International Journal of Approximate Reasoning 125 (2020) 26–48

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Tractable inference in credal sentential decision diagrams

Lilith Mattei [a], Alessandro Antonucci [a,*], Denis Deratani Mauá [b], Alessandro Facchini [a], Julissa Villanueva Llerena [b]

[a] *Istituto Dalle Molle di Studi per l'Intelligenza Artificiale, Manno-Lugano, Switzerland*
[b] *Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil*

ARTICLE INFO

Article history:
Received 27 December 2019
Received in revised form 2 June 2020
Accepted 18 June 2020
Available online 9 July 2020

Keywords:
Probabilistic graphical models
Imprecise probability
Credal sets
Probabilistic circuits
Sentential decision diagrams
Sum-product networks

ABSTRACT

Probabilistic sentential decision diagrams are logic circuits where the inputs of disjunctive gates are annotated by probability values. They allow for a compact representation of joint probability mass functions defined over sets of Boolean variables, that are also consistent with the logical constraints defined by the circuit. The probabilities in such a model are usually "learned" from a set of observations. This leads to overconfident and prior-dependent inferences when data are scarce, unreliable or conflicting. In this work, we develop the *credal sentential decision diagrams*, a generalisation of their probabilistic counterpart that allows for replacing the local probabilities with (so-called *credal*) sets of mass functions. These models induce a joint credal set over the set of Boolean variables, that sharply assigns probability zero to states inconsistent with the logical constraints. Three inference algorithms are derived for these models. These allow to compute: (i) the lower and upper probabilities of an observation for an arbitrary number of variables; (ii) the lower and upper conditional probabilities for the state of a single variable given an observation; (iii) whether or not all the probabilistic sentential decision diagrams compatible with the credal specification have the same most probable explanation of a given set of variables given an observation of the other variables. These inferences are *tractable*, as all the three algorithms, based on bottom-up traversal with local linear programming tasks on the disjunctive gates, can be solved in polynomial time with respect to the circuit size. The first algorithm is always exact, while the remaining two might induce a conservative (outer) approximation in the case of multiply connected circuits. A semantics for this approximation together with an auxiliary algorithm able to decide whether or not the result is exact is also provided together with a brute-force characterization of the exact inference in these cases. For a first empirical validation, we consider a simple application based on noisy seven-segment display images. The credal models are observed to properly distinguish between easy and hard-to-detect instances and outperform other generative models not able to cope with logical constraints.

* Corresponding author.
E-mail addresses: lilith@idsia.ch (L. Mattei), alessandro@idsia.ch (A. Antonucci), ddm@ime.usp.br (D.D. Mauá), alessandro.facchini@idsia.ch (A. Facchini), jgville@ime.usp.br (J. Villanueva Llerena).

https://doi.org/10.1016/j.ijar.2020.06.005
0888-613X/

1. Introduction

Probabilistic graphical models [1,2] are widely used in machine learning and knowledge-based decision-support systems, due to their ability to provide compact and intuitive descriptions of joint probability mass functions by exploiting conditional independence relations encoded in a graph. However, the ability to provide compact representation does not imply that inferences with the model can be computed efficiently [3–5], and practicioners need to rely on approximate inference algorithms with no guarantees.

To allow for fast and accurate inference, some authors have proposed abandoning the intuitive (declarative) semantics of graphical models in favour of a more procedural (and less transparent) representation of probability mass functions as arithmetic (or logic) circuits [6–9]. The latter has been broadly termed *tractable models*, for their ability to provide polynomial-time inference with respect to the circuit size. *Sum-product networks* (SPNs) [7] are the most popular example in this area. Remarkably, SPNs can be also intended as a probabilistic counterpart of deep neural networks and, when used for machine learning, they offer competitive performances in many tasks [10,11].

Another prominent example of tractable models are *probabilistic sentential decision diagrams* (PSDDs) [9]. Roughly speaking, a PSDD is a logical circuit representation of a joint probability mass function that assigns zero probability to the impossible states of the underlying logical constraints. Notably, PSDDs allow for enriching statistical models with knowledge about constraints in the domain without sacrificing efficient inference [12–15].

When data are scarce, conflicting or unreliable, learning sharp estimates of probability values can lead to inferences that are dominated by the choice of hyperparameters and priors. The area of *imprecise probabilities* advocate for a more flexible and robust representation of statistical models, through the use of *credal sets*, that is, sets of probability mass functions induced by a (typically finite) number of linear constraints [16]. This leads to the development of generalizations of graphical models such as *credal networks* [17], that extend Bayesian networks to allow for the representation of imprecisely specified conditional probability values.

Recently, SPNs have also been extended to the imprecise probability setting, giving rise to *Credal Sum-Product Networks* (CSPNs) [18–20]. These models allow for a richer representation of uncertainty without compromising computational tractability of inferences.

In this work, we develop the *Credal Sentential Decision Diagrams* (CSDDs), a credal-set extension of probabilistic sentential decision diagrams that allow for richer representation of uncertainty with small computational overhead. Compared to CSPNs, CSDDs allow for a more principled semantics of local credal sets.

We take advantage of the structural similarities between PSDDs and SPNs to adapt many of the algorithms originally proposed for CSPNs [18,20] for CSDDs. More specifically, a PSDD can be seen as a special type of *selective* SPNs [21], where differently from standard SPNs, *Maximum-A-Posteriori* (MAP) inference and parameter learning can be performed efficiently [22,23]. As a result we therefore deliver three algorithms for CSDDs allowing to compute: (i) *marginals*, that is, the lower and upper probabilities of an observation of an arbitrary number of model variables, (ii) *conditionals*, that is, the lower and upper probabilities of single queried variable given observations of some other variables; and (iii) *MAP robustness*, that is, checking whether or not the most probable configuration for some queried variables given an observation of the other ones is the same for all PSDDs consistent with a CSDD. Those inferences are tractable as all the algorithms only require a bottom-up traversal of the logical circuit underlying the model with local linear programming tasks to be solved on the disjunctive nodes, thus being polynomial in the circuit size. The inferences are always exact for the first task, while for the remaining two tasks the procedure delivers a conservative (outer) approximation for multiply connected circuits (see Definition 4). For these cases, a polynomial-time algorithm to check whether or not the inference is exact is also provided together with a bound on the complexity required to compute exact inference by brute force.

This paper extends a preliminary version [24] with the inclusion of the algorithm for MAP robustness, the characterization of the approximation in the multiply connected case, and an experimental validation.

The rest of the paper if organized as follows. In the next section we open the discussion with a toy example to be used along the paper to illustrate our approach. Section 3 contains background material about credal sets and PSDDs. The technical results are presented in Section 4 where we define CSDDs, and in Sections 5-7 where the three inference algorithms are derived. The results of an experimental validation are discussed in Section 8, while conclusions and outlooks are in Section 9. Proofs are in the appendix together with some additional technical material.

2. A demonstrative example

We begin the discussion with a minimalistic example to be used as an informal introduction to the basic concepts and problems considered in the paper. Formal definitions of these basics are provided in the next section. The example is used in the other sections to demonstrate the main ideas derived in our work and show how these can be applied.

Consider four-pixel black-and-white squared images in Fig. 1. These can be regarded as joint states of four Boolean variables. We assume that, out of sixteen possible configurations, only those in the top row of the Fig. 1 are permitted, while the remaining six in the bottom row are forbidden by some structural constraint (e.g., only “lines” and “points” can be depicted).

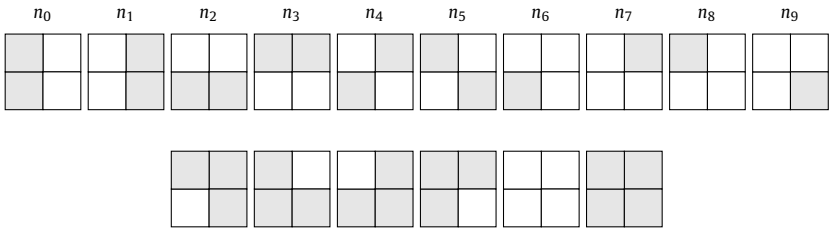

Fig. 1. Permitted (top) and un-permitted (bottom) four-pixel squared images.

Let us denote the four variables as (X_1, X_2, X_3, X_4), where X_1 corresponds to the top-left pixel and the other ones follow a clock-wise order. If black pixel corresponds to the true state of the variable, the formula implementing the constraints can be written as[1]:

$$\gamma := \left[\bigvee_{1\leq i\leq 4} X_i\right] \wedge \left[\bigvee_{1\leq i\neq j\leq 4} \neg X_i \wedge \neg X_j\right] \tag{1}$$

where the two conjunctive clauses impose, respectively, that at least one pixel is black and two pixels are white. These constraints rule out exactly the configurations in the bottom row in Fig. 1.

Consider the logic circuit in Fig. 2, where conjunctive gates are depicted in blue and they alternate with the disjunctive (red) ones. For the moment, ignore the parameters associated with the inputs of the disjunctive gates and the *top* (i.e., $\top$) inputs of the conjunctive ones. The reader can verify that the formula implemented by the circuit is equivalent to γ in Equation (1).[2]

Consider a data set of observations for the permitted configurations is available, where each configuration occurs with the counts $n_0, \ldots, n_9$, as indicated on the top of the squares in Fig. 1 for the top row. Say that we want to learn from these data a generative model, that is, a joint probability mass function over the four variables. Such a mass function should be also consistent with the logical constraints, that is, the six impossible configurations should receive zero probability.

As the sub-formulae associated to the three inputs of the disjunctive gate in the circuit output are disjoint, a joint mass function consistent with ϕ could be simply $\theta_1 I_{\phi_1} + \theta_2 I_{\phi_2} + (1-\theta_1-\theta_2) I_{\phi_3}$, where ϕ_i is the formula associated with the i-th input of the gate for each $i = 1, 2, 3$, and I denotes the indicator function of the formula in its subscript. For each $i = 1, 2, 3$, the parameter θ_i is therefore the probability of ϕ_i, that can be estimated from the data. For example, a maximum likelihood estimator would give $\theta_1 = \frac{n_2+n_6+n_9}{n}$ and $\theta_2 = \frac{n_0+n_1+n_4+n_5+n_7+n_8}{n}$ where $n = \sum_{i=0}^{9} n_i$.

More refined joint mass functions can be obtained by a recursive application of this approach to the other disjunctive gates and multiplying the contributions associated with the inputs of a conjunctive gate. In those cases the parameters should be intended as conditional probabilities for the corresponding sub-formula given by a so called *context*.[3]

Finally, for the circuit inputs, we specify indicator functions of their literals, these being replaced by a zero for *bots* (i.e., $\bot$), and by a probability mass function $\theta I_X + (1-\theta) I_{\neg X}$ for a *top* (i.e., $\top$) associated with variable X and annotated with a probability θ. Accordingly, the annotated circuit in Fig. 2 induces the joint probability mass function:

$$\begin{aligned}\theta_1 \cdot \left[I_{\neg X_1} I_{\neg X_2}\right] \cdot \left[\theta_3 I_{\neg X_3} I_{X_4} + (1-\theta_3) I_{X_3}\left[\theta_6 I_{X_4} + (1-\theta_6)\, I_{\neg X_4}\right]\right] + \theta_2 \cdot \left[\theta_4 I_{X_1} I_{\neg X_2} + (1-\theta_4) I_{\neg X_1} I_{X_2}\right] \cdot \\ \cdot \left[\theta_5 I_{X_3} I_{\neg X_4} + (1-\theta_5) I_{\neg X_3}\left[\theta_7 I_{X_4} + (1-\theta_7) I_{\neg X_4}\right]\right] + (1-\theta_1-\theta_2) \cdot \left[I_{X_1} I_{X_2}\right] \cdot \left[I_{\neg X_3} I_{\neg X_4}\right],\end{aligned} \tag{2}$$

where the variables of the indicator functions are left implicit for the sake of readability. An annotated circuit as that in Fig. 2, defining a generative model as the one in Equation (2), which is consistent with the formula γ in Equation (1), is called a *probabilistic sentential decision diagram* [9].

In this paper we are interested in developing algorithms for sensitivity analysis of the inferences in these models with respect to the parameters. This is important when only few training data are available and sharp estimates of the parameters might be not reliable. Moreover, the parameters not associated with the output disjunctive gate are conditional probabilities

[1] We assume the reader to be familiar with basic propositional logic notation. More details about that can be found in Section 3.2.

[2] To see this, notice that the logic circuit in Fig. 2 encodes formula

$$\begin{aligned}\phi := \quad & \big((\neg X_1 \wedge \neg X_2) \wedge ((\neg X_3 \wedge X_4) \vee X_3)\big) \quad \vee \\ & \big(((X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2)) \wedge ((X_3 \wedge \neg X_4) \vee \neg X_3)\big) \quad \vee \\ & \big(X_1 \wedge X_2 \wedge \neg X_3 \wedge \neg X_4\big)\end{aligned}$$

The three disjuncts are mutually exclusive. Models of the first disjuncts correspond to four-pixel squared images whose counts are n_2, n_6, n_9, models of the second disjuncts correspond to four-pixel squared images whose counts are n_0, n_1, n_4, n_5, n_7 and n_8, and finally the unique model of the third disjuncts corresponds to the four-pixel squared image whose count is n_3.

[3] Roughly, a context of a node in the circuit is the formula determined by the path leading to it and such that, joint with the underlying SDD, implies the formula associated to the node. A formal statement is given in Definition 3.

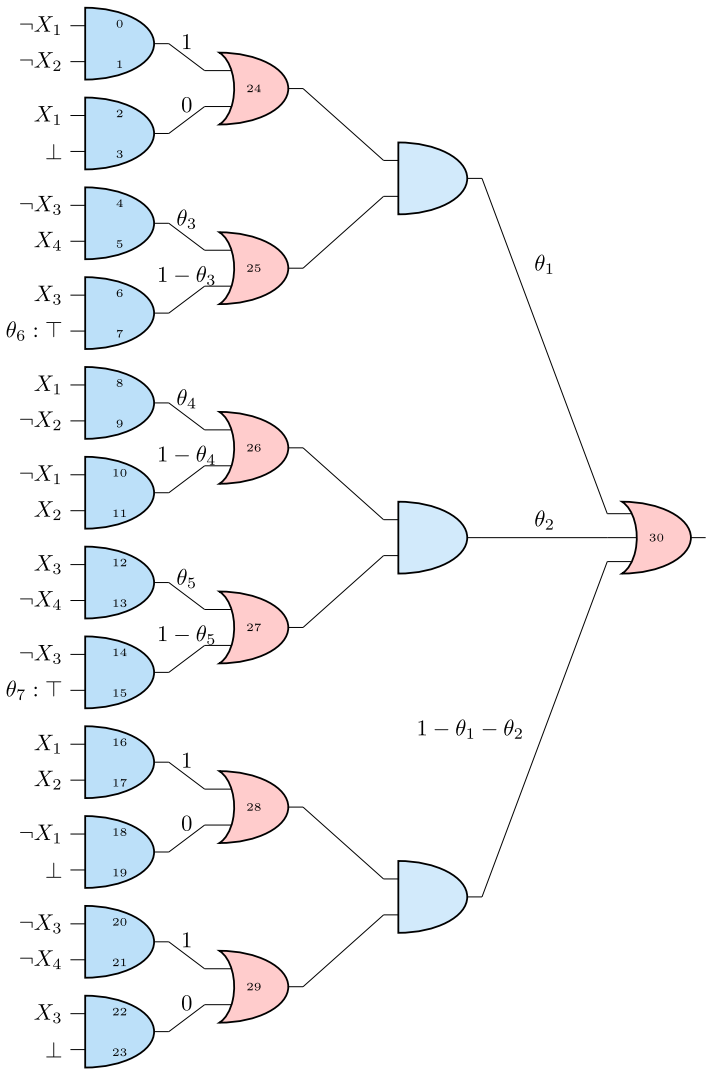

Fig. 2. A probabilistic sentential decision diagrams (PSDD) over four Boolean variables. The corresponding sentential decision diagram (SDD) is the underlying logic circuit when the probabilistic annotations of the PSDD are not considered (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

and the closer the parameter is to the input, the higher will be the number of variables involved in the conditioning event. Thus, in deep circuits, we might have very few training data to learn those parameters even if the available training data set is huge, thus making important the development of tools for sensitivity analysis. The notion of probabilistic sentential decision diagrams, together with other background concepts, are formally described in the next section.

