Approximate Inference in Logical Credal Networks

Radu Marinescu[1] , **Haifeng Qian**[2] , **Alexander Gray**[1] , **Debarun Bhattacharjya**[1] , **Francisco Barahona**[1] , **Tian Gao**[1] and **Ryan Riegel**[1]

[1]IBM Research
[2]AWS AI Labs

radu.marinescu@ie.ibm.com

Abstract

Logical Credal Networks or LCNs is a recent probabilistic logic designed for effective aggregation and reasoning over multiple sources of imprecise knowledge. An LCN specifies a set of probability distributions over all interpretations of a set of logical formulas for which marginal and conditional probability bounds on their truth values are known. Inference in LCNs involves the exact solution of a non-convex non-linear program defined over an exponentially large number of non-negative real valued variables and, therefore, is limited to relatively small problems. In this paper, we present ARIEL – a novel iterative message-passing scheme for approximate inference in LCNs. Inspired by classical belief propagation for graphical models, our method propagates messages that involve solving considerably smaller local non-linear programs. Experiments on several classes of LCNs demonstrate clearly that ARIEL yields high quality solutions compared with exact inference and scales to much larger problems than previously considered.

1 Introduction

Many real-world applications require efficient handling of uncertainty and compact representations of a wide variety of knowledge. Graphical models such as Bayesian networks [Pearl, 1988] or Markov networks [Koller and Friedman, 2009] provide a powerful framework for reasoning about uncertainty while classical (first-order) logic can naturally be used to represent compactly complex concepts and relationships that comprise expert knowledge. Therefore, probabilistic logic which combines probability and logic in a principled manner has emerged over the years as a unified framework to deal effectively with these complex applications [Nilsson, 1986; Fagin *et al.*, 1990; Heinsohn, 1994; Jaeger, 1994; Andersen and Hooker, 1994; Chandru and Hooker, 1999; Dürig and Studer, 2005; Richardson and Domingos, 2006; Getoor and Taskar, 2007; De Raedt *et al.*, 2008]. While some of these logics (e.g., [Richardson and Domingos, 2006; Getoor and Taskar, 2007; De Raedt *et al.*, 2008]) associate a single real value to the logical formulas to represent the uncertainty around their truth values, others (e.g., [Nilsson, 1986; Fagin *et al.*, 1990]) relax this requirement and allow specifying lower and upper probability bounds on logical formulas.

In practice, it is often the case that multiple sources of imprecise knowledge need to be combined to solve a problem more effectively. For example, in a realistic credit card fraud detection application, a statistical model capturing the uncertainty around historical transaction data can be combined with probabilistic logic rules expressing imprecise expert knowledge about the domain in order to predict future fraudulent transactions more accurately [Li *et al.*, 2020]. Similarly, in chemo-informatics, a more effective structural analysis of molecular materials can leverage a combination of machine learning models based on molecular fingerprinting data and expert knowledge about certain structural properties of molecules represented by probabilistic logic rules.

Logical Credal Networks or LCNs [Marinescu *et al.*, 2022] are a recent probabilistic logic specifically designed for effective aggregation and reasoning over multiple sources of imprecise knowledge. An LCN specifies a set of probability distributions over the interpretations of a set of logical formulas (propositional or first-order) for which marginal and conditional probability bounds on their truth values are known. Although the model is quite expressive and requires very few restrictions, inference in LCNs is intractable as it involves the exact solution of a non-convex non-linear constraint program defined over an exponentially large number of non-negative real valued variables. This is a serious limitation allowing to solve only relatively small problems with up to 10 atoms.

Contribution. In this paper, we present a novel iterative message-passing algorithm called ARIEL that addresses the limitation of exact inference and thus enables efficient approximate inference in LCNs. Our approach is inspired by the classical belief propagation for graphical models [Pearl, 1988; Koller and Friedman, 2009] and propagates messages in an iterative manner between the nodes of a factor graph associated with the LCN. The key novelty of our scheme is that the messages contain both lower and upper bounds on the marginal probability of the LCN's variables (i.e., atoms) and these bounds are tightened iteratively. Importantly, these messages solve considerably smaller local non-linear constraint programs as compared with those in exact inference. In addition, we show that ARIEL retains an important property of classical belief propagation, namely it yields exact re-

sults on singly-connected LCNs. We experiment and evaluate our proposed algorithm on several classes of LCNs including random as well as more realistic problem instances. Our results are quite promising and demonstrate conclusively that ARIEL is able to produce high quality solutions compared with the exact inference approach. Furthermore, we show that ARIEL scales to much larger problems than previously considered while maintaining solution quality. This is important because it allows us to tackle practical problems involving many thousands of variables.

2 Background

We review next basic concepts about Logical Credal Networks and exact inference methods for these models.

2.1 Logical Credal Networks

A Logical Credal Network (LCN) [Marinescu *et al.*, 2022] $\mathcal{L}$ is defined by a set of two types of *probability sentences*:

$$l_q \leq P(q) \leq u_q \tag{1}$$
$$l_{q|r} \leq P(q \mid r) \leq u_{q|r} \tag{2}$$

where q and r can be arbitrary propositional or first-order logic formulas[1] and $0 \leq l_q \leq u_q \leq 1$, $0 \leq l_{q|r} \leq u_{q|r} \leq 1$. Each sentence in $\mathcal{L}$ is further associated with a Boolean parameter τ indicating dependence between the atoms of q.

An LCN represents the set of probability distributions (i.e., *models*) over all interpretations that satisfy a set of constraints given explicitly by sentences (1) and (2) together with a set of implied independence constraints between the LCN's atoms. We say that an LCN is *consistent* if it has at least one model. Otherwise, it is *inconsistent*.

