5th International Symposium on Imprecise Probability: Theories and Applications, Prague, Czech Republic, 2007

Climbing the Hills of Compiled Credal Networks

Rolf Haenni

Bern University of Applied Sciences, Switzerland
rolf.haenni@bfh.ch

University of Bern, Switzerland
haenni@iam.unibe.ch

Abstract

This paper introduces a new approximate inference algorithm for credal networks. The algorithm consists of two major steps. It starts by representing the credal network as a compiled logical theory. The resulting graphical structure is the basis on which the subsequent steepest-ascent hill-climbing algorithm operates. The output of the algorithm is an inner approximation of the exact lower and upper posterior probabilities.

Keywords. Credal Networks, Bayesian Networks, Credal Sets, Approximate Inference, Logical Compilation, Hill-Climbing, Local Search.

1 Introduction

Credal networks are like discrete Bayesian networks, except that they specify closed convex sets of probability mass functions, so-called *credal sets* [36], instead of single probability mass functions. They are usually *locally* (or *separately*) specified [1, 16], i.e. every network variable is associated with a collection of local conditional credal sets, which do not interfere with each other. It is possible to view a locally specified credal network as a set of Bayesian networks with the same directed acyclic graph [17].

In general, credal sets contain an infinite number of probability mass functions, but they are normally fully specified by a finite number of extreme points. These are the vertices of a convex polytope in the corresponding multi-dimensional space. In the case of binary variables, the polytopes coincide with intervals, which restricts the maximal number of necessary extreme points to two. In a Bayesian network, each polytope is restricted to a single (extreme) point.

Inference in a locally specified credal network usually means to derive lower and upper posterior probabilities from the *strong extension* [16], i.e. from the the largest joint credal set that satisfies *strong independence* [15]. Except for the particular case of binary variables in polytree-shaped networks [30], this is computationally extremely challenging, much more than classical inference in Bayesian network. The case of general categorical variables is NP–complete for polytree-shaped networks and NP^{PP}–complete for an unbounded induced treewidth, thus making inference in credal networks very inefficient [27].

In comparison with Bayesian networks, the additional computational complexity results from the potentially unbounded number of vertices needed to describe arbitrary credal sets. This can quickly outperform the benefits of applying local computation techniques to graphical models such as Bayesian networks. Local messages propagated through a credal network (resp. through the join tree obtained from a credal network) may thus possess the richness and complexity of the (global) joint credal set [10]. In fact, inference in credal networks is essentially a global multilinear optimization problem on top of the given graphical structure [18].

Facing the inherent computational complexity of credal networks, exact inference methods are only exceptionally suitable. One exception is the abovementioned case of binary variables in polytree-shaped networks, for which a polynomial-time algorithm exists [30]. All other exact methods (e.g. vertex enumeration, global optimization, and transformation algorithms) are only applicable to very small problem instances.

For large networks, approximate inference seems to be the most natural solution. There is a general distinction between *inner* and *outer* approximations, depending on whether the resulting interval is enclosed in the exact solution or vice versa. The quest for such approximate methods is currently one of the major research topics in the imprecise probability community, as the increasing number of corresponding publications in the last couple of years demonstrates, see e.g. [2, 4, 5, 7, 8, 9, 19, 20, 31, 32].

1.1 General Ideas

In this paper, we present a new approximate method for the inference problem in credal networks. The approach results from combining the following two basic techniques:

Logical Compilation. This is an emerging inference technique for Bayesian networks [12, 13, 14, 23, 45]. The general idea is to represent the graphical structure (topology) of the Bayesian network by a propositional theory. Possible local structures within the given CPTs can be exploited to simplify corresponding sentences of the theory [12, 14]. The resulting logical encoding is then *compiled* into an appropriate logical form called d-DNNF [25, 46], which supports all necessary operations to answer arbitrary queries (conditional probabilities) in polynomial time. The computational task is thus divided into an expensive (off-line) compilation phase and a fast (on-line) query-answering phase.

Hill-Climbing. This is a generic combinatorial optimization technique, which is widely used in many AI-related fields and applications [41]. The goal is to maximize (or minimize) a function $f : X \to \mathbb{R}$ through local search, where X is usually a discrete multi-dimensional state space. Local search means to jump from one configuration in the state space to a neighboring one, until a local maximum or possibly the global maximum is reached. An obvious heuristic for the selection of the neighboring configuration is to jump to the configuration with the steepest ascent of the respective value of f (*steepest-ascent hill-climbing*). The basic hill-climbing process is usually iterated with randomly generated starting points (*random-restart hill-climbing*), thus making it an interruptible anytime algorithm.

The idea of compiling a credal network in the same way as compiling a Bayesian network is quite obvious, but to our knowledge, this is still an unexplored approach. Pointing out this possibility is one of the goals of this paper.

Applying hill-climbing or other local search algorithms to approximate inference in credal networks is also quite obvious, as some of the existing approximation algorithms have demonstrated [4, 5, 6, 20]. Most of them are oriented towards the local propagation scheme in corresponding join trees [33, 43], in which each hill-climbing step requires the updating of the affected join tree messages. The hill-climbing procedure itself is guided by the current configuration of so-called *transparent* variables, whose role consists in selecting the actual vertices in the local credal sets.

1.2 Overview and Outline

In our method, we will also exploit the benefits of local computation in join trees, but only to compile the network structure into a d-DNNF during the inward phase [13]. The necessary information for the hill-climbing procedure is then available in a very simple and compact logical structure. For the current selection of vertices, this structure can then be used to efficiently compute or update the resulting posterior probability. Moreover, without much computational overhead, it is possible to determine the currently unselected vertex (i.e. the neighboring configuration) with the steepest ascent (resp. descent), which we can use as a heuristic to improve the performance of the local search.

After all, we get a simple but yet powerful steepest-ascent, random-restart hill-climbing algorithm to approximate inference in credal networks. By running the algorithm twice, once as a maximizing and once as a minimizing procedure, it produces good inner approximations of the exact probability bounds.

With respect to existing hill-climbing techniques for credal networks, our approach appears to be considerably simpler, as no complicated management of a bidirectional *double message system* is required, like e.g. in [6]. The logical representation is also inherently predestined to exploit existing local CPT regularities in the form of *context-specific independence* [3], *logical relationships* (pure or noisy), or *determinism* [12], for which existing methods typically use so-called *probability trees* [6, 9]. Finally, from the possibility of quickly finding the neighboring configuration with the steepest ascent (respectively descent), our method is likely to converge faster towards the exact results.