3. Background

3.1. Credal sets

Consider a variable X taking its values in a finite set $\mathcal{X}$ whose generic element is denoted as x. A *probability mass function* (PMF) over X, denoted as $\mathbb{P}(X)$, is a real-valued non-negative function $\mathbb{P} : \mathcal{X} \to \mathbb{R}$ such that $\sum_{x \in \mathcal{X}} \mathbb{P}(x) = 1$. Given a function f of X, the *expectation* of f with respect to a PMF $\mathbb{P}$ is $\mathbb{P}[f] := \sum_{x \in \mathcal{X}} f(x) \cdot \mathbb{P}(x)$. A set of PMFs over X is called *credal set* (CS) and denoted as $\mathbb{K}(X)$. Here we consider CSs induced by a finite number of linear constraints. Given CS $\mathbb{K}(X)$, the bounds of the expectation with respect to $\mathbb{K}(X)$ can be computed by optimizing $\mathbb{P}[f]$ over $\mathbb{K}(X)$. For example, for the lower bound, $\underline{\mathbb{P}}[f] := \min_{\mathbb{P}(X) \in \mathbb{K}(X)} \sum_{x \in \mathcal{X}} f(x) \cdot \mathbb{P}(x)$. This is a linear programming task, whose optimum remains the same after replacing $\mathbb{K}(X)$ with its convex hull. Such optimum is attained on an extreme point of the convex closure. Moreover, if f is an indicator function, the lower expectation is called *lower* probability. Notation $\overline{\mathbb{P}}$ is used instead for the upper bounds and duality $\overline{\mathbb{P}}(f) = -\underline{\mathbb{P}}(-f)$ holds.

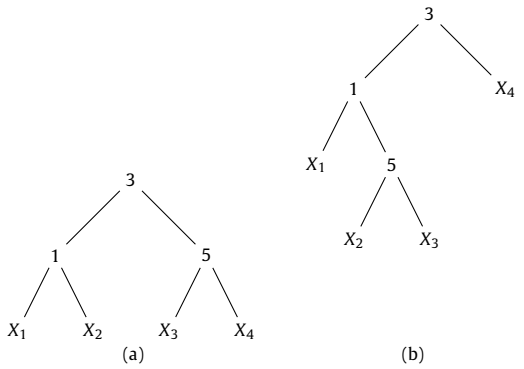

Fig. 3. Two vtrees over four variables.

In the special case of Boolean variables it is easy to see that the number of extreme points of the convex closure of a CS cannot be more than two, and the specification of a single interval constraint, say $0 \leq l \leq \mathbb{P}(x) \leq u \leq 1$ for one of the two states is a fully general CS specification.

Learning CSs from multinomial data can be done by the *imprecise Dirichlet model* (IDM) [16]. This is a generalised Bayesian approach in which a single Dirichlet prior with equivalent sample size s is replaced by the set of all the Dirichlet priors with this size. The corresponding bounds on the probabilities are

$$\mathbb{P}(x) \in \left[\frac{n(x)}{N+s}, \frac{n(x)+s}{N+s}\right] \tag{3}$$

where $n(x)$ are the number of instances of the data set, whose total size is N, such that $X = x$, for each $x \in \mathcal{X}$.

Given PMF $\mathbb{P}(X_1, X_2)$, X_1 and X_2 are *stochastically independent* if and only if $\mathbb{P}(x_1, x_2) = \mathbb{P}(x_1) \cdot \mathbb{P}(x_2)$ for each $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$. We similarly say that, given CS $\mathbb{K}(X_1, X_2)$, X_1 and X_2 are *strongly independent* if and only if stochastic independence is satisfied for each extreme point of the convex closure of the joint CS.

3.2. Sentential decision diagrams

Give a finite set of Boolean variables $\boldsymbol{X}$, a *literal* is either a Boolean variable $X \in \boldsymbol{X}$ or its negation $\neg X$. The Boolean constant always taking the value false or true is denoted, respectively, as $\bot$ and $\top$.

We start by defining a generalisation of orders on variables based on the following definition.

Definition 1 *(Vtree).* Consider a finite set $\mathbf{X}$ of Boolean variables. A *vtree* for $\mathbf{X}$ is a full binary tree v whose leaves are in one-to-one correspondence with the elements of $\mathbf{X}$. We denote by v^l (resp., v^r) the left (right) subtree of v, i.e., the vtree rooted at the left (resp., right) child of the root of v.

Two vtrees for the variables in the example in Section 2 are in Fig. 3. Note that the in-order tree traversal of a vtree induces a total order on the variables, but two distinct vtrees can induce in this way the same order (e.g., the two vtrees in Fig. 3).

Based on the notion of vtree, we provide the following definition of SDDs.

Definition 2 *(SDD).* A *sentential decision diagram* (SDD) α *normalised for vtree* v and its interpretation $\langle\alpha\rangle$ are defined inductively as follows.

- If v is a leaf, let X be the variable attached to v; then α is either a *constant*, i.e., $\alpha \in \{\bot, \top\}$, or a *literal*, i.e., $\alpha \in \{X, \neg X\}$.
- If v is not a leaf, then $\alpha = \{(p_i, s_i)\}_{i=1}^k$, where the p_i's and s_i's, called *primes* and *subs*, are SDDs normalised for v^l and v^r respectively.

The interpretation of an SDD α normalised for v, denoted as $\langle\alpha\rangle$, is a propositional sentence over the variables of v, defined as follows:

- If $\alpha \in \{\bot, \top, X, \neg X\}$: $\langle\bot\rangle = \bot$, $\langle\top\rangle = \top$ and $\langle X\rangle = X$, $\langle\neg X\rangle = \neg X$.
- If $\alpha = \{(p_i, s_i)\}_{i=1}^k$, $\langle\alpha\rangle = \bigvee_{i=1}^k \langle p_i\rangle \wedge \langle s_i\rangle$ and interpretations $\{\langle p_i\rangle\}_{i=1}^k$ form a partition.

The *sub-SDDs* of an SDD α are α itself, its primes, its subs, and the sub-SDDs of its primes and subs. A sub-SDD will be often called a *node*, more precisely a *terminal node* when it is normalized for a leaf, and a *decision node* otherwise.

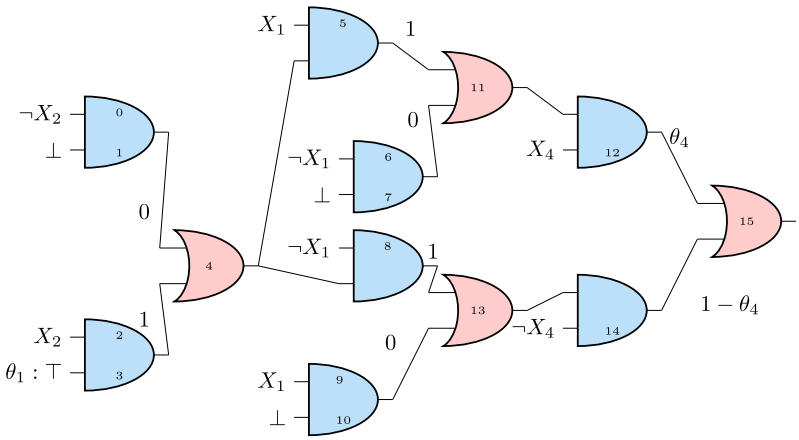

Fig. 4. A PSDD whose underlying SDD is multiply connected, normalized for the second vtree in Fig. 3, and represents formula $\phi = (X_1 \wedge X_2 \wedge X_4) \vee (\neg X_1 \wedge X_2 \wedge \neg X_4)$.

In a decision node $\{(p_i, s_i)\}_{i=1}^{k}$, the pairs (p_i, s_i)'s are called the *elements* of the node, and k is its *size*. The size of an SDD is the sum of the sizes of all its decision nodes.[4]

At the interpretation level, each decision node represents a disjunction (actually, an exclusive disjunction, as the primes form a partition), while each of its elements is a conjunction between a prime and a sub.

Example 1. Given the vtree v over the ordered pair of variables (A, P), $\alpha = \{(A, P), (\neg A, \top)\}$ is an SDD normalized for v; the interpretation of α is $\langle \alpha \rangle = (A \wedge P) \vee (\neg A \wedge \top)$, which is logically equivalent to $\phi = A \rightarrow P$.

Given the previous discussion, we can intend the SDD α as a rooted logic circuit, like the one in Fig. 2, providing a representation of the formula $\langle \alpha \rangle$. The labels on decision nodes denote the vtree nodes for which the decision node is normalized.

The following definition makes formal the notion of path in an SDD. This is needed to provide a semantics for the parameters used to annotate SDDs.

Definition 3 *(Context).* Let n be a node (either terminal or decision) of an SDD. Denote as $(p_1, s_1), \ldots, (p_l, s_l)$ a path from the root to node n. Then the conjunction of the interpretations of the primes encountered in this path, i.e., $\langle p_1 \rangle \wedge \cdots \wedge \langle p_l \rangle$, is called a *context* of n and denoted as γ_n. The context γ_n is *feasible* if and only if $s_i \neq \bot$ for each $i = 1, \ldots, l$.

By construction, each node has at least one context. The number of contexts of a node defines its *multiplicity* as follows.

Definition 4. The multiplicity of an SDD node is the number of its contexts. An SDD is *singly connected* if all of its nodes have multiplicity equal to one. Otherwise, it is *multiply connected*.

Notice that, at the circuit level, the definition of multiply connected SDD coincides with the graph-theoretical one.

Example 2. Consider SDD in Fig. 4. The terminal node with label 12 has multiplicity one and its context is $\gamma = X_1 \wedge X_2$. The decision node with label 4 (in pink in the figure) has multiplicity two and its contexts are $\gamma' = ((X_1 \wedge X_2) \wedge X_1) = X_1 \wedge X_2$ and $\gamma'' := ((\neg X_1 \wedge X_2) \wedge \neg X_1) = \neg X_1 \wedge X_2$.

The interpretation of a node is implied by its contexts and by the interpretation of the SDD it belongs to, that is, for each node n of an SDD α, for any context γ_n, we have that $\langle \alpha \rangle \wedge \gamma_n \models \langle n \rangle$.

Let us finally define a notion of topological order for the nodes of an SDD. The logic circuit underlying the SDD can be regarded as a directed graph whose arcs are oriented from the inputs to the outputs. Yet, an order in the circuit does not induce a complete order over the SDD nodes as the conjunctive gates corresponds to pairs or nodes (i.e., elements). Nevertheless, to obtain a complete order we might simply force both the nodes of an element to precede their decision node, while the terminal nodes are clearly preceding all the decision nodes.

[4] The size of an SDD depends on the number of variables, the base knowledge and the choice of the vtree. The notion of *nicety* for vtrees with respect to a given formula provides a bound on the SDD size [25]. Yet, the existence of a nice vtree is guaranteed for CNFs only.

3.3. Probabilistic sentential decision diagrams

A *probabilistic sentential decision diagram* is a parametrized SDD, where parameters are PMFs specifications on the decision nodes and on the terminal nodes labelled with constant top. A PSDD induces a joint PMF over its variables, assigning zero probability to the impossible states of the logical constraint given by the interpretation of the underlying SDD.

To turn an SDD into a PSDD, proceed as follows. For each terminal node $\top$, specify a positive parameter θ such that $0 \leq \theta \leq 1$. Notation for such terminal node is $X : \theta$, where X is the variable of the leaf vtree node for which $\top$ is normalised. Terminal nodes other than $\top$ appear as they are; for each decision node $\{(p_i, s_i)\}_{i=1}^{k}$, specify for each prime p_i a real number $\theta_i \geq 0$, such that $\sum_{i=1}^{k} \theta_i = 1$ and $\theta_i = 0$ if and only if $s_i = \bot$. Notation $\{(p_i, s_i, \theta_i)\}_{i=1}^{k}$ is used to denote such a parametrisation. The interpretation of such parametrisation is the following. Each node $n \neq \bot$ normalized for vtree node v induces a PMF $\mathbb{P}_n$ defined inductively as follows:

- if n is a terminal node whose corresponding variable in v is X, then $\mathbb{P}_n$ is a PMF over $\{\top, \bot\}$ such that:
  - if $n = X$, $\mathbb{P}_n(\top) = 1$ and $\mathbb{P}_n(\bot) = 0$
  - if $n = \neg X$, $\mathbb{P}_n(\top) = 0$ and $\mathbb{P}_n(\bot) = 1$
  - if $n = X : \theta$, $\mathbb{P}_n(\top) = \theta$ and $\mathbb{P}_n(\bot) = 1 - \theta$
- if $n = \{(p_i, s_i, \theta_i)\}_{i=1}^{k}$ is a decision node, let $(\boldsymbol{X}, \boldsymbol{Y})$ be the variables of v^l, v^r respectively. Then the joint PMF $\mathbb{P}_n(\boldsymbol{X}, \boldsymbol{Y})$ is defined as:

$$\mathbb{P}_n(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{P}_{p_i}(\boldsymbol{x}) \cdot \mathbb{P}_{s_i}(\boldsymbol{y}) \cdot \theta_i \,, \tag{4}$$

  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$, where i is the unique index such that $\mathbf{x} \models \langle p_i \rangle$.

In other words, PSDDs are SDDs with PMFs associated to each node distinct from $\bot$. It follows that sub-SDDs of a PSDD are in fact sub-PSDDs, except for terminal nodes $\bot$ (because such nodes do not induce a PMF). According to the *Base Theorem* for PSDDs [9, Theorem 1], the PMF $\mathbb{P}_n$ assigns zero probability to events which do not respect the propositional sentence associated to the SDD n. More precisely, for any instantiation $(\boldsymbol{x}, \boldsymbol{y})$ of variables $(\boldsymbol{X}, \boldsymbol{Y})$ of the vtree n is normalised for, $\mathbb{P}_n(\boldsymbol{x}, \boldsymbol{y}) > 0$ iff $(\boldsymbol{x}, \boldsymbol{y}) \models \langle n \rangle$. Moreover, the probabilities $\mathbb{P}_n(\langle p_i \rangle)$ are the parameters θ_i's of $n = \{(p_i, s_i, \theta_i)\}_{i=1}^{k}$.

