Generalized Loopy 2U: A New Algorithm for Approximate Inference in Credal Networks

Alessandro Antonucci*,[a], Yi Sun[a], Cassio P. de Campos[a], Marco Zaffalon[a]

[a] *IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Galleria 2, CH-6928 Manno (Lugano), Switzerland*

Abstract

Credal networks generalize Bayesian networks by relaxing the requirement of precision of probabilities. Credal networks are considerably more expressive than Bayesian networks, but this makes belief updating NP-hard even on polytrees. We develop a new efficient algorithm for approximate belief updating in credal networks. The algorithm is based on an important representation result we prove for general credal networks: that any credal network can be equivalently reformulated as a credal network with binary variables; moreover, the transformation, which is considerably more complex than in the Bayesian case, can be implemented in polynomial time. The equivalent binary credal network is then updated by L2U, a loopy approximate algorithm for binary credal networks. Overall, we generalize L2U to non-binary credal networks, obtaining a scalable algorithm for the general case, which is approximate only because of its loopy nature. The accuracy of the inferences with respect to other state-of-the-art algorithms is evaluated by extensive numerical tests.

Key words: credal networks, credal sets, inference algorithms, 2U, imprecise probability, Bayesian networks, loopy belief propagation

1. Introduction

Bayesian networks (Section 2.1) are probabilistic graphical models based on precise assessments for the conditional probability mass functions of the network variables given the values of their parents. As a relaxation of such

*Corresponding author

precise assessments, *credal networks* (Section 2.2) only require the conditional probability mass functions to belong to convex sets of mass functions, i.e., *credal sets*. Credal networks (CNs) are considerably more expressive than Bayesian networks (BNs),[1] and the price is an increased complexity of inference: *belief updating* in CNs is NP-hard even on polytrees [1]. The only known exception to this situation is the *2U algorithm* [2], which computes exact posterior beliefs on *binary* (i.e., with binary variables) polytree-shaped CNs in time linear in the size of the network. A *loopy* version of 2U (L2U) has been proposed for multiply connected binary CNs by [3]. Inferences based on L2U are approximate, but a good accuracy is typically observed after few iterations (Section 3).

In this paper we develop an efficient algorithm for approximate belief updating of general CNs (any topology and number of states per variable). The algorithm, which invokes L2U in its final step, is called *generalized loopy 2U* (GL2U). The GL2U algorithm is based on an important representation result that we prove in Appendix A: that any CN can be equivalently reformulated as one with binary variables. The corresponding transformation, which is considerably more complex than in the Bayesian case, is based on two distinct transformations: a *decision-theoretic specification* [4], which augments the CN with control variables enumerating the multiple mass functions owned by the nodes of the network (Section 4.2); a *binarization* procedure [5] that transforms each variable into a cluster of binary variables (Section 4.1).

We prove that the sequential application of these two transformations, originally developed for independent reasons, returns an equivalent binary representation of the initial CN (Section 5.1). Such equivalent binary CN can be finally updated by L2U. Overall, this is the generalization of the loopy version of 2U that we propose for the updating in general CNs, whose only source of approximation is the loopy propagation (Section 5.2). The algorithm, which is proved to take only polynomial time (Section 6), has been implemented in a free software. Experimental tests (Section 7) show that GL2U is comparable to that of state-of-the-art approximate methods for large CNs in terms of accuracy, being considerably faster from a computational

[1]Greater expressiveness is a consequence of the fact that Bayesian networks are a subset of credal networks. Expressiveness should not be confused with informativeness: for example, it is thanks to the greater expressiveness that credal networks can model much less informative states of knowledge (including lack of knowledge) than those Bayesian networks can model.

point of view.

2. Bayesian and Credal Networks

In this section we review the basics of Bayesian networks (BNs) and their extension to convex sets of probabilities, i.e., credal networks (CNs). Both the models are based on a collection of random variables, structured as a vector $\mathbf{X} := (X_1, \ldots, X_n)$,[2] and a directed acyclic graph (DAG) $\mathcal{G}$, whose nodes are associated with the variables of $\mathbf{X}$. In our assumptions the variables in $\mathbf{X}$ take values in finite sets. For both models, we assume the *Markov condition* to make $\mathcal{G}$ represent probabilistic independence relations between the variables in $\mathbf{X}$: every variable is independent of its non-descendant conditional on its parents. What makes BNs and CNs different is a different notion of independence and a different characterization of the conditional mass functions for each variable given the values of the parents, which will be detailed later.

Regarding notation, for each $X_i \in \mathbf{X}$, $\Omega_{X_i} = \{x_{i0}, x_{i1}, \ldots, x_{i(d_i-1)}\}$ denotes the possibility space of X_i, $P(X_i)$ is a mass function for X_i and $P(x_i)$ the probability that $X_i = x_i$, where x_i is a generic element of Ω_{X_i}. A similar notation with uppercase subscripts (e.g., X_E) denotes vectors (and sets) of variables in $\mathbf{X}$. Regarding the parents and the children of X_i with respect to the topology of $\mathcal{G}$, they are denoted respectively by Π_i and Γ_i. Finally, for each $\pi_i \in \Omega_{\Pi_i}$, $P(X_i|\pi_i)$ is the probability mass function for X_i conditional on $\Pi_i = \pi_i$.

2.1. Bayesian Networks

For BNs, a conditional mass function $P(X_i|\pi_i)$ for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$ must be defined; and the standard notion of probabilistic independence is assumed in the Markov condition. A BN can therefore be regarded as a joint probability mass function over $\mathbf{X}$ that, according to the Markov condition, factorizes as follows:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i), \tag{1}$$

for all the possible values $\mathbf{x} \in \Omega_{\mathbf{X}}$, with the values of x_i and π_i consistent with $\mathbf{x}$. In the following, we represent a BN as a pair $\langle \mathcal{G}, P(\mathbf{X}) \rangle$. Posterior

[2]The symbol ":=" is used for definitions.

beliefs about a queried variable X_q, given evidence $X_E{=}x_E$, are defined by the expression

$$P(x_q|x_E) = \frac{\sum_{x_M} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{x_M,x_q} \prod_{i=1}^{n} P(x_i|\pi_i)}, \tag{2}$$

where $X_M := \mathbf{X} \setminus (\{X_q\} \cup X_E)$, the domains of the arguments of the sums are left implicit and the values of x_i and π_i are those consistent with $\mathbf{x} = (x_q, x_M, x_E)$. Evaluating Equation (2) is an NP-hard task, but for polytrees, Pearl's belief propagation allows for efficient updating [6].

