



Handling epistemic and aleatory uncertainties in probabilistic circuits

Federico Cerutti^{1,2} · Lance M. Kaplan³ · Angelika Kimmig^{4,5} · Murat Şensoy^{6,7}

Received: 15 May 2020 / Revised: 27 March 2021 / Accepted: 27 September 2021 /
Published online: 10 January 2022

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2021

Abstract

When collaborating with an AI system, we need to assess when to trust its recommendations. If we mistakenly trust it in regions where it is likely to err, catastrophic failures may occur, hence the need for Bayesian approaches for probabilistic reasoning in order to determine the confidence (or epistemic uncertainty) in the probabilities in light of the training data. We propose an approach to Bayesian inference of posterior distributions that overcomes the independence assumption behind most of the approaches dealing with a large class of probabilistic reasoning that includes Bayesian networks as well as several instances of probabilistic logic. We provide an algorithm for Bayesian inference of posterior distributions from sparse, albeit complete, observations, and for deriving inferences and their confidences keeping track of the dependencies between variables when they are manipulated within the unifying computational formalism provided by probabilistic circuits. Each leaf of such circuits is labelled with a beta-distributed random variable that provides us with an elegant framework for representing uncertain probabilities. We achieve better estimation of epistemic uncertainty than state-of-the-art approaches, including highly engineered ones, while being able to handle general circuits and with just a modest increase in the computational effort compared to using point probabilities.

Keywords Bayesian learning · Probabilistic circuit · Imprecise probabilities

1 Introduction

Even in simple collaboration scenarios—like those in which an artificial intelligence (AI) system assists a human operator with predictions—the success of the team hinges on the human correctly deciding when to follow the recommendations of the AI system and when to override them (Bansal et al., 2019b). When that happens, the human has developed insights (i.e., a mental model) of when to trust the AI system with its recommendations

Editors: Nikos Katzouris, Alexander Artikis, Luc De Raedt, Artur d’Avila Garcez, Sebastijan Dumančić, Ute Schmid, Jay Pujara.

✉ Federico Cerutti
federico.cerutti@unibs.it

Extended author information available on the last page of the article

(Bansal et al., 2019b). If the human mistakenly trusts the AI system in regions where it is likely to err, catastrophic failures may occur. This is a strong argument in favour of Bayesian approaches to probabilistic reasoning: research in the intersection of AI and HCI has found that interaction improves when setting expectations right about what the system can do and how well it performs (Kocielnik et al., 2019; Bansal et al., 2019a). Guidelines have been produced (Amershi et al., 2019), and they recommend to *Make clear what the system can do (G1)*, and *Make clear how well the system can do what it can do (G2)*.

To identify such regions where the AI system is likely to err, we need to distinguish between (at least) two different sources of uncertainty: *aleatory* (or *aleatoric*), and *epistemic* uncertainty (Hora, 1996; Hüllermeier and Waegeman, 2019). Aleatory uncertainty refers to the variability in the outcome of an experiment which is due to inherently random effects (e.g. flipping a fair coin): no additional source of information but Laplace’s daemon¹ can reduce such a variability. Epistemic uncertainty refers to the epistemic state of the agent using the model, hence its lack of knowledge that—in principle—can be reduced on the basis of additional data samples. Particularly when considering sparse data, the epistemic uncertainty around the learnt model can significantly affect decision making (Antonucci et al., 2014; Anderson et al., 2016), for instance when used for computing an expected utility (Von Neumann and Morgenstern, 2007).

In this paper, we propose an approach to probabilistic reasoning that manipulates joint distributions of probabilities without assuming independence between the single random variables (i.e. their covariances may not be zero), and without resorting to sampling. We operate within the unifying computational formalism provided by arithmetic circuits (von zur Gathen, 1988), sometimes named *probabilistic circuits* when manipulating probabilities, or simply *circuits*. This is clearly a novel contribution as the few approaches (Rashwan et al., 2016; Jaini et al., 2016; Cerutti et al., 2019) resorting to distribution estimation via moment matching, the very same technique we also use in this work, still assume independence between the random variables when computing their joint distribution using a probabilistic circuit. Instead, we provide an algorithm for Bayesian learning from sparse—albeit complete—observations, and for probabilistic inferences that keep track of the dependencies between variables when they are manipulated within the circuit. In particular, we focus on the large class of approaches to probabilistic reasoning that rely upon algebraic model counting (AMC) (Kimmig et al., 2017) (Sect. 2.1), which has been proven to encompass probabilistic inferences under Sato (1995)’s semantics, thus covering not only Bayesian networks (Sang et al. 2005), but also probabilistic logic programming approaches such as ProbLog (Fierens et al., 2015), and others as discussed by Cerutti and Thimm (2019). We can exploit the results of Darwiche and Marquis (2002) (Sect. 2.2) who studied the succinctness relations between various types of circuits and thus their applicability to model counting. Their work, indeed, directly refers to set of models of a propositional logic theory exactly as AMC does. To stress the applicability of this setting, circuit compilation techniques (Darwiche, 2004; Choi and Darwiche, 2013; Oztok and Darwiche, 2015) are behind state-of-the-art algorithms for (1) exact and approximate inference in discrete probabilistic graphical models (Chavira and Darwiche, 2008; Kisa et al., 2014; Friedman and den Broeck, 2018); and (2) probabilistic programs (Bellodi and Riguzzi, 2013; Fierens et al., 2015). Also, learning tractable circuits is the current method of choice for discrete density estimation (Gens and Domingos, 2013; Rooshenas and Lowd, 2014;

¹ “An intelligence that, at a given instant, could comprehend all the forces by which nature is animated and the respective situation of the beings that make it up” (Laplace, 1825, p.2).

2015; Vergari et al., 2019; Liang et al., 2017). Finally, Xu et al. (2018) also used circuits to enforce logical constraints on deep neural networks.

In this paper, we label each leaf of the circuit with a beta-distributed random variable (Sect. 3). The beta distribution is a well-defined theoretical framework that specifies a distribution of probabilities representing all the possible values of a probability when the exact value is unknown. In this way, the expected value of a beta-distributed random variable relates to the aleatory uncertainty of the phenomenon, and the variance to the epistemic uncertainty: the higher the variance, the *less certain* the machine is, thus targeting directly (Amershi et al., 2019, G1 and G2). In previous work (Cerutti et al., 2019) we provided operators for manipulating beta-distributed random variables under strong independence assumptions (Sect. 4). This paper significantly extends and improves our previous approach by eliminating the independence assumption in manipulating beta-distributed random variables within a circuit.

Indeed, our main contribution (Sect. 5) is an algorithm for reasoning over a circuit whose leaves are labelled with beta-distributed random variables, with the additional piece of information describing which of those are actually independent (Sect. 5.1). This is the input to an algorithm that *shadows* the circuit by superimposing a second circuit for computing the probability of a query conditioned on pieces of evidence (Sect. 5.2) in a single feed forward. While this at first might seem unnecessary, it is actually essential when inspecting the main algorithm that evaluates such a shadowed circuit (Sect. 5.3), where a covariance matrix plays an essential role by keeping track of the dependencies between random variables while they are manipulated within the circuit. We also include discussions on memory management of the covariance matrix in Sect. 5.4.

We evaluate our approach against a set of competing approaches in an extensive set of experiments detailed in Sect. 6, comparing against leading approaches to dealing with uncertain probabilities, notably: (1) Monte Carlo sampling; (2) our previous proposal (Cerutti et al., 2019) taken as representative of the class of approaches using moment matching with strong independence assumptions; (3) Subjective Logic (Jøsang, 2016), that provides an alternative representation of beta distributions as well as a calculus for manipulating them applied already in a variety of domains, e.g. (Jøsang et al., 2006; Moglia et al., 2012; Sensoy et al., 2018); (4) Subjective Bayesian Network (SBN) on circuits derived from singly-connected Bayesian networks (Ivanovska et al., 2015; Kaplan & Ivanovska, 2016; Kaplan & Ivanovska, 2018), that already showed higher performance against other traditional approaches dealing with uncertain probabilities, such as (5) Dempster-Shafer Theory of Evidence (Dempster, 1968; Smets, 1993), and (6) replacing single probability values with closed intervals representing the possible range of probability values (Zaffalon and Fagioli, 1998). We achieve better estimation of epistemic uncertainty than state-of-the-art approaches, including highly engineered ones for a narrow domain such as SBN, while being able to handle general circuits with just a modest increase in the computational effort compared to using point probabilities.

2 Background

2.1 Algebraic model counting

Kimmig et al. (2017) introduce the task of *algebraic model counting* (AMC). AMC generalises weighted model counting (WMC) to the semiring setting and supports various types

of labels, including numerical ones as used in WMC, but also sets, polynomials, Boolean formulae, and many more. The underlying mathematical structure is that of a commutative semiring.

A *semiring* is a structure $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$, where *addition* \oplus and *multiplication* \otimes are associative binary operations over the set \mathcal{A} , \oplus is commutative, \otimes distributes over \oplus , $e^\oplus \in \mathcal{A}$ is the neutral element of \oplus , $e^\otimes \in \mathcal{A}$ that of \otimes , and for all $a \in \mathcal{A}$, $e^\oplus \otimes a = a \otimes e^\oplus = e^\oplus$. In a *commutative semiring*, \otimes is commutative as well.

Algebraic model counting is now defined as follows. Given:

- a *propositional logic theory* T over a set of variables \mathcal{V} ,
- a *commutative semiring* $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$, and
- a *labelling function* $\rho : \mathcal{L} \rightarrow \mathcal{A}$, mapping literals \mathcal{L} of the variables in \mathcal{V} to elements of the semiring set \mathcal{A} ,

compute

$$\mathbf{A}(T) = \bigoplus_{I \in \mathcal{M}(T)} \bigotimes_{l \in I} \rho(l), \tag{1}$$

where $\mathcal{M}(T)$ denotes the set of models of T .

Among others, AMC generalises the task of probabilistic inference according to Sato (1995)’s semantics (**PROB**), (Kimmig et al., 2017, Thm. 1), (Goodman, 1999; Eisner, 2002; Bacchus et al., 2009; Baras and Theodorakopoulos, 2010; Kimmig et al., 2011).

A *query* q is a finite set of algebraic literals $q \subseteq \mathcal{L}$. We denote the set of interpretations where the query is true by $\mathcal{I}(q)$,

$$\mathcal{I}(q) = \{I \mid I \in \mathcal{M}(T) \wedge q \subseteq I\} \tag{2}$$

The label of query q is defined² as the label of $\mathcal{I}(q)$,

$$\mathbf{A}(q) = \mathbf{A}(\mathcal{I}(q)) = \bigoplus_{I \in \mathcal{I}(q)} \bigotimes_{l \in I} \rho(l). \tag{3}$$

As both operators are commutative and associative, the label is independent of the order of both literals and interpretations.

In the context of this paper, we extend AMC for handling **PROB** of queries with evidence by introducing an additional division operator \oslash that defines the conditional label of a query as follows:

$$\mathbf{A}(q \mid \mathbf{E} = \mathbf{e}) = \mathbf{A}(\mathcal{I}(q \wedge \mathbf{E} = \mathbf{e})) \oslash \mathbf{A}(\mathcal{I}(\mathbf{E} = \mathbf{e})) \tag{4}$$

where $\mathbf{A}(\mathcal{I}(q \wedge \mathbf{E} = \mathbf{e})) \oslash \mathbf{A}(\mathcal{I}(\mathbf{E} = \mathbf{e}))$ returns the label of $q \wedge \mathbf{E} = \mathbf{e}$ given the label of a set of pieces of evidence $\mathbf{E} = \mathbf{e}$.

In the case of probabilities as labels, i.e. $\rho(\cdot) \in [0, 1]$, (5) presents the AMC-conditioning parametrisation \mathcal{S}_ρ for handling **PROB** of (conditioned) queries:

² Albeit \mathbf{A} has been introduced to operate over propositional logic theories, with a small abuse of notation we use it also for a finite set of algebraic literals, i.e. *query*, and a set of interpretations.

$$\begin{aligned}
\mathcal{A} &= \mathbb{R}_{\geq 0} \\
a \oplus b &= a + b \\
a \otimes b &= a \cdot b \\
e^{\oplus} &= 0 \\
e^{\otimes} &= 1 \\
\rho(f) &\in [0, 1] \\
\rho(\neg f) &= 1 - \rho(f) \\
a \oslash b &= \frac{a}{b}
\end{aligned} \tag{5}$$

A naïve implementation of (4) is clearly exponential: Darwiche (2004) introduced the first method for deriving tractable circuits (d-DNNFs) that allow polytime algorithms for clausal entailment, model counting and enumeration. Also, while for a generic AMC it is true that $\mathbf{A}(q)$ might not be an un-normalised probability distribution, we will see in the following section that the decomposability and determinism restrictions to the circuits solve the problem.

2.2 Probabilistic circuits

After defining what AMC is, we turn our attention to how we can compute it. From (1) we can see that it requires two operations \oplus and \otimes whose operands are elements of the commutative semiring \mathcal{A} that are associated to literals in a propositional logic theory T . Not only, but we explicitly need to consider the set of models $\mathcal{M}(T)$ of such a propositional logic theory T to compute the result of AMC.

AMC is thus computed by, informally speaking, *multiplying* and *adding*³ labels of propositions that belong to one of the models $\mathcal{M}(T)$ of a propositional theory T : this is a hard problem. Therefore, the difficulty of AMC does not rely in the addition or multiplication, rather in computing the models of a theory. To illustrate this, the truth table for just variables of the form like $p \wedge q \vee r \wedge \neg s$ leads to 2^4 rows.

To better manage the hard problem of computing the models $\mathcal{M}(T)$ of a propositional theory T , we can exploit the succinctness results of the knowledge compilation map by Darwiche and Marquis (2002). The restriction to two-valued variables allows us to directly compile AMC tasks to circuits without adding constraints on legal variable assignments to the theory.

In their knowledge compilation map, Darwiche and Marquis (2002) provide an overview of succinctness relationships between various types of circuits. Instead of focusing on classical, flat target compilation languages based on conjunctive or disjunctive normal forms, Darwiche and Marquis (2002) consider a richer, nested class based on representing propositional sentences using directed acyclic graphs: NNFs. A sentence in *negation normal form* (NNF) over a set of propositional variables \mathcal{V} is a rooted, directed acyclic graph where each leaf node is labeled with true (T), false (F), or a literal of a variable in \mathcal{V} , and each internal node with disjunction (\vee) or conjunction (\wedge).

An NNF is *decomposable* if for each conjunction node $\bigwedge_{i=1}^n \phi_i$, no two children ϕ_i and ϕ_j share any variable.

³ Formally speaking, AMC is computed by the using \otimes , the usual multiplication operation in **PROB**, and \oplus , the usual addition operation in **PROB**.

An NNF is *deterministic* if for each disjunction node $\bigvee_{i=1}^n \phi_i$, each pair of different children ϕ_i and ϕ_j is logically contradictory, that is $\phi_i \wedge \phi_j \vDash \perp$ for $i \neq j$. In other terms, only one child can be true at any time.⁴

The function EVAL specified in Algorithm 1 *evaluates* an NNF circuit for a commutative semiring $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$ and labelling function ρ . Evaluating an NNF representation N_T of a propositional theory T for a semiring $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$ and labelling function ρ is a *sound AMC computation* iff $\text{EVAL}(N_T, \oplus, \otimes, e^\oplus, e^\otimes, \rho) = \mathbf{A}(T)$.

Algorithm 1 Evaluating an NNF circuit N for a commutative semiring $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$ and labelling function ρ .

```

1: procedure EVAL( $N, \oplus, \otimes, e^\oplus, e^\otimes, \rho$ )
2:   if  $N$  is a true node  $\top$  then return  $e^\otimes$ 
3:   if  $N$  is a false node  $\perp$  then return  $e^\oplus$ 
4:   if  $N$  is a literal node  $l$  then return  $\rho(l)$ 
5:   if  $N$  is a disjunction  $\bigvee_{i=1}^m N_i$  then
6:     return  $\bigoplus_{i=1}^m \text{EVAL}(N_i, \oplus, \otimes, e^\oplus, e^\otimes, \rho)$ 
7:   end if
8:   if  $N$  is a conjunction  $\bigwedge_{i=1}^m N_i$  then
9:     return  $\bigotimes_{i=1}^m \text{EVAL}(N_i, \oplus, \otimes, e^\oplus, e^\otimes, \rho)$ 
10:  end if
11: end procedure

```

In particular, (Kimmig et al., 2017, Theorem 4) shows that evaluating a d-DNNF representation of the propositional theory T for a semiring and labelling function with neutral (\oplus, ρ) is a sound AMC computation. A semiring addition and labelling function pair (\oplus, ρ) is *neutral* iff $\forall v \in \mathcal{V} : \rho(v) \oplus \rho(\neg v) = e^\otimes$.

Unless specified otherwise, in the following we will refer to d-DNNF circuits labelled with probabilities or distributions of probability simply as circuits, and any addition and labelling function pair (\oplus, ρ) are neutral. Also, we extend the definition of the labelling function such that it also operates on $\{\perp, \top\}$, i.e. $\rho(\perp) = e^\oplus$ and $\rho(\top) = e^\otimes$.