We simply denote as $\mathbb{P}$ the (joint) PMF induced by the root r. PMF $\mathbb{P}_n$ induced by an internal node can be obtained by conditioning $\mathbb{P}$ on a feasible context of the considered node [9, Theorem 4]: for each feasible context γ_n of n, $\mathbb{P}_n(\cdot) = \mathbb{P}(\cdot | \gamma_n)$. The topological definitions made for SDDs extend to PSDDs. Finally, we have the following result about independence [9, Theorem 5]: according to $\mathbb{P}$, the variables inside v are independent of those outside v given context γ_n. This is the PSDD analogue of the *Markov condition* for Bayesian networks.

3.4. Inferences in PSDDs

PSDD inferences are computed with respect to the joint PMF $\mathbb{P}$. The probability of a joint state $\boldsymbol{e}$ of a set of PSDD variables $\boldsymbol{E}$ can be obtained in linear time with respect to the diagram size by the bottom-up (i.e., based on a topological order from the inputs to the output) scheme in Algorithm 1. Note that here and in the rest of the paper we assume that the nodes of the PSDD are labelled by integers from one to N following a topological order and N is therefore the output/root of the circuit. Given a vtree node v, notation $\mathbf{e}_v$ is used for the subset of $\mathbf{e}$ including only the variables of v. Note also that, as the node index n in the loop follows a topological order, the *message* $\pi(n)$, to be computed after the *else* statement, is always a combination of messages already computed.

Algorithm 1 Probability of evidence [9].

```
input: PSDD, evidence e
for n ← 1, ..., N (topological order) do
  π(n) ← 0
  if node n is terminal, n ≠ ⊥ then
    v ← leaf vtree node that n is normalized for
    π(n) ← ℙ_n(e_v)
  else
    (p_i, s_i, θ_i)_{i=1}^k ← n (decision node)
    π(n) ← Σ_{i=1}^k π(p_i) · π(s_i) · θ_i
  end if
end for
output: ℙ(e) ← π(N)
```

The computation of a conditional query is based on a similar strategy.

Regarding MAP inference, that is, the problem of finding the most probable configuration for a set of variables given an observation of the other ones, the computation proceeds very similarly, replacing the sums with maximizations [26]. More formally, given a PSDD rooted at r, and evidence $\boldsymbol{e}$ for the variables in $\boldsymbol{E}$, we are interested in finding

$\boldsymbol{x}^* := \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{P}_r(\boldsymbol{x}|\boldsymbol{e})$ for the PSDD variables other than $\boldsymbol{E}$ and denoted as $\boldsymbol{X}$. We assume the evidence consistent with the PSDD logical constraints and hence $\mathbb{P}_r(\boldsymbol{e}) > 0$. This way, the task is well-defined and it is equivalent to the maximization of the joint, that is,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{P}_r(\boldsymbol{x}, \boldsymbol{e})\,. \tag{5}$$

Algorithm 2 takes as input a PSDD rooted at r over variables $\{\mathbf{X}, \mathbf{E}\}$ (with $\mathbf{X}$ and $\mathbf{E}$ disjoint) and evidence $\mathbf{e}$ over variables $\mathbf{E}$, and computes $\max_{\mathbf{x}\in\boldsymbol{\mathcal{X}}} \mathbb{P}_r(\mathbf{x}, \boldsymbol{e})$. Correctness is implied by the following result.

Theorem 1. *The output of Algorithm 2 is the probability of the configuration of Equation (5), that is,*

$$MAP(r) = \mathbb{P}_r(\boldsymbol{x}^*, \boldsymbol{e})\,. \tag{6}$$

Finally, the arguments realizing the maximum may be obtained by backtracking the solutions of the maximizations.

Algorithm 2 MAP.

input: PSDD r, evidence $\mathbf{e}$
for $n \leftarrow 1, \ldots, N$ **do**
  $MAP(n) \leftarrow 0$
  if node n is terminal, $n \neq \bot$ **then**
    $v \leftarrow$ leaf vtree node that n is normalized for
    if $var(v) \in \mathbf{X}$ **then**
      if $n \in \{X, \neg X\}$ **then**
        $MAP(n) \leftarrow 1$
      else if $n = (\top, \theta)$ **then**
        $MAP(n) \leftarrow \max\{\theta, 1-\theta\}$
      end if
    else
      $MAP(n) \leftarrow \mathbb{P}_n(\mathbf{e})$
    end if
  else
    $(p_i, s_i, \theta_i)_{i=1}^k \leftarrow n$ (decision node)
    $MAP(n) \leftarrow \max_{i=1}^k MAP(p_i) \cdot MAP(s_i) \cdot \theta_i$
  end if
end for
output: $MAP(N)$

4. Credal sentential decision diagrams

In this section we present a generalization of PSDDs (see Section 3.3) based on the notion of credal set provided in Section 3.1. The number of variables involved in a node's context increases with the distance from the root when the SDD is singly connected (see Definition 3). As the PMFs associated with decision nodes specify probabilities conditional on the (unique) corresponding context, the amount of data used to estimate such parameters decreases rapidly with the "depth" of the node. In the case of a multiply connected circuit, deepest nodes with high multiplicity generally do not suffer from data scarcity, thanks to their multiple contexts. Nevertheless, data scarcity can affect single-multiplicity nodes in multiply connected circuits, namely when a deep, singly-connected sub-circuit is present. This justifies the need of a robust statistical learning of the parameters as the one provided by the IDM, even when data is initially abundant. This motivates the following definition of CSDDs.

Definition 5. A *credal sentential decision diagram* (CSDD) is an SDD augmented as follows.

- For each terminal node $\top$, an interval $[l, u]$ is provided such that $0 < l \leq u < 1$. Notation $X : [l, u]$, where X is the variable of the leaf vtree node that $\top$ is normalised for, is consequently adopted. Terminal nodes other than $\top$ appear as they are.
- For each decision node $n = \{(p_i, s_i)\}_{i=1}^k$, a CS $\mathbb{K}_n(P)$ is provided over a variable P, whose states are the interpretations $\langle p_i \rangle$ of the primes p_i's of n. We require that for all $\mathbb{P}(P) \in \mathbb{K}_n(P)$, for each $1 \leq i \leq k$, $\mathbb{P}(\langle p_i \rangle) = 0$ if and only if $s_i = \bot$.

According to the above definition, the CSs associated with the decision nodes assign strictly positive (lower) probability to all the states of P apart from those corresponding to a prime whose sub is $\bot$. Similarly, the intervals $[l, u]$ assigned to terminal nodes $\top$ are also CS specifications (see Section 3.1), while literal terminal nodes have attached degenerate CSs containing the single PMF induced by the same literal when regarded as a PSDD node. It follows that sub-SDDs different from $\bot$ (with their CSs) are in fact sub-CSDDs. Thanks to this requirement, it follows that each assignment of the parameters

respecting the CSDD constraints defines a *compatible* PSDD. Thus, the interpretation of a CSDD is a collection of PSDDs compatible with its constraints. This also gives a semantics for the CSDD CSs, which are regarded as conditional CSs for the variables/events in the associated nodes given a context.

Exactly as a PSDD defines a joint PMF, a CSDD defines a joint CS. Such a CS, called here the *strong extension* of the CSDD and denoted as $\mathbb{K}^r(\boldsymbol{X})$, where r is the root node of the CSDD, is defined as the convex hull of the set of joint PMFs induced by the collection of its compatible PSDDs. By definition of CSDD strong extension and by the Base Theorem for PSDDs, we have the following result.

Theorem 2 *(Base). For each node n of a CSDD, for each instantiation $\boldsymbol{z}$ of its variables $\boldsymbol{Z}$,*

$$\underline{\mathbb{P}}_n(\boldsymbol{z}) > 0 \quad \textit{iff} \quad \boldsymbol{z} \models \langle n \rangle\,, \tag{7}$$

$$\overline{\mathbb{P}}_n(\boldsymbol{z}) = 0 \quad \textit{iff} \quad \boldsymbol{z} \not\models \langle n \rangle\,, \tag{8}$$

where $\underline{\mathbb{P}}_n(\boldsymbol{z}) = \min_{\mathbb{P}(\boldsymbol{Z}) \in \mathbb{K}^n(\boldsymbol{Z})} \mathbb{P}(\boldsymbol{z})$ *and* $\overline{\mathbb{P}}_n(\boldsymbol{z}) = \max_{\mathbb{P}(\boldsymbol{Z}) \in \mathbb{K}^n(\boldsymbol{Z})} \mathbb{P}(\boldsymbol{z})$.

Example 3. Consider the PSDD in Fig. 2. This model can be converted into a CSDD by simply replacing the (precise) learning of the parameters from the data set of consistent observations in Fig. 1 with IDM-based (see Section 3.1) interval-valued estimates. The intervals associated with two of the seven parameters are:

$$\theta_1 = P(\neg X_1 \wedge \neg X_2) \in \left[\frac{n_2 + n_6 + n_9}{n+s}, \frac{n_2 + n_6 + n_9 + s}{n+s}\right] \tag{9}$$

$$\theta_6 = P(X_4 | (\neg X_1 \wedge \neg X_2) \wedge X_3) \in \left[\frac{n_2}{n_2 + n_9 + s}, \frac{n_2 + s}{n_2 + n_9 + s}\right], \tag{10}$$

while the complete set of constraints on the parameters is in the appendix.

As in PSDDs, the CSs of a CSDD are associated with conditional probabilities based on a context, which for "deep" nodes are estimated from small amounts of data consistent with the context; the use of robust estimators such as the IDM allows for CS size to be proportional to the amount of data (see Section 3.1), which leads to more conservative inferences.

Inference in a CSDD is intended as the computation of lower and upper bounds with respect to its strong extension. An important remark is that, as the extreme points of the convex hull of a set also belong to the original set, the extreme points of the strong extension are joint PMFs induced by PSDDs (whose local PMFs are compatible with the local CSs in the CSDD). As a consequence of that, a CSDD encodes the same probabilistic independence relations of a PSDD with the same underlying SDD, but based on the notion of strong independence instead of that of stochastic independence (see Section 3.1). Thus, the variables of a node are *strongly* independent from the ones outside the node when its context is given and feasible. In this sense, the relation between PSDDs and CSDDs retraces that between BNs and credal networks [17]. In the next three sections we address the problem of computing inferences in CSDDs.

5. Marginal inference in CSDDs

Recall that Algorithm 1 computes the probability of a marginal query in a PSDD. Algorithm 3 provides an extension of this procedure to CSDDs, allowing for the computation of lower/upper marginal probabilities. The procedure follows exactly the same scheme based on a topological order. Unlike Algorithm 1, every time a decision node is processed, Algorithm 3 requires the solution of a linear programming task whose feasible region is the CS associated with the decision node.

Algorithm 3 Lower probability of evidence.

input: CSDD, evidence **e**
for $n \leftarrow 1, \ldots, N$ **do**
  $\underline{\pi}(n) \leftarrow 0$
  if n is terminal, $n \neq \bot$ **then**
    $v \leftarrow$ leaf vtree node that n is normalized for
    $\underline{\pi}(n) \leftarrow \underline{\mathbb{P}}_n(\mathbf{e}_v)$
  else
    $((p_i, s_i)_{i=1}^k, \mathbb{K}_n(P)) \leftarrow n$ (decision node)
    $\underline{\pi}(n) \leftarrow \min_{[\theta_1,\ldots,\theta_k] \in \mathbb{K}_n(P)} \sum_{i=1}^k \underline{\pi}(p_i) \cdot \underline{\pi}(s_i) \cdot \theta_i$
  end if
end for
output: $\underline{\mathbb{P}}(\mathbf{e}) \leftarrow \underline{\pi}(N)$

To see why the algorithm properly computes $\underline{\mathbb{P}}(\mathbf{e})$ just regard the output of Algorithm 1 as a symbolic expression of the local probabilities involved in the CSDD local CSs. This is a multi-linear function of these probabilities subject to the linear

constraints defining the CSs. The optimizations with respect to the CSs of the terminal nodes can be done independently of the others, and in any order. Afterwards, the decision nodes whose primes and subs are (already processed) terminal nodes can be safely processed too. In turn, decision nodes whose primes and subs are already processed terminal or decision nodes can be safely processed as well, and so on. Any topological order respects such priorities. The algorithm runs in polynomial time with respect to the SDD size, as it requires the solution of a single linear programming task for each CS of the CSDD. Note that for terminal nodes the optimization is trivial as it only consists in the computation of a lower probability for a CS over a Boolean variable. An analogous procedure can also be defined for upper probabilities.

The intuition above is made formal by the next theorem, stating that the output of Algorithm 3 is indeed the lower bound of a query with respect to the strong extension of the CSDD.

Theorem 3. *Consider a CSDD and a node $n \neq \bot$ normalized for vtree v with variables $\boldsymbol{Z}$. Let $\boldsymbol{e}$ be a partial or total evidence over variables in $\boldsymbol{Z}$:*

$$\underline{\pi}(n) = \underline{\mathbb{P}}_n(\boldsymbol{e}), \tag{11}$$

where $\underline{\pi}(n)$ is the message associated to node n by Algorithm 3.

In the above theorem, there are no restrictions on the topology of the CSDD. Indeed, for any node n, the computation of $\underline{\pi}(n)$ only depends on n's predecessors with respect to a topological order. To make this clear, assume that the CSDD is multiply connected, i.e., that there exist two distinct decision nodes n and n' sharing a sub-CSDD m, say in the i-th, respectively j-th element.[5] Then m is a predecessor of both n and n'. Hence, $\underline{\pi}(m)$ will be already computed when the algorithm is about to compute $\underline{\pi}(n)$ and $\underline{\pi}(n')$, and will appear in the computations of the latter as a factor of the ith, respectively jth coefficient of two LPs over distinct local CSs attached to n, n' respectively. This means that the optimal configuration of m will not be modified in any manner during the optimizations relative to n and n', and so multiply connectedness does not compromise the operations of Algorithm 3.