2.2. Credal Sets and Credal Networks

CNs relax BNs by allowing for *imprecise probability* statements: in our assumptions, the conditional mass functions of a CN are required to belong to a *finitely generated* credal set, i.e., the convex hull of a finite number of mass functions for a certain variable. Geometrically, a credal set is a *polytope*. A credal set contains an infinite number of mass functions, but only a finite number of extreme mass functions corresponding to the *vertices* of the polytope. Updating based on a credal set is equivalent to that based only on its vertices [7]. A credal set over X will be denoted as $K(X)$ and the set of its vertices as $\mathrm{ext}[K(X)]$. Given a non-empty $\Omega_X^* \subseteq \Omega_X$, an important credal set for our purposes is the *vacuous credal set* relative to Ω_X^*, i.e., the set of all the mass functions for X assigning probability one to Ω_X^*. We denote this set by $K_{\Omega_X^*}(X)$. In the following we will use the well-known fact that the vertices of $K_{\Omega_X^*}(X)$ are the[3] $|\Omega_X^*|$ degenerate mass functions assigning probability one to the single elements of Ω_X^*. Marginalization generalizes to credal sets as follows: the marginalization $K(X)$ of a joint credal set $K(X,Y)$ to X is the convex hull of the mass functions $P(X)$ obtained from the marginalization of $P(X,Y)$ to X for each $P(X,Y) \in K(X,Y)$.

In order to specify a CN over the variables in $\mathbf{X}$ based on $\mathcal{G}$, a collection of conditional credal sets $K(X_i|\pi_i)$, one for each $\pi_i \in \Omega_{\Pi_i}$, should be provided separately for each $X_i \in \mathbf{X}$; regarding the Markov condition, we assume *strong independence* [8], i.e., standard stochastic independence to be satisfied by every vertex. A CN associated with these local specifications is said to be with *separately specified* credal sets.

Figure 1 reports a CN, whose specification requires the (separate) assessment of an unconditional credal set for X_1, and respectively two and eight

[3]The cardinality of a set Ω is denoted as $|\Omega|$.

conditional credal sets for X_2 and X_3.

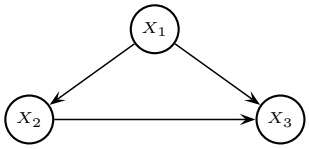

Figure 1: A separately specified CN over (X_1, X_2, X_3), with $|\Omega_{X_1}|=2$, $|\Omega_{X_2}|=|\Omega_{X_3}|=4$.

The specification becomes global considering the *strong extension* $K(\mathbf{X})$ of the CN, i.e., the convex hull of the following collection of joint mass functions:

$$\left\{ P(\mathbf{X}): \quad \begin{array}{ll} P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i) & \forall \mathbf{x} \in \Omega_{\mathbf{X}}, \\ P(X_i|\pi_i) \in K(X_i|\pi_i) & \forall \pi_i \in \Omega_{\Pi_i} \end{array} \right\}. \tag{3}$$

We represent a CN as a pair $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$, with $\mathbf{P}(\mathbf{X}) := \{P_k(\mathbf{X})\}_{k=1}^{n_v} := \text{ext}[K(\mathbf{X})]$. Clearly, for each $k = 1, \ldots, n_v$, $\langle \mathcal{G}, P_k(\mathbf{X}) \rangle$ is a BN. For this reason a CN can be regarded as a finite set of BNs. For CNs *updating* is intended as the computation of tight bounds of the posterior probabilities of a queried variable given some evidence, i.e., Equation (2) generalizes as: [4]

$$\underline{P}(x_q|x_E) = \min_{k=1,\ldots,n_v} \frac{\sum_{x_M} \prod_{i=1}^{n} P_k(x_i|\pi_i)}{\sum_{x_M, x_q} \prod_{i=1}^{n} P_k(x_i|\pi_i)}, \tag{4}$$

and similarly for the upper probabilities $\overline{P}(x_q|x_E)$. Exact updating in CNs displays high complexity. Updating in polytree-shaped CNs is NP-hard, and NP^{PP}-hard in general CNs [1]. The only known exact linear-time algorithm for updating a specific class of CNs is the 2U algorithm, which is reviewed in the next section.

3. 2U and Its Loopy Extension

Pearl's belief propagation scheme for updating polytree-shaped BNs has been extended to polytree-shaped binary CNs in [2]. The resulting algorithm, called *2-Updating* (2U), is the only known exact procedure for efficient CNs updating.

[4] Some authors call the computation in Equation (2) *basic updating* in order to emphasize that more general expected values can be also considered in that formula.

Here we briefly review some features of this algorithm by assuming the reader familiar with the main ideas of belief propagation. Let us therefore consider a polytree-shaped binary CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$. To each $X_i \in \mathbf{X}$ associate the *lower messages* $\underline{\mu}(X_i)$ and $\underline{\Lambda}_i$, which are received respectively from the parents Π_i and the children Γ_i of X_i. Given an evidence $X_E = x_E$, 2U performs a distributed calculation of these messages, from which any updated probability as in Equation (4) can be computed as follows:[5]

$$\underline{P}(x_q|x_E) = \left(1 + \left(\frac{1}{\underline{\mu}(x_q)} - 1\right)\frac{1}{\underline{\Lambda}_q}\right)^{-1}. \tag{5}$$

In order to implement the distributed computation, also the arcs of $\mathcal{G}$ should be equipped with lower and upper messages. Thus, for each pair of nodes (X_j, X_i) such that $\mathcal{G}$ has an arc from X_j to X_i, the lower messages $\underline{\mu}_i(X_j)$ and $\underline{\Lambda}^i_j$ are also provided. The messages associated to the incoming and outgoing arcs of X_i are used to compute those associated to X_i, according to the following equations:

$$\underline{\mu}(x_i) = \min_{\substack{j:X_j\in\Pi_i \\ \mu_i(x_j)\in\{\underline{\mu}_i(x_j),\overline{\mu}_i(x_j)\}}} \sum_{\pi_i\in\Omega_{\Pi_i}} \underline{P}(x_i|\pi_i)\cdot \prod_{j:X_j\in\Pi_i} \mu_i(x_j), \tag{6}$$

$$\underline{\Lambda}_i = \prod_{k:X_k\in\Gamma_i} \underline{\Lambda}^k_i, \tag{7}$$

where x_j is the state of X_j consistent with π_i. The messages associated to the arcs are indeed computed as follows:

$$\underline{\mu}_i(x_j) = \left(1 + \left(\frac{1}{\underline{\mu}(x_j)} - 1\right)\frac{1}{\prod_{l:X_l\in\Gamma_j, l\neq i}\underline{\Lambda}^l_j}\right)^{-1}, \tag{8}$$

$$\underline{\Lambda}^k_i = \min_{\substack{l:X_l\in\Pi_i, l\neq k \\ \mu_i(x_l)\in\{\underline{\mu}_i(x_l),\overline{\mu}_i(x_l)\}}} \left(\min_{\Lambda_i\in\{\underline{\Lambda}_i,\overline{\Lambda}_i\}} \underline{c}^k_i(\Lambda_i)\right), \tag{9}$$