Let us now introduce a graphical notation for circuits in this paper: Fig. 1 illustrates a d-DNNF circuit where each node has a unique integer (positive or negative) identifier. Moreover, circled nodes are labelled either with \oplus for disjunction (a.k.a. \oplus -gates) or with \otimes for conjunction (a.k.a. \otimes -gates). Leaves nodes are marked with a squared box and they are labelled with the literal, \top or \perp , as well as its label via the labelling function ρ .

Unless specified otherwise, in the following we will slightly abuse the notation by defining an $\bar{\cdot}$ operator both for variables and \top, \perp , i.e. for $x \in \mathcal{V} \cup \{\perp, \top\}$,

$$\bar{x} = \begin{cases} \neg x & \text{if } x \in \mathcal{V} \\ \perp & \text{if } x = \top \\ \top & \text{if } x = \perp \end{cases} \tag{6}$$

and for elements of the set \mathcal{A} of labels, s.t. $\overline{\overline{\rho(x)}} = \rho(\bar{x})$.

⁴ In the case ϕ_i and ϕ_j are seen as events in a sample space, the determinism can be equivalently rewritten as $\phi_i \cap \phi_j = \emptyset$ and hence $P(\phi_i \cap \phi_j) = 0$.

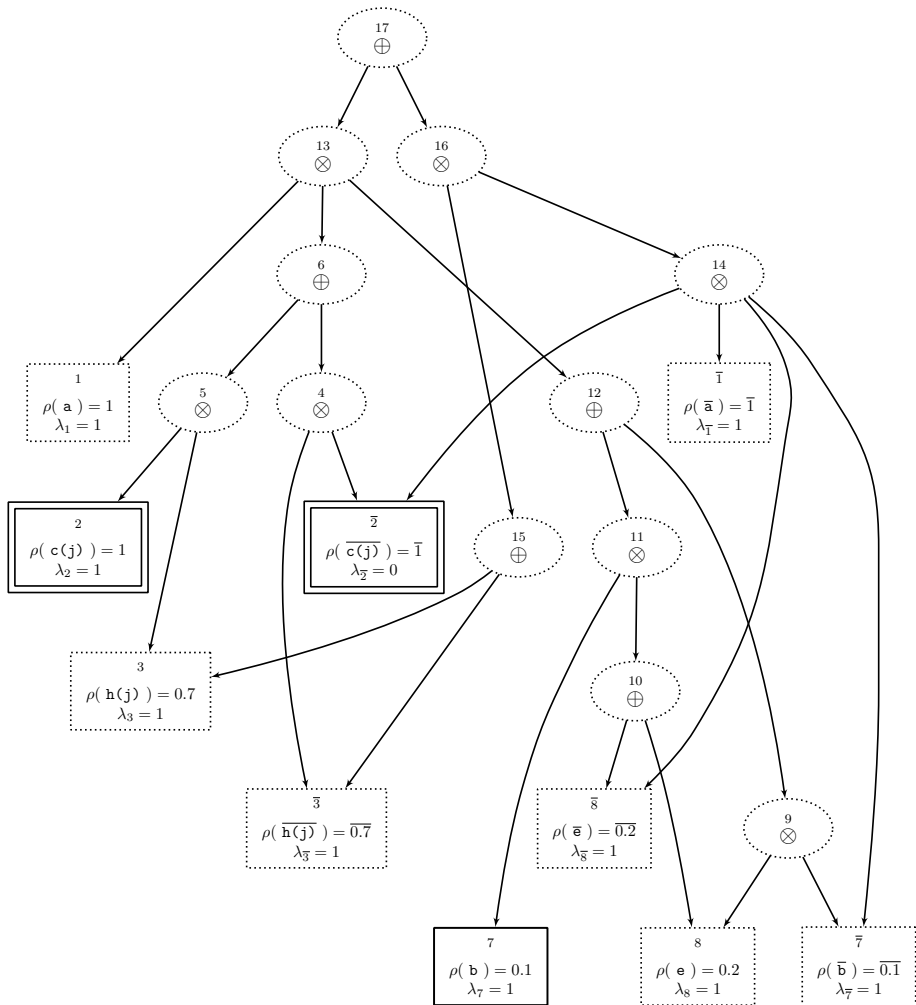


Fig. 1 Circuit computing $p(\text{calls}(\text{john}))$ for the Burglary example (Listing 1). Solid box for query, double box for evidence

Finally, each leaf node i presents an additional parameter λ_i —i.e. the indicator variable cf. (Fierens et al. 2015)—that assumes values 0 or 1, which allows one to reuse the same circuit for different purposes.

In the following, we will make use of a running example based upon the burglary example as presented in (Fierens et al., 2015, Example 6). In this way, we hope to better convey to the reader the value of our approach as the circuit derived from it using (Darwiche, 2004) will have a clear, intuitive meaning behind. However, our approach is independent from the system that employs circuit compilation for its reasoning process, as long as it can make use of d-DNNFs circuits. The d-DNNF circuit for our running example is depicted in Fig. 1 and has been derived by compiling the ProbLog (Fierens et al., 2015) code listed in Listing 1 (Fierens et al., 2015, Example 6) into a d-DNNF using the methods introduced in (Darwiche, 2004). For compactness, in the graph each literal of the program

is represented only by its initials, i.e. `burglary` becomes `b`, `hears_alarm(john)` becomes `h(j)`. ProbLog is an approach to augment⁵ Prolog programs (Kowalski, 1988; Bratko, 2001) annotating facts⁶ with probabilities: see Appendix A for an introduction. As discussed in (Fierens et al., 2015), the Prolog language admits a propositional representation of its semantics. For the example the propositional representation of Listing 1 is:

$$\begin{aligned} \text{alarm} &\leftrightarrow \text{burglary} \vee \text{earthquake} \\ \text{calls}(\text{john}) &\leftrightarrow \text{alarm} \wedge \text{hears_alarm}(\text{john}) \\ &\text{calls}(\text{john}) \end{aligned} \quad (7)$$

Figure 1 thus shows the result of the compilation of (7) in a circuit, annotated with a unique id that is either a number x or \bar{x} to indicate the node that represents the negation of the variable represented by node x ; and with weights (probabilities) as per Listing 1.

The fact that `calls(john)` is true (see line 7 of Listing 1) translates in having $\lambda_2 = 1$ for the double boxed node with index 2 in Fig. 1—that indeed is labelled with the shorthand for `calls(john)`, i.e. $c(j)$ —and $\lambda_2 = 0$ for the double boxed node with index $\bar{2}$ that is instead labelled with the shorthand for $\overline{\text{calls}(\text{john})}$, i.e. $\overline{c(j)}$.

```

1 0.1::burglary.
2 0.2::earthquake.
3 0.7::hears_alarm(john).
4 alarm :- burglary.
5 alarm :- earthquake.
6 calls(john) :- alarm, hears_alarm(john).
7 evidence(calls(john)).
8 query(burglary).

```

Listing 1: Problog code for the Burglary example, originally Example 6 in (Fierens et al., 2015).

The λ_i indicators modify the execution of the function `EVAL` (Alg. 1) in the way illustrated by Algorithm 2: note that Algorithm 2 is analogous to Algorithm 1 when all $\lambda_i = 1$. Hence, in the following, when considering the function `EVAL`, we will be referring to the one defined in Algorithm 2.

⁵ We refer readers interested in probabilistic augmentation of logical theories in general to (Cerutti and Thimm, 2019).

⁶ Albeit ProbLog allows for rules to be annotated with probabilities: rules of the form $p : h \text{ :- } b$ are translated into $h \text{ :- } b, t$ with t a new fact of the form $p : t$.

Algorithm 2 Evaluating an NNF circuit N for a commutative semiring $(\mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes)$ and labelling function ρ , considering indicators λ_i .

```

1: procedure EVAL( $N, \oplus, \otimes, e^\oplus, e^\otimes, \rho$ )
2:   if  $N$  is a true node  $\top$  then
3:     if  $\lambda_N = 1$  then return  $e^\otimes$ 
4:     else return  $e^\oplus$ 
5:   end if
6:   if  $N$  is a false node  $\perp$  then return  $e^\oplus$ 
7:   if  $N$  is a literal node  $l$  then
8:     if  $\lambda_N = 1$  then return  $\rho(l)$ 
9:     else return  $e^\oplus$ 
10:  end if
11:  if  $N$  is a disjunction  $\bigvee_{i=1}^m N_i$  then
12:    return  $\bigoplus_{i=1}^m \text{EVAL}(N_i, \oplus, \otimes, e^\oplus, e^\otimes, \rho)$ 
13:  end if
14:  if  $N$  is a conjunction  $\bigwedge_{i=1}^m N_i$  then
15:    return  $\bigotimes_{i=1}^m \text{EVAL}(N_i, \oplus, \otimes, e^\oplus, e^\otimes, \rho)$ 
16:  end if
17: end procedure

```

Finally, the ProbLog program in Listing 1 queries the value of `burglary`, hence we need to compute the probability of `burglary` given `calls(john)`,

$$p(\text{burglary} \mid \text{calls}(\text{john})) = \frac{p(\text{burglary} \wedge \text{calls}(\text{john}))}{p(\text{calls}(\text{john}))} \quad (8)$$

While the denominator of (8) is given by EVAL of the circuit in Fig. 1, we obtain the numerator $p(\text{burglary} \wedge \text{calls}(\text{john}))$ by evaluating the same circuit with $\lambda_{\top} = 0$ that is the parameter for the node labelled with `burglary` (see Fig. 2). `eval` on the circuit in Fig. 2 will thus return the value of the denominator in (8).

It is worth highlighting that computing $p(\text{query} \mid \text{evidences})$ for an arbitrary *query* and arbitrary set of *evidences* requires EVAL to be executed at least twice on slightly modified circuits.

In this paper, similarly to (Kisa et al., 2014), we are interested in learning the parameters of our circuit, i.e. the ρ function for each of the leaves nodes, or ρ in the following, thus representing it as a vector. We will learn ρ from a set of *examples*, where each *example* is an instantiation of all propositional variables: for n propositional variables, there are 2^n of such instantiations. In the case the circuit is derived from a logic program, an example is a complete interpretation of all the ground atoms. A *complete dataset* \mathcal{D} is then a sequence (allowing for repetitions) of examples, each of those is a vector of instantiations of independent Bernoulli distributions with true but unknown parameter θ . Indeed, in this case, the dataset is assumed to have been sampled from the joint Bernoulli distribution represented by a circuit whose parameters are unknown. This, for complete training datasets, translates into observing independent Bernoulli distributions, one for each (pair) of leaves. Covariances will be not null only between one leaf and its negation (see Appendix C).

From this, the *likelihood* is thus:

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{|\mathcal{D}|} p(x_i \mid \theta) \quad (9)$$

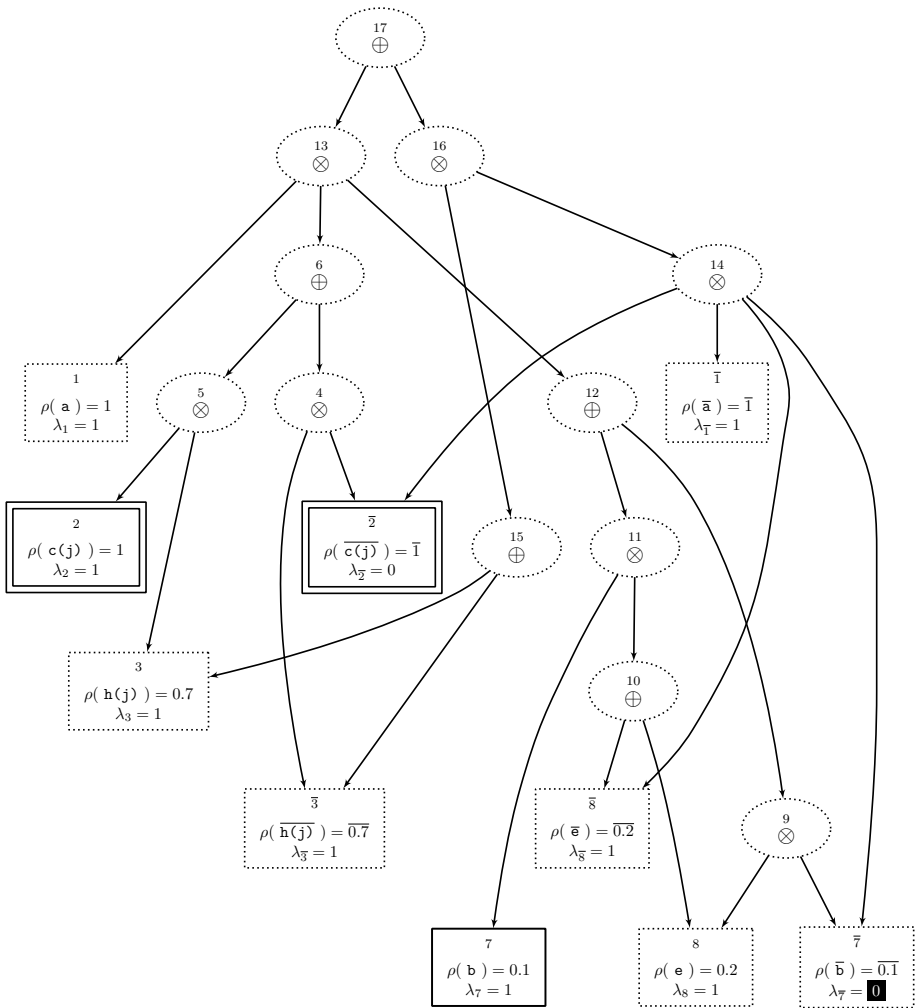


Fig. 2 Circuit computing $p(\text{burglary} \wedge \text{calls}(\text{john}))$ for the Burglary example (Listing 1). Solid box for query, double box for evidence. White over black for the numeric value that has changed from Fig. 1. In particular, in this case, λ_7 for the node labelled with burglary is set to 0

where x_i represents the i -th example in the dataset \mathcal{D} . Differently, however, from (Kisa et al., 2014), we do not search for a maximum likelihood solution of this problem, rather we provide a Bayesian analysis of it in Sect. 3.

The following analysis provides the distribution of the probabilities (second-order probabilities) for each propositional variable. For complete datasets, their joint distributions is factorised into probabilities on individual variables, meaning that the second-order probabilities for the propositional variables are statistically independent (see Appendix C). Nevertheless, it is shown that second-order probabilities of a variable and its negation are correlated because the first-order probabilities (i.e. the expected values of the distributions) sum up to one. For complete datasets, the covariances at the leaves are only non-zero between a variable and its negation.

In this paper, we propose an inference process that does not assume independent second order probabilities. Indeed, when training using incomplete data—i.e. with not all variable values visible during training—the random variables associated to the leaves of the circuits are no longer independent, hence they can have non-null covariance. The derivations of these correlations during the learning process with partial observations is left for future work. Nevertheless, the proposed inference method can accommodate such correlations without any modifications.

This seamless integration of covariance information is one of our main contributions, that separates our approach from the literature on Bayesian approach to learning parameters in circuits (Jaini et al., 2016; Rashwan et al., 2016; Zhao et al., 2016a; Zhao et al., 2016b; Trapp et al., 2019; Vergari et al., 2019). In addition, similarly to (Rashwan et al., 2016; Jaini et al., 2016) we also apply the idea of moment matching instead of using sampling.

3 A Bayesian account of uncertain probabilities

Let us now expand further (9): for simplicity, let us consider here only the case of a single propositional variable, i.e. a single binary random variable $x \in \{0, 1\}$, e.g. flipping coin, not necessary fair, whose probability is thus conditioned by a parameter $0 \leq \theta \leq 1$:

$$p(x = 1 \mid \theta) = \theta \quad (10)$$

The probability distribution over x is known as the *Bernoulli* distribution:

$$\text{Bern}(x \mid \theta) = \theta^x (1 - \theta)^{1-x} \quad (11)$$

Given a data set \mathcal{D} of i.i.d. observations $(x_1, \dots, x_N)^T$ drawn from the Bernoulli with parameter θ , which is assumed unknown, the *likelihood* of data given θ is:

$$p(\mathcal{D} \mid \theta) = \prod_{n=1}^N p(x_n \mid \theta) = \prod_{n=1}^N \theta^{x_n} (1 - \theta)^{1-x_n} \quad (12)$$

To develop a Bayesian analysis of the phenomenon, we can choose as prior the beta distribution, with parameters $\alpha = \langle \alpha_x, \alpha_{\bar{x}} \rangle$, $\alpha_x \geq 1 > 0$ and $\alpha_{\bar{x}} \geq 1 > 0$, that is conjugate to the Bernoulli:

$$\text{Beta}(\theta \mid \alpha) = \frac{\Gamma(\alpha_x + \alpha_{\bar{x}})}{\Gamma(\alpha_x)\Gamma(\alpha_{\bar{x}})} \theta^{\alpha_x - 1} (1 - \theta)^{\alpha_{\bar{x}} - 1} \quad (13)$$

where

$$\Gamma(t) \equiv \int_0^\infty u^{t-1} e^{-u} du \quad (14)$$

is the gamma function.