The *primal graph* of an LCN $\mathcal{L}$ is a directed graph G that contains *formula nodes* associated with the formulas q and r in $\mathcal{L}$'s sentences and *atomic nodes* associated with the atoms involved in those formulas, respectively. If a formula consists of a single atom then G contains a single atomic node for that formula. For type (1) sentences, there is a directed edge from each of formula q's atomic nodes to q's formula node, while in case of type (2) sentences, G contains directed edges from formula r's atomic nodes to r's formula node, a directed edge from formula node r to formula node q, and directed edges from q's formula node to its corresponding atomic nodes. In addition, for all sentences with $\tau = True$, G contains directed edges from the atomic nodes corresponding to q's atoms to the formula node q.

Given an LCN $\mathcal{L}$ and its primal graph G, the *parents* of an atomic node x, denoted by $parents(x)$, is the set of atomic nodes y such that there exists a directed path $(y \rightarrow z_1 \rightarrow \cdots \rightarrow z_k \rightarrow x)$ from y to x in G such that all intermediate nodes z_i (if any) are formula nodes. Similarly, the *descendants* of an atomic node x, denoted by $descendants(x)$, is the set of atomic nodes y such that there exists a directed path $(x \rightarrow z_1 \rightarrow \cdots \rightarrow z_k \rightarrow y)$ from x to y in G such that none of the intermediate nodes x_i (if any) is in $parents(x)$.

[1]For simplicity, we assume that the first-order logic formulas are universally quantified, do not contain functions and their variables have finite domains of values.

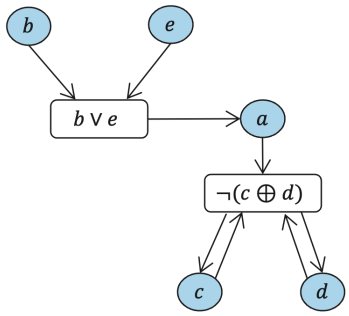

Figure 1: The primal graph of the LCN from Example 1

The Markov condition associated with an LCN $\mathcal{L}$ allows us to make additional independence assumptions between its atoms. Namely, given a model M of $\mathcal{L}$, every atom x in $\mathcal{L}$ is conditional independent of its non-descendant non-parent atoms given its parents in $\mathcal{L}$'s primal graph.

Example 1. *Consider the following propositional LCN which was inspired by the Earthquake example from [Pearl, 1988]. The sentences below state that: burglaries (b) are more common than earthquakes (e) (Eq. 3 and 4); the house alarm (a) can be triggered by a burglary or an earthquake (Eq. 5); in case of an alarm, either both Charles (c) and Dan (d) call the emergency services or neither does (Eq. 6); the alarm can also be triggered accidentally (Eq. 7).*

$$0.1 \leq P(b) \leq 0.2 \tag{3}$$
$$0.05 \leq P(e) \leq 0.1 \tag{4}$$
$$0.8 \leq P(a|b \vee e) \leq 0.9 \tag{5}$$
$$0.7 \leq P(\neg(c \oplus d)|a) \leq 0.8 \tag{6}$$
$$0.01 \leq P(a) \leq 0.08 \tag{7}$$

Figure 1 depicts the factor graph of the LCN where the round shaded nodes represent atomic nodes and the rectangular hollow nodes correspond to the formula nodes, respectively. In this case, we assume that the τ flag associated with the sentences is $True$. Therefore, we have that b and e are independent, c is conditionally independent of $\{b, e\}$ given $\{a, d\}$ and d is conditionally independent of $\{b, e\}$ given $\{a, c\}$.

2.2 Exact Inference in LCNs

Given an LCN $\mathcal{L}$ and a query formula ρ, the *marginal inference* task computes lower and upper bounds on the posterior marginal probability $P(\rho)$, denoted by $\underline{P}(\rho)$ and $\overline{P}(\rho)$, respectively. The task entails solving a non-linear program defined over a set of variables representing the probabilities of $\mathcal{L}$'s interpretations and comprising of linear constraints derived from $\mathcal{L}$'s sentences, non-linear constraints corresponding to the independence assumptions derived from $\mathcal{L}$'s Markov condition and a linear objective function corresponding to the query $P(\rho)$ which is subsequently minimized and maximized to yield the desired bounds.

Let $\vec{p} = (p_1, \ldots, p_N)$ be the vector representing the probabilities of the $N = 2^n$ interpretations of an LCN $\mathcal{L}$ with n atoms, and let $\vec{A}_\alpha = (a_1^\alpha, \ldots, a_N^\alpha)$ be a binary vector, called an *indicator vector*, such that a_j^α is 1 if formula α is true in the j-th interpretation and 0 otherwise. Since the probability

of a formula α is the sum of the probabilities of the interpretations in which α is true, we can write $P(\alpha)$ as $\vec{A}_\alpha \odot \vec{p}$ where $\odot$ is the dot-product of two vectors. Therefore, we solve:

$$\sum_{i=1}^{N} p_i = 1 \tag{8}$$

$$p_i \geq 0, \forall i = 1, \ldots, N \tag{9}$$

$$l_q \leq \vec{A}_q \odot \vec{p} \leq u_q \tag{10}$$

$$l_{q|r} \cdot \vec{A}_r \odot \vec{p} \leq \vec{A}_{q \wedge r} \odot \vec{p} \leq u_{q|r} \cdot \vec{A}_r \odot \vec{p} \tag{11}$$

$$(\vec{A}_\alpha \odot \vec{p}) \cdot (\vec{A}_\beta \odot \vec{p}) - (\vec{A}_\gamma \odot \vec{p}) \cdot (\vec{A}_\delta \cdot \vec{p}) = 0 \tag{12}$$

$$\text{minimize/maximize } \vec{A}_\rho \odot \vec{p}$$

where x_i is an atomic formula, $S_i = \{s_{i1}, \ldots, s_{ik}\}$ and $T_i = \{t_{i1}, \ldots, t_{il}\}$ are x_i's parents and non-descendants in the primal graph, $\vec{A}_q$ and $\vec{A}_{q \wedge r}$ are the indicator vectors for formulas q and $q \wedge r$, and $\vec{A}_\alpha$, $\vec{A}_\beta$, $\vec{A}_\gamma$ and $\vec{A}_\delta$ are the indicator vectors corresponding to the formulas $\alpha = (x_i \wedge s_{i1} \wedge \cdots \wedge s_{ik} \wedge t_{i1} \wedge \cdots \wedge t_{il})$, $\beta = (s_{i1} \wedge \cdots \wedge s_{ik})$, $\gamma = (x_i \wedge s_{i1} \wedge \cdots \wedge s_{ik})$, and $\delta = (s_{i1} \wedge \cdots \wedge s_{ik} \wedge t_{i1} \wedge \cdots \wedge t_{il})$.