[5] An analogous relation is provided for the upper probability $\overline{P}(x_q|x_E)$ by simply replacing the lower with the corresponding *upper messages* $\overline{\mu}(X_q)$ and $\overline{\Lambda}_q$. Similarly, any equation involving lower messages and probabilities has a corresponding upper formulation [2].

where the function to be minimized is

$$\underline{c}_i^k(\Lambda_i) = \begin{cases} \frac{1+(\Lambda_i-1)\overline{\rho}(x_i|x_k)}{1+(\Lambda_i-1)\underline{\rho}(x_i|x_k)} & \text{if} \quad \Lambda_i \geq 1 \\ \frac{1+(\Lambda_i-1)\underline{\rho}(x_i|x_k)}{1+(\Lambda_i-1)\overline{\rho}(x_i|x_k)} & \text{otherwise} \end{cases}, \tag{10}$$

with

$$\underline{\rho}(x_i|x_k) = \sum_{\substack{l:X_l\in\Pi_i, l\neq k \\ x_l\in\Omega_l}} \underline{P}(x_i|\pi_i) \prod_{l:X_l\in\Pi_i, l\neq k} \mu_i(x_l). \tag{11}$$

Note that $\underline{\rho}(x_i|x_k)$, and hence $\underline{c}_i^k(\Lambda_k)$, are also functions of $\{\mu_i(x_l)\}_{l:X_l\in\Pi_i}$. This dependence is left implicit for the sake of notation.

Overall, Equations (5)–(11) define a distributed algorithm that obeys the same principles of Pearl's belief updating algorithm. The global computation is carried out in discrete steps. At each step, some nodes are sending messages from a certain subset of active nodes, which modifies the set of active nodes for the next step and the procedure is repeated until no node is active. This condition is satisfied when some node has been updated about the global state of the network. In this state, the messages associated to the nodes are the final result of the computation.

Loopy belief propagation is a popular technique for approximate updating that applies Pearl's algorithm to multiply connected BNs: propagation is iterated until probabilities converge to a fixed value or for a given number of iterations. Similarly, multiply connected binary CNs can be updated by a loopy variant of 2U (called L2U) [3]. Initialization of variables and messages follows the same steps used in the 2U algorithm. Then nodes are repeatedly updated, until convergence is observed. The L2U algorithm, whose performances have been tested in [3], seems to be very accurate and mostly returns good results with low errors after a few iterations.

Finally, let us analyze the computational complexity of 2U (while L2U has clearly the complexity of 2U multiplied by the number of iterations). The computation of the messages in X_i is dominated by Equation (6) and Equation (9). Let $p_i := |\Pi_i|$ denote the *indegree* for the node X_i. These equations require an optimization over 2^{p_i} different configurations of the messages $\mu_i(x_j)$ (for each parent X_j, we can choose the upper or lower $\mu_i(x_j)$). Moreover, the functions to be minimized are sums with 2^{p_i} terms, and for each term p_i multiplications must be performed (we need to multiply the messages of all the parents). Finally, an additional factor p_i arises in Equation (9), because it is computed for each k. Overall, this means a time complexity

$O(p_i^2 2^{2p_i})$ locally to X_i ($p_i 2^{p_i}$ is the time to compute the function for a fixed configuration of the parent messages, 2^{p_i} is the number of combinations of parent messages and p_i is the number of times Equation (9) needs to be evaluated). However, we have noted that instead of computing Equation (9) separately for each k, we can reuse the computations that are performed in Equation (11) for distinct k's, that is, evaluations of Equation (11) altogether (for all k) can be computed in time $O(p_i 2^{p_i})$ given that a configuration of the messages is fixed (just note that from a x_k to another, we can reuse the computations and spend just constant time per term of the summation instead of $O(p_i)$ time). Hence, the final complexity of our 2U implementation is $O(p_i 2^{p_i})$ times the number of configurations of the messages, that is, $O(p_i 2^{2p_i})$.[6]

4. Transformations of Credal Networks

In this section we review two different transformations of CNs that have been recently proposed for independent reasons. Their sequential application is the basis to obtain an equivalent representation of CNs based on binary variables.

4.1. Binarization Algorithm

By definition, L2U (see Section 3) cannot be applied to a non-binary CN like the one in the example of Figure 1. To overcome this limitation, a *binarization* that transforms a CN into a binary CN has been proposed in [5].

First, each variable is equivalently represented by a cluster of binary variables. Assume d_i, which is the number of states for X_i, to be an integer power of two, and let $\tilde{d}_i := \log_2 |\Omega_{X_i}|$.[7] An obvious one-to-one correspondence between the states of X_i and the joint states of a vector of $\tilde{d}_i$ binary variables $\tilde{X}_i := (\tilde{X}_i^0, \tilde{X}_i^1, \ldots, \tilde{X}_i^{\tilde{d}_i - 1})$ is established if the joint state $(\tilde{x}_i^0, \ldots, \tilde{x}_i^{\tilde{d}_i - 1}) \in \{0,1\}^{\tilde{d}_i}$ is associated with $x_{il} \in \Omega_{X_i}$, where l is the integer whose $\tilde{d}_i$-bit

[6] This is a fast implementation as the belief propagation in standard BNs already takes time $O(p_i 2^{p_i})$ to evaluate the functions.

[7] This is not a limitation as a number of *dummy* states up to the nearest power of two can be always added. Accordingly, from now on we assume for all the variables a number of possible values equal to an integer power of two.

representation is $\tilde{x}_i^{\tilde{d}_i-1}\cdots\tilde{x}_i^1\tilde{x}_i^0$. Elements of $\tilde{X}_i$ are said *bits* of X_i and their position in the vector their *order*.

Overall, $\tilde{\mathbf{X}}$ denotes the vector of bits obtained *binarizing* all the elements of $\mathbf{X}$. We write $P(\mathbf{X}) = \tilde{P}(\tilde{\mathbf{X}})$, if $P(\mathbf{x}) = \tilde{P}(\tilde{\mathbf{x}})$ for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, where $\tilde{\mathbf{x}} \in \Omega_{\tilde{\mathbf{X}}}$ is the state corresponding to $\mathbf{x}$.