Given a beta-distributed random variable X ,

$$s_X = \alpha_x + \alpha_{\bar{x}} \quad (15)$$

is its *Dirichlet strength* and

$$\mathbb{E}[X] = \frac{\alpha_x}{s_x} \tag{16}$$

is its expected value. From (15) and (16) the beta parameters can equivalently be written as:

$$\alpha_x = \langle \mathbb{E}[X]s_x, (1 - \mathbb{E}[X])s_x \rangle. \tag{17}$$

The variance of a beta-distributed random variable X is

$$\text{var}[X] = \text{var}[1 - X] = \frac{\mathbb{E}[X](1 - \mathbb{E}[X])}{s_x + 1} \tag{18}$$

and because $X + (1 - X) = 1$, it is easy to see that

$$\text{cov}[X, 1 - X] = -\text{var}[X]. \tag{19}$$

From (18) we can rewrite s_x (15) as

$$s_x = \frac{\mathbb{E}[X](1 - \mathbb{E}[X])}{\text{var}[X]} - 1. \tag{20}$$

Considering a beta distribution prior and the binomial likelihood function, and given N observations of x such that for r observations $x = 1$ and for $s = N - r$ observations $x = 0$

$$p(\theta | \mathcal{D}, \alpha^0) = \frac{p(\mathcal{D} | \theta)p(\theta | \alpha^0)}{p(\mathcal{D})} \propto \theta^{r+\alpha_x^0-1}(1 - \theta)^{s+\alpha_x^0-1} \tag{21}$$

Hence $p(\theta | r, s, \alpha^0)$ is another beta distribution such that after normalization via $p(\mathcal{D})$,

$$p(\theta | r, s, \alpha^0) = \frac{\Gamma(r + \alpha_x^0 + s + \alpha_x^0)}{\Gamma(r + \alpha_x^0)\Gamma(s + \alpha_x^0)} \theta^{r+\alpha_x^0-1}(1 - \theta)^{s+\alpha_x^0-1} \tag{22}$$

We can specify the parameters for the prior we are using for deriving our beta distributed random variable X as $\alpha^0 = \langle a_x W, (1 - a_x)W \rangle$ where a_x is the prior assumption, i.e. $p(x = 1)$ in the absence of observations; and $W > 0$ is a prior weight indicating the strength of the prior assumption. Unless specified otherwise, in the following we will assume $\forall x, a_x = 0.5$ and $W = 2$, so to have an uninformative, uniformly distributed, prior.

The complete dataset \mathcal{D} is modelled as samples from independent Bernoulli distributions. As such, the posterior factors as a product of beta distributions representing the posterior distribution for each fact or rule as in (22) for a single fact (see Appendix C for further details). This posterior distribution enables the computation of the means and covariances for the leaves of the circuit, and because it factors, the different variables are statistically independent leading to zero covariances. Only the leaves associated to a variable and its complement exhibit nonzero covariance via (19). Now, the means and covariances of the leaves can be propagated through the circuit to determine the distribution of the queried conditional probability as described in Sect. 5.

Given an inference, like the conditioned query of our running example (8), we approximate its distribution by a beta distribution by finding the corresponding Dirichlet strength to match the computed variance. Given a random variable Z with known mean $\mathbb{E}[Z]$ and variance $\text{var}[Z]$, we can use the method of moments and (20) to determine the α parameters of a beta-distributed variable approximation Z' with mean $\mathbb{E}[Z'] = \mathbb{E}[Z]$. To ensure that the

approximated variable can be seen as a posterior beta distribution and thus that its parameters can be interpreted as observations pro and against a phenomenon further to a Bayesian update starting with a prior α^0 ,⁷ we need to impose a restriction on such a Dirichlet strength to ensure that, for the approximated random variable Z' , $\alpha_{Z'} \geq \alpha^0$:

$$s_{Z'} = \max \left\{ \frac{\mathbb{E}[Z](1 - \mathbb{E}[Z])}{\text{var}[Z]} - 1, \frac{Wa_Z}{\mathbb{E}[Z]}, \frac{W(1 - a_Z)}{1 - \mathbb{E}[Z]} \right\}. \quad (23)$$

To summarise, each time we use the method of moments to approximate a random variable Z with a beta-distributed random variable Z' such that $\mathbb{E}[Z'] = \mathbb{E}[Z]$ and $\text{var}[Z'] \approx \text{var}[Z]$. The approximation of the variance is computed using (18), that bounds together variance and the Dirichlet strength, and the constraint on the Dirichlet strength added by (23).

3.1 Subjective logic

Subjective logic (Jøsang, 2016) provides (1) an alternative, more intuitive, way of representing the parameters of beta-distributed random variables, and (2) a set of operators for manipulating them that we use to compare against our proposal in an empirical evaluation in Sect. 6. Our proposal, in fact, is inspired by subjective logic, which approximates Bayesian reasoning via a *least commitment principle*, i.e., matching the expected values, but then maximising the variance. Contrarily, in our approach not only we match the expected values but also the variances.

A subjective opinion about a proposition X is a tuple $\omega_X = \langle b_X, d_X, u_X, a_X \rangle$, representing the belief, disbelief and uncertainty that X is true at a given instance, and, as above, a_X is the prior probability that X is true in the absence of observations. These values are non-negative and $b_X + d_X + u_X = 1$. The projected probability $p(x) = b_X + u_X \cdot a_X$, provides an estimate of the ground truth probability θ .

The mapping from a beta-distributed random variable X with parameters $\alpha_X = \langle \alpha_x, \alpha_{\bar{x}} \rangle$ to a subjective opinion is:

$$\omega_X = \left\langle \frac{\alpha_x - Wa_X}{s_X}, \frac{\alpha_{\bar{x}} - W(1 - a_X)}{s_X}, \frac{W}{s_X}, a_X \right\rangle \quad (24)$$

With this transformation, the mean of X is equivalent to the projected probability $p(x)$, and the Dirichlet strength is inversely proportional to the uncertainty of the opinion:

$$\mathbb{E}[X] = p(x) = b_X + u_X a_X, \quad s_X = \frac{W}{u_X} \quad (25)$$

Conversely, a subjective opinion ω_X translates directly into a beta-distributed random variable with:

$$\alpha_X = \left\langle \frac{W}{u_X} b_X + Wa_X, \frac{W}{u_X} d_X + W(1 - a_X) \right\rangle \quad (26)$$

⁷ This is also needed by Subjective Logic (SL) (Jøsang, 2016) discussed in Sect. B, from which this work was inspired.

Subjective logic is a framework that includes various operators to indirectly determine opinions from various logical operations. In particular, we will make use of \boxplus_{SL} , \boxtimes_{SL} , and \boxdiv_{SL} , resp. summing, multiplying, and dividing two subjective opinions as they are defined in (Jøsang, 2016) (Appendix B). Those operators aim at faithfully matching the projected probabilities: for instance the multiplication of two subjective opinions $\omega_x \boxtimes_{SL} \omega_y$ results in an opinion ω_z such that $p(z) = p(x) \cdot p(y)$.

4 AMC-conditioning parametrisation with strong independence assumptions

Building upon our previous work (Cerutti et al., 2019), we allow manipulation of imprecise probabilities as labels in our circuits. Figure 3 shows an example of the circuits we will be manipulating, where probabilities from the circuit depicted in Fig. 1 have been replaced by uncertain probabilities represented as beta-distributed random variables and formalised as SL opinions, in a shorthand format listing only belief and uncertainty values.

4.1 SL AMC-conditioning parametrisation with strong independence assumptions

The straightforward approach, we first introduced in (Cerutti et al., 2019), to derive an AMC-conditioning parametrisation under complete independence assumptions at each step of the evaluation of the probabilistic circuit using subjective logic, is to use the operators \boxplus , \boxtimes , and \boxdiv . This gives rise to the SL AMC-conditioning parametrisation \mathcal{S}_{SL} , defined as follows:

$$\begin{aligned}
 \mathcal{A}_{SL} &= \mathbb{R}_{\geq 0}^4 \\
 a \oplus_{SL} b &= \begin{cases} a & \text{if } b = e^{\oplus_{SL}} \\ b & \text{if } a = e^{\oplus_{SL}} \\ a \boxplus_{SL} b & \text{otherwise} \end{cases} \\
 a \otimes_{SL} b &= \begin{cases} a & \text{if } b = e^{\otimes_{SL}} \\ b & \text{if } a = e^{\otimes_{SL}} \\ a \boxtimes_{SL} b & \text{otherwise} \end{cases} \\
 e^{\oplus_{SL}} &= \langle 0, 1, 0, 0 \rangle \\
 e^{\otimes_{SL}} &= \langle 1, 0, 0, 1 \rangle \\
 \rho_{SL}(f_i) &= \langle b_{f_i}, d_{f_i}, u_{f_i}, a_{f_i} \rangle \\
 \rho_{SL}(\neg f_i) &= \langle d_{f_i}, b_{f_i}, u_{f_i}, 1 - a_{f_i} \rangle \\
 a \circ_{SL} b &= \begin{cases} a & \text{if } b = e^{\otimes_{SL}} \\ a \boxdiv_{SL} b & \text{if defined} \\ \langle 0, 0, 1, 0.5 \rangle & \text{otherwise} \end{cases}
 \end{aligned}
 \tag{27}$$

Note that $\langle \mathcal{A}_{SL}, \oplus_{SL}, \otimes_{SL}, e^{\oplus_{SL}}, e^{\otimes_{SL}} \rangle$ does not form a commutative semiring in general. If we consider only the projected probabilities—i.e. the means of the associated beta

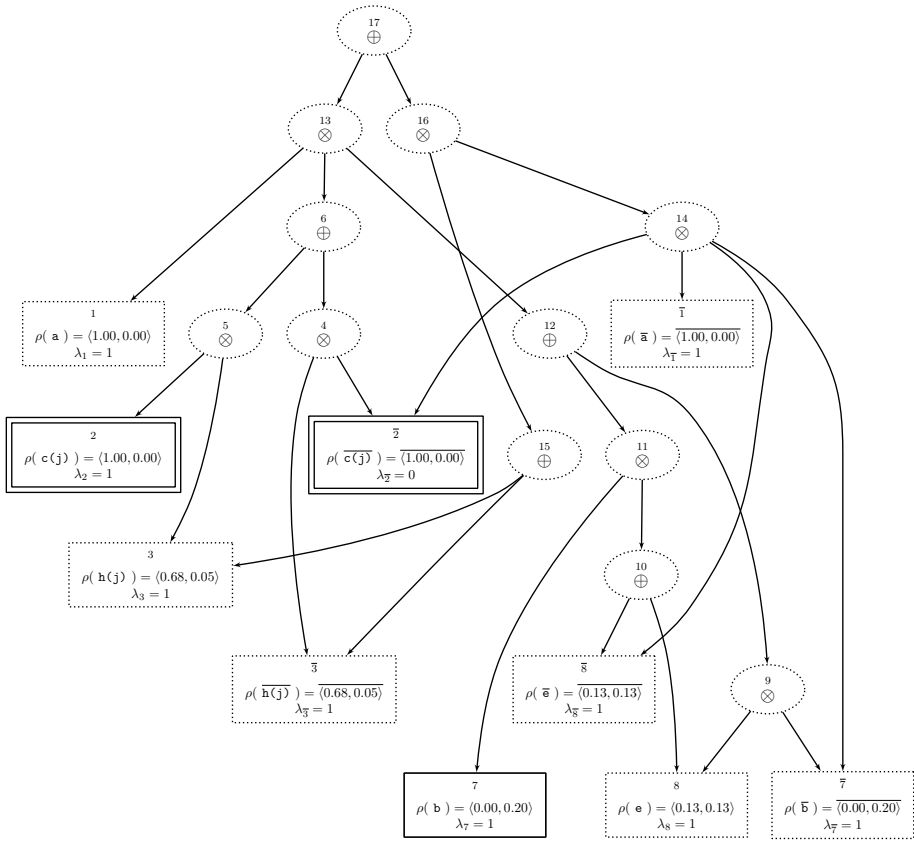


Fig. 3 Variation on the circuit represented in Fig. 1 with leaves labelled with imprecise probabilities represented as Subjective Logic opinions, listing only b_X and u_X : $d_X = 1 - b_X - u_X$, and $a_X = 0.5$. Solid box for query, double box for evidence

distributions—then \boxplus and \boxtimes are indeed commutative, associative, and \boxtimes distributes over \boxplus . However, the uncertainty of the resulting opinion depends on the order of operands.

4.2 Moment Matching AMC-conditioning parametrisation with strong independence assumptions

In (Cerutti et al., 2019) we derived another set of operators operating with moment matching: they aim at maintaining a stronger connection to beta distribution as the result of the manipulation. Indeed, while SL operators try to faithfully characterise the projected probabilities, they employ an uncertainty maximisation principle to limit the belief commitments, hence they have a looser connection to the beta distribution. Instead, in (Cerutti et al., 2019) we first represented beta distributions (and thus also SL opinions) not parametric in α , but rather parametric on mean and variance. Hence we proposed operators that manipulate means and variances, and then we transformed them back into beta distributions by moment matching.

In (Cerutti et al., 2019) we chose to represent all labels—not only of leaves—as beta distributions. Since the sum (and in the following the product as well) of two beta random variables is not necessarily a beta random variable, we follow (Kaplan and Ivanovska, 2018) and approximate the result as a beta distribution via moment matching on mean and variance.

Given X and Y independent beta-distributed random variables represented by the subjective opinion ω_X and ω_Y , the *sum* of X and Y ($\omega_X \boxplus^\beta \omega_Y$) is defined as the beta-distributed random variable Z such that:

$$\mathbb{E}[Z] = \mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{28}$$

and

$$\sigma_Z^2 = \sigma_{X+Y}^2 = \sigma_X^2 + \sigma_Y^2. \tag{29}$$

$\omega_Z = \omega_X \boxplus^\beta \omega_Y$ can then be obtained as discussed in Sect. 3, taking (23) into consideration. The same applies for the following operators as well.

The product operator between two independent beta-distributed random variables X and Y is then defined as the beta-distributed random variable Z such that $\mathbb{E}[Z] = \mathbb{E}[XY]$ and $\sigma_Z^2 = \sigma_{XY}^2$. Given X and Y independent beta-distributed random variables represented by the subjective opinion ω_X and ω_Y , the *product* of X and Y ($\omega_X \boxtimes^\beta \omega_Y$) is defined as the beta-distributed random variable Z such that:

$$\mathbb{E}[Z] = \mathbb{E}[XY] = \mathbb{E}[X] \mathbb{E}[Y] \tag{30}$$

and

$$\sigma_Z^2 = \sigma_{XY}^2 = \sigma_X^2(\mathbb{E}[Y])^2 + \sigma_Y^2(\mathbb{E}[X])^2 + \sigma_X^2\sigma_Y^2. \tag{31}$$

Finally, the conditioning-division operator between two independent beta-distributed random variables X and Y , represented by subjective opinions ω_X and ω_Y , is the beta-distributed random variable Z such that $\mathbb{E}[Z] = \mathbb{E}\left[\frac{X}{Y}\right]$ and $\sigma_Z^2 = \sigma_{\frac{X}{Y}}^2$. Given $\omega_X = \langle b_X, d_X, u_X, a_X \rangle$ and $\omega_Y = \langle b_Y, d_Y, u_Y, a_Y \rangle$ subjective opinions such that X and Y are beta-distributed random variables, the *conditioning-division* of X by Y ($\omega_X \boxdiv^\beta \omega_Y$) is defined as the beta-distributed random variable Z such that:

$$\mathbb{E}[Z] = \mathbb{E}\left[\frac{X}{Y}\right] = \mathbb{E}[X] \mathbb{E}\left[\frac{1}{Y}\right] \simeq \frac{\mathbb{E}[X]}{\mathbb{E}[Y]} \tag{32}$$

and⁸

$$\sigma_Z^2 \simeq (\mathbb{E}[Z])^2(1 - \mathbb{E}[Z])^2 \left(\frac{\sigma_X^2}{(\mathbb{E}[X])^2} + \frac{\sigma_Y^2 + \sigma_X^2}{(\mathbb{E}[Y] - \mathbb{E}[X])^2} + \frac{2\sigma_X^2}{\mathbb{E}[X](\mathbb{E}[Y] - \mathbb{E}[X])} \right) \tag{33}$$

Similarly to (27), the moment matching AMC-conditioning parametrisation \mathcal{S}^β is defined as follows:

⁸ Please note that (33) corrects a typo that is present in its version in (Cerutti et al., 2019).

$$\begin{aligned}
\mathcal{A}^\beta &= \mathbb{R}_{\geq 0}^4 \\
a \oplus^\beta b &= a \boxplus^\beta b \\
a \otimes^\beta b &= a \boxtimes^\beta b \\
e^{\oplus^\beta} &= \langle 1, 0, 0, 0.5 \rangle \\
e^{\otimes^\beta} &= \langle 0, 1, 0, 0.5 \rangle \\
\rho^\beta(f_i) &= \langle b_{f_i}, d_{f_i}, u_{f_i}, a_{f_i} \rangle \in [0, 1]^4 \\
\rho^\beta(\neg f_i) &= \langle d_{f_i}, b_{f_i}, u_{f_i}, 1 - a_{f_i} \rangle \\
a \circlearrowleft^\beta b &= a \boxdot^\beta b
\end{aligned} \tag{34}$$

As per (27), also $\langle \mathcal{A}^\beta, \oplus^\beta, \otimes^\beta, e^{\oplus^\beta}, e^{\otimes^\beta} \rangle$ is not in general a commutative semiring. Means are correctly matched to projected probabilities, therefore for them \mathcal{S}^β actually operates as a semiring. However, for what concerns variance, by using (31) and (29)—thus under independence assumption—the product is not distributive over addition: $\text{var}[X(Y + Z)] = \text{var}[X](\mathbb{E}[Y] + \mathbb{E}[Z])^2 + (\text{var}[Y] + \text{var}[Z])\mathbb{E}[X]^2 + \text{var}[X](\text{var}[Y] + \text{var}[Z]) \neq \text{var}[X](\mathbb{E}[Y]^2 + \mathbb{E}[Z]^2) + (\text{var}[Y] + \text{var}[Z])\mathbb{E}[X]^2 + \text{var}[X](\text{var}[Y] + \text{var}[Z]) = \text{var}[(XY) + (XZ)]$.