The rest of the paper is organized as follows. In Section 2, we give a short introduction to the main concepts of Bayesian and credal networks and the terminology used in this paper. Section 3 summarizes the compilation-based approach to inference in Bayesian (and credal) networks. Section 4 introduces hill-climbing and its application to compiled credal networks. This is the main part of the paper. The discussion and outlook in Section 5 concludes the paper.

2 Bayesian and Credal Networks

A *Bayesian network* (BN) is an efficient representation of a joint probability mass function over a set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of variables [38]. We assume throughout this paper that all variables $X \in \mathbf{X}$ are categorical, i.e. their associated sets Ω_X of possible values are *finite*. The network itself consists of a directed acyclic graph (DAG), which represents the

direct influences among the variables, each of them attached to one node, and a set of conditional probability tables (CPT), which quantify the strengths of these influences. The whole BN represents a *joint probability mass function* $p : \Omega_{\mathbf{X}} \to [0, 1]$ over its variables in a compact manner by

$$p(\mathbf{X}) = \prod_{X \in \mathbf{X}} p(X|\mathbf{\Pi}(X)), \tag{1}$$

where $\mathbf{\Pi}(X)$ denotes the parents of node X in the DAG. Figure 1 depicts the BN for the "Dog-Problem" [11], which is often used in the literature for illustrative purposes. It consists of five binary variables F, B, L, D, and H, with corresponding CPTs $p(F)$, $p(B)$, $p(L|F)$, $p(D|F, B)$, and $p(H|D)$.

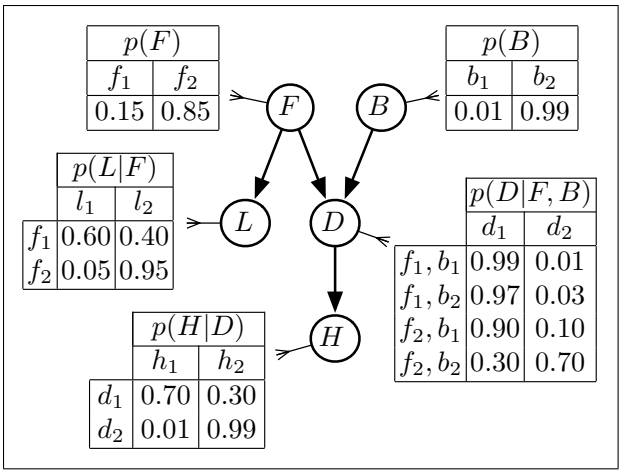

Figure 1: Example of a simple Bayesian network with five binary variables.

Inference in Bayesian networks means to compute the conditional probability $P(H{=}h \,|\, E_1{=}e_1, \ldots, E_r{=}e_r)$, or simply

$$P(h|\mathbf{e}) = \frac{P(h, \mathbf{e})}{P(\mathbf{e})}, \tag{2}$$

of a hypothesis $h \in \Omega_H$ for some observed evidence $\mathbf{e} = (e_1, \ldots, e_r) \in \Omega_{\mathbf{E}}$. We will call $H \in \mathbf{X}$ *query variable* and the elements of $\mathbf{E} = \{E_1, \ldots, E_r\} \subseteq \mathbf{X}$ *evidence variables*. To see how to solve the inference problem, let $\mathbf{Y} = \{Y_1, \ldots, Y_s\} \subseteq \mathbf{X}$ be an arbitrary subset of variables, $\mathbf{y} = (y_1, \ldots, y_s) \in \Omega_{\mathbf{Y}}$ a configuration of values $y_i \in Y_i$, and $\mathbf{Z} = \mathbf{X} \backslash \mathbf{Y}$. Then it is sufficient to compute

$$P(\mathbf{y}) = \sum_{\mathbf{z} \in \Omega_{\mathbf{Z}}} p(\mathbf{yz}) \tag{3}$$

twice, once with $\mathbf{Y} = \{H\} \cup \mathbf{E}$ and $\mathbf{y} = (h, \mathbf{e})$ to get the nominator and once with $\mathbf{Y} = \mathbf{E}$ and $\mathbf{y} = \mathbf{e}$ to get the denominator of the above formula. Note that the necessary sum-of-products involve exponentially many terms, but if the computations are performed *locally* in a join tree propagation or variable elimination process, it is almost always possible to replace it by a compact factorization [28, 33, 43]. Join trees are also useful to avoid redundant computations in the case of multiple queries or updates.

Credal networks (CN) are similar to Bayesian networks, but they relax the uniqueness assumption for the given probability values [16]. In a *locally* (or *separately*) specified CN, the CPT entries are replaced by corresponding conditional credal sets, on which no further restrictions are imposed [1]. A *credal set* for a variable $X \in \mathbf{X}$ is a closed convex set $K(X)$ of probability mass functions $p(X)$ [36]. Similarly, a *conditional credal set* $K(X|\boldsymbol{\pi})$ is a closed convex set of conditional probability mass functions $p(X|\boldsymbol{\pi})$, where $\boldsymbol{\pi} \in \Omega_{\mathbf{\Pi}(X)}$ is one particular assignment of values for the direct influences $\mathbf{\Pi}(X)$ of X. With

$$K(X|\mathbf{\Pi}(X)) = \{K(X|\boldsymbol{\pi}) : \boldsymbol{\pi} \in \Omega_{\mathbf{\Pi}(X)}\} \tag{4}$$

we denote the collection of all such conditional credal sets. This is what a CN needs to specify for all variables $X \in \mathbf{X}$.

Normally, a single conditional credal set $K(X|\boldsymbol{\pi})$ is specified and represented by a finite set

$$\mathrm{Ext}(K(X|\boldsymbol{\pi})) = \{p_1(X|\boldsymbol{\pi}), \ldots, p_m(X|\boldsymbol{\pi})\} \tag{5}$$

of extreme points $p_i(X|\boldsymbol{\pi})$. Geometrically, these extreme points are vertices of a polytope in the corresponding additive subspace of $[0, 1]^{|\Omega_X|}$. In the binary case, i.e. for $|\Omega_X| = 2$, the additive subspace of $[0, 1]^2$ is a simple straight line between $(0, 1)$ and $(1, 0)$, on which credal sets degenerate into intervals with at most two extreme points (the bounds of the intervals).