Example 4. As an example of application of Algorithm 3, assume the counts for the observations of the ten permitted four-pixel images in Fig. 1 are $n_0 = 30$, $n_1 = 8$, $n_2 = 5$, $n_3 = 17$, $n_4 = 3$, $n_5 = 0$, $n_6 = 12$, $n_7 = 2$, $n_8 = 9$, and $n_9 = 14$, this leading to a total of $n = 100$ observations. Using the IDM with $s = 1$, the PSDD in Fig. 2 becomes a CSDD whose parameters are constrained by the following constraints:

$$\theta_1 \in \left[\frac{31}{101}, \frac{32}{101}\right], \theta_2 \in \left[\frac{52}{101}, \frac{53}{101}\right], \theta_3 \in \left[\frac{12}{32}, \frac{13}{32}\right],$$
$$\theta_4 \in \left[\frac{39}{53}, \frac{40}{53}\right], \theta_5 \in \left[\frac{8}{53}, \frac{9}{53}\right], \theta_6 \in \left[\frac{5}{20}, \frac{6}{20}\right], \theta_7 \in \left[\frac{33}{45}, \frac{34}{45}\right].$$

Consider a complete evidence $(X_1 = \bot, X_2 = \bot, X_3 = \bot, X_4 = \top)$. The output of Algorithm 3 corresponds to the following minimization:

$$\min_{\substack{\theta_1 \in \left[\frac{31}{101}, \frac{32}{101}\right] \\ \theta_2 \in \left[\frac{52}{101}, \frac{53}{101}\right]}} \underline{\pi}(24) \cdot \underline{\pi}(25) \cdot \theta_1 + \underline{\pi}(26) \cdot \underline{\pi}(27) \cdot \theta_2 + \underline{\pi}(28) \cdot \underline{\pi}(29) \cdot (1 - \theta_1 - \theta_2), \tag{12}$$

where $\underline{\pi}(24)$ requires no minimization because of the sharp parameters on the arcs of node 24 and has therefore value $\underline{\pi}(0) \cdot \underline{\pi}(1) \cdot 1 = 1$, while

$$\underline{\pi}(25) = \min_{\theta_3 \in \left[\frac{12}{32}, \frac{13}{32}\right]} \underline{\pi}(4) \cdot \underline{\pi}(5) \cdot \theta_3 + \underline{\pi}(6) \cdot \underline{\pi}(7) \cdot (1 - \theta_3). \tag{13}$$

As $\underline{\pi}(6) = 0$ and $\underline{\pi}(4) \cdot \underline{\pi}(5) = 1 \cdot 1 = 1$ the result of the minimization in Equation (13) is $\frac{12}{32}$. It is an easy exercise to verify that both $\underline{\pi}(26)$ and $\underline{\pi}(28)$ are equal to zero. It follows that the output $\underline{\pi}(15)$, i.e. the lower probability $\underline{\mathbb{P}}(X_1 = \bot, X_2 = \bot, X_3 = \bot, X_4 = \top)$ has value $\frac{12}{32} \cdot \frac{31}{101} \simeq 0.1151$. Note that the complete evidence considered in this example corresponds to the four-pixel image in Fig. 1 whose count is n_6, and value returned for the lower probability looks reasonably consistent with the maximum likelihood estimate $\frac{n_6}{n} = \frac{12}{100}$.

[5] The case in which two nodes n and n' share a common sub-CSDD possibly lower than a prime or sub relies on the one treated here.

6. Conditional queries in CSDDs

In the previous section we discussed the computation by Algorithm 3 of lower (or upper) marginal probabilities in a CSDD. This corresponds to a sequence of linear programming tasks whose feasible regions are the CSs of the CSDD processed in topological order, thus taking polynomial time with respect to the diagram size. In this section we show that something similar can also be done for conditional queries.

Let $X = x$ denote the variable and state to be queried, and let $\boldsymbol{e}$ be the available evidence about other variables in a CSDD α rooted at r with variables $\boldsymbol{X}$. The task is to compute the lower conditional probability with respect to the strong extension, i.e.,

$$\underline{\mathbb{P}}(x|\boldsymbol{e}) = \min_{\mathbb{P}(\boldsymbol{X})\in\mathbb{K}^r(\boldsymbol{X})} \frac{\mathbb{P}(x,\boldsymbol{e})}{\mathbb{P}(\boldsymbol{e})}\,. \tag{14}$$

To have $\underline{\mathbb{P}}(x|\boldsymbol{e})$ well defined, we assume $\boldsymbol{e}$ to be consistent with the underlying SDD interpretation $\langle\alpha\rangle$. To see this, assume there is a total instantiation of $\boldsymbol{X}$ extending $\boldsymbol{e}$. Then, given an extreme point $\mathbb{P}(\boldsymbol{X})$ of the strong extension $\mathbb{K}^r(\boldsymbol{X})$, the Base Theorem for PSDDs tells us that $\mathbb{P}(\boldsymbol{x}) > 0$ if and only if $\boldsymbol{x} \models \langle\alpha\rangle$. This immediately yields that the denominator in the right-hand side of Equation (14) is positive for each extreme point of the strong extension $\mathbb{K}^r(\boldsymbol{X})$ if and only if $\boldsymbol{e}$ is consistent with $\langle\alpha\rangle$.

Note also that if $\boldsymbol{e} \models \neg x$, then $\mathbb{P}(x,\boldsymbol{e}) = 0$, and similarly if $\boldsymbol{e} \models x$, then $\mathbb{P}(\neg x,\boldsymbol{e}) = 0$. Otherwise both $(x,\boldsymbol{e}) = (x,\boldsymbol{e}_{\overline{v}})$ and $(\neg x,\boldsymbol{e}) = (\neg x,\boldsymbol{e}_{\overline{v}})$, and therefore $\mathbb{P}(x,\boldsymbol{e}) = \mathbb{P}(x,\boldsymbol{e}_{\overline{v}})$ and $\mathbb{P}(\neg x,\boldsymbol{e}) = \mathbb{P}(\neg x,\boldsymbol{e}_{\overline{v}})$, where v is the leaf node with variable X in the vtree the CSDD is normalized for. In the following we might therefore assume $\boldsymbol{e}_{\overline{v}} = \boldsymbol{e}$.

The task in Equation (14) corresponds to the linearly constrained minimization of a (multilinear) fractional function of the probabilities. This prevents a straightforward application of the same approach considered in the previous section. Thus, we consider instead a decision version of the optimization task in Equation (14), i.e., deciding whether or not the following inequality is satisfied for a given $\mu \in [0,1]$:

$$\underline{\mathbb{P}}(x|\boldsymbol{e}) > \mu\,. \tag{15}$$

As for the algorithm in [27], an algorithm able to solve Equation (15) for any $\mu \in [0,1]$ inside a bracketing scheme linearly converges to the actual value of the lower probability.

As $\mathbb{P}(x|\boldsymbol{e}) + \mathbb{P}(\neg x|\boldsymbol{e}) = 1$ for each $\mathbb{P}(\boldsymbol{X}) \in \mathbb{K}^r(\boldsymbol{X})$, and assuming that $\mathbb{P}(\boldsymbol{e}) > 0$, Equation (15) holds if and only if the following inequality holds:

$$\min_{\mathbb{P}(\boldsymbol{X})\in\mathbb{K}^r(\boldsymbol{X})} [(1-\mu)\mathbb{P}(x,\boldsymbol{e}) - \mu\mathbb{P}(\neg x,\boldsymbol{e})] > 0\,. \tag{16}$$

In order to define an algorithm solving the task of deciding whether or not inequality (16) is satisfied for a given $\mu \in [0,1]$ we need to define the following auxiliary quantities.

(i) For a given value of μ and any node $n \neq \bot$ normalized for vtree node v:

$$\underline{\rho}_n(\mu) := (1-2\mu)\cdot\underline{\mathbb{P}}_n(\boldsymbol{e}_v)\,. \tag{17}$$

(ii) For a given value of μ and a terminal node $n \neq \bot$:

$$\Lambda_n(\mu) := \begin{cases} \lambda_n(\mu) & \text{if } X \text{ occurs in } n \\ \underline{\rho}_n(\mu) & \text{otherwise}\,, \end{cases} \tag{18}$$

with

$$\lambda_n(\mu) := \min \begin{Bmatrix} (1-\mu)\underline{\mathbb{P}}_n(x) - \mu\overline{\mathbb{P}}_n(\neg x), \\ (1-\mu)\overline{\mathbb{P}}_n(x) - \mu\underline{\mathbb{P}}_n(\neg x) \end{Bmatrix}, \tag{19}$$

where the lower and upper probabilities in the above expression are those associated with the bounds in the CS specification for $X = \top$ and the other values are obtained by the conjugacy relation $\underline{\mathbb{P}}(x) = 1 - \overline{\mathbb{P}}(\neg x)$.

(iii) For any node n normalized for vtree node v, for $z \in \mathbb{R}$:

$$\underline{\sigma}_n(z) = \begin{cases} \overline{\mathbb{P}}_n(\boldsymbol{e}_v) & \text{if } z < 0 \\ \underline{\mathbb{P}}_n(\boldsymbol{e}_v) & \text{otherwise}\,, \end{cases} \tag{20}$$

for $n \neq \bot$, while if $n = \bot$ we set $\underline{\sigma}_n(z) = 0$ for any $z \in \mathbb{R}$.

We are ready to define Algorithm 4.

The following result proves the correctness of Algorithm 4 for singly connected CSDDs.

Algorithm 4 Lower conditional probability.

input: CSDD, μ, $X = x$, $\boldsymbol{e}$
for $n \leftarrow 1, \ldots, N$ **do**
  $\underline{\pi}(n) \leftarrow 0$
  $v \leftarrow$ vtree node that n is normalized for
  if node n is terminal, $n \neq \bot$ **then**
    $\underline{\pi}(n) \leftarrow \Lambda_n(\mu)$ as in Eq. (18)
  else
    $((p_i, s_i)_{i=1}^k, \mathbb{K}_n(P)) \leftarrow n$ (decision node)
    if X occurs in v **then**
      if X occurs in v^l **then**
        $u_i \leftarrow p_i$ and $w_i \leftarrow s_i$ for $1 \leq i \leq k$
      else if X occurs in v^r **then**
        $u_i \leftarrow s_i$ and $w_i \leftarrow p_i$ for $1 \leq i \leq k$
      end if
      $\underline{\pi}(n) \leftarrow \min_{[\theta_1,\ldots,\theta_k] \in \mathbb{K}_n(P)} \sum_{i=1}^k \underline{\pi}(u_i) \cdot \underline{\sigma}_{w_i}(\underline{\pi}(u_i)) \cdot \theta_i$
      with $\underline{\sigma}$ as in Eq. (20)
    else
      $\underline{\pi}(n) \leftarrow \underline{\rho}_n(\mu)$
    end if
  end if
end for
output: $\text{sign}[\underline{\mathbb{P}}(x|\boldsymbol{e}) - \mu] \leftarrow \text{sign}[\underline{\pi}(N)]$

Theorem 4. *Consider a singly connected CSDD and a node $n \neq \bot$ normalized for vtree node v, whose variables are $\boldsymbol{X}$. For any instantiation x of a single variable $X \in \boldsymbol{X}$ and any coherent evidence $\boldsymbol{e}$ over some or all of the remaining variables,*

$$\underline{\pi}(n) = \min_{\mathbb{P}(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})} \left[(1-\mu)\mathbb{P}(x, \boldsymbol{e}) - \mu\mathbb{P}(\neg x, \boldsymbol{e})\right], \tag{21}$$

where $\underline{\pi}(n)$ is the message of node n in Algorithm 4.

Observe that, both for terminal and decision nodes whose variables do not contain the queried variable X, the value $\underline{\pi}(n)$ does not really matter, meaning that it does not affect the computation of the messages of the nodes processed after them. Indeed, consider a node n' (terminal or decision) appearing as prime or sub in a decision node n, and assume X occurs in n but not in n'. Then the message $\underline{\pi}(n')$ will not contribute to $\underline{\pi}(n)$, but $\underline{\sigma}_{n'}(\underline{\pi}(n''))$ will, instead, where n'' is the node that, together with n', forms an element of n. An implementation of Algorithm 4 might therefore simply set $\underline{\pi}(n) = 0$ for each node n in which the queried variable does not occur, in order to avoid useless computations.

The procedure described by Algorithm 4 requires the solution of a number of linear programming tasks, whose feasible regions are the CSs associated with the CSDD, equal to the number of decision nodes. The computation of the coefficients of the objective function in these tasks requires a call of Algorithm 3 for each optimization variable to compute the quantities in Equation (20). Note also that, for each decision node $n = (\{(p_i, s_i)\}_{i=1}^k, \mathbb{K}_n(P))$ the optimization in the recursive call is performed before the one in Equation (20). As discussed before, by iterated calls of Algorithm 4, we can therefore compute lower conditional queries in polynomial time in singly connected CSDDs.

Example 5. Let us demonstrate how Algorithm 4 works in practice by considering the same CSDD, with the same training data, as in the Example 4. Consider the query $X_1 = \top$ given evidence $(X_2 = \bot, X_3 = \bot, X_4 = \top)$. Take a generic $\mu \in [0, 1]$. As the queried variable is the left-most variable in the variables ordering induced by the vtree in Fig. 3a, the output of Algorithm 4 is the result of the following minimization:

$$\min_{\substack{\theta_1 \in \left[\frac{31}{101}, \frac{32}{101}\right] \\ \theta_2 \in \left[\frac{52}{101}, \frac{53}{101}\right]}} \underline{\pi}(24) \cdot \underline{\sigma}_{25}(\underline{\pi}(24)) \cdot \theta_1 + \underline{\pi}(26) \cdot \underline{\sigma}_{27}(\underline{\pi}(26)) \cdot \theta_2 + \underline{\pi}(28) \cdot \underline{\sigma}_{29}(\underline{\pi}(28)) \cdot (1 - \theta_1 - \theta_2). \tag{22}$$

Computing $\underline{\pi}(24)$ requires no minimization because of the sharp parameters on the arcs of node 24 and its value is $\underline{\pi}(0) \cdot \underline{\sigma}_1(\underline{\pi}(0)) \cdot 1$. As node 0 is a terminal node containing the queried variable, $\underline{\pi}(0) = \lambda_0(\mu)$. The latter quantity is equal to $-\mu$ because the query $X_1 = \top$ does not agree with node 0 whose literal is $\neg X_1$. Since $\underline{\pi}(0) < 0$, $\underline{\sigma}_1(\underline{\pi}(0)) = \overline{\mathbb{P}}_1(X_2 = \bot) = 1$. Hence, $\underline{\pi}(24) = -\mu < 0$, and $\underline{\sigma}_{25}(\underline{\pi}(24)) = \overline{\mathbb{P}}_{25}(X_3 = \bot, X_4 = \top) = \frac{13}{32}$. The value of $\underline{\pi}(26)$ is the result of the following minimization:

$$\min_{\theta_4 \in \left[\frac{39}{53}, \frac{40}{53}\right]} \underline{\pi}(8) \cdot \underline{\sigma}_9(\underline{\pi}(8)) \cdot \theta_4 + \underline{\pi}(10) \cdot \underline{\sigma}_{11}(\underline{\pi}(10)) \cdot (1 - \theta_4). \tag{23}$$

Both node 8 and node 10 contain the queried variable, hence $\underline{\pi}(8) = \lambda_8(\mu) = (1 - \mu)$ and $\underline{\pi}(10) = \lambda_{10}(\mu) = -\mu$. Accordingly to the signs of the latter, $\underline{\sigma}_9(\underline{\pi}(8)) = \underline{\mathbb{P}}_9(X_2 = \bot) = 1$ and $\underline{\sigma}_{11}(\underline{\pi}(10)) = \overline{\mathbb{P}}_{11}(X_2 = \bot) = 0$. Hence, $\underline{\pi}(26) = (1 - \mu) \cdot \frac{39}{53} > 0$. Moreover, $\underline{\sigma}_{27}(\underline{\pi}(26))$ is equal to $\underline{\mathbb{P}}_{27}(X_3 = \bot, X_4 = \top)$ and hence corresponds to:

$$\min_{\theta_5 \in \left[\frac{8}{53}, \frac{9}{53}\right]} \underline{\mathbb{P}}_{12}(X_3 = \bot) \cdot \underline{\mathbb{P}}_{13}(X_4 = \top) \cdot \theta_5 + \underline{\mathbb{P}}_{14}(X_3 = \bot) \cdot \underline{\mathbb{P}}_{15}(X_4 = \top) \cdot (1 - \theta_5)$$

$$= \min_{\theta_5 \in \left[\frac{8}{53}, \frac{9}{53}\right]} \frac{33}{45} \cdot (1 - \theta_5) = \frac{33}{45} \cdot \frac{44}{53} = \frac{484}{795} .$$

One can easily verify that $\underline{\pi}(28) = 0$. Thus, the minimization of Equation (22) rewrites as the following linear programming task:

$$\min_{\substack{\theta_1 \in \left[\frac{31}{101}, \frac{32}{101}\right] \\ \theta_2 \in \left[\frac{52}{101}, \frac{53}{101}\right]}} -\mu \cdot \frac{13}{32} \cdot \theta_1 + (1 - \mu) \cdot \frac{39}{53} \cdot \frac{484}{795} \cdot \theta_2 , \tag{24}$$

whose optimum is a numerical zero for $\mu \simeq 0.657$.