To illustrate the discrepancy, let's consider node 6 in Fig. 3: the disjunction operator there is summing up probabilities that are not statistically independent, despite the independence assumption used in developing the operator. Due to the dependencies between nodes in the circuit, the error grows during propagation, and then the numerator and denominator in the conditioning operator exhibit strong correlation due to redundant operators. Therefore, (33) introduces further error leading to an overall inadequate characterisation of variance. The next section reformulates the operations to account for the existing correlations.

5 CPB: covariance-aware probabilistic inference with beta-distributed random variables

We now propose an entirely novel approach to the AMC-conditioning problem that considers the covariances between the various distributions we are manipulating. Indeed, our approach for computing Covariance-aware Probabilistic entailment with beta-distributed random variables $\boxed{\text{CPB}}$ is designed to satisfy the total probability theorem, and in particular to enforce that for any X and Y beta-distributed random variables,

$$\text{var}[(Y \otimes X) \oplus (Y \otimes \bar{X})] = \text{var}[Y] \tag{35}$$

Algorithm 3 provides an overview of $\boxed{\text{CPB}}$, that comprises three stages: (1) pre-processing; (2) circuit shadowing; and (3) evaluation. In particular, we associate each node in the circuit with a beta distribution that gives us a distribution of values between 0 and 1 that can be interpreted as a measure of imprecise probabilities, i.e., a *second-order probability*. The determination of the distributions is through moment matching via the first and second moments through (18) and (20). Effectively, the collection of nodes are treated as

Table 1 Associative table for the aProbLog code in Listing 2

Identifier	Beta parameters	Subjective Logic opinion
ω_1	Beta(∞ , 1)	$\langle 1.00, 0.00, 0.00, 0.50 \rangle$
$\overline{\omega_1}$	Beta(1, ∞)	$\langle 0.00, 1.00, 0.00, 0.50 \rangle$
ω_2	Beta(2, 18)	$\langle 0.05, 0.85, 0.10, 0.50 \rangle$
$\overline{\omega_2}$	Beta(18, 2)	$\langle 0.85, 0.05, 0.10, 0.50 \rangle$
ω_3	Beta(2, 8)	$\langle 0.10, 0.70, 0.20, 0.50 \rangle$
$\overline{\omega_3}$	Beta(8, 2)	$\langle 0.70, 0.10, 0.20, 0.50 \rangle$
ω_4	Beta(3.5, 1.5)	$\langle 0.50, 0.10, 0.40, 0.50 \rangle$
$\overline{\omega_4}$	Beta(1.5, 3.5)	$\langle 0.10, 0.50, 0.40, 0.50 \rangle$

multivariate Gaussian characterised by a mean vector and covariance matrix that it computed via the propagation process described below. When analysing the distribution for particular node (via marginalisation of the Gaussian), it is approximated via the best-fitting beta distribution through moment-matching.

5.1 Pre-processing

We assume that the circuit we are receiving has the leaves labelled with unique identifiers of beta-distributed random variables. We also allow for the specification of the covariance matrix between the beta-distributed random variables, bearing in mind that $\text{cov}[X, 1 - X] = -\text{var}[X]$, cf. (18) and (19). In our running example, we assume the ProbLog code from Listing 1 has been transformed into the aProbLog⁹ code in Listing 2.

We also expect there is a table associating the identifier with the actual value of the beta-distributed random variable. In the following, we assume that ω_1 is a reserved indicator for the Beta(∞ , 1.00) (in Subjective Logic term $\langle 1.0, 0.0, 0.0, 0.5 \rangle$). For instance, Table 1 provides the associations for code in Listing 2, and Table 2 the covariance matrix for those beta-distributed random variables that we assume being learnt from complete observations of independent random variables, and hence the posterior beta-distributed random variables are also independent (cf. Appendix C).

Algorithm 3 Solving the **PROB** problem on a circuit N_A labelled with identifier of beta-distributed random variables and the associative table A , and covariance matrix C_A .

```

1: procedure COVPROBBETA( $N_A$ ,  $C_A$ )
2:    $\widehat{N}_A := \text{SHADOWCIRCUIT}(N_A)$ 
3:   return EVALCOVPROBBETA( $\widehat{N}_A$ ,  $C_A$ )
4: end procedure

```

⁹ aProbLog (Kimmig et al., 2011) is the algebraic version of ProbLog that allows for arbitrary labels to be used.

Table 2 Covariance matrix for the associative table (Table 1) under the assumption that all the beta-distributed random variables are independent each other. We use a short-hand notation for clarity: $\sigma_i^2 = \text{cov}[\omega_i]$. Zeros are omitted

	ω_1	$\overline{\omega_1}$	ω_2	$\overline{\omega_2}$	ω_3	$\overline{\omega_3}$	ω_4	$\overline{\omega_4}$
ω_1	σ_1^2	$-\sigma_1^2$						
$\overline{\omega_1}$	$-\sigma_1^2$	σ_1^2						
ω_2			σ_2^2	$-\sigma_2^2$				
$\overline{\omega_2}$			$-\sigma_2^2$	σ_2^2				
ω_3					σ_3^2	$-\sigma_3^2$		
$\overline{\omega_3}$					$-\sigma_3^2$	σ_3^2		
ω_4							σ_4^2	$-\sigma_4^2$
$\overline{\omega_4}$							$-\sigma_4^2$	σ_4^2

```

1  $\omega_2::\text{burglary}.$ 
2  $\omega_3::\text{earthquake}.$ 
3  $\omega_4::\text{hears\_alarm}(\text{john}).$ 
4  $\text{alarm} :- \text{burglary}.$ 
5  $\text{alarm} :- \text{earthquake}.$ 
6  $\text{calls}(\text{john}) :- \text{alarm}, \text{hears\_alarm}(\text{john}).$ 
7  $\text{evidence}(\text{calls}(\text{john})).$ 
8  $\text{query}(\text{burglary}).$ 

```

Listing 2: Prolog code for the Burglary example with unique identifier for the random variables associated to the database, originally Example 6 in (Fierens et al., 2015)

5.2 Circuit shadowing

We then augment the circuit adding *shadow* nodes to superimpose a second circuit to enable the possibility to assess, in a single forward pass, both $p(\text{query} \wedge \text{evidence})$ and $p(\text{evidence})$. This can provide a benefit time-wise at the expense of memory, but more importantly it simplifies the bookkeeping of indexes in the covariance matrix as we will see below. The pseudocode is provided in Appendix D, Algorithm 5.

Figure 4 depicts the result of such an algorithm applied to our running example. The algorithm begins by focusing on the node that identifies the negation of the query we want to evaluate with this circuit, that we mark with $\text{QNODE}(\overline{N_A})$:¹⁰ indeed, to evaluate $p(\text{query} \wedge \text{evidence})$, the $\lambda_{\text{QNODE}(\overline{N_A})}$ parameter for such a node must be set to 0. In Fig. 4, $\text{QNODE}(\overline{N_A}) = \overline{7}$. The algorithm then superimposes a new circuit by creating *shadow nodes*, e.g. \widehat{c} , that will represent random variables affected by the change in the $\lambda_{\text{QNODE}(\overline{N_A})}$ parameter. Figure 4 depicts the $\overline{7}$ node right next to its shadow $\widehat{7}$. The algorithm then allocates new nodes for each and every node that would be affected by this change in $\lambda_{\text{QNODE}(\overline{N_A})}$: in Fig. 4, nodes 9, 12, 13, 14, 16, and 17.

¹⁰ In this paper we focus on a query composed by a single literal.

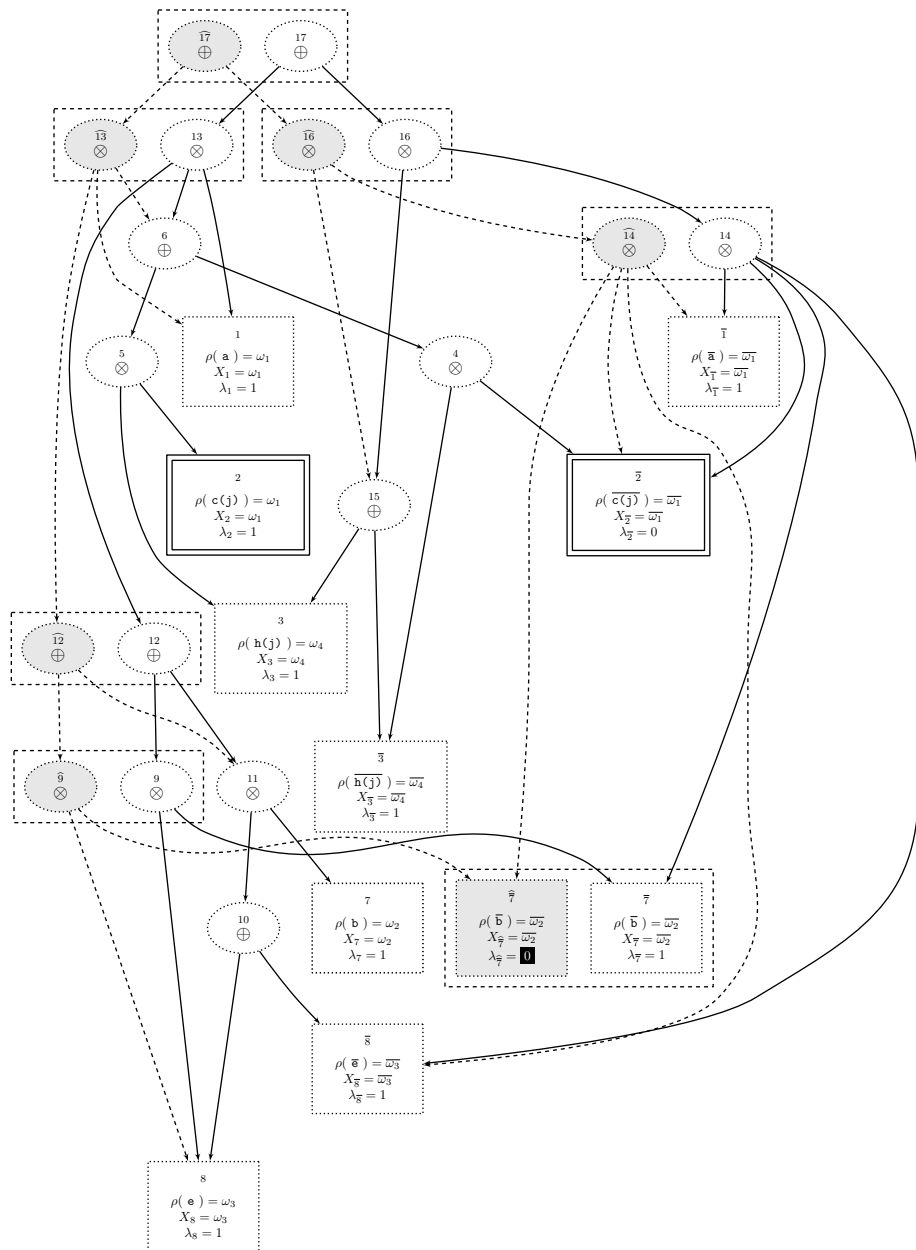


Fig. 4 Shadowing of the circuit represented in Fig. 1 according to Algorithm 5. Solid box for query, double box for evidence, in grey the shadow nodes added to the circuit. If a node has a shadow, they are grouped together with a dashed box. Dashed arrows connect shadow nodes to their children

5.3 Evaluating the shadowed circuit

Each of the nodes in the shadowed circuit (e.g. Fig. 4) has associated a (beta-distributed) random variable. In the following, and in Algorithm 4, given a node n , its associated random variable is identified as X_n . For the nodes for which exists a ρ label, its associated random variable is the beta-distributed random variable labelled via the ρ function, cf. Fig. 4.

Algorithm 4 takes a shadowed circuit and a covariance matrix, to then output means and variance of a beta-distributed random variable that approximates the probabilistic evaluation of a given query, see (44) and (45) below.

Algorithm 4 Evaluating the shadowed circuit \widehat{N}_A taking into consideration the given C_A covariance matrix.

```

1: procedure EVALCOVPROBBETA( $\widehat{N}_A, C_A$ )
2:    $means := ZEROS(|\widehat{N}_A|, 1)$ 
3:    $cov := ZEROS(|\widehat{N}_A|, |\widehat{N}_A|)$ 
4:    $nvisited := \{QNODE(\widehat{N}_A)\}$ 
5:   for  $n \in LEAVES(\widehat{N}_A) \setminus QNODE(\widehat{N}_A)$  do
6:      $nvisited := nvisited \cup \{n\}$ 
7:      $tvar := 0$ 
8:     if  $\lambda_n = 1$  then
9:        $means[n] := \mathbb{E}[X_n]$ 
10:       $tvar := \text{var}[X_n]$ 
11:     else  $means[n] := 0$ 
12:     end if
13:     for  $n' \in LEAVES(\widehat{N}_A) \setminus QNODE(\widehat{N}_A)$  do
14:        $cov[n, n'] := C_A[X_n, X_{n'}]$ 
15:     end for
16:   end for
17:    $nqueue := \widehat{N}_A \setminus nvisited$ 
18:   while  $nqueue \neq \emptyset$  do
19:      $n := n \in nqueue$  s.t.  $CHILDREN(\widehat{N}_A, n) \subseteq nvisited$ 
20:      $nqueue := nqueue \setminus \{n\}$ 
21:      $nvisited := nvisited \cup \{n\}$ 
22:     if  $n$  is a (shadowed) disjunction over  $C := CHILDREN(\widehat{N}_A, n)$  then
23:        $means[n] := \sum_{c \in C} means[X_c]$ 
24:        $cov[n, n] := \sum_{c \in C} \sum_{c' \in C} cov[c, c']$ 
25:        $cov[z, n] := cov[n, z] := \sum_{c \in C} cov[c, z] \quad \forall z \in \widehat{N}_A \setminus \{n\}$ 
26:     else if  $n$  is a (shadowed) conjunction over  $C := CHILDREN(\widehat{N}_A, n)$  then
27:        $means[n] := \prod_{c \in C} means[X_c]$ 
28:        $cov[n, n] := \sum_{c \in C} \sum_{c' \in C} \frac{means[X_n]^2}{means[X_c] means[X_{c'}]} cov[c, c']$ 
29:        $cov[z, n] := cov[n, z] := \sum_{c \in C} \frac{means[X_n]}{means[X_c]} cov[c, z] \quad \forall z \in \widehat{N}_A \setminus \{n\}$ 
30:     end if
31:   end while
32:    $r := \text{ROOT}(\widehat{N}_A)$ 
33:   return  $\left\langle \frac{means[\hat{r}]}{means[r]}, \frac{1}{means[r]^2} cov[\hat{r}, \hat{r}] + \frac{means[\hat{r}]^2}{means[r]^4} cov[r, r] - 2 \frac{means[\hat{r}]}{means[r]^3} cov[\hat{r}, r] \right\rangle$ 
34: end procedure

```

Algorithm 4 begins with building a vector of means ($means$), and a matrix of covariances (cov) of the random variables associated to the leaves of the circuit (lines 2–16) derived from the C_A covariance matrix provided as input. At lines 2 and 3 we make use of

a support function $ZEROS(X, Y)$ that returns a matrix of X rows and Y columns filled with zeroes: when $Y = 1$, this is equivalently a vector of X values. The algorithm can also be modified to handle the case where C_A is in this case, assuming independence among the variables, it is straightforward to obtain a matrix such as Table 2.

Then, Algorithm 4 proceeds to compute the means and covariances for all the remaining nodes in the circuit (lines 17–31). Here two cases arise.