If we generalize the BN of Fig. 1 to a CN, we need to replace the rows in each CPT by corresponding (conditional) credal sets. Since all involved variables are binary, it is sufficient to specify *two* extreme points for each credal set. As an example, consider $K(H|D)$, which consists of the credal sets $K(H|d_1)$ and $K(H|d_2)$, and suppose that the precise values $p(H|d_i)$ from Fig. 1 are enlarged to sets of extreme points $\mathrm{Ext}(K(H|d_i)) = \{p_1(H|d_i), p_2(H|d_i)\}$ with the following values:

	Ext($K(H\|D)$)			
	$p_1(H\|D)$		$p_2(H\|D)$	
	h_1	h_2	h_1	h_2
d_1	0.70	0.30	0.80	0.20
d_2	0.01	0.99	0.03	0.97

Note that the particularity of binary variables allows us to specify the same information more compactly

by $p(h_1|d_1) \in [0.7, 0.8]$ and $p(h_1|d_2) \in [0.01, 0.03]$ (and therefore by $p(h_2|d_1) \in [0.2, 0.3]$ and $p(h_2|d_2) \in [0.97, 0.99]$), thus making the interval-shaped credal sets more visible. This is an appealing view, in which credal sets appear to be nothing but *probability intervals* or *interval-valued probabilities* [35, 37, 44, 48], but the simplicity of this view belies the fact that credal sets are more general than probability intervals, e.g. for variables with more than two values. Interval representations are also problematical when it comes to apply Bayes' rule or to propagate them through a network [6, 16].

For a given credal network, we use $K(\mathbf{X})$ to denote its *joint credal set*. Note that its actual definition depends on how the concept of independence is adopted for credal sets. In this paper, we follow the usual convention of *strong independence* [15], which allows us to define $K(\mathbf{X})$ to be the *strong extension* of the credal network, i.e. as the largest joint credal set such that every variable $X \in \mathbf{X}$ is strongly independent [16]. This set contains all possible joint probability mass functions, if we select corresponding elements $p(X|\boldsymbol{\pi})$ from each conditional credal set $K(X|\boldsymbol{\pi})$. Formally, we can write

$$K(\mathbf{X}) = \left\{ \prod_{X \in \mathbf{X}} p(X|\mathbf{\Pi}(X)) : p(X|\boldsymbol{\pi}) \in K(X|\boldsymbol{\pi}) \right\},$$

where $\boldsymbol{\pi}$ denotes respective configurations of $\mathbf{\Pi}(X)$. Note that each element $p(\mathbf{X}) \in K(\mathbf{X})$ can be seen as the joint probability mass function of a corresponding Bayesian network (on the same DAG).

The convexity of $K(\mathbf{X})$ guarantees its extreme points to result only from combinations of extreme points of each conditional credal set $K(X|\boldsymbol{\pi})$ [17], and this allows us to rewrite the above expression as

$$\mathrm{Ext}(K(\mathbf{X})) = \mathrm{CH} \left\{ \prod_{X \in \mathbf{X}} p(X|\mathbf{\Pi}(X)) : p(X|\boldsymbol{\pi}) \in \mathrm{Ext}(K(X|\boldsymbol{\pi})) \right\},$$

where CH stands for an algorithm to compute the convex hull of a set of points in a multi-dimensional space [26]. This property reflects the fact that inference in credal networks is reducible to computations of extreme points.

Inference for a given credal set $K(\mathbf{X})$, a query $h \in \Omega_H$, and some observations $\mathbf{e} \in \Omega_{\mathbf{E}}$ means to determine tight bounds over all possible probability values $P(h|\mathbf{e})$, i.e. to compute the *lower* posterior probability

$$\underline{P}(h|\mathbf{e}) = \min\{P(h|\mathbf{e}) : p(\mathbf{X}) \in K(\mathbf{X})\}, \qquad (6)$$

and the *upper* posterior probability

$$\overline{P}(h|\mathbf{e}) = \max\{P(h|\mathbf{e}) : p(\mathbf{X}) \in K(\mathbf{X})\}. \qquad (7)$$

To compute these values under the assumption of strong independence, we can again exploit the convexity of $K(\mathbf{X})$ to restrict the necessary search space to the finite set $\mathrm{Ext}(K(\mathbf{X}))$ of extreme points [17]. Note that if N denotes the total number of involved conditional credal sets, all of them described by k extreme points, then $\mathrm{Ext}(K(\mathbf{X}))$ may possess up to N^k elements, thus making the above minimization/maximization problems very difficult tasks. Except for polytree-shaped networks with binary variables, no algorithm can handle large credal networks exactly [27, 30].

3 Compiling Bayesian Networks

The goal of compiling a Bayesian or credal network is the construction of a logical representation φ, in which all the topological and context-specific information of the network is included in a compact and easily manageable form. This construction is a one-time preparatory step, which is intended to take place off-line. The resulting logical representation φ contains two types of propositional variables, the ones linked to the CPT entries and the ones linked to the individual values of the network variables. The corresponding sets of propositions are denoted by Θ and Δ, respectively.

To compute the probability $P(\mathbf{y})$ of a configuration $\mathbf{y} = (y_1, \ldots, y_s) \in \Omega_{\mathbf{Y}}$ w.r.t. $\mathbf{Y} = \{Y_1, \ldots, Y_s\} \subseteq \mathbf{X}$, which is the basic computational task to answer arbitrary probabilistic queries (see Equation 3 in Section 2), φ is transformed into $\varphi_{\mathbf{y}} = (\varphi|\mathbf{y})^{-\Delta}$ by first *conditioning* φ on $\mathbf{y}$ and then *eliminating* (or *forgetting*) from $\varphi|\mathbf{y}$ all Δ-variables. The remaining Θ-variables in $\varphi_{\mathbf{y}}$ are all of the form $\theta_{x|\boldsymbol{\pi}}$, i.e. each of them is linked to a CPT entry $p(x|\boldsymbol{\pi})$.

To ensure that the above-mentioned computational steps are always efficient, φ must be a so-called d-DNNF [25, 46].[1] A *negation normal form* (NNF) is a rooted, directed acyclic graph, whose leaves are labeled with the literals of a propositional language.[2] All other nodes denote either a logical AND or a logical OR. d-DNNFs are NNFs satisfying two important properties called *determinism* (d) and *decomposability* (D).[3] Fig. 2 depicts the d-DNNF φ_{h_1} for the Bayesian

[1] The suggestion of using d-DNNFs as a target compilation language for Bayesian networks goes back to [23]. The mathematical explanation in [45] backups this choice.