Equations (8) and (9) ensure that $\vec{p}$ is a valid probability distribution, while Equations (10) and (11) encode the sentences of type (1) and (2) in $\mathcal{L}$. Equation 12 encodes the conditional independencies implied by the Markov condition, i.e., $P(x_i|S_i, T_i) = P(x_i|S_i)$, which must hold for all truth values of its atoms (see also [Marinescu *et al.*, 2022]).

3 Approximate Inference in LCNs

Since exact inference is not tractable for large LCNs, we introduce a new message-passing algorithm to approximate the posterior marginals of the atomic formulas in an LCN. The basic idea is to follow the classical belief propagation scheme on a factor graph associated with the LCN and propagate messages between the variable and factor nodes until convergence [Pearl, 1988; Koller and Friedman, 2009].

3.1 Incompatibility with Belief Propagation

Graphical models such as Bayesian networks [Pearl, 1988] or credal networks [Cozman, 2000] typically require a *unique-assessment* assumption, namely a variable must occur in either a marginal distribution (resp. credal set) or a conditional distribution (resp. conditional credal set) but not both, and for each conditional distribution or credal set, all possible interpretations of the parent variables must be specified. LCNs do not require the unique-assessment assumption and, therefore, the sum-product message-passing based approximate inference methods (i.e., belief propagation) which were originally developed for credal networks (i.e., 2U [Fagiuoli and Zaffalon, 1998], L2U [Antonucci *et al.*, 2010] or IPE [Ide and Cozman, 2008]) are not compatible with LCNs. Indeed, consider the following illustrative example:

$$0.2 \leq P(a) \leq 0.3 \tag{13}$$

$$0.6 \leq P(b \mid a) \leq 0.7 \tag{14}$$

$$0.1 \leq P(b \mid \neg a) \leq 0.2 \tag{15}$$

$$0.3 \leq P(b) \leq 0.4 \tag{16}$$

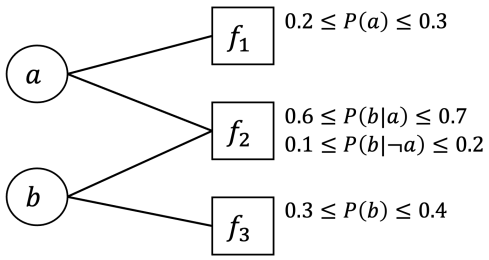

Figure 2: Factor graph for the LCN in Example 2.

Clearly, this is not a valid credal network because (16) violates the unique-assessment assumption but is legitimate for an LCN. If we query $P(b)$, the correct answer is $[0.3, 0.35]$. However, 2U or L2U yield an incorrect answer of $[0.1, 0.26]$ even though the underlying graph has a tree-like structure.

3.2 A Novel Message-Passing Scheme

We start with the definition of the factor graph associated with an LCN that underlies our message-passing scheme.

Definition 3.1 (factor graph)**.** *Given an LCN $\mathcal{L}$, the* factor graph *$\mathcal{F}$ of $\mathcal{L}$ is a bipartite graph with* variable nodes *and* factor nodes*, respectively. A variable node corresponds to an atom in $\mathcal{L}$, while a* factor node *represents one or more sentences in $\mathcal{L}$ that involve the same set of atoms. A factor node is connected to a variable node if they share the same atom.*

Example 2. *Figure 2 shows the factor graph associated with the LCN given by (13)-(16). There are 2 variable nodes (depicted as circles) corresponding to atoms $\{a, b\}$ and 3 factor nodes (depicted as squares) f_1, f_2 and f_3 corresponding to sentences (13), (14)(15) and (16), respectively.*

Given a factor graph $\mathcal{F}$, the high-level flow of our proposed message-passing scheme is similar to that of classical belief propagation, namely to iteratively pass and update messages between the variable and factor nodes of $\mathcal{F}$ until convergence. Let v and f denote a variable node and a factor node in $\mathcal{F}$, respectively. We use $N(\cdot)$ to denote the neighbors of a node in $\mathcal{F}$. Two kinds of messages will be propagated along the edges of $\mathcal{F}$, as follows:

The variable-to-factor message. The message sent by a variable node v to a neighboring factor node f is an interval $[l_{v \to f}, u_{v \to f}]$, where $0 \leq l_{v \to f} \leq u_{v \to f} \leq 1$. If node v has only one neighbor, then the message $[l_{v \to f}, u_{v \to f}]$ is just $[0, 1]$. Otherwise, the message $[l_{v \to f}, u_{v \to f}]$ is defined by:

$$l_{v \to f} = \max_{f' \in N(v) \setminus \{f\}} l_{f' \to v} \tag{17}$$

$$u_{v \to f} = \min_{f' \in N(v) \setminus \{f\}} u_{f' \to v} \tag{18}$$

where $[l_{f' \to v}, u_{f' \to v}]$ is the message sent by a neighboring factor node $f' \in N(v) \setminus \{f\}$, other than f, to v.