Let n be a \oplus -gate over C nodes, its children: hence (lines 22–35)

$$\mathbb{E}[X_n] = \sum_{c \in C} \mathbb{E}[X_c], \tag{36}$$

$$\text{cov}[X_n] = \sum_{c \in C} \sum_{c' \in C} \text{cov}[X_c, X_{c'}], \tag{37}$$

$$\text{cov}[X_n, X_z] = \sum_{c \in C} \text{cov}[X_c, X_z] \text{ for } z \in \widehat{N}_A \setminus \{n\} \tag{38}$$

with

$$\text{cov}[X, Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] \tag{39}$$

and $\text{cov}[X] \equiv \text{cov}[X, X] = \text{var}[X]$.

Let n be a \otimes -gate over C nodes, its children (lines 26–30). Following (Benaroya et al., 2005, §4.3.2) we perform a Taylor approximation: let's assume $X_n = \Pi(\mathbf{X}_C) = \prod_{c \in C} X_c$, with

$\mathbf{X}_C = (X_{c_1}, \dots, X_{c_k})^T$ and $k = |C|$.

Expanding the first two terms of the Taylor series about $\mathbb{E}[\mathbf{X}_C]$ yields:

$$\begin{aligned} X_n &\simeq \Pi(\mathbb{E}[\mathbf{X}_C]) + (\mathbf{X}_C - \mathbb{E}[\mathbf{X}_C])^T \nabla \Pi(\mathbf{X}_C) \Big|_{\mathbf{X}_C = \mathbb{E}[\mathbf{X}_C]} \\ &\simeq \mathbb{E}[X_n] + (X_{j_1} - \mathbb{E}[X_{c_1}]) \prod_{j \in C \setminus \{c_1\}} \mathbb{E}[X_c] + \dots + (X_{j_k} - \mathbb{E}[X_{c_k}]) \prod_{j \in C \setminus \{c_k\}} \mathbb{E}[X_c] \\ &= \mathbb{E}[X_n] + \sum_{c \in C} \frac{\prod_{c' \in C} \mathbb{E}[X_{c'}]}{\mathbb{E}[X_c]} (X_c - \mathbb{E}[X_c]) \\ &= \mathbb{E}[X_n] + \sum_{c \in C} \frac{\mathbb{E}[X_n]}{\mathbb{E}[X_c]} (X_c - \mathbb{E}[X_c]) \end{aligned} \tag{40}$$

Taking the expectation of both leads to approximating $\mathbb{E}[X_n]$ as $\Pi(\mathbb{E}[\mathbf{X}_C])$.

Using this approximation, then (lines 34–42 of Algorithm 4)

$$\text{cov}[X_n] \simeq \sum_{c \in C} \sum_{c' \in C} \frac{\mathbb{E}[X_n]^2}{\mathbb{E}[X_c]\mathbb{E}[X_{c'}]} \text{cov}[X_c, X_{c'}], \tag{41}$$

$$\text{cov}[X_n, X_z] \simeq \sum_{c \in C} \frac{\mathbb{E}[X_n]}{\mathbb{E}[X_c]} \text{cov}[X_c, X_z] \text{ for } z \in \widehat{N}_A \setminus \{n\}. \tag{42}$$

Finally, Algorithm 4 computes a conditioning between X_r and $X_{\hat{r}}$, with r being the root of the circuit ($r := \text{ROOT}(\widehat{N}_A)$ at line 46). This shows how critical is to keep track of the non-zero covariances where they exist. The Taylor series approximation of X_r and $\frac{1}{X_{\hat{r}}}$ about $\mathbb{E}[X_{\hat{r}}]$ and $\frac{1}{\mathbb{E}[X_r]}$ leads to

$$\frac{X_{\hat{r}}}{X_r} \simeq \frac{\mathbb{E}[X_{\hat{r}}]}{\mathbb{E}[X_r]} + \frac{1}{X_{\hat{r}}}(X_{\hat{r}} - \mathbb{E}[X_{\hat{r}}]) - \frac{\mathbb{E}[X_{\hat{r}}]}{\mathbb{E}[X_r]^2}(X_r - \mathbb{E}[X_r]), \quad (43)$$

which implies

$$\mathbb{E}\left[\frac{X_{\hat{r}}}{X_r}\right] \simeq \frac{\mathbb{E}[X_{\hat{r}}]}{\mathbb{E}[X_r]}, \quad (44)$$

$$\text{cov}\left[\frac{X_{\hat{r}}}{X_r}\right] \simeq \frac{1}{\mathbb{E}[X_r]^2} \text{cov}[X_{\hat{r}}] + \frac{\mathbb{E}[X_{\hat{r}}]^2}{\mathbb{E}[X_r]^4} \text{cov}[X_r] - 2 \frac{\mathbb{E}[X_{\hat{r}}]}{\mathbb{E}[X_r]^3} \text{cov}[X_{\hat{r}}, X_r]. \quad (45)$$

Tables 3 and 4 depict respectively the non-zero values of the *means* vector and *cov* matrix for our running example. Overall, the mean and variance for $p(\text{burglary}|\text{calls}(\text{john}))$ are 0.3571 and 0.0528, respectively. Figure 5 depicts the resulting beta-distributed random variable (solid line) against a Monte Carlo simulation.

5.4 Scalability and memory performance

Algorithm 3 returns the mean and variance of the probability for the query conditioned on the evidence. Algorithm 5 adds shadow nodes to the initial circuit formed by the evidence to avoid redundant computations in the second pass. For the sake of clarity, Algorithm 4 is presented in its most simple form. As formulated, it requires a $|\widehat{N}_A| \times |\widehat{N}_A|$ array to store the covariance values between the nodes. For large circuits, this memory requirement can significantly slow down the processing (e.g., disk swaps) or simply become prohibitive. The covariances of a particular node are only required after it is computed via lines 24–25 or 34–35 in Algorithm 4. Furthermore, these covariances are no longer needed once all the parent node values have been computed. Thus, it is straightforward to dynamically allocate/de-allocate portions of the covariance array as needed. In fact, the selection of node n to compute in line 19, which is currently arbitrary, can be designed to minimise processing time in light of the resident memory requirements for the covariance array. Such an optimisation depends on the computing architecture and complicates the presentation. Thus, further details are beyond the scope of this paper.

6 Experimental results

6.1 The benefits of considering covariances

To illustrate the benefits of Algorithm 3 (Sect. 5), we run an experimental analysis involving several circuits with unspecified labelling function. For each circuit, first labels are derived for the case of parametrisation \mathcal{S}_p (5) by selecting the ground truth probabilities from a uniform random distribution. Then, for each label, we derive a set of subjective opinions by observing N_{ins} instantiations of a random variable derived from the chosen probability, so to simulate data sparsity (Kaplan and Ivanovska 2018).

We then proceed analysing the inference on specific query nodes q in the presence of a set of evidence $E = e$ using:

Table 3 Means as computed by Algorithm 4 on our running example. In grey the shadow nodes. Values very close or equal to zero are omitted. Also, values for nodes labelled with negated variables are omitted. $\bar{7}$, i.e. the shadow of $Q_{NODE}(N_A)$, is included for illustration purposes

	X_5	X_6	$X_{\bar{7}}$	X_9	X_{10}	X_{11}	X_{12}	$X_{\bar{12}}$	X_{13}	$X_{\bar{13}}$	X_{15}	X_{17}	$X_{\bar{17}}$
μ	0.7	0.7	0	0.18	1	0.1	0.28	0.1	0.196	0.07	1	0.196	0.07

Table 4 Covariances ($\times 10^{-2}$) as computed by Algorithm 4 on our running example. In grey the shadow nodes. Values very close or equal to zero are omitted. Also, values for nodes labelled with negated variables are omitted. $\bar{7}$, i.e. the shadow of $Q_{NODE}(N_A)$, is included for illustration purposes

	X_3	X_5	X_6	X_7	$X_{\bar{7}}$	X_8	X_9	X_{11}	X_{12}	$X_{\bar{12}}$	X_{13}	$X_{\bar{13}}$	X_{17}	$X_{\bar{17}}$
X_3	3.5	3.5	3.5								1.0	0.4	1.0	0.4
X_5	3.5	3.5	3.5								1.0	0.4	1.0	0.4
X_6	3.5	3.5	3.5								1.0	0.4	1.0	0.4
X_7				0.4			-0.1	0.4	0.3	0.4	0.2	0.3	0.2	0.3
$X_{\bar{7}}$														
X_8						1.5	1.3		1.3		0.9		0.9	
X_9				-0.1		1.3	1.2	-0.1	1.1	-0.1	0.8	-0.1	0.8	-0.1
X_{11}				0.4			-0.1	0.4	0.3	0.4	0.2	0.3	0.2	0.3
X_{12}				0.3		1.3	1.1	0.3	1.5	0.3	1.0	0.2	1.0	0.2
$X_{\bar{12}}$				0.4			-0.1	0.4	0.3	0.4	0.2	0.3	0.2	0.3
X_{13}	1.0	1.0	1.0	0.2		0.9	0.8	0.2	1.0	0.2	1.0	0.3	1.0	0.3
$X_{\bar{13}}$	0.4	0.4	0.4	0.3			-0.1	0.3	0.2	0.3	0.3	0.3	0.3	0.2
X_{17}	1.0	1.0	1.0	0.2		0.9	0.8	0.2	1.0	0.2	1.0	0.3	1.0	0.3
$X_{\bar{17}}$	0.4	0.4	0.4	0.3			-0.1	0.3	0.2	0.3	0.3	0.2	0.3	0.3

- **CPB** as articulated in Sect. 5;¹¹
- S^β , cf. (34);
- S_{SL} , cf. (27);
- **MC**, a Monte Carlo analysis with 100 samples from the derived random variables to obtain probabilities, and then computing the probability of queries in presence of evidence using the parametrisation S_p .

¹¹ Source code is available at <https://github.com/federicocerutti/CPB>.

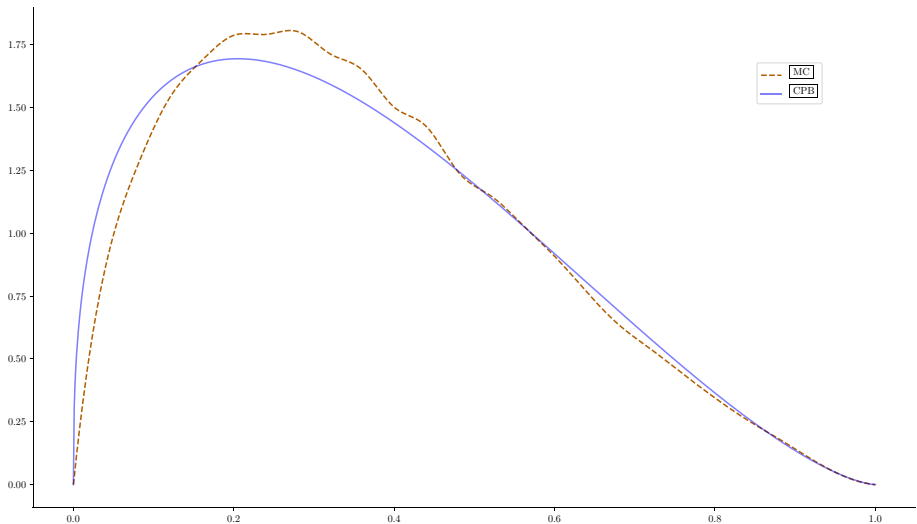


Fig. 5 Resulting distribution of probabilities for our running example using Algorithm 3 (solid line), and a Monte Carlo simulation with 100,000 samples grouped in 25 bins and then interpolated with a cubic polynomial (dashed line)

We then compare the RMSE to the actual ground truth. This process of inference to determine the marginal beta distributions is repeated 1000 times by considering 100 random choices for each label of the circuit, i.e. the ground truth, and for each ground truth 10 repetitions of sampling the interpretations used to derive the subjective opinion labels observing N_{ins} instantiations of all the variables.

We judge the quality of the beta distributions of the queries on how well its expression of uncertainty captures the spread between its projected probability and the actual ground truth probability, as also Kaplan and Ivanovska (2018) did. In simulations where the ground truths are known, such as ours, confidence bounds can be formed around the projected probabilities at a significance level of γ and determine the fraction of cases when the ground truth falls within the bounds. If the uncertainty is well determined by the beta distributions, then this fraction should correspond to the strength γ of the confidence interval (Kaplan & Ivanovska, 2018, Appendix C).

```

1  ω1::stress(X) :- person(X).
2  ω1::influences(X,Y) :- person(X), person(Y).
3  smokes(X) :- stress(X).
4  smokes(X) :- friend(X,Y), influences(Y,X), smokes(Y).
5  ω3::asthma(X) :- smokes(X).
6  person(1).
7  person(2).
8  person(3).
9  person(4).
10 friend(1,2).
11 friend(2,1).
12 friend(2,4).
13 friend(3,2).
14 friend(4,2).
15 evidence(smokes(2),true).
16 evidence(influences(4,2),false).
17 query(smokes(1)).
18 query(smokes(3)).
19 query(smokes(4)).
20 query(asthma(1)).
21 query(asthma(2)).
22 query(asthma(3)).
23 query(asthma(4)).

```

Listing 3: Smoker and Friends aProbLog code

Following (Cerutti et al. 2019), we consider the famous Friends & Smokers problem, cf. Listing 3,¹² with fixed queries and set of evidence. Table 5 provides the root mean square error (RMSE) between the projected probabilities and the ground truth probabilities for all the inferred query variables for $N_{ins} = 10, 50, 100$. The table also includes the predicted RMSE by taking the square root of the average—over the number of runs—variances from the inferred marginal beta distributions, cf. (18). Figure 6 plots the desired and actual significance levels for the confidence intervals (best closest to the diagonal), i.e. the fractions of times the ground truth falls within confidence bounds set to capture x%. Finally, Fig. 8 depicts the correlation of Dirichlet strengths between the Monte Carlo approach [MC] running with variable number of samples and the golden standard (i.e. a Monte Carlo run with 10,000 samples), as well as between the golden standard and [CPB], which is clearly independent of the number of samples used in MC. We, however, rephrased the sentence to clarify it. of the number of samples used in the Monte Carlo approach [MC]. Given X_q^g (resp. X_q) the random variable associated to the queries q computed using the golden standard (resp. computed using either [MC] or [CPB]), the Pearson's correlation coefficient displayed in Fig. 8 is given by:

$$r = \frac{\text{cov}[s_{X_q^g}, s_{X_q}]}{\sqrt{\text{cov}[s_{X_q^g}] \text{cov}[s_{X_q}]}} \quad (46)$$

¹² https://dtai.cs.kuleuven.be/problog/tutorial/basic/05_smokers.html (on 29th April 2020).

Table 5 RMSE for the queried variables in the Friends & Smokers program: A stands for Actual, P for Predicted. Best results—also considering hidden decimals—for the actual RMSE boxed. MC has been run over 100 samples

			N_{ins}	CPB	S^β	S_{SL}	MC
Friends & Smokers	10	A		0.1065	0.1065	0.1198	0.1072
		P		0.1024	0.1412	0.1060	0.1027
	50	A		0.0489	0.0489	0.0617	0.0490
		P		0.0491	0.0898	0.0587	0.0489
	100	A		0.0354	0.0354	0.0521	0.0355
		P		0.0357	0.0709	0.0487	0.0356

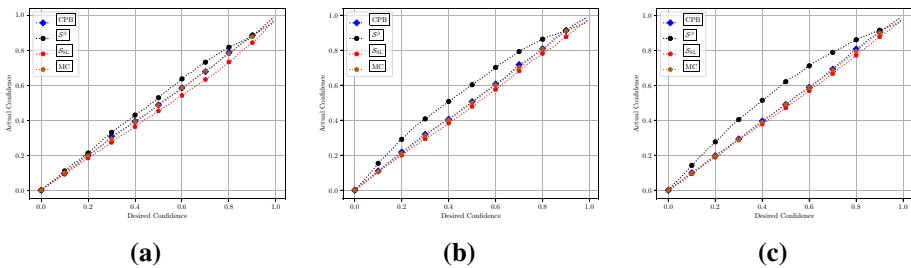


Fig. 6 Actual versus desired significance of bounds derived from the uncertainty for Smokers & Friends with: (a) $N_{ins} = 10$; (b) $N_{ins} = 50$; and (c) $N_{ins} = 100$. Best closest to the diagonal. MC has been run over 100 samples

This is a measure of the quality of the epistemic uncertainty associated with the evaluation of the circuit using MC with varying number of samples, and CPB: the closer the Dirichlet strengths are to those of the golden standard, the better the computed epistemic uncertainty represents the actual uncertainty,¹³ hence the closer the correlations are to 1 in Fig. 8 the better.

From Table 5, CPB exhibits the lowest RMSE and the best prediction of its own RMSE. As already noticed in (Cerutti et al., 2019), S^β is a little conservative in estimating its own RMSE, while S_{SL} is overconfident. This is reflected in Fig. 6, with the results of S^β being over the diagonal, and those of S_{SL} being below it, while CPB sits exactly on the diagonal, like also MC. However, MC with 100 samples does not exhibit the lowest RMSE according to Table 5, although the difference with the best one is much lower compared with S_{SL} .

