ADVANCES IN MARGINAL INFERENCE FOR CHAIN-GRAPH LOGICAL CREDAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Logical Credal Networks (LCNs) are an expressive probabilistic logic framework that combines propositional formulas with imprecise probability bounds to represent sets of probability distributions. While LCNs support cyclic dependencies and impose few syntactic restrictions, inference remains computationally challenging. Existing exact methods scale exponentially with the number of propositions, and the only available approximate approach is the ARIEL message-passing algorithm. In this paper, we exploit the chain graph structure underlying LCNs and reformulate inference as a credal network problem. Building on this formulation, we propose three approximate inference algorithms based on variable elimination, interval-valued message passing, and coordinate descent for computing posterior probability bounds. We also introduce an exact inference method that leverages a junction tree decomposition of the chain graph to construct and solve a substantially smaller nonlinear program. We establish theoretical properties of the proposed methods and evaluate them on synthetic and realistic LCN instances. The results show substantial speedups over existing approaches while maintaining high-quality probability bounds and enabling exact inference on significantly larger problems.

1 INTRODUCTION

Combining logical and probabilistic reasoning is a long-standing goal of Artificial Intelligence. Frameworks such as Markov Logic Networks (Richardson & Domingos, 2006), ProbLog (De Raedt et al., 2007), and Probabilistic Soft Logic (Bach et al., 2017) encode domain knowledge as logical rules augmented with uncertainty, but most of them require *precise* probability values or impose strong structural assumptions such as acyclicity. In many real applications, however, domain expertise yields only *ranges* of plausible probabilities and the rules of interest are neither directed nor acyclic.

Logical Credal Networks (LCNs) (Marinescu et al., 2022) were introduced to address this gap: they allow *imprecise* specifications of the form $\alpha \leq P(\phi \mid \varphi) \leq \beta$ over arbitrary propositional formulas, with no acyclicity restriction. An LCN defines a set of probability distributions consistent with its constraints and a local Markov condition. Inference amounts to computing tight lower and upper bounds on posterior probabilities of query formulas. This flexibility makes LCNs well suited to causal reasoning with partial knowledge, robust decision making, and trustworthy neuro-symbolic systems, but also makes inference considerably harder than in Bayesian or classical credal networks (Pearl, 1988; Cozman, 2000). Existing LCN inference algorithms are limited to exact constrained non-linear optimization (Marinescu et al., 2022), which is confined to small instances, and the ARIEL message-passing scheme (Marinescu et al., 2023), which is currently the *only* approximate algorithm for LCN marginal inference. The lack of a richer toolbox of scalable exact and approximate algorithms is the main obstacle to deploying LCNs at practical sizes.

Contribution. In this paper, we revisit the chain graph factorization of LCNs recently established by (Cozman, 2023) and turn it into a concrete algorithmic framework. The key idea is to approximate the LCN's distribution set $\mathcal{D}_{\mathrm{LCN}}$ with the strong extension $\mathcal{D}_{\mathrm{SE}}$ of a credal network corresponding to the chain graph factorization of the original LCN and tackle the marginal inference problem with the well-developed machinery of credal networks (Cozman, 2000; Mauá & Cozman, 2020; Ide & Cozman, 2008; Antonucci et al., 2015). This connection lets us import and extend credal-network algorithms into the LCN setting for the first time, trading a controlled relaxation for substantial scalability gains.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We therefore extend three credal networks inference algorithms: a variable elimination scheme (CVE) producing exact bounds on $\mathcal{D}_{\mathrm{SE}}$, together with a message-truncated variant CVE$_{cm}$, a fast interval-vector message-passing algorithm called IBP that produces outer bounds with native multi-valued support, and a coordinate descent algorithm denoted CDVE that computes inner bounds by optimizing over extreme points. We further introduce Credal Junction Tree (CJT), a novel exact algorithms that leverages a junction tree decomposition of the chain graph to construct and solve an exponentially smaller nonlinear program to compute posterior bounds. We establish theoretical properties of the proposed methods and evaluate them on a diverse collection of synthetic and realistic benchmark LCN instances. Experimental results demonstrate substantial improvements in computational efficiency over existing approaches, while maintaining high-quality probability bounds and, in the case of the exact method, enabling exact inference on significantly larger problems than previously possible.

The Appendix includes additional details on the algorithms as well as experimental results.

2 Background

2.1 Logical Credal Networks

A *Logical Credal Network* (LCN) (Marinescu et al., 2022) consists of a set of binary variables (or propositions) $\mathbf{X} = \{X_1, \ldots, X_n\}$ and a set of constraints or probabilistic sentences $\mathcal{S}$. Each sentence $s \in \mathcal{S}$ is a constraint of a Type-1 form $\ell \leq P(\varphi) \leq u$, or a Type-2 form $\ell \leq P(\varphi \mid \psi) \leq u$ over propositional formulas. Since a Type-2 sentence is a ratio constraint which is undefined when its conditioning event has probability zero, we consider hereafter the algebraically *cleared* form:

$$\ell \cdot P(\psi) \;\leq\; P(\varphi \wedge \psi) \;\leq\; u \cdot P(\psi), \tag{1}$$

which is total (defined for all p) and *bilinear* in p.

The *primal graph* G of an LCN contains a *proposition-node* for each of its propositions (Fig. 1). For each Type-1 sentence $\ell \leq P(\varphi) \leq u$, G contains a *formula-node* labeled by φ and bi-directed edges from φ to each proposition in φ. For each Type-2 constraint $\ell \leq P(\varphi \mid \psi) \leq u$, G contains formula nodes labeled by φ and ψ, directed edges from each proposition in ψ to ψ, a directed edge from ψ to φ, and bi-directed edges from each proposition in φ to φ, respectively (Marinescu et al., 2022).

The *structure* of an LCN is a graph that can be extracted from G as follows: (i) for each formula-node φ appearing in either a Type-1 or a Type-2 constraint, place an undirected edge between any two propositions in φ; (ii) for each pair of formula-nodes φ and ψ in a Type-2 constraint, add a directed edge from each proposition in ψ to each proposition in φ; (iii) if there is now a pair of edges $A \leftrightarrows B$ for some propositions A, B, replace both by an undirected edge; (iv) remove formula-nodes and all their edges as well as any duplicate edges (Cozman, 2023).

The parents of a proposition node X_i in G, denoted $\mathbf{pa}(X_i)$, are the proposition nodes that reach X_i via a directed path whose intermediate nodes (if any) are formula nodes. The descendants of X_i, denoted $\mathbf{de}(X_i)$, are the proposition nodes reachable from A_i via a directed path whose intermediate nodes do not belong to $\mathbf{pa}(X_i)$. An LCN is endowed with a Local Markov Condition (LMC(LCN)).

Definition 1 (LMC(LCN))**.** *Given an LCN $\mathcal{L}$ with primal graph G, every proposition X_i in $\mathcal{L}$ is independent of its non-parent non-descendant propositions conditioned on X_i's parents in G.*

Each LMC assertion $\iota : (X \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ with X a singleton imposes, for *every* configuration $(\mathbf{y}, \mathbf{z})$ of $\mathbf{Y} \cup \mathbf{Z}$, the bilinear equality $P(x, \mathbf{y}, \mathbf{z})\, P(\mathbf{z}) = P(x, \mathbf{z})\, P(\mathbf{y}, \mathbf{z})$. Writing g_s for the functional of a sentence s and h_ι for the residual of an assertion ι, the *LCN distribution set* $\mathcal{D}_{\mathrm{LCN}}$ is:

$$\mathcal{D}_{\mathrm{LCN}} = \big\{\, p \in \Delta(2^{\mathbf{X}}) \;:\; \ell_s \leq g_s(p) \leq u_s \;\forall s, \qquad h_\iota(p) = 0 \;\forall \iota \,\big\}, \tag{2}$$

where $\Delta(2^{\mathbf{X}})$ is the simplex of joint distributions of the 2^n truth assignments to $\mathbf{X}$. The *marginal inference* task for a query atom a is defined by:

$$\underline{P}(a) = \min_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a), \qquad \overline{P}(a) = \max_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a),$$

and analogously for a conditional $P(a \mid \mathbf{e})$ under evidence $\mathbf{e}$. The current state of the art consists of two approaches: the exact nonlinear optimization method of (Marinescu et al., 2022) and the approximate message-passing algorithm called ARIEL (Marinescu et al., 2023), respectively.

$0.1 \leq P(B) \leq 0.2$

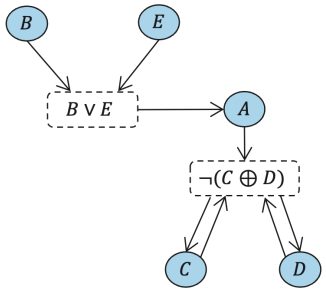
$0.05 \leq P(E) \leq 0.1$

$0.6 \leq P(A|B \vee E) \leq 0.9$

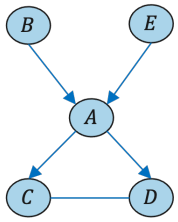
$0.5 \leq P(\neg(C \oplus D)|A) \leq 0.7$

$0.05 \leq P(A) \leq 0.1$

(a) LCN constraints



(b) LCN primal graph



(c) LCN structure (chain graph)

Figure 1: A simple LCN instance with its primal graph and corresponding structure.

Example 1. *Figure 1 shows a simple LCN with 5 propositions and 5 constraints. The corresponding primal graph and structure are given in Figures 1b and 1c, respectively. The LMC implies that: B and E are marginally independent, C is conditionally independent of B and E given C's parents $\{A, D\}$, and D is conditionally independent of B and E given $\{A, C\}$, respectively.*

2.2 Chain Graphs and Factorization

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of random variables, and let $G = (\mathcal{N}, E)$ be a graph whose nodes $\mathcal{N}$ are in one-to-one correspondence with the variables in $\mathbf{X}$. We denote a configuration of $\mathbf{X}$ by $\mathbf{x}$.

Definition 2 (chain graph)**.** *Let $G = (\mathcal{N}, E)$ be a mixed graph with $E = E_{\rightarrow} \cup E_{-}$, where $E_{\rightarrow}$ is a set of directed edges and E_{-} is a set of undirected edges. The graph G is a* chain graph *if there exists a partition $\mathcal{N} = \mathcal{T}_1 \dot{\cup} \mathcal{T}_2 \dot{\cup} \cdots \dot{\cup} \mathcal{T}_m$ such that: (i) for each i, the subgraph induced by $\mathcal{T}_i$ contains only undirected edges; (ii) for $i \neq j$, every edge between $\mathcal{T}_i$ and $\mathcal{T}_j$ is directed, and all such edges have the same orientation; (iii) the directed graph obtained by contracting each chain component $\mathcal{T}_i$ into a single node is acyclic. The sets $\mathcal{T}_1, \ldots, \mathcal{T}_m$ are called the* chain components *of G.*

If there is a directed edge from A to B in G then A is a *parent* of B and B is a *child* of A. The parents of A are denoted $\mathbf{pa}(A)$. If there is an undirected edge $A - B$ then both nodes are said to be *neighbors*. The neighbors of A are denoted by $\mathbf{ne}(A)$. The *boundary* of a node A, denoted $\mathbf{bd}(A)$, is the set $\mathbf{pa}(A) \cup \mathbf{ne}(A)$. The boundary of a set $\mathcal{B}$ of nodes is $\mathbf{bd}(\mathcal{B}) = \bigcup_{A \in \mathcal{B}} \mathbf{bd}(A) \setminus \mathcal{B}$. The Local Markov Condition (LMC(C)) for chain graphs is defined as follows:

Definition 3 (LMC(C))**.** *Let G be a chain graph. A node $A \in G$ is independent, given its boundary $\mathbf{bd}(A)$, of all nodes that are not A itself nor descendants of A nor boundary nodes of A.*

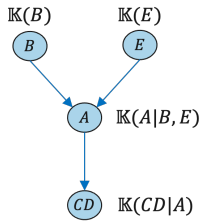
Given a chain graph G over variables $\mathbf{X}$, it can be shown that for strictly positive probabilities the local and global Markov conditions are equivalent (Cowell et al., 1999; Cozman, 2023). The implication is that the joint distribution $P(\mathbf{X})$ factorizes according to G as:

$$P(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{m} P(\mathcal{T}_i = \mathbf{x}_{\mathcal{T}_i} \mid \mathbf{pa}(\mathcal{T}_i) = \mathbf{x}_{\mathbf{pa}(\mathcal{T}_i)}), \tag{3}$$

where the chain components are ordered so that directed edges can only point from earlier to later components, $\mathbf{pa}(\mathcal{T}_i)$ denotes the set of nodes outside $\mathcal{T}_i$ with directed edges into $\mathcal{T}_i$, and $\mathbf{x}_{\mathcal{T}_i}$ and $\mathbf{x}_{\mathbf{pa}(\mathcal{T}_i)}$ are the projections of $\mathbf{x}$ onto the corresponding variables. Consequently, for an LCN whose structure is a chain graph, the local Markov condition for LCNs (Definition 1) coincides with Definition 3 and every distribution in $\mathcal{D}_{\text{LCN}}$ admits the factorization in equation 3 (Cozman, 2023).

3 From LCNs to Credal Networks

In this section, we establish the connection between LCNs and credal networks by exploiting the chain-graph factorization induced by the structure of an LCN. Specifically, we focus on LCNs whose underlying structure is a chain graph, referred to as *chain-graph LCNs*, and show how to reduce inference in LCNs to inference in a credal network. We first provide a brief overview of credal networks (Cozman, 2000).



(a) Credal network

$$\underline{P}(A=1|B=0,E=1)$$
$$\min \frac{P(a \wedge \neg b \wedge e)}{P(\neg b \wedge e)}$$
$$0.1 \le P(b) \le 0.2$$
$$0.05 \le P(e) \le 0.1$$
$$0.6 \le P(a|b \vee e) \le 0.9$$
$$0.05 \le P(a) \le 0.1$$

$$\overline{P}(A=1|B=0,E=1)$$
$$\max \frac{P(a \wedge \neg b \wedge e)}{P(\neg b \wedge e)},$$
$$0.1 \le P(b) \le 0.2$$
$$0.05 \le P(e) \le 0.1$$
$$0.6 \le P(a|b \vee e) \le 0.9$$
$$0.05 \le P(a) \le 0.1$$

(b) Local fractional LPs

B	E	A	𝕂(A|B,E)
0	0	0	[0.96,1.0]
0	0	1	[0.0,0.04]
0	1	0	[0.0,1.0]
0	1	1	[0.0, 1.0]
…	…	…	…

(c) Local credal set

Figure 2: The credal network for the chain-graph factorization of the LCN from Figure 1.