The factor-to-variable message. The message sent by a factor node f to a neighboring variable node v is also an interval $[l_{f \to v}, u_{f \to v}]$ obtained by minimizing and, respectively, maximizing the objective function $P(v)$ subject to (i) a set of linear constraints encoding f's sentences, (ii) a set of linear

constraints ensuring that, for each variable node other than v that is connected to f, its marginal probability is within the bounds given by the corresponding variable-to-factor messages, namely $l_{v' \to f} \leq P(v') \leq u_{v' \to f}, \forall v' \in N(f) \setminus \{v\}$, and (iii) a set of non-linear constraints encoding the assumption that f's atoms (other than v) are independent of each other. The latter independence assumption in the local constraint program is a mechanism to approximate the Markov condition, and the same approach is used in classical belief propagation [Pearl, 1988; Koller and Friedman, 2009].

Specifically, if f involves at most k atoms, namely $N(f) = \{v_1, \ldots, v_k\}$, then the message $[l_{f \to v}, u_{f \to v}]$ is computed by solving the following local constraint program:

$$\sum_{i=1}^{K} p_i = 1 \tag{19}$$

$$p_i \geq 0, \forall i = 1, \ldots, K \tag{20}$$

$$l_q \leq \vec{A}_q \cdot \vec{p} \leq u_q \tag{21}$$

$$l_{q|r} \vec{A}_r \odot \vec{p} \leq \vec{A}_{q \wedge r} \odot \vec{p} \leq u_{q|r} \vec{A}_r \odot \vec{p} \tag{22}$$

$$l_{v' \to f} \leq \vec{A}_{v'} \odot \vec{p} \leq u_{v' \to f}, \forall v' \in \mathcal{N}_f \tag{23}$$

$$\vec{A}_{v' \wedge v''} \odot \vec{p} = (\vec{A}_{v'} \odot \vec{p}) \cdot (\vec{A}_{v''} \odot \vec{p}), \forall v' \neq v'' \in \mathcal{N}_f \tag{24}$$

$$\text{minimize/maximize } \vec{A}_v \odot \vec{p}$$

where $\vec{p} = (p_1, \ldots, p_K)$ is the vector representing the probabilities of the $K = 2^k$ interpretations, $\mathcal{N}_f = N(f) \setminus \{v\}$ denotes f's neighbors other than v and $[l_{v' \to f}, u_{v' \to f}]$ is the message sent by a neighboring variable node $v' \in \mathcal{N}_f$ to f, respectively. Equations (21) and (22) encode sentences of type (1) and (2) in f, Equation (23) ensures that v's marginal probability is within the required bounds, while Equation (24) encodes the independence assumption between f's atoms.

Example 3. *Consider the LCN defined by the following sentence $0.3 \leq P(c \wedge (d \vee e)) \leq 0.4$. The factor graph has one factor node f corresponding to the sentence and three variable nodes for the atoms $\{c, d, e\}$, respectively. The factor-to-variable message $[l_{f \to d}, u_{f \to d}]$ is obtained by minimizing and maximizing the following non-linear program:*

$$0.3 \leq P(c \wedge (d \vee e)) \leq 0.4$$
$$l_{c \to f} \leq P(c) \leq u_{c \to f}$$
$$l_{e \to f} \leq P(e) \leq u_{e \to f}$$
$$P(c \wedge e) = P(c) \cdot P(e)$$
$$\textit{minimize/maximize } P(d)$$

Algorithm 1 which we denote hereafter by ARIEL summarizes our message-passing scheme for approximate inference in LCNs. All messages along the edges of the factor graph are first initialized with $[0, 1]$ intervals. Subsequently, the variable-to-factor and factor-to-variable messages are updated in an iterative manner until convergence (i.e., either a fixed number of iterations is exceeded or the average change in messages from one iteration to the next is below a given threshold). Finally, for each variable node v, the lower and upper bounds of the posterior marginal interval are obtained by maximizing and, respectively, minimizing the lower and upper bounds of the incoming factor-to-variable messages to v (lines 16–18), namely:

$$\underline{P}(v) = \max_{f \in N(v)} l_{f \to v} \quad \text{and} \quad \overline{P}(v) = \min_{f \in N(v)} u_{f \to v}$$

Algorithm 1 AppRoximate InfErence for LCNs (ARIEL)

Require: LCN $\mathcal{L}$
1: Create factor graph $\mathcal{F}$
2: **for all** edges $(v, f) \in \mathcal{F}$ **do**
3: Set $[l_{v \to f}, u_{v \to f}] = [l_{f \to v}, u_{f \to v}] = [0, 1]$
4: **end for**
5: **repeat**
6: ▷ Update the variable-to-factor messages
7: **for all** edges $(v, f) \in \mathcal{F}$ **do**
8: **if** $|N(v)| = 1$ **then**
9: $l = 0, u = 1$
10: **else**
11: $l = \max_{f' \in N(v) \setminus \{f\}} l_{f' \to v}$
12: $u = \min_{f' \in N(v) \setminus \{f\}} u_{f' \to v}$
13: **end if**
14: Update $[l_{v \to f}, u_{v \to f}] = [l, u]$
15: **end for**
16: ▷ Update the factor-to-variable messages
17: **for all** edges $(v, f) \in \mathcal{F}$ **do**
18: $l = \min P(v)$ subject to constraints (19)–(24)
19: $u = \max P(v)$ subject to constraints (19)–(24)
20: Update $[l_{f \to v}, u_{f \to v}] = [l, u]$
21: **end for**
22: **until** convergence
23: **for all** variable nodes v **do**
24: $\underline{P}(v) = \max_{f \in N(v)} l_{f \to v}$
25: $\overline{P}(v) = \min_{f \in N(v)} u_{f \to v}$
26: **end for**
27: **return** $[\underline{P}(v), \overline{P}(v)]$ for each atom $v \in \mathcal{L}$