Considering the execution time, Fig. 7, we can see that there is a substantial difference between CPB and MC with 100 samples.

¹³ The Dirichlet strengths are inversely proportional to the epistemic uncertainty.

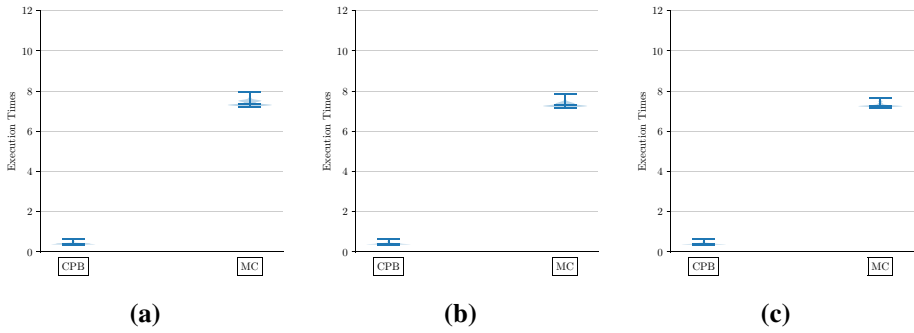


Fig. 7 Distribution of execution time for running the different algorithms for Smokers & Friends with: (a) $N_{ins} = 10$; (b) $N_{ins} = 50$; and (c) $N_{ins} = 100$. Best lowest. **MC** has been run over 100 samples

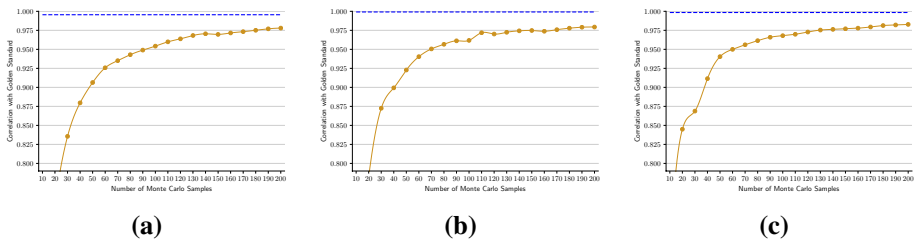


Fig. 8 Correlation of Dirichlet strengths between runs of **MC** varying the number of samples and golden standard (i.e. a Monte Carlo run with 10,000 samples) as well as between **CPB** and golden standard with cubic interpolation—that is independent of the number of samples used in **MC**—for Smokers & Friends with: (a) $N_{ins} = 10$; (b) $N_{ins} = 50$; and (c) $N_{ins} = 100$

Finally, Fig. 8 depicts the correlation of the Dirichlet strength between the golden standard, i.e. a Monte Carlo simulation with 10,000 samples, and both **CPB** and **MC**, this last one varying the number of samples used. It is straightforward to see that **MC** improves the accuracy of the computed epistemic uncertainty when increasing the number of samples considered, approaching the same level of **CPB** when considering more than 200 samples.

6.2 Comparison with other approaches for dealing with uncertain probabilities

```

1  ω2 :: n1.
2  ω3 :: n2 :- \+n1.
3  ω4 :: n2 :- n1.
4  ω5 :: n3 :- \+n2.
5  ω6 :: n3 :- n2.
6  ω7 :: n4 :- \+n2.
7  ω8 :: n4 :- n2.
8  ω9 :: n5 :- \+n3.
9  ω10 :: n5 :- n3.
10 ω11 :: n6 :- \+n3.
11 ω12 :: n6 :- n3.
12 ω13 :: n7 :- \+n6.
13 ω14 :: n7 :- n6.
14 ω15 :: n8 :- \+n5.
15 ω16 :: n8 :- n5.
16 ω17 :: n9 :- \+n5.
17 ω18 :: n9 :- n5.
18 evidence(n1, e1).
19 evidence(n4, e2).
20 evidence(n7, e3).
21 evidence(n8, e4).
22 evidence(n9, e5).
23 query(n2).
24 query(n3).
25 query(n5).
26 query(n6).

```

Listing 4: An example of aProblog code that can be seen also as a Bayesian network, cf. Fig. 12a in Appendix E. e_i are randomly assigned as either True or False.

To compare our approach against the state-of-the-art approaches for reasoning with uncertain probabilities, following (Cerutti et al., 2019) we restrict ourselves to the case of circuits representing inferences over a Bayesian network. For instance, Listing 4 shows an aProblog code that can also be interpreted as a Bayesian network. We considered three circuits and their Bayesian network representation: Net1 (Listing 4); Net2; and Net3. Figure 12 in Appendix E depicts the Bayesian networks that can be derived from such circuits. In the following, we will refer to NetX as both the circuit and the Bayesian network without distinction. We then compared [CPB] against three approaches specifically designed for dealing with uncertain probabilities in Bayesian networks: Subjective Bayesian Networks; Belief Networks; and Credal Networks.

Subjective Bayesian Network [SBN] (Ivanovska et al., 2015; Kaplan & Ivanovska, 2016; Kaplan & Ivanovska, 2018), was first proposed in (Ivanovska et al., 2015), and it is an uncertain Bayesian network where the conditionals are subjective opinions instead of dogmatic probabilities. In other words, the conditional probabilities are known within a beta distribution. [SBN] uses subjective belief propagation (SBP), which was introduced for trees in (Kaplan & Ivanovska 2016) and extended for singly-connected networks in (Kaplan & Ivanovska, 2018), that extends the Belief Propagation (BP) inference method of Pearl (1986). In BP, π - and λ -messages are passed from parents and children, respectively, to a node, i.e., variable. The node uses these messages to formulate the inferred marginal probability of the corresponding variable. The node also uses these messages to determine

the π - and λ -messages to send to its children and parents, respectively. In SBP, the π - and λ -messages are subjective opinions characterised by a projected probability and Dirichlet strength. The SBP formulation approximates output messages as beta-distributed random variables using the methods of moments and a first-order Taylor series approximation to determine the mean and variance of the output messages in light of the beta-distributed input messages. The details of the derivations are provided in (Kaplan & Ivanovska, 2016; Kaplan & Ivanovska, 2018).

Belief Networks [GBT] Smets (1993) introduced a computationally efficient method to reason over networks via Dempster-Shafer theory (Dempster, 1968). It is an approximation of a valuation-based system. Namely, a (conditional) subjective opinion $\omega_X = [b_x, b_{\bar{x}}, u_X]$ from our circuit obtained from data is converted to the following belief mass assignment: $m(x) = b_x$, $m(\bar{x}) = b_{\bar{x}}$ and $m(x \cup \bar{x}) = u_X$. Note that in the binary case, the belief function overlaps with the belief mass assignment. The method exploits the disjunctive rule of combination to compose beliefs conditioned on the Cartesian product space of the binary power sets. This enables both forward propagation and backward propagation after inverting the belief conditionals via the generalized Bayes' theorem (GBT). By operating in the Cartesian product space of the binary power sets, the computational complexity grows exponentially with respect to the number of parents.

Credal Networks [Credal] (Zaffalon & Fagioli, 1998). A credal network over binary random variables extends a Bayesian network by replacing single probability values with closed intervals representing the possible range of probability values. The extension of Pearl's message-passing algorithm by the 2U algorithm for credal networks is described in (Zaffalon & Fagioli, 1998). This algorithm works by determining the maximum and minimum value (an interval) for each of the target probabilities based on the given input intervals. It turns out that these extreme values lie at the vertices of the polytope dictated by the extreme values of the input intervals. As a result, the computational complexity grows exponentially with respect to the number of parents nodes. For the sake of comparison, we assume that the random variables we label our circuits with and elicited from the given data corresponds to a credal network in the following way: if $\omega_x = [b_x, b_{\bar{x}}, u_X]$ is a subjective opinion on the probability θ , then we have $[b_x, b_x + u_X]$ as an interval corresponding to this probability in the credal network. It should be noted that this mapping from the beta-distributed random variables to an interval is consistent with past studies of credal networks (Karlsson et al., 2008).

As before, Table 6 provides the root mean square error (RMSE) between the projected probabilities and the ground truth probabilities for all the inferred query variables for $N_{ins} = 10, 50, 100$, together with the RMSE predicted by taking the square root of the average variances from the inferred marginal beta distributions. Figure 9 plots the desired and actual significance levels for the confidence intervals (best closest to the diagonal). Figure 10 depicts the distribution of execution time for running the various algorithms, and Fig. 11 the correlation of the Dirichlet strength between the golden standard, i.e. a Monte Carlo simulation with 10,000 samples, and both [CPB] and [MC] varying the number of samples.

Table 6 shows that [CPB] shares the best performance with the state-of-the-art [SBN] and $[S^\beta]$ almost constantly. This is clearly a significant achievement considering that [SBN] is the state-of-the-art approach when dealing only with single connected Bayesian Networks with uncertain probabilities, while we can also handle much more complex problems. Consistently with Table 5, and also with (Cerutti et al., 2019), $[S^\beta]$ has lower RMSE

Table 6 RMSE for the queried variables in the various networks: A stands for Actual, P for Predicted. Best results—also considering hidden decimals—for the Actual RMSE boxed. $\boxed{\text{MC}}$ has been run over 100 samples

N_{ins}			$\boxed{\text{CPB}}$	$\boxed{S^\beta}$	S_{SL}	$\boxed{\text{MC}}$	$\boxed{\text{SBN}}$	$\boxed{\text{GBT}}$	$\boxed{\text{Credal}}$
Net1	10	A	0.1511	$\boxed{0.1511}$	0.2078	0.1517	0.1511	0.1542	0.1633
		P	0.1473	0.1864	0.1559	0.1465	0.1472	0.0873	0.2009
Net1	50	A	0.0816	$\boxed{0.0816}$	0.1237	0.0818	0.0816	0.0848	0.0827
		P	0.0802	0.1227	0.0825	0.0789	0.0794	0.0372	0.1069
Net1	100	A	$\boxed{0.0544}$	0.0544	0.0837	0.0550	0.0544	0.0601	0.0557
		P	0.0572	0.0971	0.0592	0.0564	0.0566	0.0262	0.0766
Net2	10	A	0.1389	$\boxed{0.1389}$	0.1916	0.1392	0.1389	0.1418	0.1473
		P	0.1391	0.1808	0.1457	0.1381	0.1399	0.1058	0.1856
Net2	50	A	$\boxed{0.0701}$	$\boxed{0.0701}$	0.1092	0.0702	$\boxed{0.0701}$	0.0730	0.0702
		P	0.0722	0.1148	0.0755	0.0714	0.0720	0.0486	0.0952
Net2	100	A	$\boxed{0.0534}$	0.0534	0.0901	0.0536	0.0534	0.0553	0.0537
		P	0.0533	0.0937	0.0601	0.0526	0.0531	0.0340	0.0696
Net3	10	A	$\boxed{0.1481}$	0.1481	0.2160	0.1488	0.1481	0.1511	0.1634
		P	0.1453	0.1708	0.1578	0.1438	0.1454	0.0821	0.1947
Net3	50	A	0.0737	0.0737	0.1167	0.0741	$\boxed{0.0737}$	0.0760	0.0756
		P	0.0777	0.1115	0.0780	0.0763	0.0772	0.0348	0.1003
Net3	100	A	$\boxed{0.0574}$	0.0574	0.0909	0.0578	0.0574	0.0608	0.0582
		P	0.0564	0.0882	0.0584	0.0553	0.0560	0.0239	0.0728

than $\boxed{S_{SL}}$ and it seems that $\boxed{S^\beta}$ overestimates the predicted RMSE and $\boxed{S_{SL}}$ underestimates it as $\boxed{S_{SL}}$ predicts smaller error than is realised and vice versa for $\boxed{S^\beta}$.

From visual inspection of Fig. 9, it is evident that $\boxed{\text{CPB}}$, $\boxed{\text{SBN}}$, and $\boxed{\text{MC}}$ all are very close to the diagonal, thus correctly assessing their own epistemic uncertainty. $\boxed{S^\beta}$ performance is heavily affected by the fact that it computes the conditional distributions at the very end of the process and it relies, in (33), on the assumption of independence. $\boxed{\text{CPB}}$, keeping track of the covariance between the various nodes in the circuits, does not suffer from this problem. This positive result has been achieved without substantial deterioration of the performance in terms of execution time, as displayed in Fig. 10, for which the same commentary of Fig. 7 applies.

Finally, Fig. 11 depicts the correlation of the Dirichlet strength between the golden standard, i.e. a Monte Carlo simulation with 10,000 samples, and both $\boxed{\text{CPB}}$ and $\boxed{\text{MC}}$, this last one varying the number of samples used. Like for Fig. 8, it is straightforward to see that $\boxed{\text{MC}}$ improves the accuracy of its computed epistemic uncertainty when increasing the number of samples considered, approaching the same level of $\boxed{\text{CPB}}$ when considering more than 200 samples, while $\boxed{\text{CPB}}$ performs very closely to the optimal value of 1.

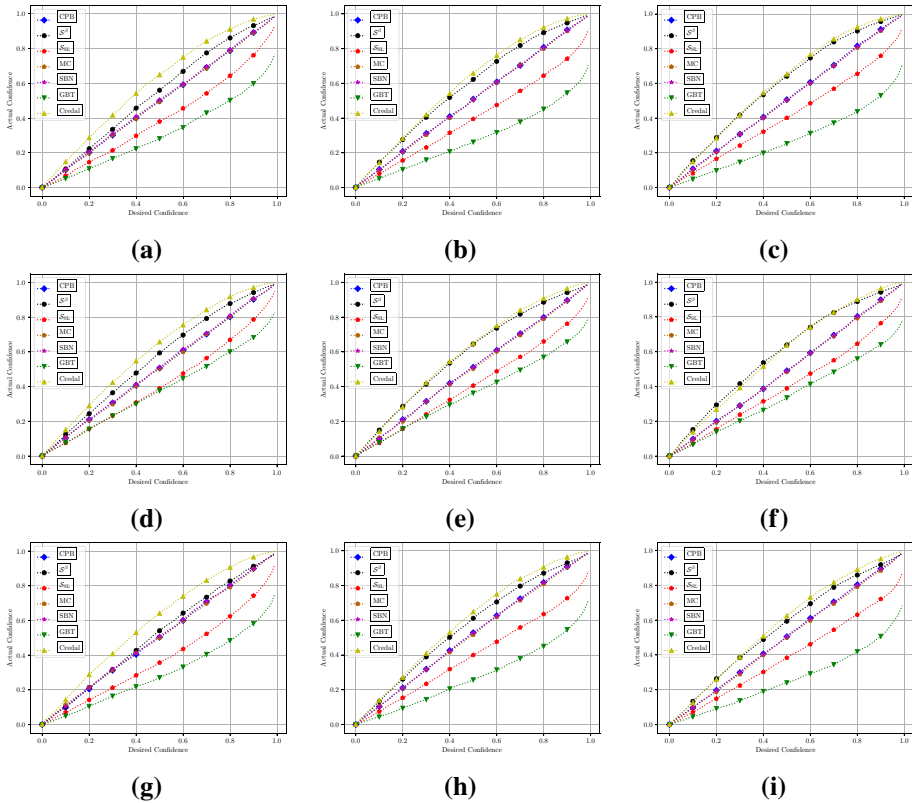


Fig. 9 Actual versus desired significance of bounds derived from the uncertainty for: (a) Net1 with $N_{ins} = 10$; (b) Net1 with $N_{ins} = 50$; (c) Net1 with $N_{ins} = 100$; (d) Net2 with $N_{ins} = 10$; (e) Net2 with $N_{ins} = 50$; (f) Net2 with $N_{ins} = 100$; (g) Net3 with $N_{ins} = 10$; (h) Net3 with $N_{ins} = 50$; (i) Net3 with $N_{ins} = 100$. Best closest to the diagonal. MC has been run over 100 samples

7 Conclusion

In this paper, we introduce (Sect. 5) an algorithm for reasoning over a probabilistic circuit whose leaves are labelled with beta-distributed random variables, with the additional piece of information describing which of those are actually independent (Sect. 5.1). This provides the input to an algorithm that *shadows* the circuit derived for computing the probability of the *pieces of evidence* by superimposing a second circuit modified for computing the probability of a given *query and the pieces of evidence*, thus having all the necessary components for computing the probability of a query conditioned on the pieces of evidence (Sect. 5.2). This is essential when evaluating such a shadowed circuit (Sect. 5.3), with the covariance matrix playing an essential role by keeping track of the dependencies between random variables while they are manipulated within the circuit. We also include discussions on memory management in Sect. 5.4.