3.1 Credal Networks

A *credal network* (CN) (Cozman, 2000) is defined by a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of discrete variables with finite domains $\mathbf{D} = \{D_1, \ldots, D_n\}$, G is a directed acyclic graph (DAG) over $\mathbf{X}$ as nodes, and $\mathbf{K} = \{K(X_i \mid \mathbf{pa}(X_i) = \pi_{ik})\}$ is a set of separately specified conditional credal sets for each variable X_i and each k-th configuration π_{ik} of its parents $\mathbf{pa}(X_i)$ in G. A credal set is a set of probability distributions (Levi, 1980) and we consider credal sets that are closed and convex with a finite number of vertices (e.g., probability intervals). The *strong extension* $\mathcal{D}_{\text{SE}}$ of a credal network is the *convex hull* (denoted CH) of all joint distributions that satisfy the following Markov property: every variable is strongly independent of its non-descendants conditional on its parents (Cozman, 2000; Mauá & Cozman, 2020).

$$\mathcal{D}_{\text{SE}} = \text{CH}\{P(\mathbf{X}) \;:\; P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i \mid \pi_{ik}),\; P(X_i \mid \pi_{ik}) \text{ is a vertex of } K(X_i \mid \pi_{ik})\} \tag{4}$$

3.2 The Chain Graph Approximation for LCNs

Given a chain-graph LCN $\mathcal{L}$ over variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, we approximate $\mathcal{D}_{\text{LCN}}$ by the strong extension $\mathcal{D}_{\text{SE}}$ of an induced credal network. The construction is based on the chain components $\mathcal{T}_1, \ldots, \mathcal{T}_m$ of the underlying chain graph. We first build a DAG G' whose nodes correspond to these chain components. Each node represents the joint state space of the variables in the corresponding chain component, that is, the Cartesian product of their domains. Thus, a node corresponds to a compound variable whenever its chain component contains multiple variables, and to a single LCN variable otherwise. Let v denote a family in G', consisting of a child node ch_v and its parent set pa_v. For each parent configuration $pa_v = \pi$, we define a local credal set $K(ch_v \mid pa_v = \pi)$ (or K_v for short) by computing lower and upper bounds on the conditional probabilities of ch_v given π. Following the standard LCN encoding of Marinescu et al. (2022), these bounds are obtained by solving a local optimization problem involving only the probabilistic sentences whose variables belong to $ch_v \cup pa_v$. The resulting fractional linear programs (LPs) are solved using the Charnes–Cooper transformation (Charnes & Cooper, 1962).

Example 2. *Figure 2a shows the credal network induced by the chain-graph factorization of the LCN from Figure 1. Any joint distribution in $\mathcal{D}_{\text{SE}}$ factorizes as $P(B) \cdot P(E) \cdot P(A \mid B, E) \cdot P(C, D \mid A)$. In Figure 2b we outline the fractional LPs for computing lower and upper bounds on $P(A = 1 \mid B = 0, E = 1)$, respectively. A partial local credal set $K(A \mid B, E)$ is given in Figure 2c.*

Given a reference credal set $\mathcal{R}$ with true query bounds $[\underline{q}^\star, \overline{q}^\star]$, an estimated interval is *exact* when it equals the true bounds, an *outer bound* when it encloses them, and an *inner bound* when it is enclosed by them. Intervals that satisfy neither property are termed *unreliable*. We can show that:

Proposition 1 (inclusion)**.** *For any chain-graph LCN, $\mathcal{D}_{\text{LCN}} \subseteq \mathcal{D}_{\text{SE}}$. Consequently, any inference algorithm that is* exact *or* outer *for $\mathcal{D}_{\text{SE}}$ computes valid* outer bounds *for $\mathcal{D}_{\text{LCN}}$.*

3.3 Family-realizable Sentences

The proposed approximation introduces two looseness gaps. First, each K_v is computed per family in isolation, so it may admit conditionals that no global $p \in \mathcal{D}_{\text{LCN}}$ realizes – it ignores sentences on other families and the cross-family LMC equalities. Formally $K_v^{\text{true}} \subseteq K_v$, sometimes strictly. Second, even with $K_v = K_v^{\text{true}}$, the free product of equation 12 lets each family pick its row independently,

which can violate a sentence whose scope spans several families. We therefore introduce a class of LCN sentences, called *family-realizable sentences*, and identify conditions when $\mathcal{D}_{\mathrm{SE}} = \mathcal{D}_{\mathrm{LCN}}$.

Definition 4 (family-realizable sentence)**.** *Let $\mathcal{L}$ be a chain-graph LCN with families $\{(\mathrm{ch}_v, \mathrm{pa}_v)\}$, and let s be a sentence with functional g_s. Say s is* family-realizable in family v *if* $\mathrm{atoms}(s) \subseteq \mathrm{ch}_v \cup \mathrm{pa}_v$ *and the value $g_s(p)$ is determined by the conditional table $P(\mathrm{ch}_v \mid \mathrm{pa}_v)$* alone *— i.e. for any two joints p, p' that induce the same conditional table at v, $g_s(p) = g_s(p')$, whatever the parents' marginals $P(\mathrm{pa}_v)$ and $P'(\mathrm{pa}_v)$ may be. Concretely, s is family-realizable in v if and only if it has one of the two forms:*

(R1) Type-1 on a root block: *s is $\ell \le P(\varphi) \le u$ with $\mathrm{atoms}(\varphi) \subseteq \mathrm{ch}_v$ and $\mathrm{pa}_v = \varnothing$. Then $P(\varphi)$ is a function of $P(\mathrm{ch}_v)$, which* is *the family's table.*

(R2) Type-2 pinning one full parent configuration: *s is $\ell \le P(\varphi \mid \psi) \le u$ with $\mathrm{atoms}(\varphi) \subseteq \mathrm{ch}_v$, $\mathrm{atoms}(\psi) \subseteq \mathrm{pa}_v$, and ψ satisfied by* exactly one *assignment $\pi^\star$ of pa_v. Then $P(\varphi \mid \psi) = P(\varphi \mid \mathrm{pa}_v{=}\pi^\star)$ is a single row of the table.*

A sentence is non-realizable *if it is family-realizable in no family. The LCN is* fully realizable *if every sentence is family-realizable in some family*[1].

Proposition 2 (sufficiency)**.** *Let $\mathcal{L}$ be a chain-graph LCN with families $\{(ch_v, pa_v)\}$ and local credal sets built by the chain-graph approximation. Suppose (i) for every family v and parent configuration π, $K_{v,\pi} = K^{\mathrm{true}}_{v,\pi}$, the projection of $\mathcal{D}_{\mathrm{LCN}}$ onto that row; and (ii) every sentence of $\mathcal{L}$ is family-realizable in some family (Definition 7). Then $\mathcal{D}_{\mathrm{SE}} = \mathcal{D}_{\mathrm{LCN}}$.*

We next present three algorithms that compute exact, outer and inner bounds for $\mathcal{D}_{\mathrm{SE}}$, respectively.

3.4 Credal Variable Elimination

Credal Variable Elimination (CVE) extends Variable Elimination (VE) for Bayesian networks (Dechter, 2003) to credal networks by propagating finite sets of non-negative functions called *potentials* instead of functions (Mauá & Cozman, 2020).

Definition 5 (potential)**.** *A* potential $\phi(\mathbf{Y})$ *over variables $\mathbf{Y}$ is a finite set of non-negative real-valued functions $\{f_1, f_2, \ldots, f_m\}$ on the joint domain of $\mathbf{Y}$. The* product *(or combination) of two potentials is defined by $\phi{\cdot}\psi = \{f{\cdot}g : f \in \phi,\ g \in \psi\}$, where $(f{\cdot}g)(\mathbf{y}, \mathbf{z}) = f(\mathbf{y}){\cdot}g(\mathbf{z})$ over the joint domain $\mathbf{Y}{\cup}\mathbf{Z}$. The* sum-marginal *is defined by $\sum_X \phi(\mathbf{Y}) = \{\sum_x f(\mathbf{y}) : f \in \phi\}$, producing functions over $\mathbf{Y} \setminus \{X\}$. A* pruning *operation is defined by $prune(\phi) = \{f \in \phi : \nexists\, g \in \phi \setminus \{f\}$ s.t. $g(\mathbf{y}) \ge f(\mathbf{y})\ \forall\, \mathbf{y}\}$.*

Given a query variable Z, an elimination order $\mathbf{o}$ such that Z appears last and initial potentials obtained from the extreme points of the conditional credal sets, each potential is assigned to the bucket of the first variable in its scope. The algorithm then processes each bucket, from first to last in the elimination ordering o but excluding Z, by multiplying all potentials in the current bucket and eliminating the bucket's variable by summation, resulting in a new potential which is first pruned by its non-maximal elements and then placed in a subsequent bucket, depending on its scope. Processing the last bucket by first multiplying all its potentials, normalizing and taking component-wise minima and maxima yields lower and upper bounds for Z (see Appendix for more details and examples).

Since the intermediate potentials generated by CVE can grow exponentially, we can approximate the potentials by fixing their cardinality to a relatively small number B (Mauá & Cozman, 2020; Marinescu et al., 2026). Specifically, we group the elements of $\phi(\mathbf{Y})$ into B clusters based on minimizing the Manhattan distance to each cluster's centroid (i.e., minimize $\sum_{i=1}^{m} |p_i - r_i|$, where p_i and r_i are the i-th components of p and r, respectively). Then, for each cluster we replace its components with their mean values. This yields an approximation method denoted by CVE_{cm}.

Theorem 1 (complexity)**.** *The time and space complexity of CVE is $O(m \cdot C \cdot d^{w^*})$ where m is the number of variables, d bounds the domain size, w^* is the treewidth, and C is the maximum potential cardinality (which can be exponential). With potential approximation, C is bounded by B.*

Theorem 2 (exactness)**.** *Let $\mathcal{L}$ be a family-realizable chain-graph LCN. Then CVE computes* exact bounds *with respect to $\mathcal{D}_{\mathrm{LCN}}$.*

[1]Note that a sentence can be attached to several families while being realizable in only one of them.

3.5 INTERVAL BELIEF PROPAGATION

Given a credal network compiled to a *factor graph* (Koller & Friedman, 2009) whose factor nodes $\{f_i\}$ store the finite sets of extreme points $\operatorname{ext} K(X_i \mid \pi)$ (one per parent configuration π of the associated variable X_i), plus evidence $\mathbf{Y} = \mathbf{y}$, our second approach called Interval Belief Propagation (IBP) computes bounds $[\underline{P}(X_i{=}x \mid \mathbf{y}),\ \overline{P}(X_i{=}x \mid \mathbf{y})]$ for every non-evidence variable X_i and every state $x \in D_{X_i}$. Unlike previous algorithms like L2U (Ide & Cozman, 2008) and GL2U (Antonucci et al., 2010) which are restricted to binary variables, IBP natively handles multi-valued variables.

Each directed edge of the factor graph carries an interval-vector message $\mu_{v\to f} = (\mathbf{l}_{v\to f}, \mathbf{u}_{v\to f}) \in [0,1]^{|D_v|} \times [0,1]^{|D_v|}$ (and symmetrically $\mu_{f\to v}$) with $\mathbf{l} \le \mathbf{u}$ componentwise. All messages are initialized to the vacuous interval $(\mathbf{0}, \mathbf{1})$. For every evidence assignment $Y_j = y_j$, the outgoing variable-to-factor messages from Y_j are clamped to the one-hot indicator $\mu_{Y_j\to f} \leftarrow (\delta_{y_j}, \delta_{y_j})$ and held fixed throughout.

At every iteration, two sweeps are performed. *Variable-to-factor* messages (skipped for evidence variables) tighten by intersecting the other incoming bounds:

$$\mathbf{l}_{v\to f} \;=\; \max_{f'\in\mathcal{N}(v)\setminus\{f\}} \mathbf{l}_{f'\to v}, \qquad \mathbf{u}_{v\to f} \;=\; \min_{f'\in\mathcal{N}(v)\setminus\{f\}} \mathbf{u}_{f'\to v}, \qquad v \notin \mathbf{Y} \tag{5}$$

(component-wise, over all neighbor factors $\mathcal{N}(v)$ other than f). *Factor-to-variable* messages extremize the target's normalized marginal over two discrete choice sets:

$$\mathbf{l}_{f_i\to v}(x) \;=\; \min_{(\mathbf{vc},\,\mathbf{cc})} \tilde{p}_{\mathbf{vc},\mathbf{cc}}(x), \qquad \mathbf{u}_{f_i\to v}(x) \;=\; \max_{(\mathbf{vc},\,\mathbf{cc})} \tilde{p}_{\mathbf{vc},\mathbf{cc}}(x), \tag{6}$$

where a *vertex combination* $\mathbf{vc}$ picks one extreme point $P_{\mathbf{vc}}(X_i \mid \pi) \in \operatorname{ext} K(X_i \mid \pi)$ for every parent configuration π, a *corner-point combination* $\mathbf{cc}$ picks $\mathbf{q}_w \in \{\mathbf{l}_{w\to f_i}, \mathbf{u}_{w\to f_i}\}$ for each other-variable $w \in \operatorname{scope}(f_i) \setminus \{v\}$, and $\tilde{p}_{\mathbf{vc},\mathbf{cc}}(x) = m_{\mathbf{vc},\mathbf{cc}}(x) / \sum_{x'} m_{\mathbf{vc},\mathbf{cc}}(x')$ with $m_{\mathbf{vc},\mathbf{cc}}$ the unnormalized marginal (see details and examples in Appendix). The two sweeps are repeated until the maximum component-wise change across all messages falls below a tolerance τ, or a cap of T iterations is reached. The final conditional marginal bounds for every variable are obtained by intersecting all factor-to-variable messages incident on it:

$$\underline{P}(X_i{=}x \mid \mathbf{y}) = \max_{f\in\mathcal{N}(X_i)} \mathbf{l}_{f\to X_i}(x), \qquad \overline{P}(X_i{=}x \mid \mathbf{y}) = \min_{f\in\mathcal{N}(X_i)} \mathbf{u}_{f\to X_i}(x). \tag{7}$$

Theorem 3 (complexity)**.** *The complexity per iteration of IBP is $O(N_f \cdot V_{\max} \cdot 2^{S_{\max}} \cdot k^{S_{\max}})$, where N_f is the number of factors, $V_{\max}$ the maximum vertex combinations per factor, $S_{\max}$ the maximum scope size, and k the maximum domain size.*

Theorem 4 (exactness)**.** *Let $\mathcal{L}$ be a family-realizable chain-graph LCN. If $\mathcal{L}$'s structure is singly connected (i.e., a tree), then IBP computes* exact bounds *with respect to $\mathcal{D}_{\mathrm{LCN}}$.*

3.6 COORDINATE DESCENT WITH VARIABLE ELIMINATION

Our third approach called Coordinate Descent with Variable Elimination (CDVE) extends the ApproxLP method (Antonucci et al., 2015) that reformulates credal marginal inference as a multilinear program and solves it via coordinate descent. In our case, for every non-evidence variable Z, CDVE runs coordinate descent twice — once with sense $\min$ and once with sense $\max$. Each run starts by initializing all $P(X_i \mid \pi_i)$ at the centroid of their respective credal sets and computing by exact Variable Elimination (VE) $p^* = \mathrm{VE}(\{P\}, Z)$, the full posterior vector on D_Z under these fixed distributions. A *sweep* then visits every (X_i, π_i): for each vertex $v \in \operatorname{ext} K(X_i \mid \pi_i)$, it temporarily sets $P(X_i \mid \pi_i) \leftarrow v$, re-evaluates VE, and accepts v (strictly, else rejects) if the target component $p^*(z)$ moves in the sense's direction. Sweeps repeat until one produces no improvement or the cap T is reached. Because each VE call returns a full vector, the two runs together yield componentwise bounds $[\underline{p}(z), \overline{p}(z)]$ for every state $z \in D_Z$ simultaneously. Evidence variables (if any) are skipped by a point-mass shortcut. The output is *inner bounds* for $\mathcal{D}_{\mathrm{SE}}$ — coordinate descent explores only a subset of feasible vertex combinations, so the returned interval is contained in the true interval.

