Cozman. Graphical models for imprecise probabilities. *International Journal of Approximate Reasoning*, 39(2-3):167–184, 2005.

[20] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 125(1-2):5–41, 2001.

[21] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In E. Onaindia and S. Staab, editors, *Second Starting AI Researchers' Symposium (STAIRS)*, pages 50–61, Amsterdam, The Netherlands, 2004. IOS Press.

[22] C. P. de Campos and F. G. Cozman. Belief updating and learning in semi-qualitative probabilistic networks. In F. Bacchus and T. Jaakkola, editors, *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 153–160, Edinburgh, Scotland, 2005.

[23] C. P. de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1313–1318, Edinburgh, United Kingdom, 2005.

[24] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In E. Horvitz and F. Jensen, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 211–219, San Francisco, California, 1996. Morgan Kaufmann.

[25] D. L. Draper and S. Hanks. Localized partial evaluation of belief networks. *Conference on Uncertainty in Artificial Intelligence*, pages 170–177, 1994.

[26] D. L. Draper. *Localized Partial Evaluation of Belief Networks*. PhD thesis, Dept. of Computer Science, University of Washington, Washington,WA, 1995.

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

[28] J. C. Ferreira da Rocha and F. G. Cozman. Inference in credal networks: branch-and-bound methods and the A/R+ algorithm. *International Journal of Approximate Reasoning*, 39(2-3):279–296, 2005.

[29] J. C. Ferreira 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, United States, 2003. Morgan Kaufmann.

[30] R. Fung and K. Chag. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Uncertainty in Artificial Intelligence 5*, pages 209–219. Morgan Kaufmann, 1990.

[31] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

[32] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, England, 1996.

[33] V. Ha and P. Haddawy. Theoretical foundations for abstraction-based probabilistic planning. In E. Horvitz and F. Jensen, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 291–298, San Francisco, California, United States, 1996. Morgan Kaufmann.

[34] T. Hailperin. *Sentential Probability Logic*. Lehigh University Press, Bethlehem, United States, 1996.

[35] J. Y. Halpern. *Reasoning about Uncertainty*. MIT Press, Cambridge, Massachusetts, 2003.

[36] M. Henrion. Propagation of uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier/North-Holland, Amsterdam, London, New York, 1988.

[37] J. S. Ide and F. G. Cozman. Generating random Bayesian networks with constraints on induced width. In *European Conference on Artificial Intelligence*, pages 323–327, Amsterdam - The Netherlands, 2004. IOS Press.

[38] J. S. Ide and F. G. Cozman. Approximate inference in credal networks by variational mean field methods. In *International Symposium on Imprecise Probabilities and Their Applications*, pages 203–212, Pittsburgh, Pennsylvania, 2005. Brightdoc.

[39] J. S. Ide and F. G. Cozman. Set-based variational methods in credal networks: the SV2U algorithm. In A. C. Garcia and F. Osório, editors, *XXV Congresso da Sociedade Brasileira de Computação*, volume V Encontro Nacional de Inteligência Artificial, pages 872–881, São Leopoldo, Rio Grande do Sul, Brazil, 2005.

[40] J. S. Ide and F. G. Cozman. IPE and L2U: Approximate algorithms for credal networks. In *Second Starting AI Researcher Symposium (STAIRS)*, pages 118–127. IOS Press, 2004.

[41] T. S. Jaakkola. Tutorial on variational approximation methods. *Advanced Mean Field Methods: Theory and Practice*, pages 129–160, 2001.

[42] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer Verlag, New York, 1996.

[43] M. I. Jordan, Z. Ghahramani, and T. S. Jaakkola. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

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

[45] Z. Li and B. D'Ambrosio. Efficient inference in Bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11:55–81, 1994.

[46] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. Turbo decoding as an instance of Pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communication*, 16(2)(CSD-99-1046):140–152, 1998.

[47] J. M. Mooij and H.J. Kappen. Sufficient conditions for convergence of loopy belief propagation. In *Conference on Uncertainty in Artificial Intelligence*, 2005.

[48] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[49] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[50] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[51] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[52] F. T. Ramos and F. G. Cozman. Anytime anyspace probabilistic inference. *International Journal of Approximate Reasoning*, 38:53–80, 2005.

[53] S. Renooij, S. Parsons, and P. Pardieck. Using kappas as indicators of strength in qualitative probabilistic networks. In T.D. Nielsen and N.L. Zhang, editors, *Seventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 87–99. Springer Verlag, 2003.

[54] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[55] L. K. Saul, T. S. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

[56] L. K. Saul and M. I. Jordan. Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 486–492. MIT Press, Cambridge, MA, 1996.

[57] H. J. Suermondt and G. F. Cooper. Initialization for the method of conditioning in Bayesian belief networks. *Artificial Intelligence*, 50(1):83–94, 1991.

[58] S. C. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In A. Darwiche and N. Friedman, editors, *Conference on Uncertainty in Artificial Intelligence*, pages 493–500, San Francisco, California, 2002. Morgan Kaufmann.

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

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

[61] P. Walley. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83:1–58, 1996.

[62] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. Technical Report CSD-99-1046, CS Department, UC Berkeley, 1999.

[63] J. Winn. *Variational Message Passing and its Applications*. PhD thesis, Department of Physics, University of Cambridge, Cambridge, UK, 2003.

[64] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Neural Information Processing Systems*, pages 689–695, 2000.

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

[66] M. Zaffalon. Conservative rules for predictive inference with incomplete data. In *Fourth International Symposium on Imprecise Probabilities and their Applications*, pages 406–415, 2005.