In our extensive experimental analysis (Sect. 6) we compare against leading approaches to compute uncertain probabilities, notably: (1) Monte Carlo sampling; (2) our previous proposal (Cerutti et al., 2019) as representative of the family of approaches using a moment matching approach with strong independence assumptions; (3) Subjective Logic

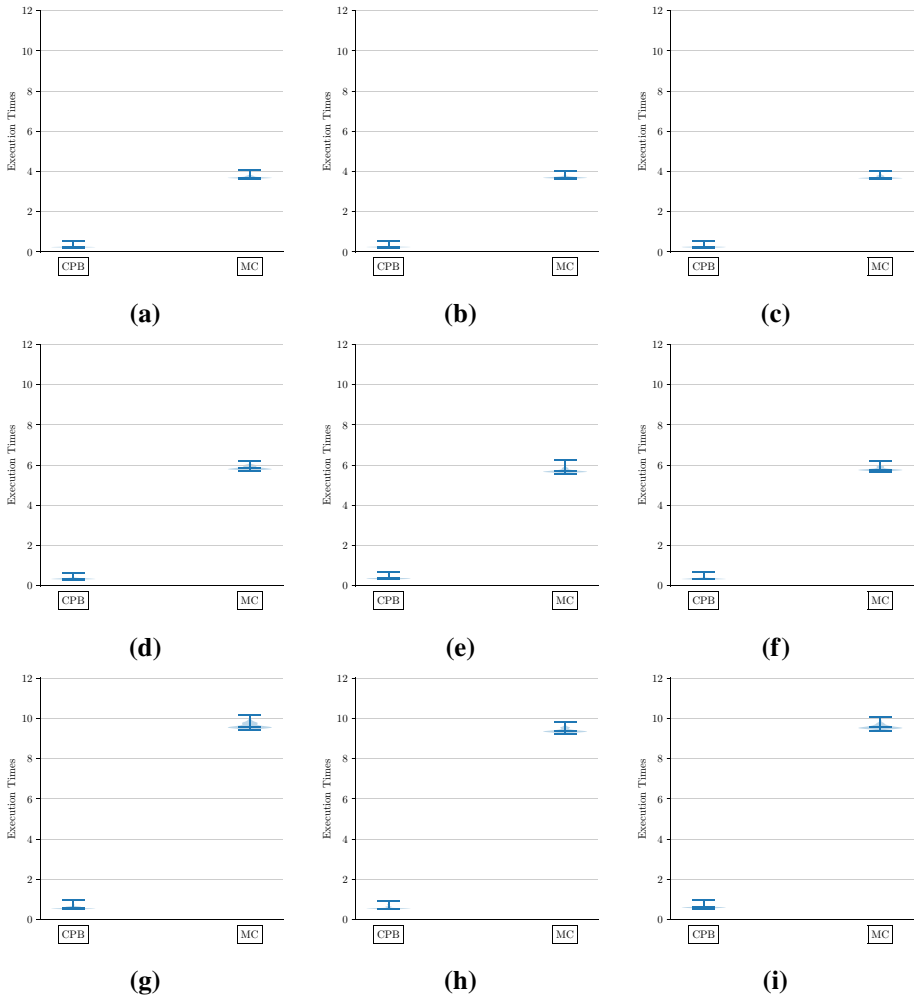


Fig. 10 Distribution of computational time for running the different algorithms for: (a) Net1 with $N_{ins} = 10$; (b) Net1 with $N_{ins} = 50$; (c) Net1 with $N_{ins} = 100$; (d) Net2 with $N_{ins} = 10$; (e) Net2 with $N_{ins} = 50$; (f) Net2 with $N_{ins} = 100$; (g) Net3 with $N_{ins} = 10$; (h) Net3 with $N_{ins} = 50$; (i) Net3 with $N_{ins} = 100$. MC has been run over 100 samples

(Jøsang, 2016); (4) Subjective Bayesian Network (SBN) (Ivanovska et al., 2015; Kaplan & Ivanovska, 2016; Kaplan & Ivanovska, 2018); (5) Dempster-Shafer Theory of Evidence (Dempster, 1968; Smets, 1993); and (6) credal networks (Zaffalon and Fagioli 1998).

We achieve the same or better results of state-of-the-art approaches for dealing with epistemic uncertainty, including highly engineered ones for a narrow domain such as SBN, while being able to handle general probabilistic circuits and with just a modest increase in the computational effort. In fact, this work has inspired us to leverage probabilistic circuits to expand second-order inference for SBN for arbitrary directed acyclic graphs whose variables are multinomials. As part of future work, we will expand our experimental investigation to consider larger models, also leveraging recent advancements in engineering

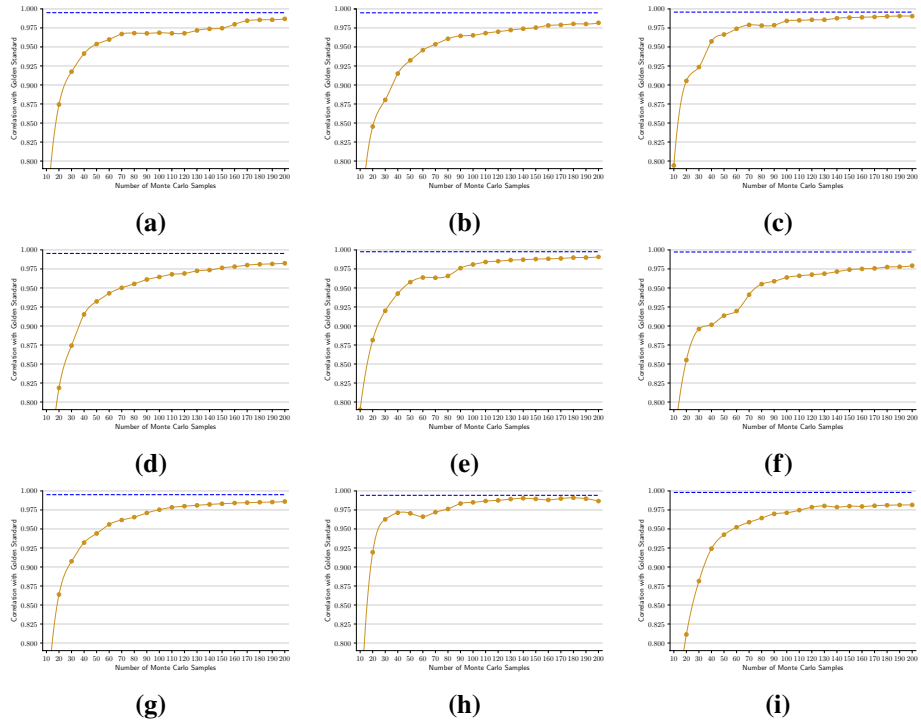


Fig. 11 Correlation of Dirichlet strengths between runs of `MC` varying the number of samples and golden standard (i.e. a Monte Carlo run with 10,000 samples) as well as between `CPB` and golden standard with cubic interpolation—that is independent of the number of samples used in `MC`—for: (a) Net1 with $N_{ins} = 10$; (b) Net1 with $N_{ins} = 50$; (c) Net1 with $N_{ins} = 100$; (d) Net2 with $N_{ins} = 10$; (e) Net2 with $N_{ins} = 50$; (f) Net2 with $N_{ins} = 100$; (g) Net3 with $N_{ins} = 10$; (h) Net3 with $N_{ins} = 50$; (i) Net3 with $N_{ins} = 100$

highly-efficient procedures over probabilistic circuits, e.g. (Peharz et al., 2020). However, as also highlighted in Figs. 7 and 10 our research-grade prototype is substantially faster than using Monte Carlo sampling for estimating variances. Indeed, we can estimate it from just one pass over the circuit (see Algorithm 4), while a Monte Carlo approach would need to go through the circuit once for each sample.

We focused our attention on probabilistic circuits derived from \mathcal{d} -DNNFs: work by Darwiche (2011), and then also by Kisa et al. (2014) has introduced Sentential Decision Diagrams (SDDs) as a new canonical formalism respectively for propositional and for probabilistic circuits. However, as we can read in (Darwiche, 2011, p. 819) SDDs is a strict subset of \mathcal{d} -DNNF, which is thus the least constrained type of propositional circuit we can safely rely on according to (Kimmig et al., 2017, Theorem 4). However, in future work we will enable our approach to efficiently make use of SDDs.

In addition, we will also work in the direction of enabling learning with partial observations—incomplete data where the instantiations of each of the propositional variables are not always visible over all training instantiations—on top of its ability of tracking the covariance values between the various random variables for a better estimation of epistemic uncertainty.

A. aProbLog

In the last years, several probabilistic variants of Prolog have been developed, such as ICL (Poole, 2000), Dyna (Eisner et al., 2005), PRISM (Sato and Kameya, 2001) and ProbLog (De Raedt et al. 2007), with its aProbLog extension (Kimmig et al., 2011) to handle arbitrary labels from a semiring. They all are based on definite clause logic (pure Prolog) extended with facts labelled with probability values. Their meaning is typically derived from Sato's distribution semantics (Sato, 1995), which assigns a probability to every literal. The probability of a Herbrand interpretation, or possible world, is the product of the probabilities of the literals occurring in this world. The success probability is the probability that a query succeeds in a randomly selected world.

For a set J of ground facts, we define the set of literals $L(J)$ and the set of interpretations $\mathcal{I}(J)$ as follows:

$$L(J) = J \cup \{\neg f \mid f \in J\} \quad (47)$$

$$\mathcal{I}(J) = \{S \mid S \subseteq L(J) \wedge \forall l \in J : l \in S \leftrightarrow \neg l \notin S\} \quad (48)$$

An algebraic Prolog (aProbLog) program (Kimmig et al., 2011) consists of:

- a commutative semiring $\langle \mathcal{A}, \oplus, \otimes, e^\oplus, e^\otimes \rangle$
- a finite set of ground algebraic facts $F = \{f_1, \dots, f_n\}$
- a finite set BK of background knowledge clauses
- a labeling function $\rho : L(F) \rightarrow \mathcal{A}$

Background knowledge clauses are definite clauses, but their bodies may contain negative literals for algebraic facts. Their heads may not unify with any algebraic fact.

For instance, in the following aProbLog program

```
alarm :- burglary.
0.05 :: burglary.
```

`burglary` is an algebraic fact with label 0.05, and `alarm :- burglary` represents a background knowledge clause, whose intuitive meaning is: in case of burglary, the alarm should go off.

The idea of splitting a logic program in a set of facts and a set of clauses goes back to Sato's distribution semantics (Sato, 1995), where it is used to define a probability distribution over interpretations of the entire program in terms of a distribution over the facts. This is possible because a truth value assignment to the facts in F uniquely determines the truth values of all other atoms defined in the background knowledge. In the simplest case, as realised in ProbLog (De Raedt et al., 2007; Fierens et al., 2015), this basic distribution considers facts to be independent random variables and thus multiplies their individual probabilities. aProbLog uses the same basic idea, but generalises from the semiring of probabilities to general commutative semirings. While the distribution semantics is defined for countably infinite sets of facts, the set of ground algebraic facts in aProbLog must be finite.

In aProbLog, the label of a complete interpretation $I \in \mathcal{I}(F)$ is defined as the product of the labels of its literals

$$\mathbf{A}(I) = \bigotimes_{l \in I} \rho(l) \tag{49}$$

and the label of a set of interpretations $S \subseteq \mathcal{I}(F)$ as the sum of the interpretation labels

$$\mathbf{A}(S) = \bigoplus_{I \in S} \bigotimes_{l \in I} \rho(l) \tag{50}$$

A query q is a finite set of algebraic literals and atoms from the Herbrand base,¹⁴ $q \subseteq L(F) \cup HB(F \cup BK)$. We denote the set of interpretations where the query is true by $\mathcal{I}(q)$,

$$\mathcal{I}(q) = \{I \mid I \in \mathcal{I}(F) \wedge I \cup BK \models q\} \tag{51}$$

The label of query q is defined as the label of $\mathcal{I}(q)$,

$$\mathbf{A}(q) = \mathbf{A}(\mathcal{I}(q)) = \bigoplus_{I \in \mathcal{I}(q)} \bigotimes_{l \in I} \rho(l) \tag{52}$$

As both operators are commutative and associative, the label is independent of the order of both literals and interpretations.

ProbLog (Fierens et al., 2015) is an instance of aProbLog with

$$\begin{aligned} \mathcal{A} &= \mathbb{R}_{\geq 0}; \\ a \oplus b &= a + b; \\ a \otimes b &= a \cdot b; \\ e^\oplus &= 0; \\ e^\otimes &= 1; \\ \delta(f) &\in [0, 1]; \\ \delta(\neg f) &= 1 - \delta(f) \end{aligned} \tag{53}$$

B. Subjective logic operators of sum, multiplication, and division

Let us recall the following operators as defined in (Jøsang, 2016). In the following, let $\omega_X = \langle b_X, d_X, u_X, a_X \rangle$ and $\omega_Y = \langle b_Y, d_Y, u_Y, a_Y \rangle$ be two subjective logic opinions.

Sum

The opinion about $X \cup Y$ (**sum**, $\omega_X \boxplus_{SL} \omega_Y$) is defined as $\omega_{X \cup Y} = \langle b_{X \cup Y}, d_{X \cup Y}, u_{X \cup Y}, a_{X \cup Y} \rangle$, where:

- $b_{X \cup Y} = b_X + b_Y$;
- $d_{X \cup Y} = \frac{a_X(d_X - b_Y) + a_Y(d_Y - b_X)}{a_X + a_Y}$;
- $u_{X \cup Y} = \frac{a_X u_X + a_Y u_Y}{a_X + a_Y}$; and
- $a_{X \cup Y} = a_X + a_Y$.

¹⁴ I.e., the set of ground atoms that can be constructed from the predicate, functor and constant symbols of the program.

Product

The opinion about $X \wedge Y$ (**product**, $\omega_X \boxtimes_{\text{SL}} \omega_Y$) is defined—under assumption of independence—as $\omega_{X \wedge Y} = \langle b_{X \wedge Y}, d_{X \wedge Y}, u_{X \wedge Y}, a_{X \wedge Y} \rangle$, where:

- $b_{X \wedge Y} = b_X b_Y + \frac{(1-a_X)a_Y b_X u_Y + a_X(1-a_Y)u_X b_Y}{1-a_X a_Y}$;
- $d_{X \wedge Y} = d_X + d_Y - d_X d_Y$;
- $u_{X \wedge Y} = u_X u_Y + \frac{(1-a_Y)b_X u_Y + (1-a_X)u_X b_Y}{1-a_X a_Y}$; and
- $a_{X \wedge Y} = a_X a_Y$.

Division

The opinion about the division of X by Y , $X \tilde{\wedge} Y$ (**division**, $\omega_X \boxdiv_{\text{SL}} \omega_Y$) is defined as $\omega_{X \tilde{\wedge} Y} = \langle b_{X \tilde{\wedge} Y}, d_{X \tilde{\wedge} Y}, u_{X \tilde{\wedge} Y}, a_{X \tilde{\wedge} Y} \rangle$ where

- $b_{X \tilde{\wedge} Y} = \frac{a_Y(b_X + a_X u_X)}{(a_Y - a_X)(b_Y + a_Y u_Y)} - \frac{a_X(1-d_X)}{(a_Y - a_X)(1-d_Y)}$;
- $d_{X \tilde{\wedge} Y} = \frac{d_X - d_Y}{1-d_Y}$;
- $u_{X \tilde{\wedge} Y} = \frac{a_Y(1-d_X)}{(a_Y - a_X)(1-d_Y)} - \frac{a_Y(b_X + a_X u_X)}{(a_Y - a_X)(b_Y + a_Y u_Y)}$; and
- $a_{X \tilde{\wedge} Y} = \frac{a_X}{a_Y}$

subject to:

- $a_X < a_Y$; $d_X \geq d_Y$;
- $b_X \geq \frac{a_X(1-a_Y)(1-d_X)b_Y}{(1-a_X)a_Y(1-d_Y)}$; and
- $u_X \geq \frac{(1-a_Y)(1-d_X)u_Y}{(1-a_X)(1-d_Y)}$.

C. Independence of posterior distributions when learning from complete observations

Let us instantiate AMC using probabilities as labels (cf. (5)) and let us consider a propositional logic theory over M variables. We can thus re-write (1) as:

$$p(T) = \sum_{I \in \mathcal{M}(T)} \prod_{m=1}^M p(l_m) \tag{54}$$

Hence, the probability of a theory is function of the probabilities of interpretations $p(I \in \mathcal{M}(T))$, where

$$p(I \in \mathcal{M}(T)) = \prod_{m=1}^M p(l_m) \tag{55}$$

Let's assume that we want to learn such probabilities from a dataset $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$, then by (55) the variables for which we are learning probabilities are independent, hence

$$p(l_1, \dots, l_M) = \prod_{m=1}^M p(l_m) \quad (56)$$

We can thus re-write the likelihood (9) as:

$$\begin{aligned} p(\mathcal{D} | \mathbf{p}_x) &= \prod_{i=1}^{|\mathcal{D}|} p(x_i | \mathbf{p}_{x_i}) \\ &= \prod_{i=1}^{|\mathcal{D}|} \prod_{m=1}^M p_{x_m}^{x_{i,m}} (1 - p_{x_m})^{1-x_{i,m}} \end{aligned} \quad (57)$$

Assuming a uniform prior, and letting r_m be the number of observations for $x_m = 1$ and s_m the number of observations for $x_m = 0$, we can thus compute the posterior as:

$$\begin{aligned} p(\mathbf{p}_x | \mathcal{D}, \boldsymbol{\alpha}^0) &\propto p(\mathcal{D} | \mathbf{p}_x) \cdot p(\mathbf{p}_x | \boldsymbol{\alpha}^0) \\ &\propto \prod_{m=1}^M p_{x_m}^{r_m + \alpha_{x_m}^0 - 1} (1 - p_{x_m})^{s_m + \alpha_{x_m}^0 - 1} \end{aligned} \quad (58)$$

which, in turns, show that the independence is maintained also considering the posterior beta distributions.