Theorem 5 (complexity)**.** *The time and space complexity of CDVE is $O(T \cdot V_{total} \cdot n \cdot k^{w^*})$ and $O(n \cdot k^{w^*})$, where V_{total} is the total number of extreme points across all credal sets, k bounds the domain size, and w^* is the induced width.*

4 EXACT LCN INFERENCE AT TREEWIDTH COST

Recall that exact inference in LCNs amounts to computing the lower and upper probabilities $\underline{P}(a) = \min_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a)$ and $\overline{P}(a) = \max_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a)$, respectively. Direct optimization over $\mathcal{D}_{\mathrm{LCN}}$ requires a nonlinear program with 2^n variables (Marinescu et al., 2022). Alternatively, compiling the LCN into a credal network and performing inference as described in Section 3 is computationally cheaper, but in general yields bounds over the strictly larger strong extension $\mathcal{D}_{\mathrm{SE}}$.

To address these limitations, we introduce Credal Junction Tree (CJT), a novel algorithm for exact inference in LCNs. CJT preserves the exactness of the nonlinear optimization approach of Marinescu et al. (2022) while requiring a nonlinear program that is exponentially smaller. The key idea is to exploit a junction-tree decomposition of the credal network induced by the chain-graph factorization. Specifically, CJT first moralizes and triangulates the underlying chain graph, producing a chordal graph G^*. It then constructs a junction tree T whose clusters $\mathcal{C}$ satisfy the running-intersection property (Dechter, 2003). For each cluster $C \in \mathcal{C}$, the algorithm introduces a local marginal variable vector $q_C \in \Delta(2^{|C|})$ representing a distribution over the variables in C. Exact inference is then reduced to the following optimization problem (see the Appendix for more details and examples):

$$
\begin{aligned}
\min \,/\, \max \quad & \rho_a(q) && \\
\text{s.t.} \quad & \textstyle\sum_i q_{C,i} = 1,\ q_{C,i} \geq 0 && \forall C \in \mathcal{C}, \\
& q_C \restriction S = q_D \restriction S && \forall (C, D) \in T,\ S = C \cap D, \\
& \ell \langle A_\psi, q_{C_h} \rangle \leq \langle A_{\varphi \wedge \psi}, q_{C_h} \rangle \leq u \langle A_\psi, q_{C_h} \rangle && \forall \text{ sentence hosted on } C_h, \\
& \langle A_a, q \rangle \langle A_b, q \rangle = \langle A_c, q \rangle \langle A_d, q \rangle && \forall \text{ LMC assertion hosted,}
\end{aligned}
$$

where A_φ is the $0/1$ indicator vector of φ over the host cluster's states, a constraint is *hosted* on any cluster containing its scope, $\langle A_\varphi, q \rangle$ is the dot-product of two vectors, and ρ_a reads $P(a)$ (or $P(a \mid \mathbf{e})$, via one auxiliary variable) off a cluster containing a and the evidence. For every pair of adjacent clusters C and D, the marginals induced on their separator S, denoted by $q_C \restriction S$ and $q_D \restriction S$, are required to coincide, ensuring global consistency. An LMC assertion whose conditioning set separates its two sides in G^* is *separator-entailed* and is omitted: the junction tree's topology already enforces it. The program is nonconvex through the bilinear LMC assertions and is solved globally.

Theorem 6 (exactness)**.** *Let $\mathcal{L}$ be a chain-graph LCN and T a junction tree of its chordal graph G^*. Assume that* (H1) *T satisfies the running-intersection property;* (H2) *every probabilistic sentence is hosted by some cluster of T;* (H3) *every LMC assertion is either hosted by a cluster or separator-entailed in G^*;* (H4) *the query and evidence variables are contained in a common cluster. Then CJT computes* exact *lower and upper probabilities over $\mathcal{D}_{\mathrm{LCN}}$.*

The formulation needs $\sum_C 2^{|C|} = O(n 2^{w+1})$ variables and $O(n 2^w)$ separator-consistency constraints making its complexity exponential in the treewidth w rather than in the number of variables n.

5 EXPERIMENTS

In this section, we evaluate the proposed marginal inference schemes on several classes of LCNs, comparing them with the exact method of Marinescu et al. (2022) and the approximate ARIEL algorithm (Marinescu et al., 2023). All algorithms are implemented in Python 3.12 and use `ipopt` 3.14 (Wächter & Biegler, 2006) for the nonlinear programs. Every run has a limit of 3600 seconds and 30 GB of memory; a dash in a table means the algorithm did not finish inside that budget.

Benchmarks. We consider two types of LCN instances. *Random LCNs* come in four topologies, each generated so that every sentence is *family-realizable*: `tree`, `polytree`, `ktree` (k=3) and `dag` (moralized treewidth ≤ 3, at most two parents per atom). Each topology is generated at two scales, *small* ($n \in \{5, \ldots, 10\}$) and *large* ($n \in \{10, 20, 30, 40, 50, 100\}$), with 10 instances per size, giving $8 \times 60 = 480$ instances. Sentence bounds are drawn as $[\max(0.01, v{-}\varepsilon), \min(1, v{+}\varepsilon)]$ with $v \sim U(0, 1)$ and $\varepsilon = 0.3$; formulas span at most three atoms; two extra marginals per instance are placed on root atoms only; the seed is 42. Every instance is accepted only after a product-distribution consistency witness is found, so all 480 are consistent by construction. *Realistic LCNs* are of two kinds: ten instances derived from real Bayesian networks (Constantinou et al., 2020) (n = 4–11, 6–24 sentences), and sixteen instances from the causal chain graphs of the US Supreme Court analysis of

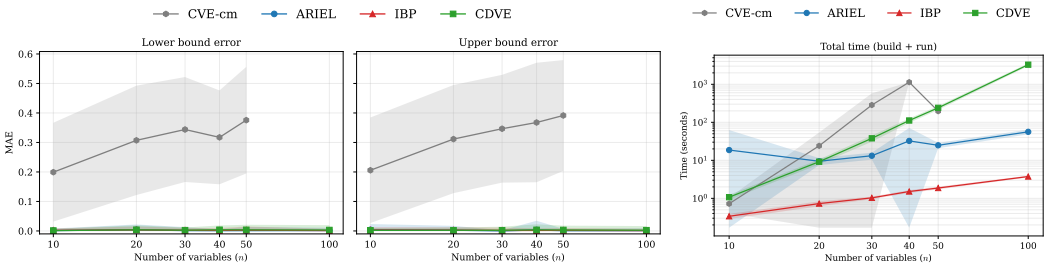


Figure 3: Large random `dag` LCNs: (left) MAEs against the CJT reference, with bands giving one standard deviation over 10 instances per size – curves stopping where the budget was exceeded; (right) total time in seconds.

Instance	Algorithm	MAE_l	MAE_u	Time (s)
Alarm (n=10), CJT ref 0.72s	ARIEL	0.000	0.000	3.92
	IBP	0.101	0.015	0.13
	CDVE	0.101	0.015	0.69
	CVE_{cm}	0.087	0.067	0.24
Credit (n=11), CJT ref 1.15s	ARIEL	0.050	0.000	3.18
	IBP	0.257	0.000	0.24
	CDVE	0.257	0.000	0.69
	CVE_{cm}	0.108	0.165	1.73
Hepatitis (n=10), CJT ref 0.56s	ARIEL	0.008	0.008	3.91
	IBP	0.069	0.080	0.75
	CDVE	0.070	0.085	1.15
	CVE_{cm}	0.114	0.184	16.53
MScaling (n=12), CJT ref 0.66s	ARIEL	0.028	0.000	8.13
	IBP	0.129	0.493	0.35
	CDVE	0.129	0.501	1.23
	CVE_{cm}	0.079	0.227	0.57
BScaling (n=32), CJT ref 2.42s	ARIEL	0.000	0.000	19.80
	IBP	0.115	0.225	0.69
	CDVE	0.115	0.225	20.55
	CVE_{cm}	0.064	0.072	9.18
PScaling (n=40), CJT ref 3.27s	ARIEL	0.000	0.000	34.72
	IBP	0.129	0.263	0.91
	CDVE	0.129	0.263	51.25
	CVE_{cm}	0.070	0.057	658.84

Table 1: Realistic LCNs: three instances derived from real Bayesian networks (top) and three Supreme Court chain-graph instances (bottom), all against the CJT reference. Time is total time including the credal-network build; the reference's own time on each instance is given with the instance name.

Ogburn et al. (2020). The latter stress two different axes: the `scaling` instances are large and sparse (n = 6–40, 11–79 sentences), whereas the `supreme_court` instances are small but extremely dense (n=10 with 345–1009 sentences).

Algorithms, reference and metrics. We compare four approximate algorithms: ARIEL (Marinescu et al., 2023), which operates directly on the LCN factor graph (10 iterations) and needs no credal-network build; IBP, interval belief propagation on the credal network (100 iterations); CDVE, coordinate descent over extreme points (50 iterations), which by construction returns *inner* bounds; and CVE_{cm}, Credal Variable Elimination with its intermediate potentials clustered to at most 10 functions using the mean representative. Plain CVE (exact over the strong extension, no clustering) is included where it terminates. All credal-network methods share the same build: the chain-graph factorization, no scope merging, and per-family local credal sets solved as linear programs. As the reference we use CJT, the junction-tree NLP formulation (using the `ipopt` backend): it is the tightest engine tractable at every size here, whereas exact inference of Marinescu et al. (2022) over the 2^n joint solves only $15/60$ of `tree_small` and none of the Supreme Court instances. We report the mean absolute error of the lower and upper marginal bounds against the reference, $\text{MAE}_l = \frac{1}{n}\sum_i |\underline{P}^*(x_i) - \underline{P}(x_i)|$ and MAE_u analogously, and wall-clock time in seconds including

the credal-network build. Note that CJT's build is not comparable to the others': it needs only the chain-graph structure and skips the vertex enumeration entirely. Recall that family realizability requires each sentence to constrain a single family conditional distribution. Thus, for singly connected family-realizable LCNs (e.g., `tree` and `polytree`), both IBP and CVE are exact. For family-realizable but loopy LCNs (e.g., `ktree` and `dag`), where moralization creates cycles, only CVE is guaranteed to be exact.

Random LCNs. Figure 3 shows the loopy `dag` LCNs at the large scale, with shaded bands giving one standard deviation over the ten instances per size (full results on all LCNs are included in the Appendix). Three findings hold across all four topologies. First, IBP dominates: its error against the reference is at or below 4×10^{-3} on every scale (small, large) and size — indistinguishable from CJT on `tree` and `polytree`, as the exactness characterization predicts — while being the fastest method everywhere, by one to three orders of magnitude at n=100. Second, CDVE matches that accuracy but not that cost: its mean error stays in the 10^{-3} range, yet its runtime grows from under a second on the small instances to 273–614 seconds on the large ones, making it slower than ARIEL at n=100. ARIEL is likewise accurate (exactly zero error on `tree` and `polytree`) but pays 25–35 seconds at scale. Third, approximating the potentials by clustering buys speed at a real cost in quality: CVE$_{cm}$ runs in a fraction of a second on the small instances but its error grows with n, from 0.08–0.11 on the small instances to 0.31–0.35 on the large ones. Plain CVE is the scalability bottleneck: it is essentially exact where it runs but costs 226–1141 seconds, times out on all of `ktree_small` and `polytree_large`, and is intractable on the large loopy LCNs.

Realistic LCNs. Table 1 reports six representative instances. These instances are chain-graph LCNs but they are not family-realizable and therefore Proposition 4 does not hold. The ordering is different here: ARIEL is clearly the tightest method, at or near zero error on four of the six instances, because it reasons on the original LCN rather than on its credal-network relaxation. IBP and CDVE agree closely and produce widths 1.4–7.7× the reference — genuine but loose outer bounds, the gap being the cost of the relaxation rather than a defect of the message passing. On running time the picture is again favourable to IBP, which is the fastest method on every instance and the only one that stays under a second at n=40, where CDVE needs 51 seconds and CVE$_{cm}$ 659.

6 Related Work

Credal networks (CNs) (Cozman, 2000) generalize Bayesian networks by replacing precise conditional probabilities with credal sets. Despite the computational hardness of exact inference (Mauá & Cozman, 2020), the area has produced a rich algorithmic toolbox, including 2U/L2U (Ide & Cozman, 2008), GL2U (Antonucci et al., 2010), and ApproxLP (Antonucci et al., 2015). In contrast, inference in LCNs remains largely unexplored, with existing methods limited to exact constrained nonlinear optimization (Marinescu et al., 2022) and the ARIEL message-passing algorithm (Marinescu et al., 2023); related work has focused on abductive reasoning (Marinescu et al., 2024) and the theoretical development of chain-graph factorizations (Cozman, 2023). Our work closes this gap by turning the latter into an algorithmic bridge between LCNs and CNs, enabling the first systematic transfer of credal-network inference methods to the LCN setting. More broadly, LCNs are related to probabilistic logic frameworks such as ProbLog (De Raedt et al., 2007), DeepProbLog (Manhaeve et al., 2021), MLNs (Richardson & Domingos, 2006), and PSL (Bach et al., 2017), but are distinguished by their support for imprecise probability assessments and minimal structural assumptions.

7 Conclusion

In this paper, we introduced a credal network perspective on Logical Credal Networks, yielding the first general framework for transferring credal network inference algorithms to the LCN setting. This connection led to a family of exact and approximate inference methods, including CVE, CVE$_{cm}$, IBP, CDVE, and CJT. In particular, CJT achieves exact LCN inference by exploiting a junction-tree decomposition to solve an exponentially smaller nonlinear program than previous exact approaches. Our theoretical and empirical results show that these methods substantially expand the scalability-accuracy trade-off space for LCN inference, thus providing a richer algorithmic toolbox for reasoning under logical and probabilistic uncertainty.

AI USE STATEMENT

We used generative AI tools to assist with language editing, grammar correction, and code development. In particular, AI-assisted coding tools were used during the implementation of the proposed algorithms. All AI-generated content was reviewed, verified, and, where applicable, extensively tested by the authors. For example, AI-generated code was manually inspected for correctness and validated through unit and integration testing. The authors take full responsibility for the accuracy, validity, and integrity of all content presented in this paper.

ETHICS STATEMENT

This work did not involve human subjects.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have included the Python code and benchmark problem instances in the supplementary material. We will open-source the code and benchmarks if the paper is accepted.

REFERENCES

A. Antonucci, C. P. de Campos, D. Huber, and M. Zaffalon. Approximate credal network updating by linear programming with applications to decision making. *International Journal of Approximate Reasoning*, 58:25–38, 2015.

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.

Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 18(1):1–67, 2017.

A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962.

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.

Robert Cowell, Philip Dawid, Steffen Lauritzen, and David Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, 1999.

Fabio Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

Fabio Gagliardi Cozman. Markov conditions and factorization in logical credal networks. In Enrique Miranda, Ignacio Montes, Erik Quaeghebeur, and Barbara Vantaggi (eds.), *Proceedings of the Thirteenth International Symposium on Imprecise Probability: Theories and Applications*, volume 215 of *Proceedings of Machine Learning Research*, pp. 130–140. PMLR, 11–14 Jul 2023. URL `https://proceedings.mlr.press/v215/cozman23a.html`.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 7, pp. 2462–2467, 2007.

R. Dechter. Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Uncertainty in Artificial Intelligence (UAI'96)*, pp. 211–219, 1996.

R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

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

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

I. Levi. *The Enterprise of Knowledge*. MIT Press, 1980.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504, 2021. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2021.103504. URL https://www.sciencedirect.com/science/article/pii/S0004370221000552.

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.