[2] Note that NNFs are *propositional directed acyclic graphs* (PDAG), for which the *simple-negation* property holds [46].

[3] NNFs, in which some propositional variables are implicitly known to be exclusive and exhaustive, should be regarded as corresponding *multi-state directed acyclic graphs* (MDAG), a generalization of PDAGs (and NNFs) to arbitrary categorical variables [47]. In the context of MDAGs, some properties (incl. determinism and decomposability) and some operations (incl. conditioning and variable elimination) are based on more gen-

network in Fig. 1 and the query $\mathbf{y} = h_1$. Note that the network node L has no impact on $P(h_1)$, which is why φ_{h_1} is not affected by variables of the form $\theta_{l_i|f_j}$ (they disappear while l_1 and l_2 are eliminated from $\varphi|h_1$). Similarly, φ_{h_1} does not contain variables of the form $\theta_{h_2|d_i}$ (they disappear while φ is conditioned on h_1).

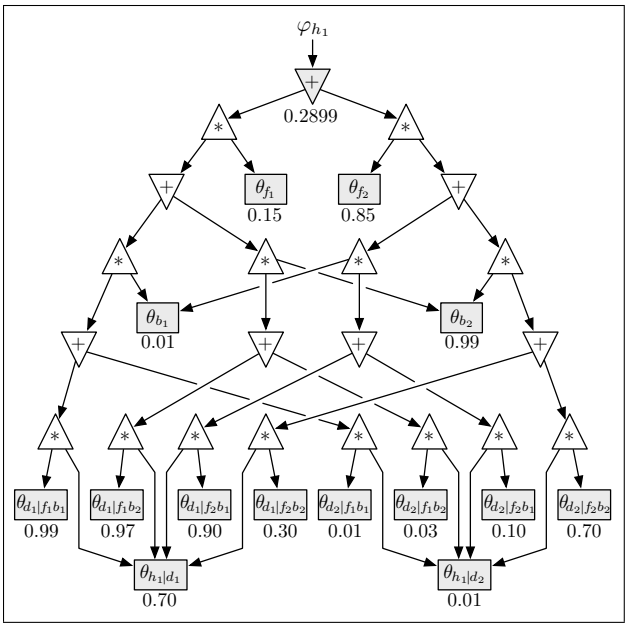

Figure 2: The d-DNNF obtained for the Bayesian network in Fig. 1 and the query $\mathbf{y} = h_1$. AND- and OR-nodes are denoted by $\triangle$ and $\triangledown$, respectively.

For a given d-DNNF $\varphi_{\mathbf{y}}$, it is easy to compute $P(\mathbf{y}) = P(\varphi_{\mathbf{y}})$ by simply propagating the conditional probabilities $p(x|\boldsymbol{\pi})$ from the leaves $\theta_{x|\boldsymbol{\pi}}$ upwards to the root of the DAG. At each OR-node, determinism allows the incoming values to be added, and at each AND-node, decomposability allows the incoming values to be multiplied, as indicated in Fig. 2 by the symbols + and $*$, respectively. The result we obtain at the root is $P(h_1) = P(\varphi_{h_1}) = 0.2899$.

Computing probabilities is thus another efficient operation supported by d-DNNFs. In other words, any given compiled Bayesian network φ allows us to efficiently compute all possible simple queries $P(\mathbf{y}) = P(\varphi_{\mathbf{y}})$. This in turn enables the efficient computation of all possible general queries $P(h|\mathbf{e})$, namely in terms of two simple queries $P(h, \mathbf{e}) = P(\varphi_{h,\mathbf{e}})$ and $P(\mathbf{e}) = P(\varphi_{\mathbf{e}})$. Note that $\varphi_{h,\mathbf{e}}$ is often simpler than $\varphi_{\mathbf{e}}$. Moreover, it is very likely that $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ (or any pair of related d-DNNFs) share a substantial number of common subgraphs.[4] In Fig. 2, for example, it turns out that φ_{f_1,h_1} corresponds to the left subgraph of the root node of φ_{h_1}, whereas only three additional nodes are required to construct φ_{b_1,h_1}, two of them pointing to respective subgraphs of φ_{h_1}. The sharing of common subgraphs is important, as it allows the bottom-up computation of several probabilities in one single pass. We will heavily exploit this when it comes to realize the selection of the steepest ascent in the random-restart hill-climbing algorithm of the following section.

For the compilation itself, there are two distinct classes of methods. The methods of the first class start from encoding the Bayesian network as a CNF ψ, which is then converted into a d-DNNF $\varphi = \text{CNF2dDNNF}(\psi)$, e.g. by using Darwiche's compiler [22, 24]. This is the classical compilation approach in the literature [12, 14, 42, 45].

The more recent methods of the second class, called *tabular compilation* methods [13], avoid the detour over a CNF. The idea is to run a simple variable elimination procedure over all network variables. More generally spoken, it is the application of the *fusion* (or *bucket elimination*) algorithm to a particular type of *semiring valuations* [34], in which the semiring consists of all Boolean functions w.r.t. the variables $\Theta \cup \Delta$ (respectively of all classes of equivalent logical representations). For appropriate input valuations, it is easy to show that the output of the algorithm is indeed a d-DNNF. The fact that any algebra of semiring valuations satisfies the general valuation algebra axioms (see Theorem 2 in [34]) allows this type of compilation to fully exploit the principle of *local computation*. The worst-case complexity (for both time and space) is thus identical to standard join tree algorithms for Bayesian networks, i.e. exponential in the network's induced treewidth (= size of the largest node in the join node). In fact, one can look at tabular compilation as a standard inward propagation in a join tree, where the evolving d-DNNF keeps trace of the effected computations [13, 21].

4 Hill-Climbing in Compiled Credal Networks

Let's assume now that a given credal network is compiled in the same way as a corresponding Bayesian network, i.e. as if the attached credal sets were precise values. We will now show how to use the resulting d-DNNF φ as a starting position for the inner approximation of lower and upper posterior probabilities $\underline{P}(h|\mathbf{e})$ and $\overline{P}(h|\mathbf{e})$, respectively. If the hypothesis h and the evidence $\mathbf{e}$ are given, the first step is clear, namely to transform φ into corresponding d-DNNFs $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ (see previous section). Note that the same $\varphi_{\mathbf{e}}$ can be used for several hypotheses as long as $\mathbf{e}$ remains unchanged.

eral definitions, but their basic functionalities and properties remain the same.