A DAG $\tilde{\mathcal{G}}$ associated to the variables $\tilde{\mathbf{X}}$ can be obtained from $\mathcal{G}$ as follows: (i) two nodes of $\tilde{\mathcal{G}}$ corresponding to bits of different variables in $\mathbf{X}$ are connected by an arc if and only if there is an arc with the same direction between the related variables in $\mathbf{X}$; (ii) an arc connects two nodes of $\tilde{\mathcal{G}}$ corresponding to bits of the same variable of $\mathbf{X}$ if and only if the order of the bit associated to the node from which the arc departs is lower than the order of the bit associated to the remaining node. An example of this transformation is depicted in Figure 2.

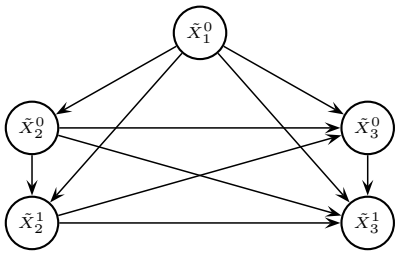

Figure 2: The binarization of the CN in Figure 1.

Finally, regarding the quantification of the conditional credal sets, we have:

$$\underline{\tilde{P}}(\tilde{x}_i^j|\tilde{\pi}_i^j) := \min_{k=1,\ldots,n_v} \tilde{P}_k(\tilde{x}_i^j|\tilde{\pi}_i^j), \tag{12}$$

where the index k is defined like in Equation (4). Denoting by $\tilde{\Pi}_i$ the parents of $\tilde{X}_i^j$ corresponding to the binarization of Π_i, i.e., those that are not in the same cluster of $\tilde{X}_i^j$, the probabilities to be minimized on the right-hand side are:

$$\tilde{P}_k(\tilde{x}_i^j|\tilde{x}_i^{j-1},\ldots,\tilde{x}_i^0,\tilde{\pi}_i) \propto \sum\nolimits_l^* P_k(x_{il}|\pi_i), \tag{13}$$

where the sum $\sum^*$ is restricted to the states $x_{il} \in \Omega_{X_i}$ such that $l \bmod 2^{j+1}$ is the integer whose $(j+1)$-bit representation is $\tilde{x}_i^j,\ldots,\tilde{x}_i^1,\tilde{x}_i^0$, π_i is the joint state of the parents of X_i corresponding to the joint state $\tilde{\pi}_i$ for the bits of the parents of X_i, symbol $\propto$ denotes proportionality, and the relations are considered for each $i = 1,\ldots,n$, $j = 0,\ldots,\tilde{d}_i - 1$, and $\pi_i \in \Omega_{\Pi_i}$. E.g., from

Equation (13) we have $P(\tilde{X}_2^0 = 0|\tilde{x}_1^0) \propto [P(x_{20}|x_1) + P(x_{22}|x_1)]$ and $P(\tilde{X}_2^1 = 1|\tilde{X}_2^0 = 0, \tilde{x}_1^0) \propto P(x_{22}|x_1)$ for the CN in Figure 2. If both the states of $\tilde{X}_i^j$ produce zero in Equation (13), the corresponding conditional mass functions can be arbitrarily specified (we set a degenerate mass function). Note that minimization in Equation (12) can be obtained by simply considering the vertices of $K(X_i|\pi_i)$ in Equation (13).

The overall procedure returns a well-defined CN, which is called the *binarization* of the original CN. Given an updating problem on a CN as in Equation (4), we can consider the corresponding problem on its binarization. E.g., the computation of $\underline{P}(x_{33}|x_{10})$ for the CN in Figure 1 corresponds to $\underline{P}(\tilde{X}_3^0 = 1, \tilde{X}_3^1 = 1|\tilde{X}_1^0 = 0)$. According to Theorem 2 in [5] this is an *outer approximation* (i.e., the posterior interval includes that of the original updating problem), which can be approximately estimated by L2U.

This approach entails a twofold approximation: (i) the approximation introduced by the binarization and (ii) that due to the loopy propagation. Approximation (i) can be regarded as originated by replacing each credal set of the original network with an enclosing polytope with a fixed number of vertices.[8] The latter number cannot be controlled and could be too low to lead to a satisfactory approximation of the original credal set, which in turns leads approximation (i) to be quite crude. In the next section, we recall an independently developed transformation that will be used to remove approximation (i).

4.2. Decision-theoretic specification

In [4], a general graphical language for CNs based on the so-called *decision-theoretic specification* (DTS) has been proposed. A DTS of a CN is obtained augmenting the original CN by a number of *control nodes*, used to enumerate the vertices of the conditional credal sets. That turns the original nodes into precise-probability ones, while the control nodes can be formulated as standard chance nodes with vacuous credal sets.

Let us briefly describe this transformation in the case of a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$. First, we obtain from $\mathcal{G}$ a second DAG $\mathcal{G}'$ defined over a wider domain $\mathbf{X}' := (X_1, \ldots, X_{2n})$. This is done by iterating, for each $i = 1, \ldots, n$, the following operations: (i) add a node X_{i+n}; (ii) draw an arc from each parent of X_i to X_{i+n}; (iii) delete the arcs connecting the parents of X_i with X_i; (iv) draw an

[8] Remember that a credal set over a binary variable cannot have more than two vertices.

arc from X_{i+n} to X_i. An example of this transformation is shown in Figure 3.

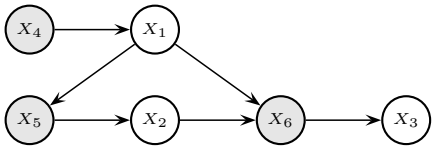

Figure 3: The output of the transformation described in Section 4.2 for the CN in Figure 1. The nodes added to the original network are in gray.

Note that, for each $i = 1, \ldots, n$, $\Pi'_{i+n} = \Pi_i$, i.e., the parents of X_{i+n} in $\mathcal{G}'$ are the parents of X_i in $\mathcal{G}$, and also $\Pi'_i = X_{i+n}$, i.e., X_{i+n} is the only parent of X_i in $\mathcal{G}'$ and is therefore called the *control variable* of X_i.

We assume a one-to-one correspondence between the possible states of a control variable X_{i+n} and the collection of all the (distinct) extreme mass functions of *all* the conditional credal sets specified over X_i, i.e., $\Omega_{X_{i+n}} := \bigcup_{\pi_i \in \Omega_{\Pi_i}} \text{ext}[K(X_i|\pi_i)]$, for each $i = 1, \ldots, n$. As an example, assuming the number of vertices for the credal sets of the CN in Figure 1 equal to the number of possible states of the relative variables, we have that X_4 in Figure 3 is a binary variable, whose states correspond to the two vertices of $K(X_1)$; X_5 has eight possible states corresponding to the four vertices of $K(X_2|x_1)$ and the four of $K(X_2|\neg x_1)$; X_6 has 32 possible states corresponding to the vertices, four per each set, of the conditional credal sets over X_3.