The assumption of singly connected topology is crucial for the proof of Theorem 4. Yet, nothing prevents us from applying Algorithm 4 to a multiply connected CSDD. Considered the last iteration of the algorithm leading to the value of μ for which the output of Algorithm 4 is a numerical zero. The CSs associated with nodes of multiplicity higher than one have been used more than once as the feasible region of a linear programming tasks during the recursive calls of the algorithm. If the optima of those linear programming tasks corresponds to different extreme points of the same CS, we might have that an outer approximation has been introduced, i.e., the estimate of the lower (upper) probability returned by the algorithm is smaller (greater) than the exact one. Vice versa, if this is not the case, we might conclude that the algorithm returned an exact inference. To check this, we only need to store the extreme points of the CSs leading to the optima of the different linear programming tasks executed by the algorithm. In other words, no additional computational costs are required to decide whether or not the output of the algorithm is exact. Moreover, if an approximation has been introduced, a simple brute-force approach to the computation of the exact solution consists in running the same inferential task in the PSDDs compatible with the input CSDD and such that: (i) the PMFs of the nodes with multiplicity one and of the nodes with multiplicity more than one in case all the linear programming tasks have the same optimum are just the extreme points of the CSs that led to the optimum; (ii) the PMFs for the other nodes are any possible extreme points of the CSs, each with its multiplicity. This represents a brute-force algorithm involving a number of PSDD inference tasks exponential in the number of credal sets such as in (ii). These ideas are clarified by the following example.

Example 6. Consider a CSDD over the PSDD structure in Fig. 4, whose CSs are all *precise* (i.e., made of a single PMF) apart from specifications for each node except for node 3 for which we assume a CS induced by the constraint $\theta_1 \in [l, u]$. Consider the conditional query $X_1 = \top$ given evidence $X_3 = \top$. For a given $\mu \in]0, 1[$, it is straightforward to verify that the messages $\underline{\pi}(n)$ of terminal nodes $n \in \{0, 1, 2, 3, 4, 7, 10, 12, 14\}$ are all equal to zero, while $\underline{\pi}(5) = \underline{\pi}(9) = 1 - \mu$ and $\underline{\pi}(6) = \underline{\pi}(8) = -\mu$. Consider now the decision nodes 11 and 13, sharing node 4. We have:

$$\underline{\pi}(11) = \underline{\pi}(5) \cdot \underline{\sigma}_4(\underline{\pi}(5)) \cdot 1 + \underline{\pi}(6) \cdot \underline{\sigma}_7(\underline{\pi}(6)) \cdot 0 = (1 - \mu) \cdot \underline{\mathbb{P}}_4(X_3 = \top) , \tag{25}$$

and

$$\underline{\pi}(13) = \underline{\pi}(8) \cdot \underline{\sigma}_4(\underline{\pi}(8)) \cdot 1 + \underline{\pi}(9) \cdot \underline{\sigma}_1 0(\underline{\pi}(9)) \cdot 0 = -\mu \cdot \overline{\mathbb{P}}_4(X_3 = \top) . \tag{26}$$

The two optimizations in Equations (25) and (26) with respect to θ_1 give divergent values, i.e., $\theta_1 = l$ in the first case and $\theta_1 = u$ in the second. This is not consistent with the definition of strong extension in Section 4 and it would lead to an approximate value of the lower probability smaller than the exact one because of fewer constraints.

7. Robustness of MAP inference in PSSDs

CSDD can be also intended as tool for sensitivity analysis in PSDDs. Here we show how to evaluate the *robustness* of a MAP inference in a PSDD. Let us first apply Algorithm 2 to a PSDD rooted at r with evidence $\mathbf{e}$. We might ask ourselves whether or not the resulting configuration is sensitive to variations in the PSDD parameters. In order to do so, we also consider a CSDD the PSDD is consistent with. If all the PSDDs consistent with this CSDD have the same optimal configuration, and hence this is equal to the one obtained in the original PSDD, we say that the MAP inference is *robust*. The following definition formalizes this idea.

Definition 6. Given a PSDD r over variables $\{\mathbf{X}, \mathbf{E}\}$ - with $\mathbf{X}$ and $\mathbf{E}$ disjoint - and an evidence $\mathbf{e}$ over variables $\mathbf{E}$, $\mathbf{x}^* := \arg\max_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} \mathbb{P}_r(\mathbf{x}, \boldsymbol{e})$ is robust with respect to a CSDD with which r is consistent if:

$$\max_{\mathbf{x} \neq \mathbf{x}^*} \max_{\mathbb{P} \in \mathbb{K}^r} \frac{\mathbb{P}(\mathbf{x}, \mathbf{e})}{\mathbb{P}(\mathbf{x}^*, \mathbf{e})} < 1 . \tag{27}$$

If $(\mathbf{x}^*, \mathbf{e})$ is inconsistent with r, we say that the inference is not robust by definition and gives to the maximum in Equation (27) a reference value one.

Algorithm 5 is a subroutine used to decide the robustness of a MAP instance. It takes as input a CSDD rooted at r over variables $\{\mathbf{X}, \mathbf{E}\}$ – with $\mathbf{X}$ and $\mathbf{E}$ disjoint – and an evidence $\mathbf{e}$ over variables $\mathbf{E}$, and computes $\max_{\mathbf{x}\in\boldsymbol{\mathcal{X}}} \overline{\mathbb{P}}_r(\mathbf{x}, \boldsymbol{e})$.

Algorithm 5 "Credal MAP".

input: CSDD r, evidence $\mathbf{e}$
for $n \leftarrow 1, \ldots, N$ **do**
  $M(n) \leftarrow 0$
  if node n is terminal **then**
    $v \leftarrow$ leaf vtree node that n is normalized for
    if $var(v) \in \mathbf{X}$ **then**
      if $n \in \{X, \neg X\}$ **then**
        $M(n) \leftarrow 1$
      else if $n = (X, [l, u])$ **then**
        $M(n) \leftarrow \max\{u, 1-l\}$
      end if
    else
      $M(n) \leftarrow \overline{P}_n(\mathbf{e}_v)$
    end if
  else
    $((p_i, s_i)_{i=1}^k, \mathbb{K}_n(P)) \leftarrow n$ (decision node)
    $M(n) \leftarrow \max_{1\le i\le k} \overline{\theta_i} \cdot M(p_i) \cdot M(s_i)$ with $\overline{\theta_i} := \max_{\mathbb{K}_n} \theta_i$
  end if
end for
output: $M(N)$

The following theorem gives a semantics for the output of Algorithm 5.

Theorem 5. *Consider a CSDD and a node $n \neq \bot$ normalized for vtree node v whose variables are $\{\boldsymbol{X}, \boldsymbol{E}\}$, with $\boldsymbol{X}$ and $\boldsymbol{E}$ disjoint. Let $\boldsymbol{e}$ be a total evidence over variables $\boldsymbol{E}$. Then:*

$$M(n) = \max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \overline{\mathbb{P}}_n(\boldsymbol{x}, \boldsymbol{e}) \,. \tag{28}$$

Algorithm 6 is used to decide the robustness of a MAP inference $\mathbf{x}^* \in \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{P}_r(\boldsymbol{x}, \boldsymbol{e})$ in PSDD r in the following way. Following a topological order, each node n is processed and gives message $V(n)$, which is a relaxed version of the left-hand side of Equation (27), in which we do not require the configurations $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ to be distinct from $\boldsymbol{x}^*$ (with the adequate restrictions to n's variables). Observe that the message of decision nodes n not realized by $(\boldsymbol{x}_v^*, \boldsymbol{e}_v)$ is 0. In fact, this value does not matter: the contribution of such nodes will be taken into account - as Credal-MAP message - when processing the first higher decision node consistent with (the adequate restriction of) $(x^*, \boldsymbol{e})$.

Because of the previously relaxed constraint, the message of the root $V(r)$ is greater or equal than 1. If $V(r) > 1$, we can conclude that $\mathbf{x}^*$ is not robust. If $V(r) = 1$, we need to re-take into account the constraint. In order to do so, we observe:

- if $\mathbf{x}^*$ is the only configuration realizing the maximum, we can state its robustness;
- if $\mathbf{x}^*$ is between several configurations realizing the maximum, we can say that it is *weakly* robust;
- if $\mathbf{x}^*$ does not realize the maximum, we conclude that it is not robust.

Note that Equation (27) holds if and only if the first situation occurs.

The following theorem states the correctness of Algorithm 6 for singly connected CSDDs.

Theorem 6. *Let r be a singly connected CSDD over variables $\{\boldsymbol{X}, \boldsymbol{E}\}$, with $\boldsymbol{X}$ and $\boldsymbol{E}$ disjoint. Consider an evidence $\boldsymbol{e}$ over variables $\boldsymbol{E}$ and an instance $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{P}_r(\boldsymbol{x}, \boldsymbol{e})$ obtained by applying Algorithm 2 to a consistent PSDD. For each node $n \neq \bot$ in r normalized for vtree node v:*

$$V(n) = \max_{\boldsymbol{x}_v \in val(\boldsymbol{X}_v)} \max_{\mathbb{P}\in\mathbb{K}^n} \frac{\mathbb{P}(\boldsymbol{x}_v, \boldsymbol{e}_v)}{\mathbb{P}(\boldsymbol{x}_v^*, \boldsymbol{e}_v)} \,. \tag{29}$$

The motivations for which we are not in measure to state the theorem for general CSDDs are analogous to the ones for conditional inference. In the induction step of the previous proof, in the case of $i \neq j$, we perform a maximization on the numerator and a minimization on the denominator, this being possible because nodes on the numerator and nodes on the denominator have distinct CSs. Nevertheless, this does not prevent the algorithm from selecting several distinct optimal sub-configurations in the case of a multiple node possibly shared by p_i and p_j, or s_i and s_j, when the CSDD is multiply connected. Thus, exactly as in the case of conditional queries, we obtain an outer approximation meaning

Algorithm 6 Robustness.

input: CSDD r, evidence $\mathbf{e}$, $\mathbf{x}^* \in arg\max_{\mathbf{x} \in val(\mathbf{X})} \mathbb{P}_r(\mathbf{x}, \mathbf{e})$
for $n \leftarrow 1, \ldots, N$ **do**
  $V(n) \leftarrow 1$
  $v \leftarrow$ leaf vtree node that n is normalized for
  if node n is terminal **then**
    if $n = (X : [l, u])$ **then**
      $V(n) \leftarrow \max\{1, \frac{1-l}{l}\}$ when $\mathbf{x}^*_v = \top$
      $V(n) \leftarrow \max\{1, \frac{u}{1-u}\}$ when $\mathbf{x}^*_v = \bot$
    end if
  else
    $(\{(p_i, s_i)\}_{i=1}^k, \mathbb{K}_n(P)) \leftarrow n$ (decision node)
    if $x^*_v \mathbf{e}_v \models \langle n \rangle$ **then**
      $V(n) \leftarrow \max\{V(p_j) \cdot V(s_j), \max_{1 \le i \le k, i \neq j} U_{i,j}\}$ // j is the unique index such that $\mathbf{x}^*_{v,l}\mathbf{e}_{v,l} \models \langle p_j \rangle$ and $U_{i,j} := \max_{\mathbb{K}_n} \frac{\theta_i \cdot M(p_i) \cdot M(s_i)}{\theta_j \cdot \underline{P}_{p_j}(\mathbf{x}^*_{v,l}, \mathbf{e}_{v,l}) \cdot \underline{P}_{s_j}(\mathbf{x}^*_{v,r}, \mathbf{e}_{v,r})}$
    end if
  end if
end for
output: $V(N)$

Fig. 5. Digits represented by a seven-segment display.

that the output of Algorithm 6 might be greater than the left-hand side of Equation (27). In other words, for multiply connected models, if the algorithm says that the configuration is robust we are certain, while it might be the case that the algorithm says that the configuration is not robust, while this is not the case. This might be therefore intended as a conservative approximation. Finally, exactly as in the conditional case, we might decide whether or not the algorithm returned an approximation by simply inspecting the extreme points of the CSs with multiplicity higher than one leading to the optima of the linear programs solved during the execution of the algorithm and, in case of approximation, run a brute-force algorithm exponential in the number of CSs for which different tasks gave different optimal extreme points.

8. Experiments

As a first application of the algorithms derived in the previous section, we consider a simple machine learning task involving logical constraints over the model variables. The problem consists in the identification of the digit depicted by a seven-segment display (Fig. 5), whose segments might occasionally fail to turn on. More specifically, given an input digit to be displayed, the control unit activates the corresponding set of segments in the display; each segment can however fail to be switched on independently with an identical probability. We note that while this scenario is relatively simple, it can easily be extended to more complex and realistic scenarios involving a large number of components/devices, whose interdependence is described as a logical function, and whose probability of failures is interconnected in a complicated way.

Our setup can be described by fourteen Boolean variables: say that $\boldsymbol{X} := (X_1, \ldots, X_7)$ are the *hidden* states of the segments as decided by the control unit, and $\boldsymbol{O} := (O_1, \ldots, O_7)$ are the *observable* states of the segments as depicted in the display. Let us also assume that the true state of these Boolean variables corresponds to the segment on.