Radu Marinescu, Haifeng Qian, Alexander Gray, Debarun Bhattacharjya, Francisco Barahona, Tian Gao, and Ryan Riegel. Approximate inference in logical credal networks. In *32nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

Radu Marinescu, Fabio Cozman, Debarun Bhattacharjya, Junkyu Lee, and Alexander Gray. Abductive reasoning in logical credal networks. In *38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Radu Marinescu, Fabio Cozman, Denis Mauá, Debarun Bhattacharjya, Junkyu Lee, and Alexander G. Gray. Branch and bound search for exact MAP inference in credal networks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DTqbEtXXP3.

Denis Mauá and Fabio Cozman. Thirty years of credal networks: Specifications, algorithms and complexity. *International Journal of Approximate Reasoning*, 1(126):133–137, 2020.

Elizabeth L. Ogburn, Ilya Shpitser, and Youjin Lee. Causal inference, social networks, and chain graphs, 2020. URL https://arxiv.org/abs/1812.04990.

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

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

A PRELIMINARIES

A Logical Credal Network specifies imprecise probabilistic knowledge by bounding the probabilities of propositional formulas: a *sentence* is either $\ell \leq P(\varphi) \leq u$ (Type-1) or $\ell \leq P(\varphi \mid \psi) \leq u$ (Type-2), over propositional formulas φ, ψ on binary atoms. Together with the independence assumptions that the Local Markov Condition derives from the program's structure, a set of sentences determines a closed set $\mathcal{D}_{\mathrm{LCN}}$ of joint distributions, and the inference task is to compute $\underline{P}(a) = \min_{p \in \mathcal{D}_{\mathrm{LCN}}} P_p(a)$ and $\overline{P}(a) = \max_{p \in \mathcal{D}_{\mathrm{LCN}}} P_p(a)$ for each atom a.

Two families of inference methods exist. The *direct* exact method of Marinescu et al. (2022) writes one variable per joint state and optimizes over $\mathcal{D}_{\mathrm{LCN}}$ itself; it is exact when its solver certifies global optimality, but its 2^n formulation confines it to small instances. The *credal-network* family described in this paper (CVE, IBP, CDVE) compiles the LCN into a chain-graph credal network with interval local credal sets, and propagates messages; it scales, but it optimizes over the strong extension $\mathcal{D}_{\mathrm{SE}} \supseteq \mathcal{D}_{\mathrm{LCN}}$.

CJT belongs to neither family. It does not compile a credal network: it never reads the per-family interval bounds and never enumerates extreme points, consuming only the chain graph's *structure* and the LCN's own sentences and independence assertions. Consequently it optimizes over a relaxation of $\mathcal{D}_{\mathrm{LCN}}$ built from cluster marginals, not over $\mathcal{D}_{\mathrm{SE}}$ — and under the hypotheses of Theorem 6 that relaxation is exact. Its cost is $\sum_C 2^{|C|} = O(n\, 2^{w+1})$, exponential in treewidth rather than in n.

A.1 NOTATION

Throughout, $\mathbf{V}$ is a set of n binary atoms, $\Delta(2^{\mathbf{V}})$ the simplex of joint distributions over the 2^n truth assignments, and $p \restriction A$ the marginal of p on an atom subset $A \subseteq \mathbf{V}$. For a propositional formula φ we write $\mathrm{atoms}(\varphi)$ for its atoms and $\omega \models \varphi$ for satisfaction.

A.2 THE CREDAL SET $\mathcal{D}_{\mathrm{LCN}}$

A Type-1 sentence $\ell \le P(\varphi) \le u$ contributes the two linear rows $\ell \le \sum_{\omega \models \varphi} p(\omega) \le u$. A Type-2 sentence $\ell \le P(\varphi \mid \psi) \le u$ is read in *cleared* form,

$$\ell \cdot P(\psi) \;\le\; P(\varphi \wedge \psi) \;\le\; u \cdot P(\psi), \tag{8}$$

which avoids a division and is the form the implementation imposes.

Remark 1 (The cleared reading is the one in force)**.** *The cleared rows equation 8 are vacuously satisfied on $\{p : P(\psi) = 0\}$, so they admit joints that the ratio reading $\ell \le P(\varphi \wedge \psi)/P(\psi) \le u$ excludes. The two readings therefore define different credal sets. Every statement in this paper is made for the cleared reading, matching our implementation as well.*

The Local Markov Condition attaches to the program a set of conditional independence assertions $(X \perp\!\!\!\perp Y \mid S)$ with X a singleton. Each is imposed as the family of *joint* equalities

$$P(\mathbf{x}, \mathbf{y}, \mathbf{s})\, P(\mathbf{s}) = P(\mathbf{x}, \mathbf{s})\, P(\mathbf{y}, \mathbf{s}), \tag{9}$$

one for each configuration $(\mathbf{y}, \mathbf{s})$ of $Y \cup S$. Writing g_s for the functional of a sentence s and h_ι for the residual of an assertion ι,

$$\mathcal{D}_{\mathrm{LCN}} = \big\{\, p \in \Delta(2^{\mathbf{V}}) \;:\; \ell_s \le g_s(p) \le u_s \;\forall s, \qquad h_\iota(p) = 0 \;\forall \iota \,\big\}, \tag{10}$$

and the inference targets are $\underline{P}(a) = \min_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a)$, $\overline{P}(a) = \max_{p \in \mathcal{D}_{\mathrm{LCN}}} P(a)$.

The following elementary lemma is the workhorse of our theoretical contribution: it is what licenses imposing a constraint on a single cluster.

Lemma 1 (Locality)**.** *Every constraint in equation 10 depends on p only through its marginal on a fixed atom set, its* scope*: a sentence s through $p \restriction \mathrm{atoms}(s)$, and an LMC assertion $(X \perp\!\!\!\perp Y \mid S)$ through $p \restriction (X \cup Y \cup S)$. Consequently, if p and p' satisfy $p \restriction A = p' \restriction A$ for the scope A of a constraint c, then c holds for p iff it holds for p'.*

Proof. Let c be a constraint with scope A. There are three cases.

Type-1. $P(\varphi) = \sum_{\omega \models \varphi} p(\omega)$. Since $\mathrm{atoms}(\varphi) \subseteq A$, the truth value of φ at ω is determined by $\omega \restriction A$; grouping the sum by the value of $\omega \restriction A$ gives $P(\varphi) = \sum_{\alpha \models \varphi} (p \restriction A)(\alpha)$, where α ranges over assignments of A. This is a function of $p \restriction A$ alone.

Type-2. By the previous case both $P(\varphi \wedge \psi)$ and $P(\psi)$ are functions of $p \restriction A$ with $A \supseteq \mathrm{atoms}(\varphi) \cup \mathrm{atoms}(\psi)$; the rows equation 8 are therefore functions of $p \restriction A$.

LMC. Each factor in equation 9 is a probability of a conjunction of literals over $X \cup Y \cup S$, hence by the Type-1 case a function of $p \restriction (X \cup Y \cup S)$; a product of such functions is again one.

In each case the constraint's truth value is determined by $p \restriction A$, which is precisely the stated consequence. □

A.3 CHAIN GRAPH, STRONG EXTENSION, AND THE TWO GAPS

The LCN pipeline moralizes and simplifies the program's structure graph into a *chain graph* whose nodes are chain components; an undirected component collapses into a single *compound* node (rendered with a −-joined name, e.g. `C−D`). This yields a factorization

$$P(\mathbf{V}) = \prod_{v} P(\mathrm{ch}_v \mid \mathrm{pa}_v), \tag{11}$$

one factor per node v, where ch_v is the (possibly multi-atom) chain component at v and pa_v its parents. The *family scope* at v is $\mathrm{ch}_v \cup \mathrm{pa}_v$; root families have $\mathrm{pa}_v = \varnothing$.

The credal-network based methods replace each factor by an interval local credal set $K_{v,\pi}$ per parent configuration π and propagate over the *strong extension*:

$$\mathcal{D}_{\mathrm{SE}} = \left\{ \prod_v P_v \;:\; P_v \in K_{v,\pi} \text{ chosen independently for each parent configuration } \pi \right\} \;\supseteq\; \mathcal{D}_{\mathrm{LCN}}. \tag{12}$$

The inclusion is generally strict, for two reasons that it is worth naming since they are exactly what CJT avoids:

- **Gap A (local-set widening).** Each K_v is computed per family in isolation, so it may admit conditionals that no global $p \in \mathcal{D}_{\mathrm{LCN}}$ realizes: it ignores sentences on *other* families and the cross-family LMC equalities.
- **Gap B (strong-extension widening).** Even with exact local sets, the free product in equation 12 lets each family pick its row independently, which can violate a sentence whose scope spans several families.

CJT reads neither the interval bounds nor the extreme points of any $K_{v,\pi}$; it uses equation 11 only for its *scopes*. Neither gap therefore arises, and the engine's target is $\mathcal{D}_{\mathrm{LCN}}$ itself. This is the structural reason it is the only engine in the credal-network directory whose bound character is stated against $\mathcal{D}_{\mathrm{LCN}}$ rather than $\mathcal{D}_{\mathrm{SE}}$.

Definition 6 (Bound character)**.** *Let $\mathcal{R}$ be a reference set (here $\mathcal{D}_{\mathrm{LCN}}$ or $\mathcal{D}_{\mathrm{SE}}$) with true bounds $[\underline{q}^\star, \overline{q}^\star]$ for the query at hand. An engine's interval $[\underline{q}, \overline{q}]$ is* exact w.r.t. $\mathcal{R}$ *if $[\underline{q}, \overline{q}] = [\underline{q}^\star, \overline{q}^\star]$; a valid* outer *bound if $[\underline{q}, \overline{q}] \supseteq [\underline{q}^\star, \overline{q}^\star]$; an* inner *bound if $[\underline{q}, \overline{q}] \subseteq [\underline{q}^\star, \overline{q}^\star]$; and* unreliable *if neither containment is guaranteed. An outer bound is sound as a guarantee; an inner bound is not.*

B FAMILY-REALIZABLE SENTENCES

Which sentences can a *single* family's local set enforce? A family's degrees of freedom are the conditional probabilities $P(\mathrm{ch}_v{=}c \mid \mathrm{pa}_v{=}\pi)$; the parents' marginal $P(\mathrm{pa}_v)$ is not one of them — it is determined elsewhere in the network, and is a free variable during the local solve. A sentence that constrains a quantity depending on $P(\mathrm{pa}_v)$ therefore cannot be enforced locally, and the free product equation 12 may violate it. This is the mechanism behind Gap B, and the following definition isolates it.

Definition 7 (Family-realizable sentence)**.** *Let $\mathcal{L}$ be a chain-graph LCN with families $\{(\mathrm{ch}_v, \mathrm{pa}_v)\}$, and let s be a sentence with functional g_s. Say s is* family-realizable in family v *if $\mathrm{atoms}(s) \subseteq \mathrm{ch}_v \cup \mathrm{pa}_v$ and the value $g_s(p)$ is determined by the conditional table $P(\mathrm{ch}_v \mid \mathrm{pa}_v)$* alone *— i.e. for any two joints p, p' that induce the same conditional table at v, $g_s(p) = g_s(p')$, whatever the parents' marginals $P(\mathrm{pa}_v)$ and $P'(\mathrm{pa}_v)$ may be. Concretely, s is family-realizable in v if and only if it has one of the two forms:*

(R1) Type-1 on a root block: *s is $\ell \le P(\varphi) \le u$ with $\mathrm{atoms}(\varphi) \subseteq \mathrm{ch}_v$ and $\mathrm{pa}_v = \varnothing$. Then $P(\varphi)$ is a function of $P(\mathrm{ch}_v)$, which* is *the family's table.*

(R2) Type-2 pinning one full parent configuration: *s is $\ell \le P(\varphi \mid \psi) \le u$ with $\mathrm{atoms}(\varphi) \subseteq \mathrm{ch}_v$, $\mathrm{atoms}(\psi) \subseteq \mathrm{pa}_v$, and ψ satisfied by* exactly one *assignment $\pi^\star$ of pa_v. Then $P(\varphi \mid \psi) = P(\varphi \mid \mathrm{pa}_v{=}\pi^\star)$ is a single row of the table.*

A sentence is non-realizable *if it is family-realizable in no family. The LCN is* fully realizable *if every sentence is family-realizable in some family.*

The next lemma is what makes Definition 7 a *criterion* rather than a description: it shows the two forms (R1)–(R2) exhaust the $P(\mathrm{pa}_v)$-invariant sentences, so realizability can be decided syntactically.

Lemma 2 (Invariance criterion)**.** *Let v be a family with $\mathrm{pa}_v \neq \varnothing$ and let s be a sentence with $\mathrm{atoms}(s) \subseteq \mathrm{ch}_v \cup \mathrm{pa}_v$. Write π for assignments of pa_v, $w_\pi = P(\mathrm{pa}_v{=}\pi)$ for the parents' marginal, and $T(\cdot \mid \pi) = P(\mathrm{ch}_v{=}\cdot \mid \mathrm{pa}_v{=}\pi)$ for the table.*

(i) *If s is Type-1, $\ell \le P(\varphi) \le u$, then $P(\varphi) = \sum_\pi w_\pi\, T(\varphi_\pi \mid \pi)$ where $T(\varphi_\pi \mid \pi) := \sum_{c\,:\,(c,\pi)\models\varphi} T(c \mid \pi)$. This is $P(\mathrm{pa}_v)$-invariant for all tables iff the map $\pi \mapsto T(\varphi_\pi \mid \pi)$ is constant for every table — which fails whenever $|\{\pi\}| \ge 2$ and φ is non-trivial. Hence no Type-1 sentence is realizable in a non-root family.*

Table 2: Sentence classification. Every sentence is family-realizable in exactly one family, so the instance is fully realizable.

sentence	form	attached to families	realizable in	why
s_0, s_5, s_6	Type-1 $P(x_3)$, $P(\neg x_3)$	all five	$x_3 \mid \varnothing$	(R1) root block
s_1	Type-2, $\psi = x_3$	x_4, x_2, x_1, x_0	$x_4 \mid x_3$	(R2) $\lvert\Pi_\psi\rvert$=1
s_2	Type-2, $\psi = \neg x_3 \wedge x_4$	x_2, x_1	$x_2 \mid x_3, x_4$	(R2) $\lvert\Pi_\psi\rvert$=1
s_3	Type-2, $\psi = x_3 \wedge x_4 \wedge x_2$	x_1	$x_1 \mid x_2, x_3, x_4$	(R2) $\lvert\Pi_\psi\rvert$=1
s_4	Type-2, $\psi = x_3 \wedge x_4 \wedge \neg x_1$	x_0	$x_0 \mid x_1, x_3, x_4$	(R2) $\lvert\Pi_\psi\rvert$=1

(ii) *If s is Type-2 with $\mathrm{atoms}(\psi) \subseteq \mathrm{pa}_v$ and $\Pi_\psi := \{\pi : \pi \models \psi\}$, then on $\{P(\psi) > 0\}$*

$$P(\varphi \mid \psi) = \sum_{\pi \in \Pi_\psi} \lambda_\pi \, T(\varphi_\pi \mid \pi), \qquad \lambda_\pi = \frac{w_\pi}{\sum_{\pi' \in \Pi_\psi} w_{\pi'}}, \tag{13}$$

a convex combination of table rows with weights λ determined by $P(\mathrm{pa}_v)$. If $|\Pi_\psi| = 1$ this is one row, hence invariant; if $|\Pi_\psi| \geq 2$ it is invariant for all tables iff the rows agree, so s is not *realizable.*

Proof. (i) Condition the Type-1 probability on the parent block. Since $\mathrm{atoms}(\varphi) \subseteq \mathrm{ch}_v \cup \mathrm{pa}_v$, the truth value of φ at a joint assignment is determined by $(\mathrm{ch}_v, \mathrm{pa}_v)$, so by Lemma 1 $P(\varphi) = \sum_\pi P(\mathrm{pa}_v{=}\pi)\, P(\varphi \mid \mathrm{pa}_v{=}\pi) = \sum_\pi w_\pi T(\varphi_\pi \mid \pi)$, which is the stated formula. This is an affine function of the weight vector w on the simplex, and an affine function on a simplex is constant iff all its coefficients coincide; hence invariance for *every* table forces $\pi \mapsto T(\varphi_\pi \mid \pi)$ to be constant. Choose a table with $T(\varphi_{\pi_1} \mid \pi_1) = 1$ and $T(\varphi_{\pi_2} \mid \pi_2) = 0$ for two distinct parent configurations $\pi_1 \neq \pi_2$ (possible whenever φ is satisfiable and refutable within ch_v, i.e. non-trivial); then the coefficients differ and invariance fails. So s is not realizable in v.