[4] This is a consequence of the linear running time of conditioning, which restrains the number of newly created nodes.

4.1 The Hill-Climbing Algorithm

To realize the approximation of $\underline{P}(h|\mathbf{e})$ and $\overline{P}(h|\mathbf{e})$ as a hill-climbing algorithms, the next thing to do is to define an appropriate search space. For this, we make use of the fact that both $\underline{P}(h|\mathbf{e})$ and $\overline{P}(h|\mathbf{e})$ result from corresponding extreme points of the joint credal set $K(\mathbf{X})$, i.e. from elements of the set $\mathrm{Ext}(K(\mathbf{X}))$. This set in turn is determined by the extreme points $\mathrm{Ext}(K(X|\boldsymbol{\pi}))$ of the local credal sets $K(X|\boldsymbol{\pi})$ at each node of the network (see Section 2).

To access individual elements of $\mathrm{Ext}(K(\mathbf{X}))$, we employ a strategy that is similar to the use of *transparent variables* in [4, 6], but here we will not integrate them as explicit nodes into the network structure. The idea is thus to consider discrete variables $T_{X|\boldsymbol{\pi}}$, one for each local credal set $K(X|\boldsymbol{\pi})$, where the role of each $T_{X|\boldsymbol{\pi}}$ is to select an extreme point of the credal set $K(X|\boldsymbol{\pi})$. If $k_{X|\boldsymbol{\pi}} = |\mathrm{Ext}(K(X|\boldsymbol{\pi}))|$ denotes the number of extreme points of the credal set $K(X|\boldsymbol{\pi})$, then $\Omega_{T_{X|\boldsymbol{\pi}}} = \{1, \ldots, k_{X|\boldsymbol{\pi}}\}$ is the set of possible values of $T_{X|\boldsymbol{\pi}}$. Furthermore, if $\mathbf{T}$ denotes the set of all such variables $T_{X|\boldsymbol{\pi}}$, then

$$\Omega_{\mathbf{T}} = \prod_{T_{X|\boldsymbol{\pi}} \in \mathbf{T}} \Omega_{T_{X|\boldsymbol{\pi}}} \tag{8}$$

denotes the set of all configurations with respect to $\mathbf{T}$. For a specific configuration $\mathbf{t} = \mathbf{s}t\mathbf{u} \in \Omega_{\mathbf{T}}$, in which t denotes the value of the transparent variable $T_{X|\boldsymbol{\pi}}$ in $\mathbf{t}$, we can write $p_t(X|\boldsymbol{\pi}) \in \mathrm{Ext}(K(X|\boldsymbol{\pi}))$ to select the corresponding extreme point of the credal set $K(X|\boldsymbol{\pi})$. Similarly, we write $p_{\mathbf{t}}(\mathbf{X})$ for the selected joint probability mass function, $P_{\mathbf{t}}(h|\mathbf{e})$ for induced posterior probabilities, and $P_{\mathbf{t}}(\varphi_{h,\mathbf{e}})$ and $P_{\mathbf{t}}(\varphi_{\mathbf{e}})$ for probabilities of a compiled network. This formal setting allows us to rephrase the definitions of lower and upper posterior probabilities in Equation 6 and 7 by

$$\underline{P}(h|\mathbf{e}) = \min_{\mathbf{t} \in \Omega_{\mathbf{T}}} P_{\mathbf{t}}(h|\mathbf{e}) = \min_{\mathbf{t} \in \Omega_{\mathbf{T}}} \frac{P_{\mathbf{t}}(\varphi_{h,\mathbf{e}})}{P_{\mathbf{t}}(\varphi_{\mathbf{e}})}, \tag{9}$$

$$\overline{P}(h|\mathbf{e}) = \max_{\mathbf{t} \in \Omega_{\mathbf{T}}} P_{\mathbf{t}}(h|\mathbf{e}) = \max_{\mathbf{t} \in \Omega_{\mathbf{T}}} \frac{P_{\mathbf{t}}(\varphi_{h,\mathbf{e}})}{P_{\mathbf{t}}(\varphi_{\mathbf{e}})}, \tag{10}$$

respectively, i.e. $\Omega_{\mathbf{T}}$ is the discrete search space, on which the following steepest-ascent, random-restart hill-climbing procedure operates. The details of the procedure are shown in Algorithm 1, which deserves some additional explanations:

- Lines 2–3 describe the preparation phase. Line 4 sets the current global maximum $P_{\max}$ to 0.
- The outer loop (lines 5–12) describes the "random-restart" part of the algorithm. It starts by selecting a random configuration $\mathbf{t} \in \Omega_{\mathbf{T}}$ in Line 8 and ends by updating the current value for the global maximum. We assume the existence of a global variable *maxRuns*, which determines the number of passes.
- The actual hill-climbing takes place in the inner loop (lines 7–10). The crucial step for this is the selection of $\mathbf{t}$'s best neighbor in the search space $\Theta_{\mathbf{T}}$ by calling the function `BestNeighbor` (Line 9). This is the "steepest-ascent" part of the algorithm, which will later be discussed in further details. If no neighbor improves the current local maximum $P_{\mathbf{t}} = P_{\mathbf{t}}(h|\mathbf{e})$, we expect `BestNeighbor` to return *nil*.[5]
- The current value of the local maximum is updated in Line 8. This involves the bottom-up computation of the probabilities $P_{\mathbf{t}}(\varphi_{h,\mathbf{e}})$ and $P_{\mathbf{t}}(\varphi_{\mathbf{e}})$ based on the current selection of extreme points $p_{\mathbf{t}}(X|\boldsymbol{\pi})$, from which the actual values $p_t(\theta_{x|\boldsymbol{\pi}})$ of all variables $\theta_{x|\boldsymbol{\pi}} \in \Theta$ are extracted. Note that only those parts of $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ need to be processed, which are affected by the transition from the old to the new configuration. Of course, common subgraphs of $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ are processed in one single pass.