We create synthetic data as follows. Given digit j, the corresponding configuration of $\boldsymbol{X}$ is provided by the formula $\delta_j(\boldsymbol{X})$ as in Table 1. Then, for each $i = 1, \ldots, 7$, if X_i is false, we also set O_i false, while if X_i is true, O_i might be false with a given failure probability p_f. Such mechanism obeys the formula:

$$\phi := \wedge_{i=1}^{7}(O_i \rightarrow X_i) \wedge \left(\vee_{j=0}^{9}\delta_j(X_1, \ldots, X_7)\right). \tag{30}$$

Given formula ϕ in Equation (30), we use the algorithm proposed in [28] to build an SDD α normalized for a vtree such that, for each $i = 1, \ldots, 7$, the pair (X_i, O_i) corresponds to a pair of leaves with the same parent and with a so-called *balanced* shape. The resulting SDD has a multiply connected structure, 128 nodes (82 of them decision nodes) and maximum number of elements for decision node equal to eight.

Given a training data set $\mathcal{D}$ of size d, generated according to the above described procedure, we can obtain from α a PSDD or a CSDD. In the first case we use a Bayesian procedure, with Perks' prior and equivalent sample size $s = 1$, to learn PMFs associated with the decision nodes and the non-bot terminal nodes. In the second case, IDM with the same equivalent sample size is used to learn the CSs.

As a rival setup we consider a *hidden Markov model* (HMM) whose hidden variables are those in $\boldsymbol{X}$, while the observations are those in $\boldsymbol{O}$. The model is trained from the same data set $\mathcal{D}$ and with the same prior as the PSDD. A credal extension of HMMs, perfectly analogous to the one we presented here for PSDDs, has been proposed in [29]. Thus, we can also quantify the HMM parameters as CSs obtained by IDM with the same equivalent sample size. We refer to this model as IHMM, while HMM is its precise counterpart.

Table 1
Digits configuration as disjunctive formulae.

j	$\delta_j(\boldsymbol{X})$
0	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge \neg X_7$
1	$\neg X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge \neg X_7$
2	$X_1 \wedge X_2 \wedge \neg X_3 \wedge X_4 \wedge X_5 \wedge \neg X_6 \wedge X_7$
3	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge X_7$
4	$\neg X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$
5	$X_1 \wedge \neg X_2 \wedge X_3 \wedge X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$
6	$X_1 \wedge \neg X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge X_7$
7	$X_1 \wedge X_2 \wedge X_3 \wedge \neg X_4 \wedge \neg X_5 \wedge \neg X_6 \wedge \neg X_7$
8	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge \neg X_7$
9	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5 \wedge X_6 \wedge X_7$
0	$X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge \neg X_5 \wedge X_6 \wedge X_7$

Given a test instance $(\boldsymbol{x}', \boldsymbol{o}')$, generated by the same mechanism discussed for the training set, we therefore have four different models to perform reasoning. As a first task, we predict, given the observation $\boldsymbol{o}'$, the most probable configuration of X'_i for each $i = 1, \ldots, 7$. In the PSDD, this is prediction is driven by the conditional inference $P(X'_i = 1|\boldsymbol{o}')$. The same can be done with the HMM by the classical *filtering* algorithm (we create a different HMM for each i such that X_i and O_i are always the last elements of the sequence). For the CSDD, Algorithm 4 is used instead to compute posterior intervals $[\underline{P}(X'_i = 1|\boldsymbol{o}'), \overline{P}(X'_i = 1|\boldsymbol{o}')]$, while the same task can be solved in polynomial time also in IHMMs by the (credal) filtering algorithm proposed in [29]. With 0/1 losses, the rule to decide whether or not the segment X'_i is on according to a PSDD or HMM is simply whether or not the probability of the true state is larger than half, the segment being off otherwise. For CSDDs and IHMMs, we say that the segment is certainly on, if the lower conditional probability is more than half, and certainly off if the upper probability is less than half. If none of the two above cases is satisfied, we say that we are in a condition of *indecision* between the two options. This is an example of so-called *credal classifier* [30], which suspends the judgement about the actual state of the segment when the available information is not sufficient to take a determinate decision.

In summary, given $\boldsymbol{o}'$, we classify each segment separately by using: (i) PSDDs and HMMs as standard classifiers, whose performance is described by the accuracy, i.e., the percentage of segments whose state was properly recognized; (ii) CSDDs and IHMMs as credal classifiers, whose performance is described by the u_{80} utility-based performance measure, which is commonly used to evaluate the performance of credal classifiers as it balances the quality of the prediction and the lack of informativeness associated to indeterminate classifications and it is considered a proper measure to compare the performance of credal classifiers against the accuracy of a standard classifier [31].

In our experiments we consider training sets of size $d \in \{10, 15, 20, 50, 100\}$ and test the four models trained with these data with a test set of size $d' = 140$. Different failure probabilities $p_f \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$ are also considered.

The CSDD inference algorithms have been implemented by the authors in Python together with the necessary data structures.[6] The PySDD library was used to build the SDDs associated with a formula.[7] The PyPSDD library was used instead to validate the consistency between PSDDs and CSDDs.[8] The iHMM library was finally used instead for experiments with HMMs/IHMMs.[9]

Fig. 6 depicts five plots showing the accuracies of the four different models as a function of p_f for different training set sizes d. The behaviour is clear PSDDs/CSDDs models outperform HMMs/IHMMs most of the times, with the differences being typically narrower for low failure probabilities. This is expected and the gap between the two models should be intended as the effect of the additional information about the logical constraints in Equation (30), that is not available to the HMMs/IHMMs. The smaller gap for low failure probabilities can be also explained by noticing that the emission term $P(O_j|X_j)$ involved in the parametrization of HMMs/IHMMs takes almost diagonal form for low failure probabilities and, in these cases, the observation of O_j induces a high probability for the same state of X_j, thus making irrelevant the effect of the logical constraints. Moreover, we notice that the CSDD tends to outperform the PSDD for larger failure probabilities. This is also expected: increasing the noise level in the data promptly induces a degradation of the PSDD accuracy, while the CSDD is able to contain that effect by allowing for indeterminate classifications of some segments.

Credal classifiers are typically used as preprocessing systems able to distinguish easy-to-classify instances for which the output of the standard method is considered sufficiently reliable, from the hard-to-classify ones, for which other dedicated and typically more demanding/expensive techniques should be invoked. Such a separation is naturally provided by the classifier, as it corresponds to the difference between the instances for which the output of the classifier is determinate and the other ones. A typical description of such discriminative power is the difference between the accuracies of the precise counterpart of a credal classifier on these two sets of instances. In Fig. 7, we plot the so-called *determinate* and

[6] https://github.com/alessandroantonucci/pycsdd.
[7] https://github.com/wannesm/PySDD.
[8] https://github.com/art-ai/pypsdd.
[9] https://github.com/denismaua/ihmm.

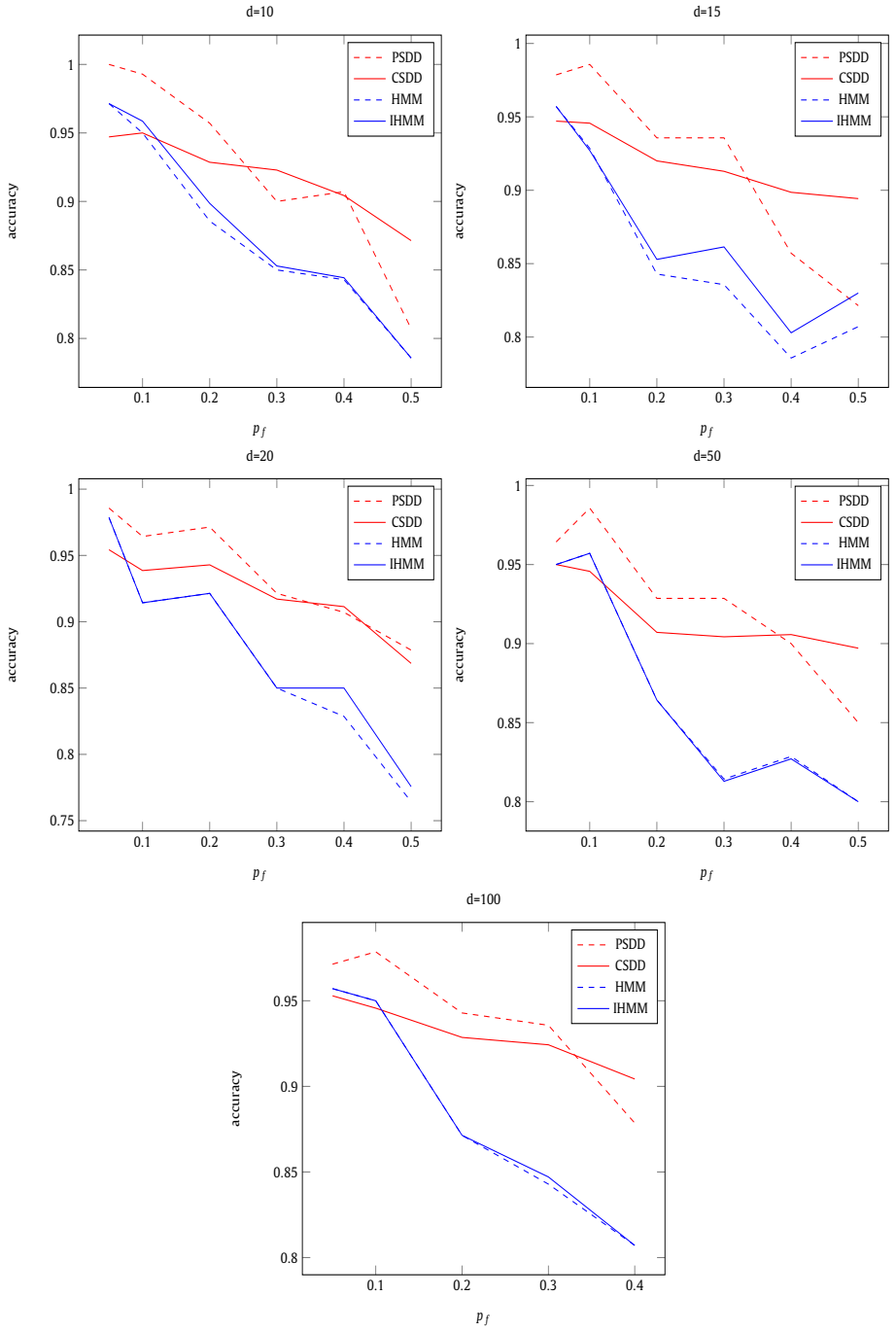

Fig. 6. Accuracies.

indeterminate accuracies of the PSDD, i.e., the accuracy of the PSDD on the instances (i.e., segments) for which the credal classifier was determinate or indeterminate. As expected, the CSDD is properly able to distinguish these two sets and keeps a level of accuracy very close to one even for high perturbation levels (the perturbation only affecting the determinacy, i.e., the percentage of determinate classifications).

Finally, for a validation of Algorithm 6, we perform an analysis analogous to that in Fig. 7 but at the level of joint configuration of the hidden variables corresponding to a particular digit. In practice, we compute the MAP configuration of $\boldsymbol{X} = \boldsymbol{x}^*$ given $\boldsymbol{o}'$ in the PSDD and use Algorithm 6 to check whether or not the configuration was robust. The corresponding determinate and indeterminate, joint, accuracies are reported in Fig. 8 only for $d \geq 20$ as for lower training set size the amount of detected digits is very low in both cases. As expected the behaviour is analogous to that in Fig. 7.

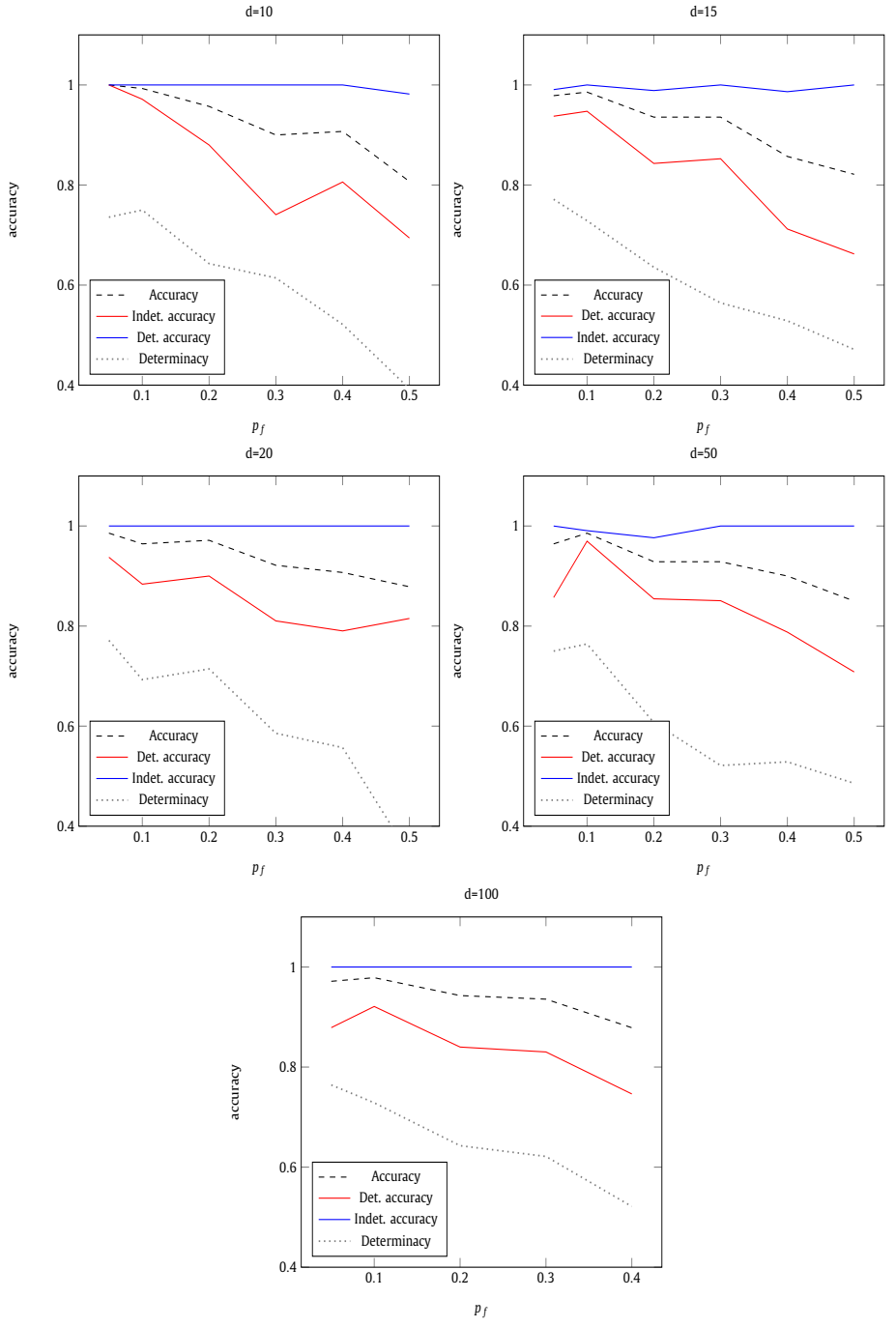

Fig. 7. PSDD determinate vs. indeterminate accuracies.