(ii) With $\mathrm{atoms}(\psi) \subseteq \mathrm{pa}_v$, the event ψ is a union of parent configurations, $\{\psi\} = \bigcup_{\pi \in \Pi_\psi} \{\mathrm{pa}_v{=}\pi\}$, a disjoint union. Hence $P(\psi) = \sum_{\pi \in \Pi_\psi} w_\pi$ and, by the computation in (i) restricted to Π_ψ, $P(\varphi \wedge \psi) = \sum_{\pi \in \Pi_\psi} w_\pi T(\varphi_\pi \mid \pi)$. Dividing gives equation 13, and λ is a probability vector supported on Π_ψ.

If $|\Pi_\psi| = 1$, say $\Pi_\psi = \{\pi^\star\}$, then $\lambda_{\pi^\star} = 1$ and $P(\varphi \mid \psi) = T(\varphi_{\pi^\star} \mid \pi^\star)$, a single row — invariant, and this is (R2). If $|\Pi_\psi| \geq 2$, then equation 13 is a non-degenerate convex combination: pick $\pi_1 \neq \pi_2$ in Π_ψ and a table with $T(\varphi_{\pi_1} \mid \pi_1) = 1$, $T(\varphi_{\pi_2} \mid \pi_2) = 0$. Taking w concentrated near π_1 gives $P(\varphi \mid \psi) \to 1$, and near π_2 gives $\to 0$, so the value genuinely depends on $P(\mathrm{pa}_v)$ and s is not realizable. Equivalently, $\partial P(\varphi \mid \psi)/\partial \lambda_\pi = T(\varphi_\pi \mid \pi)$ is not constant across Π_ψ. □

B.1 A FIVE-VARIABLE CHAIN-GRAPH EXAMPLE

We show next and example of family-realizability on a five-atom chain graph.

Example 3. *Consider the following family-realizable k-tree LCN with n=5 atoms k=3:*

$s_0 : 0.01 \leq P(x_3) \leq 0.399975$
$s_1 : 0.01 \leq P(x_4 \mid x_3) \leq 0.442867$
$s_2 : 0.638553 \leq P(x_2 \mid \neg x_3 \wedge x_4) \leq 1.0$
$s_3 : 0.01 \leq P(\neg x_1 \mid x_3 \wedge x_4 \wedge x_2) \leq 0.307066$
$s_4 : 0.673756 \leq P(\neg x_0 \mid x_3 \wedge x_4 \wedge \neg x_1) \leq 1.0$
$s_5 : 0.748845 \leq P(\neg x_3) \leq 0.954666$
$s_6 : 0.141404 \leq P(x_3) \leq 0.247704$

The chain graph has five singleton-block families and there is a single LMC assertion:

$$x_3 \mid \varnothing, \quad x_4 \mid x_3, \quad x_2 \mid x_3, x_4, \quad x_1 \mid x_2, x_3, x_4, \quad x_0 \mid x_1, x_3, x_4; \qquad (x_0 \perp\!\!\!\perp x_2 \mid x_1, x_3, x_4).$$

Table 2 classifies every sentence. Two points are worth drawing out. First, the attached *column (what the scope-subset test collects) is much larger than the* realizable-in *column: the root marginals s_0, s_5, s_6 are attached to all five families but realizable only in the root family x_3, which is exactly why Definition 7 quantifies existentially over families. Second, each of s_1, s_2, s_3, s_4 conditions on a* full *conjunction of signed literals over all of its family's parents, so $|\Pi_\psi| = 1$ and (R2) applies — this is what the random problem generator guarantees for every family-realizable problem class.*

Measured bounds. *The exact method with a global nonlinear solver (e.g., SCIP) certifies all ten solves (gap* 0*), and the new* `CJT` *reproduces every endpoint:*

engine	$P(x_0)$	$P(x_1)$	$P(x_2)$	$P(x_3)$	$P(x_4)$
`Exact` *(global)*	$[0,1]$	$[0,1]$	$[0,1]$	$[0.141404, 0.247704]$	$[0.001414, 0.921219]$
`CJT` *(SCIP)*	$[0,1]$	$[0,1]$	$[0,1]$	$[0.141404, 0.247704]$	$[0.001414, 0.921219]$

The vacuous $[0,1]$ *on* x_0, x_1, x_2 *is genuine, not a solver artifact: CJT exhibits explicit feasible witnesses attaining both endpoints (e.g. setting* $P(x_2 \mid \neg x_3 \wedge x_4) = 0$ *makes* s_3*'s conditioning event unreachable, freeing* x_1*). The junction tree has maximal cliques* $\{x_0, x_1, x_3, x_4\}$ *and* $\{x_1, x_2, x_3, x_4\}$ *— treewidth* 3 *— and the separator between them is exactly* $\{x_1, x_3, x_4\}$*, the conditioning set of the lone LMC assertion; this is the certificate that hypotheses H1–H4 of Theorem 6 hold.*

C THE LCN SET VERSUS THE STRONG EXTENSION

We now relate $\mathcal{D}_{\mathrm{LCN}}$ and $\mathcal{D}_{\mathrm{SE}}$.

Proposition 3 (Inclusion)**.** *Let* $\mathcal{L}$ *be a chain-graph LCN. Then* $\mathcal{D}_{\mathrm{LCN}} \subseteq \mathcal{D}_{\mathrm{SE}}$*. Consequently, for every query,* $[\underline{q}_{\mathrm{LCN}}, \overline{q}_{\mathrm{LCN}}] \subseteq [\underline{q}_{\mathrm{SE}}, \overline{q}_{\mathrm{SE}}]$*: any engine that is exact or outer w.r.t.* $\mathcal{D}_{\mathrm{SE}}$ *is automatically a valid* outer *bound w.r.t.* $\mathcal{D}_{\mathrm{LCN}}$*.*

Proof. Let $p \in \mathcal{D}_{\mathrm{LCN}}$. Since $\mathcal{L}$ is a chain graph and p satisfies every LMC equality, the chain-graph factorization theorem Cozman (2023) gives the factorization equation 11: $p = \prod_v P_p(\mathrm{ch}_v \mid \mathrm{pa}_v)$, where $P_p(\cdot \mid \cdot)$ denotes the conditionals induced by p (defined arbitrarily on parent configurations of zero probability, which do not affect the product).

Fix a family v and a parent configuration π with $P_p(\mathrm{pa}_v{=}\pi) > 0$. The row $P_p(\mathrm{ch}_v \mid \mathrm{pa}_v{=}\pi)$ is a conditional table attainable by the joint p, and p satisfies every sentence of $\mathcal{L}$ — in particular every sentence whose atom scope lies inside $\mathrm{ch}_v \cup \mathrm{pa}_v$. But $K_{v,\pi}$ was defined as the set of rows attainable by *some* joint satisfying exactly those in-scope sentences. Hence $P_p(\mathrm{ch}_v \mid \mathrm{pa}_v{=}\pi) \in K_{v,\pi}$: the local set is a *relaxation* of the projection, so it contains it. For a configuration with $P_p(\mathrm{pa}_v{=}\pi) = 0$ the row is unconstrained by p and may be chosen to be any element of $K_{v,\pi}$ (non-empty, since p itself witnesses feasibility of the in-scope sentences) without changing the product.

Thus p is exhibited as a product $\prod_v \prod_\pi$ of rows drawn from the respective $K_{v,\pi}$, which is precisely membership in $\mathcal{D}_{\mathrm{SE}}$ as defined in equation 12. The bound inclusion follows because minimizing and maximizing a fixed functional over a larger set yields a wider interval. □

Proposition 4 (Sufficiency)**.** *Let* $\mathcal{L}$ *be a chain-graph LCN with local sets built by chain-graph factorization. Suppose*

(A) no Gap A: *for every family* v *and parent configuration* π*,* $K_{v,\pi} = K^{\mathrm{true}}_{v,\pi}$*, the projection of* $\mathcal{D}_{\mathrm{LCN}}$ *onto that row; and*

(B) full realizability: *every sentence of* $\mathcal{L}$ *is family-realizable in some family (Definition 7).*

Then $\mathcal{D}_{\mathrm{SE}} = \mathcal{D}_{\mathrm{LCN}}$. Single-connectedness of the primal graph is not required.

Proof. Proposition 3 gives $\mathcal{D}_{\mathrm{LCN}} \subseteq \mathcal{D}_{\mathrm{SE}}$, so only $\mathcal{D}_{\mathrm{SE}} \subseteq \mathcal{D}_{\mathrm{LCN}}$ needs proof. Let $q = \prod_v \prod_\pi P_{v,\pi}$ with each $P_{v,\pi} \in K_{v,\pi}$ be an arbitrary element of $\mathcal{D}_{\mathrm{SE}}$. We verify that q satisfies both families of constraints in equation 10.

LMC equalities. By construction q is a product of the form equation 11 over the families of the chain graph. By the converse direction of the chain-graph factorization theorem (Lauritzen, Thm. 3.34: factorization implies the global Markov property of the chain graph, and this direction needs no positivity assumption), q satisfies every conditional independence entailed by the chain graph — in particular every LMC assertion, since the LMC is by definition read off that same graph. Hence $h_\iota(q) = 0$ for all ι. Note this step is where the "loopiness contributes nothing" claim comes from: a product over the chain-graph families satisfies the LMC whether or not the graph is singly connected.

Sentences. Let s be a sentence. By (B) it is family-realizable in some family v, so by Definition 7 the value $g_s(q)$ is determined by the conditional table of q at v alone. That table is the collection of rows $\{P_{v,\pi}\}_\pi$, and by (A) each $P_{v,\pi} \in K^{\text{true}}_{v,\pi}$, i.e. each row is realized by *some* $p^{(\pi)} \in \mathcal{D}_{\text{LCN}}$. It remains to see that this forces $g_s(q) \in [\ell_s, u_s]$. Two cases, matching the two clauses of Definition 7.

If s has form (R2), then $g_s(q) = P_{v,\pi^\star}(\varphi_{\pi^\star})$ is the *single* row $\pi^\star$. That row equals the corresponding row of some $p^{(\pi^\star)} \in \mathcal{D}_{\text{LCN}}$; by Definition 7 again (applied to $p^{(\pi^\star)}$, whose table has this same row) $g_s(p^{(\pi^\star)}) = g_s(q)$, and $p^{(\pi^\star)} \in \mathcal{D}_{\text{LCN}}$ satisfies s. Hence $g_s(q) \in [\ell_s, u_s]$.

If s has form (R1), then $\text{pa}_v = \varnothing$, so the family has the single (empty) parent configuration and its table is the whole marginal $P(\text{ch}_v) = P_{v,\varnothing}$. The same argument applies verbatim with $\pi^\star$ the empty configuration.

In both cases the key point is that realizability makes g_s a function of *one row*, and a row certified by (A) to be realized inside $\mathcal{D}_{\text{LCN}}$ carries its constraint value with it — so the free, independent choice of the *other* rows and families cannot disturb it. Therefore every sentence holds at q, giving $q \in \mathcal{D}_{\text{LCN}}$. □

We now give an example of a loopy LCN on which CVE is exact.

Example 4. *Single-connectedness is not necessary for exactness. Consider the following family-realizable k-tree LCN with n=2 atoms and k=3:*

$$s_0\text{:}\ 0.084382 \le P(x_1) \le 0.684382 \qquad s_2\text{:}\ 0.092785 \le P(x_3 \mid x_1 \wedge x_2) \le 0.692785$$
$$s_1\text{:}\ 0.01 \le P(\neg x_2 \mid \neg x_1) \le 0.572656 \quad s_3\text{:}\ 0.068242 \le P(x_0 \mid x_1 \wedge x_3) \le 0.668242$$

*The seed 3-clique is $\{x_1, x_2, x_3\}$ and x_0 attaches to the 2-clique $\{x_1, x_3\}$, so the moralized graph contains the triangle $x_1x_2x_3$: the instance is **loopy**, decidedly not singly connected. It is also fully realizable (each ψ is a full conjunction over its family's parents; s_0 is a root marginal). Measured, with all eight SCIP solves* `confirmed` *at gap 0:*

engine	$P(x_0)$	$P(x_1)$	$P(x_2)$	$P(x_3)$
`Exact` *(global)*	$[0, 1]$	$[0.0844, 0.6844]$	$[0.1349, 0.9968]$	$[0, 1]$
`CJT` *(SCIP)*	$[0, 1]$	$[0.0844, 0.6844]$	$[0.1349, 0.9968]$	$[0, 1]$
CVE	$[0, 1]$	$[0.0844, 0.6844]$	$[0.1349, 0.9968]$	$[0, 1]$

CVE is exact on every atom *of a loopy LCN.*

Remark 2 (What loopiness *does* cost)**.** *Loopiness is not free — but its price is paid in* cost*, not accuracy. On a five-atom k-tree LCN of Example 3, the credal network builds essentially instantly and is tiny ($8 \cdot 8 \cdot 4 \cdot 1 \cdot 2 = 512$ vertex combinations across the five families), yet CVE did not terminate within 200 s: it stalls inside the pruning stage during elimination.* `CJT` *solves the same instance in about a second. So on bounded-treewidth loopy classes the practical recommendation — prefer* `CJT` *— stands, but the reason is tractability, not looseness.*

D Detailed Description of Credal Variable Elimination

We present next three inference algorithms that leverage the chain graph factorization of an LCN to compute marginal probability bounds. The first, Credal Variable Elimination (CVE), extends classical variable elimination Dechter (1996) to credal networks by propagating credal potentials, represented as finite sets of non-negative functions Mauá & Cozman (2020).

D.1 Credal Networks

A **credal network** is a pair $\langle G, \mathcal{K} \rangle$ where G is a directed acyclic graph (DAG) over variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{D_1, \ldots, D_n\}$, and $\mathcal{K}$ is a collection of conditional credal sets $K(X_i \mid \text{Pa}(X_i) = \pi)$, one for each variable X_i and each configuration π of its parents $\text{Pa}(X_i)$ in G.