Algorithm 1: `ApproxUpperProb`$(\varphi, h, \mathbf{e}, \mathbf{T})$

```
1  begin
2      φ_{h,e} ← (φ|h,e)^{-Δ};
3      φ_e ← (φ|e)^{-Δ};
4      P_max ← 0;
5      for i ← 1 to maxRuns do
6          t ← RandomConfiguration(T);
7          repeat
8              P_t ← P_t(φ_{h,e}) / P_t(φ_e);
9              t ← BestNeighbor(t, T, P_t, φ_{h,e}, φ_e);
10         until t = nil;
11         P_max ← max{P_max, P_t};
12     return P_max;
13 end
```

The corresponding minimization algorithm, i.e. the approximation of the lower posterior probability $\underline{P}(h|\mathbf{e})$, is almost identical, except for the initialization of the global maximum (Line 4), the selection of the best neighbor (Line 9), and the updating of the global maximum (Line 11). In the rest of this paper, we will therefore restrict our discussion to the maximization problem.

[5] To avoid getting stuck on a plateau (flat part of the search space), the algorithm should allow so-called *sideway moves* to states with equal values. This may cause infinite loops, but they can be avoided by keeping track of previously visited plateau states. For simplicity, we do not explicitly take care of these details in the proposed algorithm.

4.2 Selecting the Best Neighbor Efficiently

Let us now take a closer look at the problem of selecting the best neighbor of the actual configuration $\mathbf{t}$. For this, suppose that $t \in \Omega_{T_{X|\boldsymbol{\pi}}}$ is the current value of a transparent variable $T_{X|\boldsymbol{\pi}} \in \mathbf{T}$ in the actual configuration $\mathbf{t} = \mathbf{s}t\mathbf{u}$. Every configuration $\mathbf{t}' = \mathbf{s}t'\mathbf{u}$ with $t' \in \Omega_{T_{X|\boldsymbol{\pi}}}$ and $t' \neq t$ is then a possible neighbor of $\mathbf{t}$ in $\Omega_{\mathbf{T}}$. Selecting the best neighbor, i.e. the neighbor with the most significant improvement with respect to the actual local maximum $P_{\mathbf{t}} = P_{\mathbf{t}}(h|\mathbf{e})$, means thus to compute $P_{\mathbf{t}'} = P_{\mathbf{t}'}(h|\mathbf{e})$ for all such configurations $\mathbf{t}'$ and all transparent variables $T_{X|\boldsymbol{\pi}} \in \mathbf{T}$. The following algorithm shows a naïve solution for this simple idea.

Algorithm 2: `BestNeighbor`($\mathbf{t}$,$\mathbf{T}$,$P_{\mathbf{t}}$,$\varphi_{h,\mathbf{e}}$,$\varphi_{\mathbf{e}}$)

```
1  begin
2      t_max ← t;
3      foreach T_{X|π} ∈ T do
4          t ← value of T_{X|π} in t;
5          foreach t' ∈ Ω_{T_{X|π}} \ {t} do
6              t' ← replace t by t' in t;
7              P_{t'} ← P_{t'}(φ_{h,e}) / P_{t'}(φ_e);
8              if P_{t'} > P_t then
9                  t_max ← t';
10                 P_t ← P_{t'};
11     if t = t_max then return nil;
12     else return t_max;
13 end
```

The problem with this naïve solution is the repetitive probability calculation in the inner loop (Line 7). This can be avoided by pre-compiling $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ according to the following Shannon decomposition, in which $\varphi_{\mathbf{y}}$ denotes a general instantiation of φ to a vector $\mathbf{y}$ and $X \in \mathbf{X}$ the network variable affected by the current transparent variable $T_{X|\boldsymbol{\pi}}$:

$$\begin{aligned} P_{\mathbf{t}'}(\varphi_{\mathbf{y}}) &= \sum_{x \in \Omega_X} P_{\mathbf{t}'}(\theta_{x|\boldsymbol{\pi}}) P_{\mathbf{t}'}(\varphi_{\mathbf{y}}|\theta_{x|\boldsymbol{\pi}}) \\ &= \sum_{x \in \Omega_X} p_{t'}(x|\boldsymbol{\pi}) P_{\mathbf{t}}(\varphi_{\mathbf{y}}|\theta_{x|\boldsymbol{\pi}}). \end{aligned} \tag{11}$$

Note that in the second line of Equation 11, it is no longer necessary to explicitly generate the neighboring configurations $\mathbf{t}'$. In other words, if we first derive from $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ all possible instantiations $\varphi_{h,\mathbf{e}}|\theta_{x|\boldsymbol{\pi}}$ and $\varphi_{\mathbf{e}}|\theta_{x|\boldsymbol{\pi}}$, respectively, we can use Equation 11 to directly obtain the probabilities $P_{\mathbf{t}'}(h|\mathbf{e})$ of all neighboring configurations $\mathbf{t}'$, i.e. without actually generating them. In Algorithm 2, this can be realized by skipping Line 6 and by replacing the right hand side of Line 7 by corresponding versions of Equation 11.

4.3 Recapitulation and Complexity Analysis

To conclude this section, let's first recapitulate the individual steps of the proposed method and then discuss their respective running time complexities.

To make the above steepest-ascent scheme work for a given hypothesis h and the evidence $\mathbf{e}$, we first need to transform the compiled network φ into $\varphi_{h,\mathbf{e}}$ and $\varphi_{\mathbf{e}}$ and then into $\varphi_{h,\mathbf{e}}|\theta_{x|\boldsymbol{\pi}}$ and $\varphi_{\mathbf{e}}|\theta_{x|\boldsymbol{\pi}}$ for all $\theta_{x|\boldsymbol{\pi}} \in \Theta$. The result is a collection

$$\begin{aligned} \Phi_{h|\mathbf{e}} = &\{\varphi_{h,\mathbf{e}}, \varphi_{\mathbf{e}}\} \cup \{\varphi_{h,\mathbf{e}}|\theta_{x|\boldsymbol{\pi}} : \theta_{x|\boldsymbol{\pi}} \in \Theta\} \\ &\cup \{\varphi_{\mathbf{e}}|\theta_{x|\boldsymbol{\pi}} : \theta_{x|\boldsymbol{\pi}} \in \Theta\} \end{aligned} \tag{12}$$

of d-DNNFs, which are likely to overlap heavily. This is illustrated in Fig. 3 in the form of a d-DNNF with multiple roots.