9. Conclusions

We have introduced a new class of imprecise probabilistic graphical models based on a credal set extension of *probabilistic sentential diagrams*. Three efficient algorithms for marginal, conditional and MAP queries are derived. The first algorithm is exact for any topology, while the second and the third might induce a conservative approximation in the multiply connected case. Yet, a fast procedure to test whether or not an approximation has been also derived. An empirical validation on a synthetic setup shows that the credal extension allows to properly distinguish between easy-to-classify and hard-to-classify instances. Regarding the multiply connected case, whether or not for conditional queries and for the robustness of a MAP task, exact inferences can be efficiently computed remains an open question to be addressed as a future work.

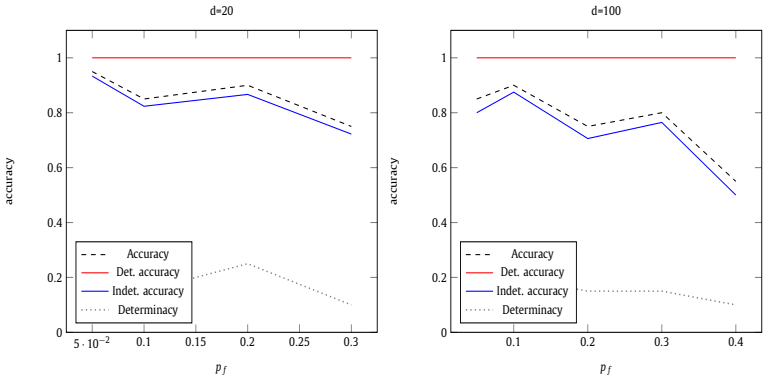

Fig. 8. PSDD determinate vs. indeterminate joint accuracies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proofs

Proof of Theorem 1. If r is a terminal PSDD, it is easy to see the correctness of the algorithm. Suppose that r is a decision node, $r = (p_i, s_i, \theta_i)_{i=1}^k$. For a given $\mathbf{x} \in val(\mathbf{X})$, $\mathbf{xe}$ is a total instantiation of its variables. By definition of PSDDs distribution, $\mathbb{P}_r(\mathbf{xe}) = \sum_{i=1}^k \mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l) \cdot \mathbb{P}_{s_i}(\mathbf{x}_l\mathbf{e}_r) \cdot \theta_i$. Now, remember that for each $\mathbf{x}$, $\mathbf{x}_l\mathbf{e}_l$ realizes a unique prime, so this maximum is of the form $\mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l) \cdot \mathbb{P}_{s_i}(\mathbf{x}_l\mathbf{e}_r) \cdot \theta_i$ for a unique $1 \leq i \leq k$. Hence,

$$\begin{aligned}\max_{\mathbf{x}\in val(\mathbf{X})} \mathbb{P}_r(\mathbf{x}, \boldsymbol{e}) &= \max_{1\leq i\leq k} \max_{\mathbf{x}\in val(\mathbf{X})} \mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l) \cdot \mathbb{P}_{s_i}(\mathbf{x}_r\mathbf{e}_r) \cdot \theta_i \\ &= \max_{1\leq i\leq k} \theta_i \cdot [\max_{\mathbf{x}_l\in val(\mathbf{X}_l)} \mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l)] \cdot [\max_{\mathbf{x}_r\in val(\mathbf{X}_r)} \mathbb{P}_{s_i}(\mathbf{x}_r\mathbf{e}_r)] \\ &= \max_{1\leq i\leq k} \theta_i \cdot MAP(p_i) \cdot MAP(s_i) \quad \square\end{aligned}$$

Proof of Theorem 2. Base case: Let n be a terminal node normalized for leaf vtree node v. Let X be the variable of leaf v and $\mathbf{x}$ an instantiation of X. If $n = X$, on one hand $\top \models X$ and $\underline{\mathbb{P}}_X(\top) = 1$, on the other hand $\bot \not\models X$ and $\overline{\mathbb{P}}_X(\bot) = 0$. Similarly for $n = \neg X$. If $n = (X : [\alpha, \beta])$, $\underline{\mathbb{P}}_n(\top) = \alpha$ and $\underline{\mathbb{P}}_n(\bot) = 1 - \beta$, which are both strictly positive, and remember that this node's interpretation is $\top$, so that both $\top$ and $\bot$ trivially model the node. Induction step: Let v be an internal vtree node and assume the statement of the theorem true for CSDD nodes normalized for v's descendant. Let $n = (\{(p_i, s_i)\}_{i=1}^k, \mathbb{K}_n)$ be a decision node normalized for v. Let $\mathbf{X}$ and $\mathbf{Y}$ be the left respectively right variables of v. Now, for any instantiation $\mathbf{xy}$ of $\mathbf{XY}$:

$$\begin{aligned}\underline{\mathbb{P}}_n(\mathbf{xy}) &= \min_{\mathbb{P}(\mathbf{XY})\in\mathbb{K}^n(\mathbf{XY})} \mathbb{P}(\mathbf{xy}) \\ &= \min_{\mathbb{P}_n(\mathbf{XY})\in\mathbb{K}^n(\mathbf{XY})} \sum_{i=1}^k \mathbb{P}_{p_i}(\mathbf{x}) \cdot \mathbb{P}_{s_i}(\mathbf{y}) \cdot \theta_i \\ &= \min_{[\theta_1,\ldots,\theta_k]\in\mathbb{K}_n(P)} \sum_{i=1}^k \underline{\mathbb{P}}_{p_i}(\mathbf{x}) \cdot \underline{\mathbb{P}}_{s_i}(\mathbf{y}) \cdot \theta_i.\end{aligned}$$

Similarly, we can derive

$$\overline{\mathbb{P}}_n(\mathbf{xy}) = \max_{[\theta_1,\ldots,\theta_k]\in\mathbb{K}_n(P)} \sum_{i=1}^k \overline{\mathbb{P}}_{p_i}(\mathbf{x}) \cdot \overline{\mathbb{P}}_{s_i}(\mathbf{y}) \cdot \theta_i.$$

We have that $\mathbf{xy} \models \langle n \rangle$ if and only if $\mathbf{y} \models \langle s_j \rangle$ for the unique $1 \leq j \leq k$ such that $\mathbf{x} \models \langle p_j \rangle$. By induction hypothesis, this happens if and only if $\underline{\mathbb{P}}_{p_j}(\mathbf{x}) \cdot \underline{\mathbb{P}}_{s_j}(\mathbf{y}) > 0$. This is equivalent to $\min_{[\theta_1,\ldots,\theta_k]\in\mathbb{K}_n(P)} \sum_{i=1}^k \underline{\mathbb{P}}_{p_i}(\mathbf{x}) \cdot \underline{\mathbb{P}}_{s_i}(\mathbf{y}) \cdot \theta_i > 0$ (observe that, because $\mathbf{y} \models \langle s_i \rangle$, $s_i \neq \bot$ and hence by definition θ_i is constrained to be strictly positive). Similarly $\mathbf{xy} \not\models \langle n \rangle$ if and only if

$\mathbf{y} \not\models \langle s_j \rangle$ for the unique $1 \leq j \leq k$ such that $\mathbf{x} \models \langle p_j \rangle$. By induction hypothesis, this happens if and only if $\overline{\mathbb{P}}_{p_j}(\mathbf{x}) \cdot \overline{\mathbb{P}}_{s_j}(\mathbf{y}) = 0$. By definition of j and by induction hypothesis, $\overline{\mathbb{P}}_{p_i}(\mathbf{x}) = 0$ for all $i \neq j$, making $\max_{[\theta_1,\ldots,\theta_k] \in \mathbb{K}_n(P)} \sum_{i=1}^{k} \overline{\mathbb{P}}_{p_i}(\mathbf{x}) \cdot \overline{\mathbb{P}}_{s_i}(\mathbf{y}) \cdot \theta_i = 0$. □

Proof of Theorem 3. If n is a terminal node, the theorem is true by definition of Algorithm 3 (the computation of $\underline{\mathbb{P}}_n(\mathbf{e})$ is immediate). Let $n = (\{(p_i, s_i)\}_{i=1}^{k}, \mathbb{K}_n(P))$ be a decision node and assume that the theorem holds for n's primes and subs. If l and r are the left, respectively right sub-vtree of v, we have that:

$$
\begin{aligned}
\underline{\mathbb{P}}_n(\mathbf{e}) &= \min_{\mathbb{P}(\mathbf{Z}) \in \mathbb{K}^n(\mathbf{Z})} \mathbb{P}(\mathbf{e}) \\
&\overset{(1)}{=} \min_{\mathbb{P}_n(\mathbf{Z}) \in \mathbb{K}^n(\mathbf{Z})} \sum_{i=1}^{k} \mathbb{P}_{p_i}(\mathbf{e}_l) \cdot \mathbb{P}_{s_i}(\mathbf{e}_r) \cdot \theta_i \\
&\overset{(2)}{=} \min_{[\theta_1,\ldots,\theta_k] \in \mathbb{K}_n(P)} \sum_{i=1}^{k} \min_{\mathbb{P}_{p_i}(\mathbf{Z}_l) \in \mathbb{K}^{p_i}} \mathbb{P}_{p_i}(\mathbf{e}_l) \cdot \min_{\mathbb{P}_{s_i}(\mathbf{Z}_r) \in \mathbb{K}^{s_i}} \mathbb{P}_{s_i}(\mathbf{e}_r) \cdot \theta_i \\
&\overset{(3)}{=} \min_{[\theta_1,\ldots,\theta_k] \in \mathbb{K}_n(P)} \sum_{i=1}^{k} \underline{\pi}(p_i) \cdot \underline{\pi}(s_i) \cdot \theta_i
\end{aligned}
$$

(1) is because optima are attained in extreme points, plus [9, Theorem 7]. In (2) we move the minimizations concerning $\mathbb{P}_{p_i}(\mathbf{e}_l)$ and $\mathbb{P}_{s_i}(\mathbf{e}_r)$ inside the sum. This can be done because these minimizations are done over two distinct CSs (the strong extension of the sub-CSDD rooted at p_i and the strong extension of the sub-CSDD rooted at s_i) and then, with the obtained values, solve the LP over the CS $\mathbb{K}_n(P)$ attached to node n. Hence, the induction hypothesis applies in (3), knowing again that the argument used in (1) applies to nodes p_i and s_i, for all $1 \leq i \leq k$. □

Proof of Theorem 4. Let n be a node normalized for a vtree node v in the input CSDD. If X does not occur in v, $\mathbb{P}(x, \boldsymbol{e}) = \mathbb{P}(\neg x, \boldsymbol{e}) = \mathbb{P}(\boldsymbol{e})$ for all $\mathbb{P}(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})$. The result of the right hand side minimization is then $(1 - 2\mu) \cdot \mathbb{P}(\boldsymbol{e})$, i.e., $\underline{\rho}_n(\mu)$.

Now assume that X occurs in v.

If v is a leaf, n is a terminal node. As optimal values are attained on the borders of the domain, the left hand side of Equation (16) rewrites exactly as $\lambda_n(\mu)$. Hence, for a terminal node, the result of the right hand side minimization is $\Lambda(n)$, thus the base case is proved. Assume now that the Theorem is true for nodes normalized for v's sub-vtrees.

Consider a decision node $n = (\{(p_i, s_i)\}_{i=1}^{k}, \mathbb{K}_n(P))$ (normalized for v) and assume that X occurs in the left sub-vtree of v, v^l, the case when X occurs in the right sub-vtree being *mutatis mutandis* the same. The right hand side of the equality to be proven can be rewritten as

$$
\begin{aligned}
\min_{\mathbb{P}(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})} [(1-\mu)\mathbb{P}(x, \boldsymbol{e}) - \mu\mathbb{P}(\neg x, \boldsymbol{e})] &\overset{(1)}{=} \min_{\mathbb{P}_n(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})} [(1-\mu)\mathbb{P}_n(x, \boldsymbol{e}) - \mu\mathbb{P}_n(\neg x, \boldsymbol{e})] \\
&\overset{(2)}{=} \min_{\mathbb{P}_n(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})} \left[(1-\mu) \sum_{i=1}^{k} \mathbb{P}_{p_i}(x, \boldsymbol{e}_l)\mathbb{P}_{s_i}(\boldsymbol{e}_r)\theta_i - \mu \sum_{i=1}^{k} \mathbb{P}_{p_i}(\neg x, \boldsymbol{e}_l)\mathbb{P}_{s_i}(\boldsymbol{e}_r)\theta_i \right] \\
&= \min_{\mathbb{P}_n(\boldsymbol{X}) \in \mathbb{K}^n(\boldsymbol{X})} \left[\sum_{i=1}^{k} [(1-\mu)\mathbb{P}_{p_i}(x, \boldsymbol{e}_l) - \mu\mathbb{P}_{p_i}(\neg x, \boldsymbol{e}_l)] \cdot \mathbb{P}_{s_i}(\boldsymbol{e}_r) \cdot \theta_i \right] \\
&\overset{(3)}{=} \min_{(\theta_1,\ldots,\theta_k) \in \mathbb{K}_n(P)} \left[\sum_{i=1}^{k} \min_{\mathbb{P}_{p_i}(\boldsymbol{X}_l) \in \mathbb{K}^{p_i}(\boldsymbol{X}_l)} \left[(1-\mu)\mathbb{P}_{p_i}(x, \boldsymbol{e}_l) - \mu\mathbb{P}_{p_i}(\neg x, \boldsymbol{e}_l) \right] \cdot \min_{\mathbb{P}_{s_i}(\boldsymbol{X}_r) \in \mathbb{K}^{s_i}(\boldsymbol{X}_r)} \mathbb{P}_{s_i}(\boldsymbol{e}_r) \cdot \theta_i \right] \\
&\overset{(4)}{=} \min_{(\theta_1,\ldots,\theta_k) \in \mathbb{K}_n(P)} \left[\sum_{i=1}^{k} \min_{\mathbb{P}(\boldsymbol{X}_l) \in \mathbb{K}^{p_i}(\boldsymbol{X}_l)} [(1-\mu)\mathbb{P}(x, \boldsymbol{e}_l) - \mu\mathbb{P}(\neg x, \boldsymbol{e}_l)] \cdot \min_{\mathbb{P}(\boldsymbol{X}_r) \in \mathbb{K}^{s_i}(\boldsymbol{X}_r)} \mathbb{P}(\boldsymbol{e}_r) \cdot \theta_i \right] \\
&\overset{(5)}{=} \min_{(\theta_1,\ldots,\theta_k) \in \mathbb{K}_n(P)} \left[\sum_{i=1}^{k} \underline{\pi}(p_i) \cdot \underline{\sigma}_{s_i}(\underline{\pi}(p_i)) \cdot \theta_i \right]
\end{aligned}
$$

where equalities (1) and (4) are because optimal values are attained in extreme points of the strong extension, (2) is thanks to Theorem [9, Theorem 6]. Equality (3) is because the strong extensions of p_i and s_i are distinct, thus the optimization can be performed separately. Note that here the singly connectedness assumption is necessary, as explained in the last part of this section. Equality (5) is by induction hypothesis plus $\underline{\sigma}_{s_i}$'s definition. □

Proof of Theorem 5. If n is a terminal, it is easy to see the correctness of the algorithm. Suppose that n is a decision node, $n = (\{(p_i, s_i)\}_{i=1}^{k}, \mathbb{K}_n)$. For a given $\mathbf{x} \in val(\mathbf{X})$, $\mathbf{xe}$ is a total instantiation of its variables. Since the maximum on $\mathbb{K}^n$ is realized on extreme points we can consider PSDDs probability distributions when computing the maximum. Remember that each considered instantiation of $\mathbf{X}$ selects a unique branch $1 \leq i \leq k$ of n. With the same reasoning adopted in the proof of Algorithm 2, we can argue that

$$\begin{aligned}\max_{\mathbf{x}\in val(\mathbf{X})} \max_{\mathbb{P}\in\mathbb{K}^n} \mathbb{P}(\mathbf{x}, \boldsymbol{e}) &= \max_{1\leq i\leq k} \max_{\mathbf{x}\in val(\mathbf{X})} \max_{\mathbb{K}_{n,i}} \theta_i \cdot \max_{\mathbb{K}^{p_i}} \mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l) \cdot \max_{\mathbb{K}^{s_i}} \mathbb{P}_{s_i}(\mathbf{x}_r\mathbf{e}_r) \\ &= \max_{1\leq i\leq k} \max_{\mathbb{K}_{n,i}} \theta_i \cdot [\max_{\mathbf{x}_l\in val(\mathbf{X}_l)} \max_{\mathbb{K}^{p_i}} \mathbb{P}_{p_i}(\mathbf{x}_l\mathbf{e}_l)] \cdot [\max_{\mathbb{K}^{s_i}} \max_{\mathbf{x}_r\in val(\mathbf{X}_r)} \mathbb{P}_{s_i}(\mathbf{x}_r\mathbf{e}_r)] \\ &= \max_{1\leq i\leq k} \max_{\mathbb{K}_{n,i}} \theta_i \cdot M(p_i) \cdot M(s_i) \quad \square\end{aligned}$$

Proof of Theorem 6. Base case: Let n be a terminal node.