Example 4. *Continuing our example LCN defined by (13)-(16), the message $[l_{f_2 \to b}, u_{f_2 \to b}]$ is obtained by solving:*

$$0.6 \leq P(b \mid a) \leq 0.7$$
$$0.1 \leq P(b \mid \neg a) \leq 0.2$$
$$l_{a \to f_2} \leq P(a) \leq u_{a \to f_2}$$
$$\textit{minimize/maximize } P(b)$$

The following messages are obtained upon convergence:

$$l_{f_1 \to a} = 0.2\,,\ u_{f_1 \to a} = 0.3$$
$$l_{a \to f_1} = 0.2\,,\ u_{a \to f_1} = 0.6$$
$$l_{a \to f_2} = 0.2\,,\ u_{a \to f_2} = 0.3$$
$$l_{f_2 \to a} = 0.2\,,\ u_{f_2 \to a} = 0.6$$
$$l_{f_2 \to b} = 0.2\,,\ u_{f_2 \to b} = 0.35$$
$$l_{b \to f_2} = 0.3\,,\ u_{b \to f_2} = 0.4$$
$$l_{b \to f_3} = 0.2\,,\ u_{b \to f_3} = 0.35$$
$$l_{f_3 \to b} = 0.3\,,\ u_{f_3 \to b} = 0.4$$

Finally, the lower and upper bounds for $P(b)$ are given by:

$$\underline{P}(b) = \max(l_{f_2 \to b}, l_{f_3 \to b}) = \max(0.2, 0.3) = 0.3$$
$$\overline{P}(b) = \min(u_{f_2 \to b}, u_{f_3 \to b}) = \min(0.35, 0.4) = 0.35$$

which match the results of exact inference in this case.

3.3 Properties

We show next that ARIEL computes exact posterior marginal probability intervals for singly-connected LCNs.

Definition 3.2. *Let $\mathcal{L}$ be an LCN with primal graph G. We call $\mathcal{L}$* singly-connected *if G does not contain directed cycles.*

Theorem 1 (correctness)**.** *Given a singly-connected LCN with n atoms denoted by $V = \{v_1, \ldots, v_n\}$, ARIEL computes exact posterior marginal lower and upper bounds $\underline{P}(v_i)$ and $\overline{P}(v_i)$, for each atom $v_i \in V$.*

Note that Theorem 1 holds even if the LCN contains additional marginal probability sentences that break the unique-assessment assumption. However, in general, LCNs may contain directed cycles and, in this case, there is no guarantee on correctness. This is also true for the classical loopy belief propagation algorithms in probabilistic graphical models [Pearl, 1988; Koller and Friedman, 2009].

Theorem 2 (complexity)**.** *Let $\mathcal{L}$ be an LCN such that its factor graph has n variable nodes (atoms) and m factor nodes. Assuming a fixed number of iterations i, the complexity of Algorithm 1 is $O(i \cdot n \cdot m \cdot Q)$, where Q bounds the complexity of solving the local non-linear programs at factor nodes. Each of these non-linear programs involves 2^t variables, where t bounds the number of atoms in a factor node.*

3.4 Comparison with Other Algorithms

The bound tightening operation at variable nodes is conceptually similar to a recent inference algorithm for Logical Neural Networks [Riegel *et al.*, 2020]. The message computation at factor nodes bears some similarity to the 2U and L2U algorithms for credal networks which make similar independence assumption to approximate the Markov condition [Ide and Cozman, 2008]. ARIEL is also related with the IPE algorithm for credal networks [Ide and Cozman, 2008] which selects the tightest bounds over a number of polytree subgraphs.

3.5 Handling Complex Query Formulas

Let ρ be a non-atomic query formula and let $\{v_1, \ldots, v_k\}$ be its atoms. In this case, following the message propagation outlined by Algorithm 1, the marginal $P(\rho)$ can be approximated by solving the non-linear constraint program:

$$\sum_{i=1}^{K} p_i = 1 \tag{25}$$

$$p_i \geq 0, \forall i = 1, \ldots, K \tag{26}$$

$$l_{v' \to f'} \leq \vec{A}_{v'} \odot \vec{p} \leq u_{v' \to f'}, \forall v' \in N(f') \tag{27}$$

$$\vec{A}_{v' \wedge v''} \odot \vec{p} = (\vec{A}_{v'} \odot \vec{p}) \cdot (\vec{A}_{v''} \odot \vec{p}), \forall v' \neq v'' \in N(f') \tag{28}$$

$$\text{minimize/maximize } \vec{A}_{\rho} \odot \vec{p}$$

where f' is an auxiliary factor node corresponding to ρ, $N(f') = \{v_1, \ldots, v_k\}$ and $\vec{A}_{\rho}$ is ρ's indicator vector.

4 Experiments

We evaluate empirically our ARIEL scheme for approximate inference and compare it with exact inference on several classes of LCNs. The competing algorithms were implemented in Python 3.8 and used the `ipopt` 3.12 solver [Wächter and Biegler, 2006] with default settings to handle the non-linear constraint programs. We ran all experiments on a 2.2GHz Intel Core processor with 32GB of RAM.

size	CPU time (sec)		Errors	
n, e	Exact	ARIEL	MAE$_l$	MAE$_u$
`polytree`				
5, 2	0.38	1.69	0.0001±0.00	0.0000±0.00
6, 2	0.76	1.99	0.0000±0.00	0.0000±0.00
7, 2	4.94	2.45	0.0000±0.00	0.0000±0.00
8, 2	64.96	2.92	0.0000±0.00	0.0000±0.00
9, 2	837.52	3.29	0.0000±0.00	0.0000±0.00
10, 2	4252.04	3.79	0.0000±0.00	0.0000±0.00
`dag`				
5, 2	9.12	1.79	0.0425±0.06	0.0949±0.11
6, 2	5.63	2.06	0.0332±0.05	0.0395±0.04
7, 2	193.07	3.13	0.0551±0.07	0.0555±0.06
8, 2	3625.37	3.71	0.0339±0.05	0.0506±0.08
9, 2	7251.62	4.07	0.0586±0.08	0.0658±0.09
10, 2	17195.05	4.33	0.1071±0.14	0.0640±0.09
`random`				
5, 2	9.48	1.87	0.0431±0.05	0.0386±0.05
6, 2	3.98	2.29	0.0466±0.06	0.0372±0.06
7, 2	567.05	3.54	0.0428±0.07	0.0356±0.06
8, 2	3235.67	3.98	0.0567±0.08	0.0348±0.06
9, 2	6263.68	5.23	0.0647±0.08	0.0509±0.07
10, 2	11310.88	5.85	0.1053±0.12	0.0938±0.12

Table 1: Results obtained on `polytree`, `dag`, and `random` LCNs. Average CPU time in seconds and MAEs for the posterior marginal lower and upper bounds on each variable. Time limit is 24 hours.