Each credal set $K(X_i \mid \pi)$ is a closed convex set of probability distributions over X_i, specified by its *extreme points* (vertices):

$$\operatorname{ext} K(X_i \mid \pi) = \{P_1(X_i \mid \pi), \ldots, P_v(X_i \mid \pi)\}.$$

D.2 Strong Extension

The **strong extension** of a separately specified credal network is the convex hull of all joint distributions formed by combining one extreme point per local credal set:

$$D_{\text{SE}}(\mathbf{X}) = \text{CH}\Big\{P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i \mid \pi_i) : P(X_i \mid \pi_i) \in \operatorname{ext} K(X_i \mid \pi_i)\Big\}. \tag{14}$$

Theorem 7 (Mauá & Cozman, 2020)**.** *Any combination of extreme points from the local credal sets is an extreme point of the strong extension. Thus the strong extension can be built directly from the local extreme points.*

D.3 Marginal Inference

Given a query variable Z and evidence $\mathbf{Y} = \mathbf{y}$, the goal is to compute:

$$\underline{P}(Z = z \mid \mathbf{Y} = \mathbf{y}) = \min_{P \in \operatorname{ext} K(\mathbf{X})} \frac{\sum_{\mathbf{x}'} \prod_i P(x_i \mid \pi_i)}{\sum_{z'} \sum_{\mathbf{x}'} \prod_i P(x_i \mid \pi_i)}, \tag{15}$$

where the sums in numerator and denominator are over the marginalized variables $\mathbf{X}' = \mathbf{X} \setminus (\{Z\} \cup \mathbf{Y})$, and similarly for the upper probability $\overline{P}$ with $\max$.

D.4 Potentials

Definition 8 (Potential)**.** *A **potential** $\phi(\mathbf{Y})$ over variables $\mathbf{Y}$ is a finite set of non-negative real-valued functions $\{f_1, f_2, \ldots, f_m\}$ on the joint domain of $\mathbf{Y}$. Each function f_k maps configurations of $\mathbf{Y}$ to $\mathbb{R}_{\geq 0}$.*

Three operations on potentials are central to the algorithm:

Product. The product of $\phi(\mathbf{Y})$ and $\psi(\mathbf{Z})$ is:

$$\phi \cdot \psi = \{f \cdot g : f \in \phi,\ g \in \psi\},$$

where $(f \cdot g)(\mathbf{y}, \mathbf{z}) = f(\mathbf{y}) \cdot g(\mathbf{z})$ over the joint domain $\mathbf{Y} \cup \mathbf{Z}$ (with broadcasting for non-shared variables).

Sum-Marginal. The sum-marginal of $\phi(\mathbf{Y})$ with respect to $X \subseteq \mathbf{Y}$ is:

$$\sum_X \phi(\mathbf{Y}) = \Big\{\sum_x f(\mathbf{y}) : f \in \phi\Big\},$$

producing functions over $\mathbf{Y} \setminus \{X\}$.

Pruning. The pruning operation removes dominated functions. We say g *dominates* f when g is pointwise at least f and strictly greater somewhere:

Definition 9 (Dominance and pruning)**.**

$$\operatorname{prune}(\phi) = \big\{f \in \phi : \nexists\, g \in \phi \setminus \{f\} \text{ s.t. } g(\mathbf{y}) \geq f(\mathbf{y})\ \forall \mathbf{y} \text{ and } g(\mathbf{y}') > f(\mathbf{y}') \text{ for some } \mathbf{y}'\big\}.$$

Because a dominated f can only ever attain the min (resp. a dominating g the max) of a quantity that is monotone in the function values, discarding f reduces the cardinality of the potential without affecting the min/max bounds.

Algorithm 1 Credal Variable Elimination (CVE)

Require: Extreme points $\operatorname{ext}\mathcal{K}(X_i \mid \operatorname{pa}(X_i))$ for each variable X_i; query variable Z; evidence $\mathbf{Y} = \mathbf{y}$
Ensure: Lower and upper bounds $[\underline{P}(Z{=}z \mid \mathbf{y}),\ \overline{P}(Z{=}z \mid \mathbf{y})]$ for each state z of Z

1: $\Gamma \leftarrow \emptyset$ ▷ the set of potentials
2: **for** each variable $X_i \in \mathbf{X}$ **do**
3: $\quad\phi_{X_i} \leftarrow \{\text{one chosen } P(X_i \mid \pi) \in \operatorname{ext}\mathcal{K}(X_i \mid \pi) \text{ per parent config } \pi\}$
4: $\quad\Gamma \leftarrow \Gamma \cup \{\phi_{X_i}\}$ ▷ $|\phi_{X_i}| = \prod_\pi |\operatorname{ext}\mathcal{K}(X_i \mid \pi)|$ CPT vertices
5: **for** each evidence variable $Y_j \in \mathbf{Y}$ **do**
6: $\quad\Gamma \leftarrow \Gamma \cup \{\{\delta_{y_j}\}\}$ ▷ indicator potential; the CPTs are not sliced

7: $\mathbf{o} \leftarrow (X_{\sigma(1)}, \ldots, X_{\sigma(n-1)})$, an elimination ordering over $\mathbf{X} \setminus \{Z\}$ ▷ *excludes* Z
8: **for** each X_i in ordering $\mathbf{o}$ **do**
9: $\quad\Gamma_{X_i} \leftarrow \{\phi \in \Gamma : X_i \in \operatorname{scope}(\phi)\}$
10: $\quad\Gamma \leftarrow \Gamma \setminus \Gamma_{X_i}$
11: $\quad$**if** $\Gamma_{X_i} = \emptyset$ **then**
12: $\quad\quad$**continue**
13: $\quad\psi \leftarrow \prod\{\phi \in \Gamma_{X_i}\}$ ▷ combine all potentials in the bucket
14: $\quad\psi' \leftarrow \sum_{X_i} \psi$ ▷ marginalize out X_i; cardinality unchanged
15: $\quad\psi'' \leftarrow \operatorname{prune}(\psi')$ ▷ remove dominated functions (Def. 9)
16: $\quad\Gamma \leftarrow \Gamma \cup \{\psi''\}$

17: $\phi_{\text{final}} \leftarrow \prod\{\phi \in \Gamma\}$ ▷ remaining potentials, scope $\{Z\}$
18: $\underline{P}(z) \leftarrow 1,\ \overline{P}(z) \leftarrow 0$ for each state z
19: **for** each $f \in \phi_{\text{final}}$ with $\sum_{z'} f(z') > 0$ **do**
20: $\quad\mathbf{p} \leftarrow f / \sum_{z'} f(z')$ ▷ normalize *this* vertex
21: $\quad$**for** each state z **do**
22: $\quad\quad\underline{P}(z) \leftarrow \min(\underline{P}(z),\ \mathbf{p}(z))$
23: $\quad\quad\overline{P}(z) \leftarrow \max(\overline{P}(z),\ \mathbf{p}(z))$
24: **return** $[\underline{P}(z),\ \overline{P}(z)]$ for each state z

D.5 Algorithm

CVE answers a query on one variable Z by a single bucket-elimination pass in which Z is eliminated *last*. Algorithm 1 states the pass in full; the paragraphs that follow it explain the three steps that carry the credal content.

One structural fact must be stated first, because it shapes the input and the output. The variables of the credal network are *chain-graph node families*, not the individual atoms of the underlying LCN: a variable X carries a set of atoms $\operatorname{atoms}(X)$ and has domain size $2^{|\operatorname{atoms}(X)|}$ (the truth table over its atoms, most-significant bit first). A variable with a single atom is called *singleton*; one with several is *compound*. The pass therefore returns bounds on the *states* of Z; when Z is singleton its state-1 bound is the atom marginal directly, and when Z is compound a further projection is needed to report per-atom marginals (Section D.7).

Initial potentials are explicit vertex enumerations. Line 3 is where the credal information enters. A family X_i with parent configurations $\pi_1, \ldots, \pi_r$ contributes one function per element of $\operatorname{ext}\mathcal{K}(X_i \mid \pi_1) \times \cdots \times \operatorname{ext}\mathcal{K}(X_i \mid \pi_r)$: each function is a complete CPT obtained by choosing one extreme point in every column, so $|\phi_{X_i}|$ is the *product* of the per-column vertex counts. The potential is therefore an explicit enumeration of that family's contribution to the strong extension, which is what makes CVE a strong-extension method.

The query is eliminated last. The ordering on line 9 omits Z entirely, so no step ever marginalizes the query and its bucket survives to the end. This is the decisive scheduling choice: it is what makes the answer exact with respect to the strong extension, and it is exactly what a single shared two-pass

ordering (as in CCTE) gives up. Answering all marginals means repeating the pass once per target, each with its own ordering, since nothing is shared between targets.

Normalization *is* the evidence conditioning. There is no separate division by $P(\mathbf{y})$. With the indicator potentials of line 7 in the product, a surviving function of ϕ_{final} satisfies $f(z) = P(Z{=}z, \mathbf{y})$ for one selection of extreme points, so $f(z)/\sum_{z'} f(z') = P(Z{=}z \mid \mathbf{y})$ for that same selection. Because each vertex is normalized *before* the min/max is taken, the result is the correct set of per-vertex conditionals — not the invalid ratio $\min P(Z, \mathbf{y})/\max P(\mathbf{y})$ of a separately minimized numerator and maximized denominator. Vertices of zero mass are skipped.

D.6 Elimination Ordering

Two heuristics are supported; both exclude the query Z, which is what keeps Z's bucket alive.

- **Topological order**: the DAG's *forward* topological order, restricted to the non-query, non-evidence variables, with the evidence variables appended at the end. Evidence variables are still eliminated — they are pinned by their indicator potentials rather than removed from the problem.
- **Min-fill**: the greedy interaction-graph heuristic, eliminating at each step the variable that adds the fewest fill-in edges, with an alphabetical tiebreak, and with Z held out of the ordering (it still counts as a neighbour when fill is scored). This aims at a small induced width w^*.

D.7 Projecting a Compound Query onto its Atoms

When Z is singleton, its atom marginal is the state-1 entry of the bounds returned by Algorithm 1. When Z is compound, with m atoms and $k = 2^m$ states, each atom a is recovered from the state bounds $[\underline{P}(z), \overline{P}(z)]$ by two linear programs over the states S_a in which a is true:

$$\underline{P}(a) = \min\Big\{ \sum_{z \in S_a} p_z \;:\; \sum_z p_z = 1,\;\; \underline{P}(z) \le p_z \le \overline{P}(z) \Big\},$$

and symmetrically $\overline{P}(a)$ with $\max$. The feasible region is the box $\prod_z [\underline{P}(z), \overline{P}(z)]$ intersected with the probability simplex, which is an *outer relaxation* of the true state polytope of Z. Atom bounds on a compound query can therefore be looser than the state bounds CVE actually computed; the running example in Section D.8 shows this at its most extreme.

D.8 Running Example

Example 5 (Alarm Network)**.** *The LCN* `examples/alarm.lcn` *over atoms A, B, C, D, E factorizes into four credal-network nodes: the singletons B (Burglary), E (Earthquake), A (Alarm), and the* compound *node C-D carrying the two atoms C, D (they are merged because the sentence $0.5 \le P(\neg(C \oplus D) \mid A) \le 0.7$ constrains them jointly). The compound node has $2^2 = 4$ states, ordered $CD \in \{00, 01, 10, 11\}$ most-significant bit first. The DAG is*

$$B \to A, \quad E \to A, \quad A \to C\text{-}D.$$

The local credal sets (extreme points from LRS vertex enumeration) are:

Node	***Parent Config***	***#***	***Vertices***
B	—	*2*	$[0.8, 0.2]$, $[0.9, 0.1]$
E	—	*2*	$[0.9, 0.1]$, $[0.95, 0.05]$
A	$B{=}0, E{=}0$	*2*	$[0.956, 0.044]$, $[1.0, 0.0]$
A	$B{=}1, E{=}0$	*2*	$[0.0, 1.0]$, $[1.0, 0.0]$
A	$B{=}0, E{=}1$	*2*	$[0.0, 1.0]$, $[1.0, 0.0]$
A	$B{=}1, E{=}1$	*2*	$[0.0, 1.0]$, $[1.0, 0.0]$
C-D	$A{=}0$	*4*	$[0, 0, 0, 1]$, $[0, 0, 1, 0]$, $[0, 1, 0, 0]$, $[1, 0, 0, 0]$
C-D	$A{=}1$	*11*	$[0, 0, 0.3, 0.7]$, $[0, 0.3, 0, 0.7]$, …, $[0.7, 0, 0, 0.3]$

Initial potentials. *Line 3 of Algorithm 1 takes one extreme point per parent configuration, so the potential sizes are the* products *of the counts above:*

$$|\phi_B| = 2, \quad |\phi_E| = 2, \quad |\phi_A| = 2^4 = 16, \quad |\phi_{C\text{-}D}| = 4 \times 11 = 44,$$

for $2 + 2 + 16 + 44 = 64$ *functions in total across the four potentials.*

Query: $P(B{=}1)$ **(no evidence).** *The forward topological order with the target removed is* $\mathbf{o} = (E, A, C\text{-}D)$*; note* B *never appears in it.*

Step 1: eliminate E*.* *The bucket is* ϕ_E *(scope* $\{E\}$*, 2 functions) and* ϕ_A *(scope* $\{A, B, E\}$*, 16 functions). Their product has* $2 \times 16 = 32$ *functions over* $\{A, B, E\}$*; summing out* E *leaves 32 functions over* $\{A, B\}$*, and pruning removes none of them.*

Step 2: eliminate A*.* *The bucket is the new potential over* $\{A, B\}$ *(32 functions) together with* $\phi_{C\text{-}D}$ *(scope* $\{C\text{-}D, A\}$*, 44 functions), giving* $32 \times 44 = 1408$ *functions over* $\{C\text{-}D, A, B\}$*. Summing out* A *leaves 1408 functions over* $\{C\text{-}D, B\}$*; again none are dominated.*

Step 3: eliminate C**-**D*.* *Summing out* C*-*D *leaves 1408 functions over* $\{B\}$*, and here pruning bites: only* 328 *survive.*

Result. *Each surviving* f *over* $\{B\}$ *is normalized,* $p(B{=}1) = f(1)/(f(0) + f(1))$*, and the componentwise min/max over the* 328 *normalized vectors gives*

$$\underline{P}(B{=}1) = 0.10, \qquad \overline{P}(B{=}1) = 0.20,$$

recovering exactly the interval asserted by the sentence $0.1 \le P(B) \le 0.2$*, as it must since* B *is a root.*

The compound node illustrates the projection gap. *Running the driver over all four nodes returns* $P(A{=}1) \in [0, 0.312]$ *and* $P(E{=}1) \in [0.05, 0.10]$*, but the state bounds for* C*-*D *come out vacuous,* $[\underline{p}, \overline{p}] = [\mathbf{0}, \mathbf{1}]$ *on all four states. The per-atom linear programs of Section D.7 can then only return* $\overline{P}(C{=}1), P(D{=}1) \in [0, 1]$*: with every state interval equal to* $[0, 1]$*, the box–simplex relaxation carries no information, even though the joint constraint on* $C \oplus D$ *does. This is the outer relaxation of the compound-node projection at its worst.*

D.9 Extracting Singleton Marginals from Compound Variables

LCN compiles groups of correlated atoms into **compound variables** whose states are joint assignments to the constituent atoms. The CVE algorithm described above produces bounds $[\underline{P}(X_i{=}s), \overline{P}(X_i{=}s)]$ for each state s of every compound variable X_i, but downstream users typically want bounds on the individual atoms. Because the lower and upper bound of each compound state can come from *different* extreme distributions (different functions surviving the pruning step), one cannot recover singleton bounds by simply summing compound bounds componentwise — that would be valid but loose. Instead, for each singleton atom we solve a small linear program over the compound marginal polytope.

The compound variable CD**.** In the alarm network, C and D are merged into a single compound variable CD with $k = 2^2 = 4$ states, using big-endian bit encoding:

$$s{=}0 \equiv (C{=}0, D{=}0), \quad s{=}1 \equiv (C{=}0, D{=}1), \quad s{=}2 \equiv (C{=}1, D{=}0), \quad s{=}3 \equiv (C{=}1, D{=}1).$$

The CVE run produces bounds $[\underline{P}(CD{=}s), \overline{P}(CD{=}s)]$ for each $s \in \{0, 1, 2, 3\}$.