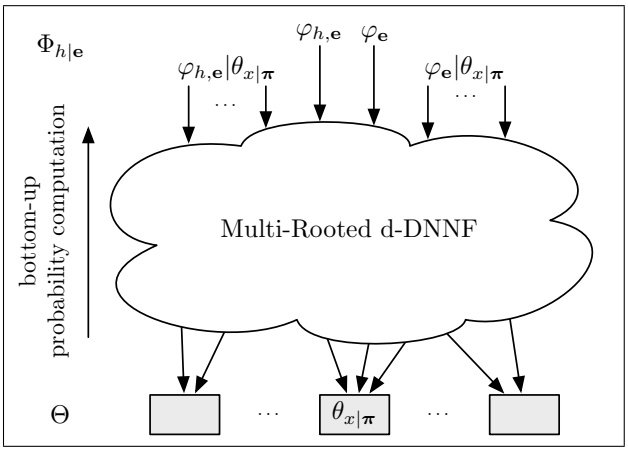

Figure 3: Probability computations in a multi-rooted d-DNNF with overlapping subgraphs.

To always keep the involved probabilities at each root up-to-date during the hill-climbing, we need to do the bottom-up probability computation only once at each hill-climbing step (i.e. at Line 8 of Algorithm 1), namely for the entire multi-rooted d-DNNF. The decision about the steepest ascent with respect to the current configuration $\mathbf{t}$ follows then from applying Equation 11 to all values t' that are incompatible with $\mathbf{t}$.

As discussed earlier, the worst-case running time and space complexity of the compilation phase is $O(2^d)$, where d denotes the network's induced treewidth for the given variable ordering. This is equivalent to the complexity of standard join tree algorithms for Bayesian networks. In other words, if $s = |\varphi|$ denotes the size (= number of edges) of the d-DNNF φ, then s reflects roughly the number of basic arithmetic operations (additions and multiplications) to be performed in the inward phase of a corresponding join tree propagation algorithm. Note that in the presence of strong local regularities in the form of context-specific independence, (pure or noisy) logical relationships, or

scarce CPTs, it is not untypical for the size s and therefore for the problem-specific complexity of the compilation phase to be much more favorable than $O(2^d)$.

The second preparatory step for the actual hill-climbing algorithm is the element-wise computation of the set $\Phi_{h|\mathbf{e}}$. For a given d-DNNF φ of size s, computing one such element requires $O(s)$ steps, which is a consequence of the fact that both conditioning and the particular type of variable elimination run in $O(s)$ time for d-DNNFs [25, 46]. Thus the total running time of the second step is $O(s{\cdot}|\Phi_{h|\mathbf{e}}|)$ and therewith $O(s{\cdot}|\Theta|)$, where $|\Theta|$ itself is proportional to both the number of network variables $n = |\mathbf{X}|$ and the corresponding maximal cardinality $c = \max\{|\Omega_X| : X \in \mathbf{X}\}$. This means that the worst-case running time of the entire preparatory phase is $O(c{\cdot}n{\cdot}2^d)$. This shows that the preparatory phase only depends on the network parameters c, d, and n, but not on the concrete local credal sets.

To analyze the running time of the actual hill-climbing algorithm, let S denote the total size of the multi-rooted d-DNNF on which the algorithm operates. Note that probability computations are supported by d-DNNFs in linear time, i.e. if K denotes the total number of extreme points over all locally specified credal sets (which correlates with the number of basic steps in the selection of the steepest ascent), then each individual hill-climbing step runs in $O(S{+}K)$ time. Since S is likely to be much larger than K, we can assume that the running time of the entire hill-climbing procedure is simply $O(\mathit{maxRuns}{\cdot}S)$. Due to the overlapping areas in the multi-rooted d-DNNF, S itself is often of the same order of magnitude as s.

5 Discussion and Conclusion

The method presented in this paper is a new technique to approximate inference in credal networks. The core of the approach is the idea of compiling the network into an appropriate logical form φ, which allows us to efficiently accomplish all necessary computational steps to answer probabilistic queries. Compilation techniques are increasingly applied to Bayesian networks, but the proposal to apply them to credal networks and to combine them with local search techniques is original.

With respect to existing approximation techniques for credal networks, let's point out some of the most important strengths of our approach.

- *Simplicity.* To make our approach work, only few simple procedures need to be implemented. The most important procedure is the compilation itself. For this, e.g. by using Nenok [39, 40], a generic framework for local computations in (semiring) valuation algebras, only few lines of code are necessary to handle the construction of the d-DNNF φ. Further procedures to implement are the operation of conditioning $\varphi|\mathbf{y}$ and the variable elimination $\varphi^{-\Delta}$. Both of them can be realized by simple recursions. The same holds for computing (and updating) the involved probabilities in the multi-rooted d-DNNF $\Phi_{h|\mathbf{e}}$, which turns out to be a classical postorder (bottom-up) traversal of a directed acyclic graph.

- *Flexibility.* The compiled logical form can be seen as a general recipe with precise instructions for the computation of all sorts of probabilities w.r.t. a given network. This is a very flexible and powerful starting position, which allows us to do all sorts of different things very easily, e.g. the efficient selection of the steepest ascent. The same structure could thus be used to solve other problems such as MAP or MPE.

- *Efficiency.* For a given multi-rooted d-DNNF, the updating of the probabilities during the hill-climbing process and the selection of the steepest ascent can be realized without any redundancy. The avoidance of redundancy can be enforced by exploiting local regularities already at the logical level. In fact, this is one of the key arguments for applying compilation techniques to Bayesian networks [12].

A couple of key questions have not yet been addressed in this paper. As corresponding implementations and testbeds are currently under development, we are not yet ready to say much about the empirical performance of the proposed method compared to existing methods. Other open questions concern the implementation of more sophisticated local search techniques such as *stochastic hill-climbing*, *simulated annealing*, or *genetic algorithms* [41]. These problems will be attacked in our subsequent work.

Acknowledgements

This research supported by the *Swiss National Science Foundation*, Project No. PP002-102652/1, and *The Leverhulme Trust*. Thanks to Michael Wachter for helpful discussions at the origin of this paper.

References

[1] A. Antonucci and M. Zaffalon. Locally specified credal networks. In *PGM'06, 3rd European Workshop on Probabilistic Graphical Models*, pages 25–34, Prague, Czech Republic, 2006.

[2] A. Antonucci, M. Zaffalon, J. S. Ide, and F. G. Cozman. Binarization algorithms for approximate updating in credal nets. In *STAIRS'06, 3rd European Starting AI Researcher Symposium*, pages 120–131, Riva del Garda, Italy, 2006.

[3] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *UAI'96, 12th Conference on Uncertainty in Artificial Inteligence*, pages 115–123, Portland, USA, 1996.