Finally, in order to obtain a well-defined CN over $\mathbf{X}'$ associated to $\mathcal{G}'$, we quantify the conditional credal sets as follows. For each $i = 1, \ldots, n$, we set $K'(X_i|x_{i+n}) := P(X_i)_{x_{i+n}}$, where $P(X_i)_{x_{i+n}}$ is the element of $\text{ext}[K(X_i|\pi_i)]$ corresponding to x_{i+n}. Regarding the control nodes $\{X_{i+n}\}_{i=1}^n$, we set instead $K'(X_{i+n}|\pi'_{i+n}) := K_{\Omega^{\pi_i}_{X_{i+n}}}(X_i)$, where $\Omega^{\pi_i}_{X_{i+n}} := \text{ext}[K(X_i|\pi_i)] \subseteq \Omega_{X_{i+n}}$.

The CN returned by this transformation will be denoted as $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$, and its strong extension as $K'(\mathbf{X}')$. Remarkably, $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$ provides an equivalent representation of $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ being that $K'(\mathbf{X}) = K(\mathbf{X})$ as stated by Theorem 2 in [4], where $K'(\mathbf{X})$ is the marginalization of $K'(\mathbf{X}')$ to $\mathbf{X}$. Note also that, if some nodes in the original CN are quantified by precise probabilities, the above results can be derived without introducing the control nodes for these nodes.

Finally let us to remark that, by construction, the quantification of the

conditional credal sets for the nodes of $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$ is either precise (for the nodes of the original CN) or vacuous (for the control nodes). This property will be exploited in the next section in order to obtain a binary equivalent representation of the CN.

5. Exact Binarization & GL2U

Now we present the original contributions of this paper, consisting of a general representation result (Section 5.1), the definition of the GL2U algorithm (Section 5.2), the study of its computational complexity (Section 6), and its empirical evaluation (Section 7).

5.1. Exact binarization

Consider the sequential application of the transformations detailed in Section 4.2 and Section 4.1. Thus, given a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$, obtain $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$ by a DTS, and hence $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$ through binarization. The latter CN is said the *exact binarization* of the first, a terminology justified by the following result.

Theorem 1. *Consider a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ and its exact binarization $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$. Let $K(\mathbf{X})$ and $\tilde{K}'(\tilde{\mathbf{X}}')$ be their corresponding strong extensions. Then:*

$$K(\mathbf{X}) = \tilde{K}'(\tilde{\mathbf{X}}), \tag{14}$$

with $\tilde{K}'(\tilde{\mathbf{X}})$ marginalization of $\tilde{K}'(\tilde{\mathbf{X}}')$ to $\tilde{\mathbf{X}}$.

According to Equation (14), $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$ is an equivalent binary representation of $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$. It should be pointed out that, even if we focus on the case of CNs with separately specified credal sets, Theorem 1 holds also for so-called *non-separately specified* CNs, for which a DTS can be provided as well. Similarly, the algorithm presented in the next section can be applied to any CN, separately or non-separately specified.

5.2. GL2U

Theorem 1 is a basis for the solution of general inference problems, as stated by the following straightforward corollary.

Corollary 1. *Any inference problem on a CN can be equivalently computed in its exact binarization.*

According to Corollary 1, we can consider a so-called *generalized* L2U algorithm (GL2U), where given an updating problem on a CN as in Equation (4), we solve by L2U the corresponding updating problem on the exact binarization of the original CN. The overall procedure is still approximate, but differently from the case without DTS considered in [5], the only source of approximation is the loopy component.

It is important to observe that, if the queried node X_q in the original CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ is not binary, the corresponding updating problem in the exact binarization $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$ requires the multiple query of the binary nodes in the cluster $(\tilde{X}_q^0, \tilde{X}_q^1, \ldots, \tilde{X}_q^{\tilde{d}_q - 1})$. This task cannot be directly performed by L2U, which is designed for querying single nodes only.

In [5], this task has been solved by augmenting the binarized network with a binary *dummy child*, which is in the state *true* if and only if its parents, i.e., the elements of the cluster corresponding to X_q, are in the joint state corresponding to x_q. This technique makes L2U less accurate because of the problem of *convergence error* discussed at the end of this section.

A slightly different binarization can be considered for X_q. The node is transformed into a cluster of d_q (instead of $\tilde{d}_q = \log_2 d_q$) binary variables, say $(\tilde{Y}_q^0, \tilde{Y}_q^1, \ldots, \tilde{Y}_q^{d_q - 1})$, such that each binary variable corresponds to a different state of X_q. The topology of the cluster is completely connected as for the standard binarization (see Section 4.1 or the example in Figure 4). For the quantification of the conditional probabilities for the binary variables, we assume the following constraint:

$$P'(\tilde{Y}_q^i = 1) = P(X_q = x_{qi}), \tag{15}$$

for each $i = 0, 1, \ldots, d_i - 1$, where $P'(\tilde{Y}_q^i)$ is obtained taking the product of all the conditional mass functions associated to the nodes of the cluster and then marginalizing over $\tilde{Y}_q^i$. Accordingly, the query $X_q = x_{qi}$ in the original CN corresponds to query $\tilde{Y}_q^i = 1$ in the exact binarization, a task that can be achieved by L2U.

Finally, let us explain why this approach tends to make the inferences for the first two states of X_q, i.e., those on $\tilde{Y}_q^0$ and $\tilde{Y}_q^1$ more accurate. In fact, according to Equation (6), 2U assumes mutual independence between the parents of X_i. In general, this assumption is violated by L2U on multiply connected binary CNs. This originates a *convergence error* [9] during the loopy propagation. Consider this error for a queried variable X_q with four possible states, whose special binarization is in Figure (4). L2U produces a

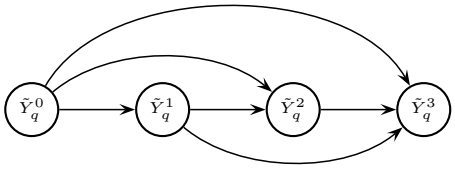

Figure 4: The binarization of a queried variable X_q with four possible states.

convergence error in the computation of the messages associated to $\tilde{Y}_q^2$ by assuming the independence of $\tilde{Y}_q^0$ and $\tilde{Y}_q^1$, and the situation is even worst for $\tilde{Y}_q^3$, while this error is not present in the computation of the messages for the nodes associated to the first two states of X_q.

Accordingly, we expect GL2U to be more accurate in computing the inferences for the first two states of the queried variable. Clearly, if other states of X_q are queried, a simple permutation of the elements of Ω_q is required before making the inferences. A numerical analysis of this feature is reported in Table 3.