Linear program for $P(C{=}1)$**.** The set of compound states where $C{=}1$ is $T_C = \{2, 3\}$. The lower bound $\underline{P}(C{=}1)$ is the optimum of the linear program

$$\begin{aligned} \underline{P}(C{=}1) \;=\; \min_{\mathbf{p} \in \mathbb{R}^4} \quad & \sum_{s \in T_C} p_s \;=\; p_2 + p_3 \\ \text{subject to} \quad & \sum_{s=0}^{3} p_s = 1, \\ & \underline{P}(CD{=}s) \;\le\; p_s \;\le\; \overline{P}(CD{=}s), \quad s = 0, 1, 2, 3, \\ & p_s \ge 0, \quad s = 0, 1, 2, 3. \end{aligned}$$

The upper bound $\overline{P}(C{=}1)$ is obtained by replacing $\min$ with $\max$ (same feasible region, same objective). The complementary bounds for $C{=}0$ follow from probabilistic duality:

$$\underline{P}(C{=}0) = 1 - \overline{P}(C{=}1), \qquad \overline{P}(C{=}0) = 1 - \underline{P}(C{=}1).$$

Linear program for $P(D{=}1)$. For D, the set of compound states where $D{=}1$ is $T_D = \{1, 3\}$. Two more LPs with the *same* feasible region but objective $\sum_{s \in T_D} p_s = p_1 + p_3$ yield $\underline{P}(D{=}1)$ and $\overline{P}(D{=}1)$; the bounds for $D{=}0$ follow by duality.

General case. For a compound variable with atoms $A_1, \ldots, A_m$ there are $k = 2^m$ states. For each un-observed atom A_j (atoms already present in the evidence are skipped), the set of states where $A_j{=}1$ is

$$T_{A_j} \;=\; \big\{\, s \in \{0, \ldots, k-1\} \;:\; \text{bit } (m{-}1{-}j) \text{ of } s \text{ equals } 1 \,\big\},$$

using big-endian encoding. Two LPs per atom (minimize and maximize $\sum_{s \in T_{A_j}} p_s$ over the compound marginal polytope) give $[\underline{P}(A_j{=}1), \overline{P}(A_j{=}1)]$.

D.10 Proofs of Theorem 1 and 2

The complexity bound of Theorem 1 follows immediately from Mauá & Cozman (2020).

First, CVE is known to be exact on $\mathcal{D}_{\mathrm{SE}}$ Mauá & Cozman (2020). Given that $\mathcal{L}$ is family-realizable there are no cross-family constraints and by Proposition 2 we have that $\mathcal{D}_{\mathrm{SE}} = \mathcal{D}_{\mathrm{LCN}}$. It follows easily that CVE computes exact marginals with respect to $\mathcal{D}_{\mathrm{LCN}}$.

E Detailed Description: Interval Belief Propagation

E.1 Factor Graph

Given a credal network $\langle G, \mathcal{K} \rangle$, we construct a **factor graph**: a bipartite graph with variable nodes and factor nodes. Each variable X_i in the credal network corresponds to a variable node with cardinality $|D_i|$. Each conditional credal set $K(X_i \mid \mathrm{Pa}(X_i))$ defines a factor node f_i whose scope is $\{X_i\} \cup \mathrm{Pa}(X_i)$. The factor stores the extreme points of $K(X_i \mid \pi)$ for all parent configurations π.

E.2 Interval Messages

Definition 10 (Interval Message)**.** *For a variable X with k states, an **interval message** is a pair of vectors $(\mathbf{l}, \mathbf{u}) \in \mathbb{R}^k \times \mathbb{R}^k$ where $0 \le l_x \le u_x \le 1$ for each state $x \in D_X$. The interval $[l_x, u_x]$ bounds the probability $P(X = x)$ from the message sender's perspective.*

Messages are of two types:

- $\mu_{v \to f} = (\mathbf{l}_{v \to f}, \mathbf{u}_{v \to f})$: message from variable v to factor f.
- $\mu_{f \to v} = (\mathbf{l}_{f \to v}, \mathbf{u}_{f \to v})$: message from factor f to variable v.

All messages are initialized to the vacuous interval $([0, \ldots, 0], [1, \ldots, 1])$.

E.3 Evidence

Evidence is a (possibly empty) set of observed variable assignments

$$\mathbf{Y} = \mathbf{y} = \{\, Y_1 = y_1,\; Y_2 = y_2,\; \ldots,\; Y_m = y_m \,\}.$$

For each evidence variable $Y_j = y_j$, the outgoing variable-to-factor message is clamped to the one-hot indicator $\delta_{y_j} \in \{0, 1\}^{|D_{Y_j}|}$ (entry y_j equals 1, all others equal 0):

$$\mu_{Y_j \to f} \;\leftarrow\; (\delta_{y_j},\, \delta_{y_j}) \qquad \text{for every factor } f \in \mathcal{N}(Y_j).$$

The interval collapses to a single point, pinning Y_j to y_j. Two consequences for the main loop:

- **Clamped messages are held fixed.** During the variable-to-factor update phase, evidence variables are *skipped* — their outgoing messages are never recomputed.
- **Factor-to-variable updates use the clamped input.** When a factor f computes an outgoing message, any incoming message from an evidence variable Y_j has $\mathbf{l}_{Y_j \to f} = \mathbf{u}_{Y_j \to f} = \delta_{y_j}$. The corner-point enumeration of Section E.4.2 then contributes no degrees of freedom for Y_j: both corners select the same vector $\mathbf{q}_{Y_j} = \delta_{y_j}$, and the marginal sum in Equation 16 is restricted to assignments consistent with $Y_j = y_j$.

After convergence, the extracted marginal bounds at every non-evidence variable X_i are bounds on the *conditional* marginal $P(X_i \mid \mathbf{Y} = \mathbf{y})$.

For compound variables that bundle multiple atoms, if an atom is itself an evidence variable, the singleton-marginal extraction LP (described in the companion CCTE document) skips that atom, since its bound is trivially $[\mathbb{1}[y_j{=}1],\ \mathbb{1}[y_j{=}1]]$.

E.4 Factor-to-Variable Message Computation

The factor-to-variable message from factor f_i (associated with node X_i, parents $\mathrm{Pa}(X_i)$) to target variable $v \in \mathrm{scope}(f_i)$ is computed by scanning over all *vertex combinations* and *corner-point combinations*, as defined below.

Let $\mathcal{O} = \mathrm{scope}(f_i) \setminus \{v\}$ be the other variables. For each $w \in \mathcal{O}$, let $(\mathbf{l}_w, \mathbf{u}_w) = \mu_{w \to f_i}$ be the incoming message.

E.4.1 Vertex Combinations (VC)

Factor f_i stores a conditional credal set $K(X_i \mid \pi)$ for each parent configuration $\pi \in \Omega_{\mathrm{Pa}(X_i)}$. Each credal set has a finite number of extreme points (vertices):

$$\mathrm{ext}\, K(X_i \mid \pi) = \{P_1(X_i \mid \pi),\ P_2(X_i \mid \pi),\ \ldots,\ P_{n_\pi}(X_i \mid \pi)\}.$$

Definition 11 (Vertex Combination)**.** *A **vertex combination** $\mathbf{vc} = (j_1, j_2, \ldots, j_m)$ selects one vertex per parent configuration: vertex j_k from configuration π_k. It specifies a complete conditional distribution $P_{\mathbf{vc}}(X_i \mid \pi_k)$ for every parent configuration.*

The total number of vertex combinations is $\prod_{k=1}^{m} n_{\pi_k}$.

Example. If factor f_A has parents $\{B, E\}$ with 4 parent configurations $(B{=}0, E{=}0)$, $(B{=}1, E{=}0)$, $(B{=}0, E{=}1)$, $(B{=}1, E{=}1)$, and each has 2 vertices, then there are $2^4 = 16$ vertex combinations. The combination $\mathbf{vc} = (1, 1, 1, 1)$ selects vertex 1 from every parent configuration, while $\mathbf{vc} = (2, 1, 2, 1)$ selects vertex 2 from $(B{=}0, E{=}0)$ and $(B{=}0, E{=}1)$, and vertex 1 from the other two.

E.4.2 Corner-Point Combinations (CC)

For each "other" variable $w \in \mathcal{O}$, the incoming interval message provides a lower bound vector $\mathbf{l}_w$ and an upper bound vector $\mathbf{u}_w$. To compute the factor-to-variable message, we must evaluate the marginal at the extreme points of these intervals.

Definition 12 (Corner-Point Combination)**.** *A **corner-point combination** $\mathbf{cc} = (c_1, c_2, \ldots, c_{|\mathcal{O}|})$ where each $c_j \in \{0, 1\}$ selects either the* lower *($c_j = 0$) or* upper *($c_j = 1$) bound vector for each other variable w_j:*

$$\mathbf{q}_{w_j} = \begin{cases} \mathbf{l}_{w_j} & \text{if } c_j = 0 \\ \mathbf{u}_{w_j} & \text{if } c_j = 1 \end{cases}$$

The total number of corner-point combinations is $2^{|\mathcal{O}|}$.

Example. When computing $f_A \to B$ (target is B), the other variables are $\mathcal{O} = \{A, E\}$. If the incoming messages are $\mu_{A \to f_A} = ([0.7, 0.0], [1.0, 0.3])$ and $\mu_{E \to f_A} = ([0.9, 0.05], [0.95, 0.1])$, then there are $2^2 = 4$ corner-point combinations selecting $\mathbf{q}_A \in \{[0.7, 0.0], [1.0, 0.3]\}$ and $\mathbf{q}_E \in \{[0.9, 0.05], [0.95, 0.1]\}$.

Algorithm 2 Interval Belief Propagation (IBP)

Require: LCN $\mathcal{L}$ with variables $\{X_1, \ldots, X_n\}$, evidence $\mathbf{E} = \mathbf{e}$, iterations T, threshold τ
Ensure: Bounds $[\underline{P}(Y_i{=}y|\mathbf{e}),\ \overline{P}(Y_i{=}y|\mathbf{e})]$ for non-evidence variables Y_i and states y

Build credal network $\mathcal{C}$ using chain graph factorization of $\mathcal{L}$
Build factor graph and initialize factors from extreme points of each $\mathbb{K}(Y_j \mid \mathbf{pa}(Y_j))$
Set $\mu_{v\to f} \leftarrow (\mathbf{0}, \mathbf{1})$ and $\mu_{f\to v} \leftarrow (\mathbf{0}, \mathbf{1})$
for each evidence variable $E_i = e_i$ **do**
  Set $\mu_{E_i\to f} \leftarrow (\delta_{e_i}, \delta_{e_i})\ \forall f \in \mathcal{N}(Y_j)$
for iteration $t = 1, \ldots, T$ **do**
  $\delta \leftarrow 0$
    ▷ VARIABLE-TO-FACTOR UPDATES
  for each v ($\notin \mathbf{E}$), each $f \in \mathcal{N}(v)$ **do**
    for each value x **do**
      $l_{v\to f}(x) \leftarrow \max_{f'\in\mathcal{N}(v)\setminus\{f\}} l_{f'\to v}(x)$
      $u_{v\to f}(x) \leftarrow \min_{f'\in\mathcal{N}(v)\setminus\{f\}} u_{f'\to v}(x)$
    Update δ
    ▷ FACTOR-TO-VARIABLE UPDATES
  for each f_i, each variable $v \in \mathrm{scope}(f_i)$ **do**
    $\mathbf{l}_{\text{new}} \leftarrow \mathbf{1}, \quad \mathbf{u}_{\text{new}} \leftarrow \mathbf{0}$
    for each vertex combination **vc** **do**
      for each corner-point combination **cc** **do**
        Compute $\mathbf{m}(v)$ using **vc** and **cc**
        $\mathbf{p} \leftarrow \mathbf{m}/\sum_x m(x)$
        $\mathbf{l}_{\text{new}} \leftarrow \min(\mathbf{l}_{\text{new}}, \mathbf{p})$
        $\mathbf{u}_{\text{new}} \leftarrow \max(\mathbf{u}_{\text{new}}, \mathbf{p})$
    $\mu_{f\to v} \leftarrow (\mathbf{l}_{\text{new}}, \mathbf{u}_{\text{new}})$
    Update δ
  if $\delta < \tau$ **then break**
    ▷ EXTRACT MARGINAL BOUNDS
for each non evidence variable Y_i **do**
  for each value $y \in D_{Y_i}$ **do**
    $\underline{P}(Y_i{=}y \mid \mathbf{E}{=}\mathbf{e}) \leftarrow \max_{f\in\mathcal{N}(Y_i)} l_{f\to Y_i}(y)$
    $\overline{P}(Y_i{=}x \mid \mathbf{E}{=}\mathbf{e}) \leftarrow \min_{f\in\mathcal{N}(Y_i)} u_{f\to Y_i}(y)$
return $[\underline{P}(Y_i{=}y \mid \mathbf{E}{=}\mathbf{e}),\ \overline{P}(Y_i{=}y \mid \mathbf{E}{=}\mathbf{e})]$

E.4.3 MARGINAL COMPUTATION

For each vertex combination **vc**, define the induced nonnegative factor

$$\phi_{i,\mathbf{vc}}(x_i, \pi) = P_{\mathbf{vc}}(X_i{=}x_i \mid \pi),$$

viewed as a function on all variables in $\mathrm{scope}(f_i)$. Then, for a given $(\mathbf{vc}, \mathbf{cc})$ pair, the unnormalized marginal over v is:

$$m(v{=}x) = \sum_{\mathbf{x}_\mathcal{O}} \phi_{i,\mathbf{vc}}(v{=}x, \mathbf{x}_\mathcal{O}) \cdot \prod_{w\in\mathcal{O}} \mathbf{q}_w(x_w), \tag{16}$$

where $\mathbf{x}_\mathcal{O}$ ranges over all assignments to $\mathcal{O} = \mathrm{scope}(f_i) \setminus \{v\}$. This treats the child and parent variables uniformly as arguments of the same factor. Expanding Equation 16 when $v = X_i$ recovers the child-target expression, while expanding it when $v \in \mathrm{Pa}(X_i)$ recovers the parent-target expression.

The marginal is normalized: $p(x) = m(x)/\sum_{x'} m(x')$.

The factor-to-variable message then takes the componentwise minimum and maximum over all $(\mathbf{vc}, \mathbf{cc})$ pairs:

$$l_{f\to v}(x) = \min_{\mathbf{vc},\,\mathbf{cc}} p(x),$$
$$u_{f\to v}(x) = \max_{\mathbf{vc},\,\mathbf{cc}} p(x).$$

Our second approach, called Interval Belief Propagation (IBP) and outlined in Algorithm 2, is an iterative message-passing algorithm for approximate marginal inference that propagates *interval messages* on the *factor graph* associated with the credal network $\mathcal{C}$ corresponding to the chain-graph factorization of the input LCN. Unlike previous algorithms like L2U Ide & Cozman (2008) and GL2U Antonucci et al. (2010) which are restricted to binary variables, IBP natively handles multi-valued variables.