Measure of Performance. In all experiments we report the average CPU time in seconds and the mean absolute error (MAE) for the posterior marginal lower and upper bounds obtained for each propositional variable. More specifically, for each problem instance, we define the errors as:

$$\text{MAE}_l = \frac{1}{n} \cdot \sum_{i=1}^{n} |\underline{P}^*(x_i) - \underline{P}(x_i)|$$

$$\text{MAE}_u = \frac{1}{n} \cdot \sum_{i=1}^{n} |\overline{P}^*(x_i) - \overline{P}(x_i)|$$

where n is the number of variables, $\underline{P}^*(x_i)$ and $\overline{P}^*(x_i)$ are the exact posterior marginal lower bounds on $P(x_i)$ for each variable x_i, while $\underline{P}(x_i)$ and $\overline{P}(x_i)$ are the approximate marginal bounds computed by ARIEL.

4.1 Random LCNs

For our purpose, we generated several classes of random LCNs with n propositional variables $\{x_1, \ldots x_n\}$, called `polytree`, `dag`, and `random`, respectively, having m sentences of the following types:

$$(a)\ \ l \leq P(x_i) \leq u$$
$$(b)\ \ l \leq P(x_i|x_j) \leq u,\ \ x_i \neq x_j$$
$$(c)\ \ l \leq P(x_i|x_j \wedge x_k) \leq u\ \ x_i \neq x_j \neq x_k$$

Specifically, for `polytree` LCNs, there are $m = n$ sentences of types (a), (b) and (c) generated randomly such that the corresponding primal graph is a polytree (i.e., there are no directed cycles). The `dag` instances contain $m = n$ randomly generated sentences of types (a), (b) and (c) such that the primal graph is a directed acyclic graph (DAG). For `random` LCNs we also have $m = n$ sentences of types (a), (b) and (c) generated randomly without the acyclicity requirement. In all cases, we added e additional sentences of type (a).

We note that for type (a) we generate only one sentence $P(x_i)$ while for types (b) and (c) we generate sentences for all truth values of x_j and x_k, namely $P(x_i|x_j)$, $P(x_i|\neg x_j)$, $P(x_i|x_j \wedge x_k)$, $P(x_i|x_j \wedge \neg x_k)$, $P(x_i|\neg x_j \wedge x_k)$ and $P(x_i|\neg x_j \wedge \neg x_k)$, respectively. We selected the probability bounds l and u of the sentences uniformly at random between 0 and 1 such that $u - l \leq 0.6$. Furthermore, we ensured that all problem instances generated were consistent.

Table 1 summarizes the results obtained on `polytree`, `dag` and `random` instances of varying sizes with $n \in \{5, 6, \ldots, 10\}$ and $e = 2$. Each data point in the table represents an average over 10 random instances generated for that particular problem size. The second and third columns report the running times of the exact and approximate inference algorithms, while the fourth and fifth columns give the errors MAE_l and MAE_u on the posterior marginal bounds computed by ARIEL using a maximum of 10 iterations and a 10^{-6} threshold for convergence (whichever comes first).

When looking at the solution quality, especially on the `polytree` problems, we can see that the absolute errors are very small (close to zero) thus verifying the correctness of ARIEL on singly-connected LCNs (the discrepancies are caused by the numerical precision for representing real numbers as well as the default tolerances used by the `ipopt` solver). For `dag` and `random` problems, the errors are also small and suggest that the approximate posterior marginals are fairly close to the exact ones. In terms of running time, we can see that, as expected, ARIEL scales much better to larger problems compared with the exact algorithm. For example, on `dag` and `random` problems of size 10, the algorithm is already almost 4 orders of magnitude faster while producing relatively good quality solutions.

Furthermore, Table 2 reports results obtained on much larger `random` LCNs with up to 100,000 propositional variables. We can see that ARIEL was able to solve all these problem instances while the exact algorithm could not go beyond the smallest problem size. Specifically, it exceeded the 30 hour time limit for the $n = 100$ case and ran out of memory while building the non-linear program for $n > 1000$, respectively. This demonstrates clearly that ARIEL overcomes the major limitation of exact inference for LCNs and thus allows us to tackle efficiently much larger problems possibly involving many thousands of variables.

In Figure 3 we plot the mean absolute errors MAE_l and MAE_u as a function of the number of iterations used by ARIEL for solving the `polytree` and `dag` benchmarks of size $n = 6$, respectively. ARIEL's convergence threshold was set to 0. We can see that the algorithm is able to converge to relatively small errors after less than 10 iterations. A similar pattern was also observed on the other benchmarks.

size n	CPU time (sec) Exact	ARIEL
10	11422.28	5.86
100	-	82.02
1,000	na	820.73
10,000	na	8509.23
100,000	na	89668.61

Table 2: CPU time in seconds obtained on large `random` LCNs.