[4] A. Cano, J. Cano, and S. Moral. Convex sets of probabilities propagation by simulated annealing on a tree of cliques. *IPMU'94, 5th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, LNCS 945, pages 978–983, Paris, France, 1994. Springer.

[5] A. Cano, J. M. Fernández-Luna, and S. Moral. Computing probability intervals with simulated annealing and probability trees. *Journal of Applied Non-Classical Logics*, 12(2):151–171, 2002.

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

[7] A. Cano and S. Moral. A genetic algorithm to approximate convex sets of probabilities. In *IPMU'96, 6th international Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 859–864, Granada, Spain, 1996.

[8] A. Cano and S. Moral. A review of propagation algorithms for imprecise probabilities. *ISIPTA'99, 1st Internation Symposium on Imprecise Probabilities and Their Applications*, pages 51–60, Ghent, Belgium, 1999.

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

[10] J. E. Cano, S. Moral, and J. F. Verdegay-López. Propagation of convex sets of probabilities in directed acyclic networks. *Uncertainty in Intelligent Systems*, pages 15–26. North-Holland, 1993.

[11] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[12] M. Chavira and A. Darwiche. Compiling Bayesian networks with local structure. In *IJCAI'05, 19th International Joint Conference on Artificial Intelligence*, Edinburgh, U.K., 2005.

[13] M. Chavira and A. Darwiche. Compiling Bayesian networks using variable elimination. In *IJCAI'07, 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.

[14] M. Chavira, A. Darwiche, and M. Jaeger. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, 42(1–2):4–20, 2006.

[15] I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5(2):165–181, 2000.

[16] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

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

[18] F. G. Cozman and C. P. de Campos. Local computation in credal networks. In *ECAI'04, 16th European Conference on Artificial Intelligence, Workshop 22 on "Local Computation for Logics and Uncertainty"*, pages 5–11, Valencia, Spain, 2004.

[19] J. C. F. da Rocha and F. G. Cozman. Inference in credal networks with branch-and-bound algorithms. *ISIPTA'03, 3rd International Symposium on Imprecise Probabilities and Their Applications*, pages 480–493, Lugano, Switzerland, 2003.

[20] J. C. F. da Rocha, F. G. Cozman, and C. P. de Campos. Inference in polytrees with sets of probabilities. *UAI'03, 19th Conference on Uncertainty in Artificial Intelligence*, pages 217–224, Acapulco, Mexico, 2003.

[21] A. Darwiche. A differential approach to inference in Bayesian networks. *UAI'00, 16th Conference on Uncertainty in Artificial Intelligence*, pages 123–132, Stanford, USA, 2000.

[22] A. Darwiche. A compiler for deterministic, decomposable negation normal form. In *AAAI'02, 18th National Conference on Artificial Intelligence*, pages 627–634, Edmonton, Canada, 2002.

[23] A. Darwiche. A logical approach to factoring belief networks. *KR'02, 8th International Conference on Principles and Knowledge Representation and Reasoning*, pages 409–420, Toulouse, France, 2002.

[24] A. Darwiche. New advances in compiling CNF to decomposable negational normal form. In *ECAI'04, 16th European Conference on Artificial Intelligence*, Valencia, Spain, 2004.

[25] A. Darwiche and P. Marquis. A knowlege compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[26] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2nd edition, 2000.

[27] C. P. de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. *IJCAI'05, 19th International Joint Conference on Artificial Intelligence*, pages 1313–1318, Edinburgh, U.K., 2005.

[28] R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.

[29] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

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

[31] J. S. Ide and F. G. Cozman. IPE and L2U: Approximate algorithms for credal networks. In *STAIRS'04, 2nd European Starting AI Researcher Symposium*, pages 118–127, Valencia, Spain, 2004.

[32] J. S. Ide and F. G. Cozman. Approximate inference in credal networks by variational mean field methods. *ISIPTA'05, 4th International Symposium on Imprecise Probabilities and Their Applications*, pages 203–212, Pittsburgh, USA, 2005.

[33] J. Kohlas and P. P. Shenoy. Computation in valuation algebras. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5: Algorithms for Uncertainty and Defeasible Reasoning, pages 5–39. Kluwer Academic Publishers, Dordrecht, Netherlands, 2000.

[34] J. Kohlas and N. Wilson. Exact and approximate local computation in semiring induced valuation algebras. Technical Report 06–06, University of Fribourg, Switzerland, 2006.

[35] H. E. Kyburg. Interval-valued probabilities. *The Imprecise Probabilities Project*. IPP Home Page, available at http://ippserv.rug.ac.be, 1998.

[36] I. Levi. *The Enterprise of Knowledge*. The MIT Press, Cambridge, USA, 1980.

[37] J. Pearl. On probability intervals. *International Journal of Approximate Reasoning*, 2(3):211–216, 1988.

[38] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, USA, 1988.

[39] M. Pouly. Implementation of a generic architecture for local computation. *ECAI'04, 16th European Conference on Artificial Intelligence, Workshop 22 on "Local Computation for Logics and Uncertainty"*, pages 31–37, Valencia, Spain, 2004.

[40] M. Pouly. NENOK 1.1 user guide. Technical Report 06–02, Department of Informatics, University of Fribourg, Switzerland, 2006.

[41] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[42] T. Sang, P. Beame, and H. Kautz. Solving Bayesian networks by weighted model counting. In *AAAI'05, 20th National Conference on Artificial Intelligence*, volume 1, pages 475–482, Pittsburgh, USA, 2005.

[43] P. P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. *UAI'88, 4th Conference on Uncertainty in Artificial Intelligence*, pages 169–198, Minneapolis, USA, 1988.

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

[45] M. Wachter and R. Haenni. Logical compilation of Bayesian networks. Technical Report iam-06-006, University of Bern, Switzerland, 2006.

[46] M. Wachter and R. Haenni. Propositional DAGs: a new graph-based language for representing Boolean functions. *KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning*, pages 277–285, Lake District, U.K., 2006. AAAI Press.

[47] M. Wachter and R. Haenni. Multi-state directed acyclic graphs. *CanAI'07, 20th Canadian Conference on Artificial Intelligence*, LNAI 4509, pages 464–475, Montréal, Canada, 2007.

[48] K. Weichselberger. The theory of interval-probability as a unifying concept for uncertainty. *International Journal of Approximate Reasoning*, 24(2–3):149–170, 2000.