D. Algorithm for shadowing a given circuit

In Algorithm 5 we make use of a stack data structure with associated pop and push functions (cf. lines 3, 5, 8, 16): that is for ease of presentation as the algorithm does not require a stack.

Algorithm 5 Shadowing the circuit N_A .

```

1: procedure SHADOWCIRCUIT( $N_A$ )
2:    $\widehat{N}_A := N_A$ 
3:    $links := \text{STACK}()$ 
4:   for  $p \in \text{PARENTS}(\widehat{N}_A, \text{QNODE}(\widehat{N}_A))$  do
5:      $\text{PUSH}(links, \langle \text{QNODE}(\widehat{N}_A), p \rangle)$ 
6:   end for
7:   while  $\neg \text{empty}(links)$  do
8:      $\langle c, p \rangle := \text{POP}(links)$ 
9:      $\widehat{N}_A := \widehat{N}_A \cup \{c\}$ 
10:    if  $\widehat{p} \notin \widehat{N}_A$  then
11:       $\widehat{N}_A := \widehat{N}_A \cup \{\widehat{p}\}$ 
12:       $\text{CHILDREN}(\widehat{N}_A, \widehat{p}) := \text{CHILDREN}(\widehat{N}_A, p)$ 
13:    end if
14:     $\text{CHILDREN}(\widehat{N}_A, \widehat{p}) := (\text{CHILDREN}(\widehat{N}_A, \widehat{p}) \setminus \{c\}) \cup \{c\}$ 
15:    for  $p' \in \text{PARENTS}(N_A, p)$  do
16:       $\text{PUSH}(links, \langle p, p' \rangle)$ 
17:    end for
18:  end while
19:  return  $\widehat{N}_A$ 
20: end procedure

```

E. Bayesian networks derived from aProbLog programs

Figure 12 depicts the Bayesian networks that can be derived from the three circuits considered in the experiments described in Sect. 6.2.

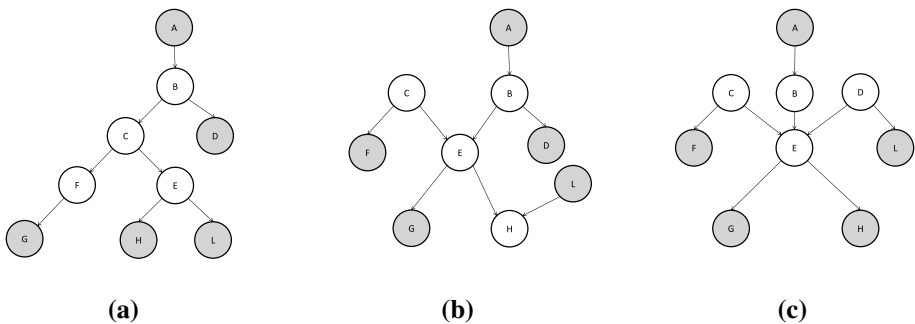


Fig. 12 Network structures tested where the exterior gray variables are directly observed and the remaining are queried: **(a)** Net1, a tree; **(b)** Net2, singly connected network with one node having two parents; **(c)** Net3, singly connected network with one node having three parents

Acknowledgements We thank the anonymous reviews whose comments improve the first draft submitted for consideration to this journal. This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This work was performed using the computational facilities of the Advanced Research Computing at Cardiff (ARCCA) Division, Cardiff University.

References


- Amershi, S., Weld, D., Vorvoreanu, M., Fournery, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., Horvitz, E. (2019). Guidelines for human-AI interaction. In *Conference on human factors in computing systems - proceedings, association for computing machinery*, New York, New York, USA, pp. 1–13. <https://doi.org/10.1145/3290605.3300233>.
- Anderson, R., Hare, N., Maskell, S. (2016). Using a bayesian model for confidence to make decisions that consider epistemic regret. In *19th International conference on information fusion*, pp. 264–269.
- Antonucci, A., Karlsson, A., Sundgren, D. (2014). Decision making with hierarchical credal sets. In A. Laurent, O. Strauss, B. Bouchon-Meunier, R. R. Yager (Eds.) *Information processing and management of uncertainty in knowledge-based systems*, pp. 456–465.
- Bacchus, F., Dalmao, S., & Pitassi, T. (2009). Solving #Sat and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34, 391–442. <https://doi.org/10.1613/jair.2648>.
- Bansal, G., Nushi, B., Kamar, E., Lasecki, W., Weld, D., Horvitz, E. (2019a). Beyond accuracy: The role of mental models in human-AI team performance. In *HCOMP, AAAI*, <https://www.microsoft.com/en-us/research/publication/beyond-accuracy-the-role-of-mental-models-in-human-ai-team-performance/>.
- Bansal, G., Nushi, B., Kamar, E., Weld, D. S., Lasecki, W. S., Horvitz, E. (2019b). Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *AAAI*, pp. 2429–2437. <https://doi.org/10.1609/aaai.v33i01.33012429>
- Baras, J. S., & Theodorakopoulos, G. (2010). Path problems in networks. *Synthesis Lectures on Communication Networks*, 3, 1–77. <https://doi.org/10.2200/S00245ED1V01Y201001CNT003>.
- Bellodi, E., & Riguzzi, F. (2013). Expectation maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis*, 17(2), 343–363.
- Benaroya, H., Han, S. M., Han, S. M., & Nagurka, M. (2005). *Probability models in engineering and science*. CRC Press.
- Bratko, I. (2001). *Prolog programming for artificial intelligence*. Addison Wesley.
- Cerutti, F., Kaplan, L. M., Kimmig, A., Sensoy, M. (2019). Probabilistic logic programming with beta-distributed random variables. In *The thirty-third AAAI conference on artificial intelligence, AAAI 2019, the thirty-first innovative applications of artificial intelligence conference, IAAI 2019, the ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019*, Honolulu, Hawaii, USA, January 27–February 1, 2019, AAAI Press, pp. 7769–7776, <https://doi.org/10.1609/aaai.v33i01.33017769>
- Cerutti, F., & Thimm, M. (2019). A general approach to reasoning with probabilities. *International Journal of Approximate Reasoning*, 111, 35–50. <https://doi.org/10.1016/j.ijar.2019.05.003>.
- Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6), 772–799.
- Choi, A., & Darwiche, A. (2013). Dynamic minimization of sentential decision diagrams. In *Proceedings of the 27th AAAI conference on artificial intelligence, AAAI 2013*, AAAI Press, AAAI'13, pp. 187–194.
- Darwiche, A. (2004). New advances in compiling CNF to decomposable negation normal form. In *Proceedings of the 16th European conference on artificial intelligence*, IOS Press, NLD, ECAI04, pp. 318–322.
- Darwiche, A. (2011). SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of the twenty-second international joint conference on artificial intelligence—Volume Two*, AAAI Press, IJCAI'11, pp. 819–826.
- Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1), 229–264.

- De Raedt, L., Kimmig, A., Toivonen, H. (2007). ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th international joint conference on artificial intelligence*, pp. 2462–2467. <https://irias.kuleuven.be/handle/123456789/146072>.
- Dempster, A. P. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society Series B (Methodological)* 30(2), 205–247. <http://www.jstor.org/stable/2984504>
- Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th annual meeting of the association for computational linguistics, association for computational linguistics*, Philadelphia, Pennsylvania, USA, pp. 1–8. <https://doi.org/10.3115/1073083.1073085>, <https://www.aclweb.org/anthology/P02-1001>
- Eisner, J., Goldlust, E., Smith, N. A. (2005). Compiling comp ling: Practical weighted dynamic programming and the dyna language. In *Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT '05*, pp. 281–290. <https://doi.org/10.3115/1220575.1220611>.
- Fierens, D., den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., et al. (2015). Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(03), 358–401. <https://doi.org/10.1017/S1471068414000076>.
- Friedman, T., den Broeck, G. (2018). Approximate knowledge compilation by online collapsed importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi., R. Garnett (Eds.) *Advances in neural information processing systems* 31, Curran Associates, Inc., pp. 8024–8034, <http://papers.nips.cc/paper/8026-approximate-knowledge-compilation-by-online-collapsed-importance-sampling.pdf>
- Gens, R., Domingos, P. (2013) Learning the structure of sum-product networks. In: S. Dasgupta, D. McAllester (Eds.) *30th International conference on machine learning, ICML 2013, PMLR, Atlanta, Georgia, USA, Proceedings of Machine Learning Research*, vol 28, pp. 1910–1917, <http://proceedings.mlr.press/v28/gens13.html>
- Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–606, <https://www.aclweb.org/anthology/J99-4004>
- Hora, S. C. (1996). Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety* 54(2), 217–223, treatment of Aleatory and Epistemic Uncertainty [https://doi.org/10.1016/S0951-8320\(96\)00077-4](https://doi.org/10.1016/S0951-8320(96)00077-4)
- Hüllermeier, E., & Waegeman, W. (2019). Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction. 1910.09457.
- Ivanovska, M., Jøsang, A., Kaplan, L., Sambo, F. (2015). Subjective networks: Perspectives and challenges. In *Proceedings of the 4th international workshop on graph structures for knowledge representation and reasoning, Buenos Aires, Argentina*, pp. 107–124.
- Jaini, P., Rashwan, A., Zhao, H., Liu, Y., Banijamali, E., Chen, Z., Poupart, P. (2016). Online algorithms for sum-product networks with continuous variables. In *Conference on probabilistic graphical models*, pp. 228–239.
- Jøsang, A. (2016). *Subjective logic: A formalism for reasoning under uncertainty*. Springer.
- Jøsang, A., Hayward, R., Pope, S. (2006). Trust network analysis with subjective logic. In *Proceedings of the 29th Australasian computer science conference-volume, 48*, pp. 85–94.
- Kaplan, L., Ivanovska, M. (2016). Efficient subjective Bayesian network belief propagation for trees. In *19th International conference on information fusion*, pp. 1300–1307.
- Kaplan, L., & Ivanovska, M. (2018). Efficient belief propagation in second-order Bayesian networks for singly-connected graphs. *International Journal of Approximate Reasoning*, 93, 132–152.
- Karlsson, A., Johansson, R., Andler, S. F. (2008). An empirical comparison of Bayesian and credal networks for dependable high-level information fusion. In *International conference on information fusion (FUSION)*, pp. 1–8.
- Kimmig, A., Van den Broeck, G., De Raedt, L. (2011). An algebraic prolog for reasoning about possible worlds. In *Proceedings of the twenty-fifth AAAI conference on artificial intelligence*, pp. 209–214, <http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3685>.
- Kimmig, A., Van den Broeck, G., & De Raedt, L. (2017). Algebraic model counting. *Journal of Applied Logic*, 22, 46–62. <https://doi.org/10.1016/j.jal.2016.11.031>.
- Kisa, D., den Broeck, G., Choi, A., Darwiche, A. (2014). Probabilistic sentential decision diagrams. In *Proceedings of the fourteenth international conference on principles of knowledge representation and reasoning, AAAI Press, KR'14*, pp. 558–567.
- Kocielnik, R., Amershi, S., Bennett, P. N. (2019). Will you accept an imperfect AI? Exploring designs for adjusting end-user expectations of AI systems. In *Conference on human factors in computing systems—proceedings, association for computing machinery*, New York, USA, pp. 1–14, <https://doi.org/10.1145/3290605.3300641>.

- Kowalski, R. A. (1988). The early years of logic programming. *Communications of the ACM*, 31(1), 38–43.
- Laplace, P. S. (1825). *A philosophical essay on probabilities*. Springer, translator Andrew I. Dale, Published in 1995.
- Liang, Y., Bekker, J., Van Den Broeck, G. (2017). Learning the structure of probabilistic sentential decision diagrams. In *Uncertainty in artificial intelligence - proceedings of the 33rd conference, UAI 2017*, <http://starai.cs.ucla.edu/papers/LiangUAI17.pdf>
- Moglia, M., Sharma, A. K., & Maheepala, S. (2012). Multi-criteria decision assessments using subjective logic: Methodology and the case of urban water strategies. *Journal of Hydrology*, 452–453, 180–189.
- Oztok, U., & Darwiche, A. (2015). A top-down compiler for sentential decision diagrams. In *Proceedings of the 24th international conference on artificial intelligence*, AAAI Press, IJCAI'15, pp. 3141–3148.
- Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3), 241–288.
- Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Van Den Broeck, G., Kersting, K., Ghahramani, Z. (2020). Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In H. D. Iii, A. Singh (Eds.) *Proceedings of the 37th international conference on machine learning*, PMLR, *proceedings of machine learning research*, vol. 119, pp. 7563–7574, <http://proceedings.mlr.press/v119/peharz20a.html>
- Poole, D. (2000). Abducting through negation as failure: Stable models within the independent choice logic. *The Journal of Logic Programming*, 44(1), 5–35. [https://doi.org/10.1016/S0743-1066\(99\)00071-0](https://doi.org/10.1016/S0743-1066(99)00071-0).
- Rashwan, A., Zhao, H., Poupart, P. (2016). Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Artificial intelligence and statistics*, pp 1469–1477.
- Rooshenas, A., & Lowd, D. (2014). Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st international conference on international conference on machine learning - volume 32, JMLR.org, ICML'14*, pp. I-710–I-718.
- Sang, T., Bearne, P., Kautz, H. (2005). Performing bayesian inference by weighted model counting. In *Proceedings of the 20th National Conference on Artificial Intelligence—volume 1*, pp. 475–481.
- Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th international conference on logic programming (ICLP-95)*.
- Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15(1), 391–454. <http://dl.acm.org/citation.cfm?id=1622845.1622858>
- Sensoy, M., Kaplan, L., Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *32nd Conference on neural information processing systems (NIPS 2018)*.
- Smets, P. (1993). Belief functions: The disjunctive rule of combination and the generalized Bayesian theorem. *International Journal of Approximate Reasoning*, 9, 1–35.
- Trapp, M., Peharz, R., Ge, H., Pernkopf, F., Ghahramani, Z. (2019). Bayesian learning of sum-product networks. In *Advances in neural information processing systems*, pp. 6344–6355.
- Van Allen, T., Singh, A., Greiner, R., & Hooper, P. (2008). Quantifying the uncertainty of a belief net response: Bayesian error-bars for belief net inference. *Artificial Intelligence*, 172(4), 483–513.
- Vergari, A., Di Mauro, N., & Esposito, F. (2015). Simplifying, regularizing and strengthening sum-product network structure learning. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, & C. Soares (Eds.), *Machine learning and knowledge discovery in databases* (pp. 343–358). Cham: Springer International Publishing.
- Vergari, A., Molina, A., Peharz, R., Ghahramani, Z., Kersting, K., & Valera, I. (2019). Automatic Bayesian density analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 5207–5215.
- Von Neumann, J., & Morgenstern, O. (2007). *Theory of games and economic behavior (commemorative edition)*. Princeton: Princeton University Press.
- von zur Gathen, J. (1988). Algebraic complexity theory. *Annual Review of Computer Science*, 3(1), 317–348. <https://doi.org/10.1146/annurev.cs.03.060188.001533>
- Xu, J., Zhang, Z., Friedman, T., Liang, Y., Van Den Broeck, G. (2018) A semantic loss function for deep learning with symbolic knowledge. In *35th International conference on machine learning, ICML 2018*, 12, 8752–8760, <http://starai.cs.ucla.edu/papers/XuICML18.pdf>
- Zaffalon, M., & Fagioli, E. (1998). 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 106(1), 77–107.
- Zhao, H., Adel, T., Gordon, G., Amos, B. (2016a). Collapsed variational inference for sum-product networks. In *International conference on machine learning*, pp. 1310–1318.
- Zhao, H., Poupart, P., Gordon, G. (2016b). A unified approach for learning the parameters of sum-product networks. In *Proceedings of the 30th international conference on neural information processing systems*, Curran Associates Inc., Red Hook, NY, USA, NIPS'16, pp. 433–441.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Federico Cerutti^{1,2}  · Lance M. Kaplan³ · Angelika Kimmig^{4,5} · Murat Şensoy^{6,7}

Lance M. Kaplan
lance.m.kaplan.civ@army.mil

Angelika Kimmig
angelika.kimmig@cs.kuleuven.be

Murat Şensoy
murat.sensoy@ozyegin.edu.tr

¹ Department of Information Engineering, University of Brescia, Brescia, Italy

² Crime and Security Research Institute, Cardiff University, Cardiff, UK

³ US DEVCOM Army Research Laboratory, Adelphi, MD, USA

⁴ Department of Computer Science, KU Leuven, Leuven, Belgium

⁵ Leuven.AI - KU Leuven Institute for AI, Leuven, Belgium

⁶ Blue Prism AI Labs, London, UK

⁷ Department of Computer Science, Ozyegin University, Istanbul, Turkey