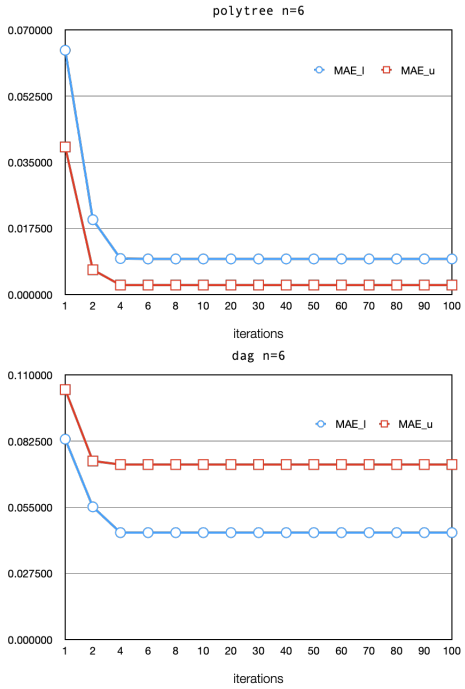

Figure 3: Mean absolute errors MAE_l and MAE_u vs number of iterations used by ARIEL for the `polytree` (top) and `dag` (bottom) benchmarks of size $n = 6$.

LCN	CPU time (sec) Exact	ARIEL	Errors MAE_l	MAE_u
Toy	0.14	0.98	0.0000±0.00	0.0000±0.00
Earth	0.32	1.31	0.0000±0.00	0.0000±0.00
Cancer	0.33	1.93	0.0014±0.00	0.0036±0.01
Asia	52.23	2.69	0.0005±0.00	0.0162±0.04
Credit	4334.66	2.96	0.0336±0.03	0.0490±0.09
Engine	14006.99	3.07	0.0737±0.08	0.0259±0.04
Suicide	5850.50	3.47	0.0318±0.07	0.0191±0.04
Tank	11415.05	4.16	0.0234±0.04	0.0355±0.02
Alarm	10783.91	2.77	0.0421±0.06	0.0578±0.09
Hepatitis	11178.05	5.16	0.0177±0.03	0.0574±0.06

Table 3: Results obtained on real-world LCNs. CPU time in seconds and MAEs for the posterior marginal lower and upper bounds on each variable. Time limit is 24 hours.

4.2 Real-World LCNs

Table 3 displays the results obtained on LCNs derived from real-world Bayesian networks with binary variables [Constantinou *et al.*, 2020]. More specifically, each of these

LCNs contains sentences of the form $l \leq P(x_i) \leq u$ and $l \leq P(x_i|\pi_i) \leq u$, respectively, where x_i is a variable and $\pi_i = y_{i1} \wedge \cdots \wedge y_{ik}$ is the conjunction of the propositional variables corresponding to a particular configuration of the parents $\{y_{i1}, \ldots y_{ik}\}$ of x_i in the Bayesian network. The bounds l and u were selected such that $u - l \leq 0.4$, while the numbers of variables and sentences ranged between 4 and 10, and between 6 and 24, respectively. The complete specification of each of these LCNs is included in the supplementary material. We can see again that ARIEL outperformed dramatically the exact inference method in terms of running time while maintaining a relatively good solution quality. For example, on the Tank and Hepatitis problem instances, ARIEL is almost 4 orders of magnitude faster than its competitor, while the corresponding posterior bounds start to differ at the second decimal point compared with the exact ones.

4.3 Application to Chemistry

We describe next a possible application of LCNs and approximate inference to the chemistry domain. Specifically, we consider a binary molecular classification task using imprecise expert knowledge as well as molecular fingerprinting data [Fernández de Gortari *et al.*, 2017]. In this case, the class variable is denoted by $y \in \{0, 1\}$ while the molecule structure is represented by a set of n binary features (or fingerprints) $F = \{f_1, \ldots, f_n\}$ indicating the presence or absence of certain molecular substructures. The LCN is defined by the following types of sentences: (a) $l_0 \leq P(\neg y) \leq u_0$ and $l_1 \leq P(y) \leq u_1$ representing the class probability, (b) $l_{i0} \leq P(f_i|\neg y) \leq u_{i0}$ and $l_{i1} \leq P(f_i|y) \leq u_{i1}$ representing the conditional probability of feature $f_i \in F$ being present given the class, and (c) $l \leq P(y|\mathcal{F}(f_{i1}, f_{i2}, \ldots, f_{ik})) \leq u$ and $l \leq P(\neg y|\mathcal{F}(f_{i1}, f_{i2}, \ldots, f_{ik})) \leq u$, where $\{f_{i1}, \ldots, f_{ik}\}$ is a subset of features and $\mathcal{F}(f_{i1}, \ldots, f_{ik})$ is the conjunction of their respective values, respectively (e.g., $\mathcal{F}(f_1, f_2) = f_1 \wedge \neg f_2$). The latter represents imprecise knowledge from one or more domain experts stating that the class can be determined by specific combinations of features.

Given a new molecule for which only a subset of features $\{f_1, \ldots, f_k\}$ is observed, the task is to compute the posterior marginal probability of its class, namely $P(y|f_1, \ldots, f_k)$ and $P(\neg y|f_1, \ldots, f_k)$. For our purpose, we consider a database containing 1298 molecules with $n = 985$ features. The corresponding LCN has 986 propositional variables, 2 sentences of type (a), 985 sentences of type (b) and 6 sentences of type (c), respectively. The observed features are translated into k additional constraints of the form $P(f_j) = 1.0$ or $P(\neg f_j) = 1.0$ $\forall j = 1..k$, depending on whether f_j is absent or present. Table 4 summarizes the results obtained for different sizes of the observed feature subset. Specifically, for each value of k, we generate 10 configurations of k features selected uniformly at random from F and report the average running time. As before, we see that ARIEL was able to solve all problem instances in a little over 5 minutes on average, while the exact method ran out of memory in all test cases.

In summary, we can conclude that the proposed approximate inference algorithm ARIEL is the only inference scheme that can tackle practical larger scale LCNs.

size k	CPU time (sec) Exact	ARIEL
10	na	324.51
20	na	322.01
50	na	329.58
100	na	339.47
200	na	362.70
500	na	410.43

Table 4: CPU time in seconds for the chemistry application.