- If $var(n) \in \mathbf{X}$:
  - if $n \in \{X, \neg X\}$, then if $\mathbf{x}_v^* \models n$ the maximization clearly reduces to 1, while if $\mathbf{x}_v^* \not\models n$, the expression is not defined and we refer to the convention;
  - if $n = (X : [l, u])$: if $\mathbf{x}_v^* = \top$,

$$\max_{\theta\in[l,u]} \{\frac{\mathbb{P}_n(\top)}{\mathbb{P}_n(\top)}, \frac{\mathbb{P}_n(\bot)}{\mathbb{P}_n(\top)}\} = \max\{\max_{\theta\in[l,u]} \tfrac{\theta}{\theta}, \max_{\theta\in[l,u]} \tfrac{1-\theta}{\theta}\} = \max\{1, \tfrac{1-l}{l}\},$$

  otherwise, if $\mathbf{x}_v^* = \bot$,

$$\max_{\theta\in[l,u]} \{\frac{\mathbb{P}_n(\bot)}{\mathbb{P}_n(\bot)}, \frac{\mathbb{P}_n(\top)}{\mathbb{P}_n(\bot)}\} = \max\{\max_{\theta\in[l,u]} \tfrac{1-\theta}{1-\theta}, \max_{\theta\in[l,u]} \tfrac{\theta}{1-\theta}\} = \max\{1, \tfrac{u}{1-u}\}.$$

- If $var(n) \in \mathbf{E}$: if $\mathbf{e} \models n$, the fraction reduces to 1, while if $\mathbf{e} \not\models n$, again the expression is not defined hence we refer to the convention.

Induction step: Let $n = ((p_i, s_i)_{i=1}^{k}, \mathbb{K}_n(P))$ be a decision node. If $\mathbf{x}_v^*\mathbf{e}_v \not\models \langle n \rangle$, $V(n) = 1$, in accord with the convention. Assume now that $\mathbf{x}_v^*\mathbf{e}_v \models \langle n \rangle$. Since $\mathbf{x}^*$ is fixed, there is a unique $1 \leq j \leq k$ such that $\mathbf{x}_{v,l}^*\mathbf{e}_{v,l} \models p_j$. Then (as usual, we can perform the optimization on the extreme points of the strong extension)

$$\begin{aligned}&\max_{\mathbf{x}_v\in val(\mathbf{X}_v)} \max_{\mathbb{P}_n\in\mathbb{K}^n} \frac{\mathbb{P}_n(\mathbf{x}_v, \mathbf{e}_v)}{\mathbb{P}_n(\mathbf{x}_v^*, \mathbf{e}_v)} = \max_{\mathbb{P}_n\in\mathbb{K}^n} \frac{\max_{\mathbf{x}_v\in val(\mathbf{X}_v)} \mathbb{P}_n(\mathbf{x}_v, \mathbf{e}_v)}{\mathbb{P}_{p_j}(\mathbf{x}_{v^l}^*, \mathbf{e}_{v^l}) \cdot \mathbb{P}_{s_j}(\mathbf{x}_{v^r}^*, \mathbf{e}_{v^r}) \cdot \theta_j} \\ &= \max_{\mathbb{K}^n} \frac{\max_{1\leq i\leq k} \theta_i \max_{\mathbf{x}_{v^l}} \mathbb{P}_{p_i}(\mathbf{x}_{v^l}, \mathbf{e}_{v^l}) \cdot \max_{\mathbf{x}_{v^r}} \mathbb{P}_{s_i}(\mathbf{x}_{v^r}, \mathbf{e}_{v^r})}{\mathbb{P}_{p_j}(\mathbf{x}_{v^l}^*, \mathbf{e}_{v^l}) \cdot \mathbb{P}_{s_j}(\mathbf{x}_{v^r}^*, \mathbf{e}_{v^r}) \cdot \theta_j}\end{aligned}$$

Now, for $1 \leq i \leq k$, if $i = j$ the above expression simplifies and becomes

$$\max_{\mathbb{P}_{p_j}\in\mathbb{K}^{p_j}} \frac{\max_{\mathbf{x}_{v^l}} \mathbb{P}_{p_j}(\mathbf{x}_{v^l}, \mathbf{e}_{v^l})}{\mathbb{P}_{p_j}(\mathbf{x}_{v^l}^*, \mathbf{e}_{v^l})} \cdot \max_{\mathbb{P}_{s_j}\in\mathbb{K}^{s_j}} \frac{\max_{\mathbf{x}_{v^r}} \mathbb{P}_{s_j}(\mathbf{x}_{v^r}, \mathbf{e}_{v^r})}{\mathbb{P}_{s_j}(\mathbf{x}_{v^r}^*, \mathbf{e}_{v^r})}$$

that is, by induction hypothesis,

$$V(p_j) \cdot V(s_j).$$

If we fix a $i \neq j$ instead, the optimizations might be performed independently since the CSs above and below are distinct:

$$= \max_{\mathbb{K}_n} \frac{\theta_i \max_{\mathbf{x}_{v^l}} \max_{\mathbb{P}_{p_i}\in\mathbb{K}^{p_i}} \mathbb{P}_{p_i}(\mathbf{x}_{v^l}, \mathbf{e}_{v^l}) \cdot \max_{\mathbf{x}_{v^r}} \max_{\mathbb{P}_{s_i}\in\mathbb{K}^{s_i}} \mathbb{P}_{s_i}(\mathbf{x}_{v^r}, \mathbf{e}_{v^r})}{\theta_j \cdot \underline{\mathbb{P}}_{p_j}(\mathbf{x}_{v^l}^*, \mathbf{e}_{v^l}) \cdot \underline{\mathbb{P}}_{s_j}(\mathbf{x}_{v^r}^*, \mathbf{e}_{v^r})}$$

that is,

$$= \max_{\mathbb{K}_n} \frac{\theta_i \cdot M(p_j) \cdot M(s_j)}{\theta_j \cdot \underline{\mathbb{P}}_{p_j}(\mathbf{x}_{v^l}^*, \mathbf{e}_{v^l}) \cdot \underline{\mathbb{P}}_{s_j}(\mathbf{x}_{v^r}^*, \mathbf{e}_{v^r})},$$

which completes the proof. $\square$

Appendix B. CSDD quantification for Example 3

$$\theta_1 = P(\neg X_1 \wedge \neg X_2) \in \left[\frac{n_{\theta_1}}{n+s}, \frac{n_{\theta_1}+s}{n+s}\right]$$

$$\theta_2 = P((X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X2)) \in \left[\frac{n_{\theta_2}}{n+s}, \frac{n_{\theta_2}+s}{n+s}\right]$$

$$\theta_3 = P(\neg X_3 | \neg X_1 \wedge \neg X_2) \in \left[\frac{n_{\theta_3}}{n_{\theta_1}+s}, \frac{n_{\theta_3}+s}{n_{\theta_1}+s}\right]$$

$$\theta_4 = P(X_1 | (X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2)) \in \left[\frac{n_{\theta_4}}{n_{\theta_2}+s}, \frac{n_{\theta_4}+s}{n_{\theta_2}+s}\right]$$

$$\theta_5 = P(X_3 | (X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2)) \in \left[\frac{n_{\theta_5}}{n_{\theta_2}+s}, \frac{n_{\theta_5}+s}{n_{\theta_2}+s}\right]$$

$$\theta_6 = P(X_4 | (\neg X_1 \wedge \neg X_2) \wedge X_3) \in \left[\frac{n_{\theta_6}}{n_{\theta_1} - n_{\theta_3}+s}, \frac{n_{\theta_6}+s}{n_{\theta_1} - n_{\theta_3}+s}\right]$$

$$\theta_7 = P(X_4 | (X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2) \wedge \neg X_3) \in \left[\frac{n_{\theta_7}}{n_{\theta_2} - n_{\theta_5}+s}, \frac{n_{\theta_7}+s}{n_{\theta_2} - n_{\theta_5}+s}\right]$$

where

$$n_{\theta_1} = n_2 + n_6 + n_9$$

$$n_{\theta_2} = n_0 + n_1 + n_4 + n_5 + n_7 + n_8$$

$$n_{\theta_3} = n_6$$

$$n_{\theta_4} = n_0 + n_5 + n_8$$

$$n_{\theta_5} = n_1 + n_5$$

$$n_{\theta_6} = n_2$$

$$n_{\theta_7} = n_0 + n_4$$

References

[1] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[2] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, 2009.
[3] D. Roth, On the hardness of approximate reasoning, Artif. Intell. 82 (1–2) (1996) 273–302.
[4] J. Kwisthout, H.L. Bodlaender, L.C. van der Gaag, The necessity of bounded treewidth for efficient inference in Bayesian networks, in: ECAI, vol. 215, 2010, pp. 237–242.
[5] C.P. de Campos, New complexity results for MAP in Bayesian networks, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011, pp. 2100–2106.
[6] D. Lowd, P. Domingos, Learning arithmetic circuits, in: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, 2008, pp. 383–392.
[7] H. Poon, P. Domingos, Sum-product networks: a new deep architecture, in: 2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops, IEEE, 2011, pp. 689–690.
[8] T. Rahman, P. Kothalkar, V. Gogate, Cutset networks: a simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML-PKDD, 2014, pp. 630–645.
[9] D. Kisa, G. Van den Broeck, A. Choi, A. Darwiche, Probabilistic sentential decision diagrams, in: Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning, 2014.
[10] R. Peharz, A. Vergari, K. Stelzner, A. Molina, M. Trapp, K. Kersting, Z. Ghahramani, Probabilistic deep learning using random sum-product networks, preprint arXiv:1806.01910, 2018.
[11] R. Peharz, A. Vergari, K. Stelzner, A. Molina, M. Trapp, X. Shao, K. Kersting, Z. Ghahramani, Random sum-product networks: a simple and effective approach to probabilistic deep learning, in: Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence, UAI, 2019.
[12] A. Choi, G.V. den Broeck, A. Darwiche, Tractable learning for structured probability spaces: a case study in learning preference distributions, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI, 2015, pp. 2861–2868.
[13] A. Choi, N. Tavabi, A. Darwiche, Structured features in naive Bayes classification, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI, 2016, pp. 3233–3240.
[14] A. Choi, Y. Shen, A. Darwiche, Tractability in structured probability spaces, in: Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 3480–3488.
[15] Y. Shen, A. Choi, A. Darwiche, A tractable probabilistic model for subset selection, in: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, 2017.
[16] P. Walley, Inferences from multinomial data: learning about a bag of marbles, J. R. Stat. Soc. B 58 (1) (1996) 3–34.
[17] F.G. Cozman, Credal networks, Artif. Intell. 120 (2000) 199–233.
[18] D. Mauá, F.G. Cozman, D. Conaty, C.P. de Campos, Credal sum-product networks, in: Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications, 2017, pp. 205–216.
[19] D.D. Mauá, D. Conaty, F.G. Cozman, K. Poppenhaeger, C.P. de Campos, Robustifying sum-product networks, Int. J. Approx. Reason. 101 (2018) 163–180.

[20] J.V. Llerena, D.D. Mauá, Robust analysis of MAP inference in selective sum-product networks, in: Proceedings of the 11th International Symposium on Imprecise Probabilities: Theories and Applications, 2019, pp. 430–440.
[21] R. Peharz, R. Gens, P. Domingos, Learning selective sum-product networks, in: Workshop on Learning Tractable Probabilistic Models, 2014.
[22] R. Peharz, R. Gens, F. Pernkopf, P. Domingos, On the latent variable interpretation in sum-product networks, IEEE Trans. Pattern Anal. Mach. Intell. (2016) 1–14.
[23] D. Conaty, D.D. Mauá, C.P. de Campos, Approximations complexity of maximum a posteriori inference in sum-product networks, in: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, 2017, pp. 322–331.
[24] L. Mattei, D. Soares, A. Antonucci, D. Mauà, A. Facchini, Exploring the space of probabilistic sentential decision diagrams, in: 3rd Workshop of Tractable Probabilistic Modeling, 2019.
[25] A. Darwiche, SDD: a new canonical representation of propositional knowledge bases, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI'11, AAAI Press, 2011, pp. 819–826.
[26] J. Bekker, J. Davis, A. Choi, A. Darwiche, G. Van den Broeck, Tractable learning for complex probability queries, in: Advances in Neural Information Processing Systems, 2015, pp. 2242–2250.
[27] G. de Cooman, F. Hermans, A. Antonucci, M. Zaffalon, Epistemic irrelevance in credal nets: the case of imprecise Markov trees, Int. J. Approx. Reason. 51 (9) (2010) 1029–1052.
[28] A. Choi, A. Darwiche, Dynamic minimization of sentential decision diagrams, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.
[29] D.D. Mauá, A. Antonucci, C.P. de Campos, Hidden Markov models with set-valued parameters, Neurocomputing 180 (2016) 94–107.
[30] M. Zaffalon, The naive credal classifier, J. Stat. Plan. Inference 105 (1) (2002) 5–21.
[31] M. Zaffalon, G. Corani, D. Mauá, Evaluating credal classifiers by utility-discounted predictive accuracy, Int. J. Approx. Reason. 53 (8) (2012) 1282–1301.