6. Complexity Issues

According to the discussion in the previous section, the computational time required by GL2U to update a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ is basically that required by L2U to update $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$. Let us initially consider a single iteration of the algorithm. As noted in Section 3, locally to a node X, the complexity is $O(p \cdot 2^{2p})$, where p is the indegree of X. It can be checked that $\tilde{X}_i^{\tilde{d}_i - 1}$ has the maximum indegree among the $\tilde{d}_i$ binary nodes in the cluster $\tilde{X}_i$; similarly, $\tilde{X}_{i+n}^{\tilde{d}_{i+n}-1}$ has the maximum indegree among the $\tilde{d}_{i+n}$ nodes of $\tilde{X}_{i+n}$. Note also that the number of nodes in $\tilde{\Pi}_i$ is $\sum_{j:X_j \in \Pi_i} \tilde{d}_j$. Therefore, the indegrees of $\tilde{X}_i^{\tilde{d}_i - 1}$ and $\tilde{X}_{i+n}^{\tilde{d}_{i+n}-1}$ are respectively $\tilde{d}_i + \tilde{d}_{i+n} - 1$ and $\tilde{d}_{i+n} + \sum_{j:X_j \in \Pi_i} \tilde{d}_j - 1$. Thus, considering that by definition $2^{\tilde{d}_i} = d_i$, we conclude that, locally to the node X_i, the complexity of the algorithm can be written as:

$$O\left((d_{i+n} \cdot d_i)^2 \cdot \log_2(d_{i+n} \cdot d_i) + (d_{i+n} \cdot \prod_j d_j)^2 \cdot \log_2(d_{i+n} \cdot \prod_j d_j) \right), \quad (16)$$

where the product over j is intended over all the parents of X_i. Note that d_i and $\prod_j d_j$ are respectively the number of rows and columns associated to the conditional probability table $P(X_i|\Pi_i)$, while d_{i+n} is the overall number of vertices associated to the conditional credal sets specified for X_i. Because both $d_{i+n} \cdot d_i$ and $d_{i+n} \cdot \prod_j d_j$ are smaller than $d = d_i \cdot d_{i+n} \cdot \prod_j d_j$, the complexity of GL2U locally to a node is $O(d^2 \log_2 d)$.

Note also that any iteration of 2U is linear in the size of the network (if we assume that d does not grow as fast as the number of nodes, which is the most natural behavior), and the size of the exact binarization grows of a factor at most equal to $2 \cdot \max_{i=1}^{2n} \tilde{d}_i$ with respect to the original network. The factor depends (i) on the decision-theoretic transformation that doubles the number of nodes, and on (ii) the binarization that makes of each node $X_i \in \mathbf{X}'$ a cluster of binary nodes $\tilde{X}_i$ whose size depends on the logarithm $\tilde{d}_i$ of its number of states d_i. We can approximate the global complexity by assuming that the local sizes of the network (that is, the maximum number of states of each variable, the maximum number of vertices of the credal sets associated to these variables, and the maximum number of parents) are not so large (which is in fact a normal situation), and we conclude that any iteration of GL2U is roughly linear in the size of the network.

7. Numerical Tests

In order to test the performance of GL2U, we employ a benchmark made of different CNs with random topology, either multiply and singly connected, and two classical (multiply connected) models, namely the *Alarm* and the *Insurance* networks [10, 11]. The maximum indegree for the networks with random topology is limited to 5. The number of states for the Alarm and the Insurance networks is the same as in their original specifications, while for the other networks the number of states is randomly chosen between 2 and 8. All the models are quantified by randomly generated conditional credal sets with a fixed number of vertices, whose number is ranging from 2 to 8 for each network.

We compute both unconditional and conditional inferences on these networks by a Python/C++ implementation of GL2U.[9] The resulting inferences are compared with the approximate *iterated local search* method [12] (ILS)

[9]This software is freely available at `www.idsia.ch/∼sun/gl2u-ii.htm`.

Network features			# of	GL2U		ILS	
topology	nodes	vertices	queries	MSE	time	MSE	time
Multi	6	2	167	0.114	0.11	**0.016**	0.11
Multi	6	4	149	0.169	0.17	**0.036**	0.15
Multi	10	2	402	0.118	0.14	**0.047**	1.02
Multi	10	4	205	0.183	0.22	**0.052**	0.56
Multi	20	2	526	**0.123**	0.27	0.164	5.93
Multi	20	4	399	**0.194**	0.36	0.243	12.62
Polytree	10	2	515	0.106	0.11	**0.017**	0.18
Polytree	10	4	449	0.181	0.11	**0.037**	0.16
Polytree	10	8	422	0.257	0.32	**0.048**	0.19
Polytree	20	2	316	**0.117**	0.07	0.159	0.16
Polytree	30	2	813	**0.112**	0.17	0.176	0.23
Polytree	30	4	693	**0.178**	0.31	0.260	0.41
Alarm	37	2	432	**0.100**	0.07	0.223	0.30
Alarm	37	4	360	**0.139**	0.09	0.328	0.47
Insurance	27	2	200	**0.143**	0.09	0.197	42.23
Insurance	27	4	199	**0.238**	0.20	0.366	65.91

Table 1: Random unconditional queries.

and the exact method presented in [13]. Tables 1 and 2 report these comparisons. The number of inferences over each type of network is shown in the fourth column. Note that such a number varies because we only compare inferences where the exact solution can be computed. The *mean square-error* (MSE) is evaluated with respect to the exact solution and the average time of inferences (in seconds) is also reported.

The results for the unconditional inferences are in Table 1. Remarkably, for networks with more than 10 nodes, the inferences of GL2U are always more accurate, while ILS is more accurate only with small networks. In fact, it seems that the accuracy of GL2U is not so related to the network size as that of ILS, and this makes GL2U more accurate for larger models.

A similar behavior is observed also for conditional queries (see Table 2). In these cases, an evidence consisting in the observation of three leaf nodes is randomly generated (for networks with 6 nodes, just one node is observed).

From our tests we also note that the approximated lower (and similarly upper) probability returned by GL2U is equally likely to be greater or smaller than the exact value.

Network features			# of	GL2U			ILS	
topology	nodes	vertices	queries	MSE	time	iter	MSE	time
Polytree	10	2	34	0.107	0.09	19.6	**0.024**	0.19
Polytree	20	2	51	**0.112**	0.11	26.1	0.218	0.14
Multi	6	2	50	0.173	0.73	26.7	**0.024**	0.14
Multi	6	4	32	0.170	0.09	12.7	**0.050**	0.13
Multi	20	2	51	**0.144**	0.26	28.9	0.191	4.07
Multi	20	4	36	**0.204**	0.79	5.7	0.271	3.40

Table 2: Random queries with evidence. The seventh column report the average number of iterations before L2U converge.