5 Related Work

Nilsson's probabilistic logic (NL) [Nilsson, 1986; Nilsson, 1994] is perhaps the first system in which the truth values of logical sentences (or formulas) can range between 0 and 1 and are interpreted as the probability of those sentences being true. The formalism however allows probability bounds but does not permit specifying independence relations between propositions which typically leads to overly large posterior probability intervals. Bayesian logic (BL) [Andersen and Hooker, 1994] combines probabilistic logic and Bayesian networks in order to capture conditional independence relations among propositions. Markov Logic Networks (MLN) [Richardson and Domingos, 2006] apply the ideas of a Markov network to first-order logic where the weights attached to the logic formulas are used to define a joint probability distribution over all possible interpretations and thus enable uncertain inference. Probabilistic Soft Logic (PSL) [Getoor and Taskar, 2007] combines Markov networks with *soft* or real-valued logic (e.g., Lukasiewicz logic). Probabilistic Logic Programs (PLP) [De Raedt *et al.*, 2008] and Stochastic Logic Programs (SLP) [Cussens, 2000] are logic programs in which some of the facts are annotated with probabilities. We emphasize that MLN, PSL, PLP, SLP do not allow probability bounds on logic formulas, BL constrains the formulas to a specific structure and NL does not consider independence assumptions. In contrast, LCNs overcome these shortcomings.

6 Conclusions

We propose a new iterative message-passing scheme called ARIEL for approximate inference in LCNs. The algorithm works by propagating messages between the nodes of a factor graph representation of the LCN. These messages are lower and upper bounds on marginal probabilities of the variables and their computation involves solving considerably smaller local non-linear programs compared with exact inference. The empirical evaluation on several classes of LCNs including random and more realistic problem instances shows promising results, particularly in scaling to much larger problems while producing good quality solutions.

Potential future directions include extending to temporal models, further algorithmic innovations for learning LCNs from data, and experiments on a wider array of applications. We also plan to investigate extensions to optimization tasks such as MAP and Marginal MAP inference in LCNs using search-based algorithms similar to those developed for graphical models (e.g., [Marinescu *et al.*, 2018]).

References

[Andersen and Hooker, 1994] Kent Andersen and John Hooker. Bayesian logic. *Decision Support Systems*, 11(2):191–210, 1994.

[Antonucci *et al.*, 2010] Alessandro Antonucci, Yi Sun, Cassio De Campos, and Marco Zaffalon. Generalized loopy 2U: A new algorithm for approximate inference in credal networks. *International Journal of Approximate Reasoning*, 51(5):474–484, 2010.

[Chandru and Hooker, 1999] Vijay Chandru and John Hooker. *Optimization Methods for Logical Inference*. John Wiley & Sons, 1999.

[Constantinou *et al.*, 2020] Anthony Constantinou, Yang Liu, Kiattikun Chobtham, Zhigao Guo, and Neville Kitson. The bayesys data and bayesian network repository. Technical report, Bayesian Artificial Intelligence research lab, Queen Mary University of London, London, UK, 2020.

[Cozman, 2000] Fabio Cozman. Generalizing variable-elimination in bayesian networks. In *Workshop on Probabilistic reasoning in Bayesian networks at SBIA/Iberamia 2000*, pages 21–26, 2000.

[Cussens, 2000] James Cussens. Stochastic logic programs: Sampling, inference and applications. In *Uncertainty in Artificial Intelligence (UAI)*, pages 115–122, 2000.

[De Raedt *et al.*, 2008] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton. *Probabilistic Inductive Logic Programming - Theory and Applications*. Springer, 2008.

[Dürig and Studer, 2005] Michael Dürig and Thomas Studer. Probabilistic abox reasoning: Preliminary results. In *Description Logics*, pages 104–111, 2005.

[Fagin *et al.*, 1990] Ronald Fagin, Joseph Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1-2):78–128, 1990.

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

[Fernández de Gortari *et al.*, 2017] Eli Fernández de Gortari, César García-Jacas, Karina Martinez-Mayorga, and José Medina-Franco. Database fingerprint (dfp): an approach to represent molecular databases. *Journal of Cheminformatics*, 9(1):1–9, 2017.

[Getoor and Taskar, 2007] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2007.

[Heinsohn, 1994] Jochen Heinsohn. Probabilistic description logics. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, pages 311–318, 1994.

[Ide and Cozman, 2008] Jaime Ide and Fabio Cozman. Approximate algorithms for credal networks with binary variables. *International Journal of Approximate Reasoning*, 48(1):275–296, 2008.

[Jaeger, 1994] Manfred Jaeger. Probabilistic reasoning in terminological logics. In *Principles of Knowledge Representation and Reasoning*, pages 305–316. Elsevier, 1994.

[Koller and Friedman, 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Li *et al.*, 2020] Shuang Li, Lu Wang, Ruizhi Zhang, Xiaofu Chang, Xuqin Liu, Yao Xie, Yuan Qi, and Le Song. Temporal logic point processes. In *International Conference on Machine Learning*, pages 5990–6000. PMLR, 2020.

[Marinescu *et al.*, 2018] Radu Marinescu, Junkyu Lee, Rina Dechter, and Alexander Ihler. AND/OR search for marginal MAP. *Journal or Artificial Intelligence Research (JAIR)*, 63(1):875 – 921, 2018.

[Marinescu *et al.*, 2022] Radu Marinescu, Haifeng Qian, Alexander Gray, Debarun Bhattacharjya, Francisco Barahona, Tian Gao, Ryan Riegel, and Pravinda Sahu. Logical credal networks. In *36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[Nilsson, 1986] Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[Nilsson, 1994] Nils Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, 59(1-2):39–42, 1994.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[Riegel *et al.*, 2020] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[Wächter and Biegler, 2006] Andreas Wächter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.