We should also remark that, in some cases, GL2U returns a vacuous posterior interval, even if the exact inference is completely different. This situation has occurred only for conditional queries, in 13% of the cases. A similar behavior is also reported to happen in loopy belief propagation for BNs in computing conditional queries [9], but there the number of loops is usually smaller, and such situation is less often observed. A further analysis of the performances of L2U in order to profile these instances should be therefore considered as a future work. Clearly, algorithms which are not based on loopy techniques like those in [12, 14] are not suffering problems of this kind.

As a further remark, let us recall that, according to the discussion reported at the end of Section 5.2, we have always permuted the states of the queried node in order to rank the queried state in one of the first two positions. This makes the convergence error smaller; in fact in the tests with non-binary queried variables we have inferences for the first two states 18.1% more accurate than for the other states in the unconditional cases, while the gain is 39.4% in the conditional case. Table 3 reports some detail about this comparison.

Moreover, the running time and the amount of allocated memory for ILS rapidly increases with the size of the net, that makes unfeasible a solution for large nets, which can be instead quickly updated by GL2U (see Figure 5).

As far as we know other existing algorithms besides ILS are at least exponential in the treewidth of the moralized graph and suffer from the same complexity issues. In fact, comparisons have been done also with the *hill climbing* (HC) algorithm in [14]. Compared to ILS, HC is slower and run out of memory with smaller network (see again Figure 5).

Network features			MSE	
topology	nodes	vertices	first two states	other states
Multi	6	2	**0.114**	0.159
Multi	6	4	**0.169**	0.239
Multi	10	2	**0.118**	0.142
Multi	10	4	**0.183**	0.210
Multi	20	2	**0.123**	0.137
Multi	20	4	**0.194**	0.197
Polytree	10	2	**0.106**	0.155
Polytree	10	4	**0.181**	0.211
Polytree	10	8	0.257	**0.254**
Polytree	20	2	**0.117**	0.168
Polytree	30	2	**0.112**	0.150
Polytree	30	4	**0.178**	0.220

Table 3: Accuracy of GL2U for inference on the first two states and on the other states of the queried variable.

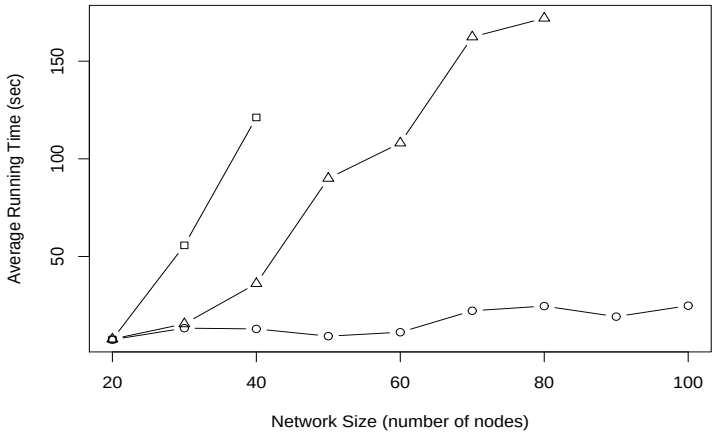

Figure 5: Average running time versus network size for HC (squares), ILS (triangles) and GL2U (circles). The HC and ILS sequences are shorter because these algorithms run out of memory for large CNs (8GB were used).

8. Conclusions

This paper has proposed a new approximate algorithm for CN updating. This task is achieved augmenting the network by a number of nodes, enumerating the vertices of the credal sets, then transforming the CN in a corresponding network over binary variables, and updating such binary CN by the loopy version of 2U. The procedure applies to any CN, without restrictions related to the topology or the number of possible states, and the only approximation is due to the loopy propagation. Empirical analysis shows that GL2U is a competitive procedure for approximate inference in CNs both in terms of accuracy and scalability. The algorithm is purely distributed and allows for simultaneous updating of all the variables in the net: these characteristics are usually not shared by optimization algorithms not based on propagation. Moreover, the computational complexity of GL2U makes it possible to solve large nets, which cannot be updated by other algorithms, mainly because of excessive memory consumptions.

Acknowledgments

Work partially supported by the Swiss NSF grants 200021-113820/1 and 200020-116674/1, and Hasler Foundation grant 2233. Third author thanks also the project "Ticino in Rete".

A. Proofs

Lemma 1. *Consider a CN with a single node X and vacuous $K(X) := K_{\Omega_X^*}(X)$, where $\Omega_X^* \subseteq \Omega_X$. Let $\tilde{K}(\tilde{X})$ denote the strong extension of its binarization (as in Section 4.1). Then:*

$$\tilde{K}(\tilde{X}) = K(X). \tag{17}$$

Proof 1 (Proof of Lemma 1). *Consider a generic $\tilde{P}_*(\tilde{X}) \in \text{ext}[\tilde{K}(\tilde{X})]$, where $\tilde{X} := (\tilde{X}^0, \ldots, \tilde{X}^{\tilde{d}-1})$ with $\tilde{d}{:=}\log_2 |\Omega_X|$. A corresponding mass function $P_*(X) := \tilde{P}_*(\tilde{X})$ can be therefore defined. Thus:*

$$\tilde{P}_*(\tilde{x}) = \prod_{j=0}^{\tilde{d}-1} \tilde{P}_*(\tilde{x}^j | \tilde{x}^{j-1}, \ldots, \tilde{x}^0), \tag{18}$$

for each $\tilde{x} \in \Omega_{\tilde{X}}$ such that $(\tilde{x}^0, \ldots, \tilde{x}^{\tilde{d}-1}) = \tilde{x}$. For each $j = 0, \ldots, \tilde{d} - 1$ and each possible value of their parents, the conditional mass functions

$\tilde{P}_*(\tilde{X}^j|\tilde{x}^{j-1},\ldots,\tilde{x}^0)$ *are vertices of their corresponding conditional credal sets because of Proposition 1 of [4].*

Thus, the values of the conditional probabilities on the right-hand side of Equation (18) are obtained by a minimization as in Equation (12). The values to be minimized are obtained from Equation (13), where the conditional probabilities on the right-hand side are the vertices of $K(X)$*, i.e., the* $m := |\Omega_X^*|$ *degenerate extreme mass functions of the vacuous credal set* $K_{\Omega_X^*}(X)$*. This means that there is only a non-zero term in the sum in Equation (13) and therefore each vertex of* $K_{\Omega_X^*}$ *produces a degenerate conditional mass function for the corresponding binary variable. Consequently, also the extreme values returned by Equation (12) will be degenerate.*

We can therefore conclude that, according to Equation (18), also $\tilde{P}_*(\tilde{X})$ *and hence* $P_*(X)$ *are degenerate mass functions. Let* $x_* \in \Omega_X$ *be the state of* X *such that* $P_*(x_*) = 1$*. Considering Equation (18) for* $\tilde{x}_* \in \Omega_{\tilde{X}}$*, we conclude that all the conditional probabilities on the right-hand side are equal to one. Considering the highest order bit, according to Equation (13) and denoting by* $P_k(X)$ *a vertex of* $\Omega_*(X)$*, we have* $\tilde{P}_*(\tilde{x}_*^{\tilde{d}-1}|\tilde{x}_*^{\tilde{d}-2},\ldots,\tilde{x}_*^0) = P_k(x_*) = 1$*, that requires* $x_* \in \Omega_X^*$*. Thus,* $P_*(X) \in \mathrm{ext}[K(X)]$*, that implies* $\mathrm{ext}[\tilde{K}(\tilde{X})] \subseteq \mathrm{ext}[K(X)]$*, and finally* $\tilde{K}(\tilde{X}) \subseteq K(X)$*. On the other side,* $\tilde{K}(\tilde{X}) \supseteq K(X)$ *because of Theorem 2 in [5], and hence the thesis.*

Proof 2 (Proof of Theorem 1). *Given a* $\tilde{P}'_*(\tilde{\mathbf{X}}') \in \mathrm{ext}[\tilde{K}'(\tilde{\mathbf{X}}')]$*, the following factorization holds:*

$$\tilde{P}'_*(\tilde{\mathbf{x}}') = \prod_{i=1}^{2n}\prod_{j=0}^{\tilde{d}_i-1}\tilde{P}'_*(\tilde{x}_i^j|\tilde{\pi}_i^j) = \prod_{i=1}^{2n}\tilde{P}'_*(\tilde{x}_i^0,\ldots,\tilde{x}_i^{\tilde{d}_i-1}|\tilde{\pi}'_i), \tag{19}$$

for each $\tilde{\mathbf{x}}' \in \Omega_{\tilde{\mathbf{X}}'}$*, where the values of the other variables are consistent with* $\tilde{\mathbf{x}}$*, and the last equality follows from chain rule. Equation (19) defines* $P'_*(X_i|\pi'_i) := \tilde{P}'_*(\tilde{X}_i^0,\ldots,\tilde{X}_i^{\tilde{d}_i-1}|\tilde{\pi}'_i)$*.*

As noted in Section (4.2), for each $i = 1,\ldots,n$ *and* $\pi_i \in \Omega_{\Pi_i}$*,* $K'(X_i|\pi'_i)$ *is a credal set made of a single point. Thus, as a corollary of Theorem 1 in [5], we have* $P'_*(X_i|\pi'_i) \in \mathrm{ext}[K'(X_i|\pi'_i)]$*, being in fact the only element of this credal set. Similarly, for each* $i = 1,\ldots,n$*, the credal set* $K'(X_{i+n}|\pi'_{i+n})$ *is vacuous.*

Let us regard this credal set as a CN made of a single node. We can invoke Lemma 1 and obtain from $\tilde{P}'_*(\tilde{X}_{i+n}|\tilde{\pi}'_{i+n}) \in \mathrm{ext}[\tilde{K}'(\tilde{X}_{i+n}|\tilde{\pi}'_{i+n})]$ *that* $P'_*(X_{i+n}|\pi'_{i+n}) \in \mathrm{ext}[K'(X_{i+n}|\pi'_{i+n})]$*. Overall, we proved that* $P'_*(\mathbf{X}')$ *is a combination of local vertices of the credal sets of* $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}')\rangle$*.*

Thus, $P'_*(\mathbf{X}') \in \text{ext}[K'(\mathbf{X}')]$*, from which* $\text{ext}[\tilde{K}'(\tilde{\mathbf{X}}')] \subseteq \text{ext}[K'(\mathbf{X}')]$*, and finally* $\tilde{K}'(\tilde{\mathbf{X}}') \subseteq K'(\mathbf{X}')$*. According to Lemma 1 in [5],* $\tilde{K}'(\tilde{\mathbf{X}}') \supseteq K'(\mathbf{X}')$*. Thus,* $\tilde{K}'(\tilde{\mathbf{X}}') = K'(\mathbf{X}')$*. Marginalizing on both the sides we get* $\tilde{K}'(\tilde{\mathbf{X}}) = K'(\mathbf{X})$*. But Theorem 2 in [4] states* $K(\mathbf{X}) = K'(\mathbf{X})$*, from which the thesis.*

References

[1] C. P. de Campos, F. G. Cozman, The inferential complexity of Bayesian and credal networks, in: Proceedings of the International Joint Conference on Artificial Intelligence, Edinburgh, 2005, pp. 1313–1318.

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

[3] J. S. Ide, F. G. Cozman, IPE and L2U: Approximate algorithms for credal networks, in: Proceedings of the Second Starting AI Researcher Symposium, IOS Press, Amsterdam, 2004, pp. 118–127.

[4] A. Antonucci, M. Zaffalon, Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks., International Journal of Approximate Reasoning 49 (2) (2008) 345–361.

[5] A. Antonucci, M. Zaffalon, J. S. Ide, F. G. Cozman, Binarization algorithms for approximate updating in credal nets, in: L. Penserini, P. Peppas, A. Perini (Eds.), Proceedings of the third European Starting AI Researcher Symposium, IOS Press, Amsterdam, 2006, pp. 120–131.

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

[7] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, New York, 1991.

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

[9] J. Bolt, Bayesian networks: aspects of approximate inference, Ph.D. thesis, Utrecht University, Department of Information and Computer Science (2008).

[10] I. Beinlich, H. J. Suermondt, R. M. Chavez, G. F. Cooper, The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in: II European Conference on Artificial Intelligence in Medicine, Springer-Verlag, Berlin, 1989, pp. 247–256.

[11] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29(2-3) (1997) 213–244.

[12] J. C. da Rocha, F. G. Cozman, C. P. de Campos, Inference in polytrees with sets of probabilities, in: Conference on Uncertainty in Artificial Intelligence, Acapulco, 2003, pp. 217–224.

[13] C. P. de Campos, F. G. Cozman, Inference in credal networks through integer programming, in: Proceedings of the Fifth International Symposium on Imprecise Probability: Theories and Applications, Action M Agency, Prague, 2007, pp. 145–154.

[14] A. Cano, M. Gómez, S. Moral, J. Abellán, Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks, International Journal of Approximate Reasoning 44 (3) (2007) 261–280.