E.5 DETAILED RUNNING EXAMPLE: ALARM NETWORK

We trace the IBP algorithm on the alarm network from file `alarm.lcn`:

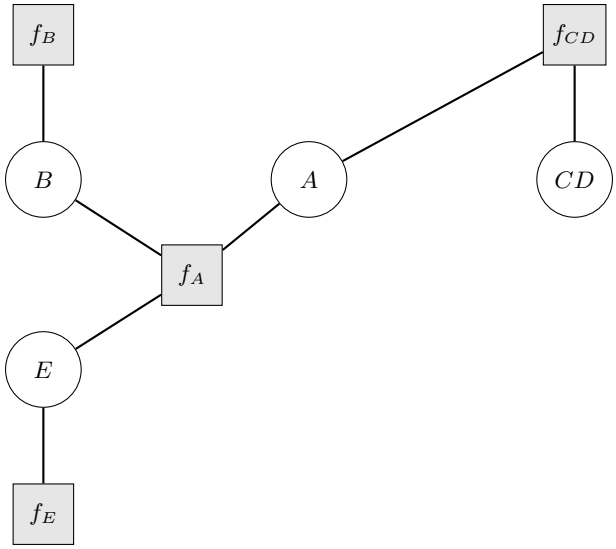
```
s1: 0.1 <= P(B) <= 0.2
s2: 0.05 <= P(E) <= 0.1
```

```
s3: 0.6 <= P(A | (B or E)) <= 0.9
s4: 0.5 <= P(not(C xor D) | A) <= 0.7 ; True
s5: 0.05 <= P(A) <= 0.1
```

After the chain graph factorization and LRS vertex enumeration, the credal network has four nodes: B (binary), E (binary), A (binary), and CD (4-valued, representing the compound variable for C and D). The DAG structure is $B \to A$, $E \to A$, $A \to CD$.

E.6 Factor Graph

The factor graph is a bipartite graph with 4 variable nodes and 4 factor nodes:



Each factor's scope and adjacency:

Factor	Node	Parents	Scope
f_B	B	—	$\{B\}$
f_E	E	—	$\{E\}$
f_A	A	B, E	$\{A, B, E\}$
f_{CD}	CD	A	$\{CD, A\}$

Variable adjacency: $B \leftrightarrow \{f_B, f_A\}$, $E \leftrightarrow \{f_E, f_A\}$, $A \leftrightarrow \{f_A, f_{CD}\}$, $CD \leftrightarrow \{f_{CD}\}$.

E.7 Credal Factors (Extreme Points)

Factor f_B: $K(B)$, no parents.

Vertex	$P(B{=}0)$	$P(B{=}1)$
v_1	0.80	0.20
v_2	0.90	0.10

Factor f_E: $K(E)$, no parents.

Vertex	$P(E{=}0)$	$P(E{=}1)$
v_1	0.90	0.10
v_2	0.95	0.05

Factor f_A: $K(A \mid B, E)$, parents $\{B, E\}$. Four parent configurations, each with 2 vertices ($2^4 = 16$ vertex combinations):

Parent config	Vertex	$P(A{=}0 \mid \cdot)$	$P(A{=}1 \mid \cdot)$
$B{=}0, E{=}0$	v_1	0.9556	0.0444
$B{=}0, E{=}0$	v_2	1.0000	0.0000
$B{=}1, E{=}0$	v_1	0.0000	1.0000
$B{=}1, E{=}0$	v_2	1.0000	0.0000
$B{=}0, E{=}1$	v_1	0.0000	1.0000
$B{=}0, E{=}1$	v_2	1.0000	0.0000
$B{=}1, E{=}1$	v_1	0.0000	1.0000
$B{=}1, E{=}1$	v_2	1.0000	0.0000

Factor f_{CD}: $K(CD \mid A)$, parent $\{A\}$. Two parent configurations: $A{=}0$ (4 vertices) and $A{=}1$ (11 vertices). For brevity, we show a subset:

Config	Vtx	$P(CD{=}0)$	$P(CD{=}1)$	$P(CD{=}2)$	$P(CD{=}3)$
$A{=}0$	v_1	0.00	0.00	0.00	1.00
$A{=}0$	v_2	0.00	0.00	1.00	0.00
$A{=}0$	v_3	0.00	1.00	0.00	0.00
$A{=}0$	v_4	1.00	0.00	0.00	0.00
$A{=}1$	v_1	0.00	0.00	0.30	0.70
$A{=}1$	v_2	0.00	0.30	0.00	0.70
$\cdots$ (9 more)					

There are $4 \times 11 = 44$ vertex combinations for f_{CD}.

E.8 Query: $P(B{=}1)$, No Evidence

E.8.1 Iteration 1: Factor-to-Variable Messages

Message $f_B \to B$. Factor f_B has scope $\{B\}$ with no parents and no other variables ($\mathcal{O} = \emptyset$).

- **Vertex combinations:** 2 (one per vertex of $K(B)$, since there is only one parent configuration—the empty one).
- **Corner-point combinations:** $2^0 = 1$ (no other variables).

Computation for each **vc**:

$$\begin{aligned} \textbf{vc} = (1): &\quad m = v_1 = [0.80,\ 0.20], \quad p = m/1.0 = [0.80,\ 0.20]. \\ \textbf{vc} = (2): &\quad m = v_2 = [0.90,\ 0.10], \quad p = m/1.0 = [0.90,\ 0.10]. \end{aligned}$$

Result:

$$\mu_{f_B \to B} = \big(\underbrace{[\,0.80,\ 0.10\,]}_{\mathbf{l}=\min},\ \underbrace{[\,0.90,\ 0.20\,]}_{\mathbf{u}=\max} \big).$$

Message $f_E \to E$. Analogous computation:

$$\mu_{f_E \to E} = \big([\,0.90,\ 0.05\,],\ [\,0.95,\ 0.10\,] \big).$$

Message $f_A \to A$. Factor f_A has scope $\{A, B, E\}$, target is A, so $\mathcal{O} = \{B, E\}$. Specializing Equation 16 to this target gives:

$$m(A{=}x) = \sum_{\pi \in \{0,1\}^2} P_{\mathbf{vc}}(A{=}x \mid B{=}\pi_B, E{=}\pi_E) \cdot \mathbf{q}_B(\pi_B) \cdot \mathbf{q}_E(\pi_E).$$

In iteration 1, the incoming messages are vacuous: $\mu_{B \to f_A} = ([0, 0], [1, 1])$, $\mu_{E \to f_A} = ([0, 0], [1, 1])$.

- **Vertex combinations:** $2^4 = 16$ (2 vertices per each of 4 parent configs).
- **Corner-point combinations:** $2^2 = 4$ (lower or upper for B and E).

When both incoming messages are vacuous, many $(\mathbf{vc}, \mathbf{cc})$ pairs produce $m = [0, 0]$ (zero total) because $\mathbf{q}_B = [0, 0]$ or $\mathbf{q}_E = [0, 0]$ zeroes out the product. Only $\mathbf{cc} = (1, 1)$ (both upper bounds $= [1, 1]$) produces nonzero marginals.

Detailed example: $\mathbf{vc} = (1, 1, 1, 1)$ (vertex 1 for all configs), $\mathbf{cc} = (1, 1)$ ($\mathbf{q}_B = [1, 1]$, $\mathbf{q}_E = [1, 1]$):

$$\begin{aligned} m(A{=}0) &= 0.9556 \cdot 1 \cdot 1 + 0.0000 \cdot 1 \cdot 1 + 0.0000 \cdot 1 \cdot 1 + 0.0000 \cdot 1 \cdot 1 = 0.9556, \\ m(A{=}1) &= 0.0444 \cdot 1 \cdot 1 + 1.0000 \cdot 1 \cdot 1 + 1.0000 \cdot 1 \cdot 1 + 1.0000 \cdot 1 \cdot 1 = 3.0444, \\ p &= [0.9556, 3.0444]/4.0 = [0.2389,\ 0.7611]. \end{aligned}$$

After scanning all 16 vertex combinations (with $\mathbf{cc} = (1, 1)$), the resulting message is wide because vacuous inputs provide no information.

Message $f_A \to B$. Target is B (a parent), $\mathcal{O} = \{A, E\}$. This is Case 2. With vacuous incoming messages, the message to B is similarly wide.

Messages from f_{CD}. Factor f_{CD} has scope $\{CD, A\}$ with 44 vertex combinations and $2^1 = 2$ corner-point combinations per target variable. In the first iteration with vacuous incoming messages, these produce wide bounds.

E.8.2 Iteration 1: Variable-to-Factor Messages

After the factor-to-variable messages are computed, the variable-to-factor messages are updated by *tightening*: each variable sends to a factor the intersection of messages from all *other* factors.

Message $B \to f_A$. Variable B has neighbors $\{f_B, f_A\}$. The message to f_A excludes f_A itself:

$$\mu_{B \to f_A} = \mu_{f_B \to B} = \big([0.80,\ 0.10],\ [0.90,\ 0.20] \big).$$

This is a significant tightening from the vacuous $([0, 0], [1, 1])$: factor f_B directly constrains B's marginal, and this information is now passed to f_A.

Message $E \to f_A$. Similarly, E has neighbors $\{f_E, f_A\}$:

$$\mu_{E \to f_A} = \mu_{f_E \to E} = \big([0.90,\ 0.05],\ [0.95,\ 0.10] \big).$$

Message $A \to f_{CD}$. Variable A has neighbors $\{f_A, f_{CD}\}$:

$$\mu_{A \to f_{CD}} = \mu_{f_A \to A}.$$

E.8.3 Iteration 2: Recomputing $f_A \to A$ with Informative Messages

Now f_A receives informative messages from B and E. The computation of $f_A \to A$ uses:

$$\begin{aligned} \mathbf{q}_B &\in \{[0.80,\ 0.10],\ [0.90,\ 0.20]\}, \\ \mathbf{q}_E &\in \{[0.90,\ 0.05],\ [0.95,\ 0.10]\}. \end{aligned}$$

Detailed example: $\mathbf{vc} = (1, 1, 1, 1)$, $\mathbf{cc}$: $\mathbf{q}_B = [0.80, 0.10]$, $\mathbf{q}_E = [0.90, 0.05]$:

$$\begin{aligned} m(A{=}0) &= 0.9556 \cdot 0.80 \cdot 0.90 + 0.0000 \cdot 0.10 \cdot 0.90 \\ &\quad + 0.0000 \cdot 0.80 \cdot 0.05 + 0.0000 \cdot 0.10 \cdot 0.05 = 0.6880, \\ m(A{=}1) &= 0.0444 \cdot 0.80 \cdot 0.90 + 1.0000 \cdot 0.10 \cdot 0.90 \\ &\quad + 1.0000 \cdot 0.80 \cdot 0.05 + 1.0000 \cdot 0.10 \cdot 0.05 = 0.1670, \\ p &= [0.6880,\ 0.1670]/0.8550 = [\,0.8047,\ 0.1953\,]. \end{aligned}$$

Another example: $\mathbf{vc} = (1, 1, 1, 1)$, $\mathbf{cc}$: $\mathbf{q}_B = [0.90, 0.20]$, $\mathbf{q}_E = [0.95, 0.10]$:

$$\begin{aligned} m(A{=}0) &= 0.9556 \cdot 0.90 \cdot 0.95 + 0.0 \cdot 0.20 \cdot 0.95 \\ &\quad + 0.0 \cdot 0.90 \cdot 0.10 + 0.0 \cdot 0.20 \cdot 0.10 = 0.8170, \\ m(A{=}1) &= 0.0444 \cdot 0.90 \cdot 0.95 + 1.0 \cdot 0.20 \cdot 0.95 \\ &\quad + 1.0 \cdot 0.90 \cdot 0.10 + 1.0 \cdot 0.20 \cdot 0.10 = 0.3380, \\ p &= [0.8170,\ 0.3380]/1.1550 = [\,0.7074,\ 0.2926\,]. \end{aligned}$$

As the incoming messages become more informative (tighter intervals), the factor explores a more constrained space and the message to A tightens accordingly.

E.8.4 Final Marginal Extraction

After convergence, the marginal bounds for B are:

$$\underline{P}(B{=}x) = \max_{f \in \{f_B, f_A\}} l_{f \to B}(x), \quad \overline{P}(B{=}x) = \min_{f \in \{f_B, f_A\}} u_{f \to B}(x).$$

Factor f_B provides the tightest bounds for B since it directly encodes $K(B)$:

$$P(B{=}1) \in [0.10,\ 0.20].$$

E.9 Query with Evidence: $P(B{=}1 \mid E{=}1)$

We now rerun IBP with evidence $E = 1$. According to Section E.3, the outgoing messages from E are clamped to the one-hot indicator for state 1:

$$\mu_{E \to f_E} \ = \ \mu_{E \to f_A} \ = \ \big([0,\ 1],\ [0,\ 1]\big),$$

and E is skipped in every subsequent variable-to-factor update.

E.9.1 Iteration 1: $f_B \to B$ and $f_E \to E$.

f_B is not adjacent to E, so its outgoing message is identical to the no-evidence case:

$$\mu_{f_B \to B} = \big([\,0.80,\ 0.10\,],\ [\,0.90,\ 0.20\,]\big).$$

$f_E \to E$ is also unchanged by the clamping of $\mu_{E \to f_E}$ (the factor receives evidence on its target; the outgoing message still reflects $K(E)$). However, this message will be intersected with the clamped $\mu_{E \to f_E}$ only at the extraction step, so in practice $P(E)$ is pinned to $[0, 1]$ on state 1.

E.9.2 Iteration 1: $f_A \to A$ with E clamped.

Target is A, $\mathcal{O} = \{B, E\}$. In iteration 1, B still sends the vacuous message $\mu_{B \to f_A} = ([0, 0], [1, 1])$, but $\mu_{E \to f_A} = ([0, 1], [0, 1])$. The corner-point enumeration over $\mathcal{O}$ now collapses along the E axis since both corners select $\mathbf{q}_E = [0, 1]$, leaving only the two B corners free. Using Equation 16:

$$m(A{=}x) = \sum_{\pi_B, \pi_E \in \{0,1\}} P_{\mathbf{vc}}(A{=}x \mid B{=}\pi_B, E{=}\pi_E) \cdot \mathbf{q}_B(\pi_B) \cdot \underbrace{\mathbf{q}_E(\pi_E)}_{0 \text{ if } \pi_E = 0}.$$

Only the $\pi_E = 1$ terms survive. With $\mathbf{vc} = (1, 1, 1, 1)$ (vertex 1 for every parent configuration) and $\mathbf{q}_B = [1, 1]$ (the upper corner, since B's message is vacuous):

$$\begin{aligned} m(A{=}0) &= \underbrace{0.9556 \cdot 1 \cdot 0}_{B=0,E=0} + \underbrace{0.0000 \cdot 1 \cdot 0}_{B=1,E=0} + \underbrace{0.0000 \cdot 1 \cdot 1}_{B=0,E=1} + \underbrace{0.0000 \cdot 1 \cdot 1}_{B=1,E=1} = 0, \\ m(A{=}1) &= \underbrace{0.0444 \cdot 1 \cdot 0}_{B=0,E=0} + \underbrace{1.0000 \cdot 1 \cdot 0}_{B=1,E=0} + \underbrace{1.0000 \cdot 1 \cdot 1}_{B=0,E=1} + \underbrace{1.0000 \cdot 1 \cdot 1}_{B=1,E=1} = 2.0, \\ p &= [0,\ 2.0]/2.0 = [0,\ 1]. \end{aligned}$$

Under $\mathbf{vc} = (2, 2, 1, 1)$ (vertex 2 for $B{=}0, E{=}0$ and $B{=}1, E{=}0$, vertex 1 for $B{=}0, E{=}1$ and $B{=}1, E{=}1$), the same calculation still gives $p = [0, 1]$ because the $\pi_E = 0$ terms are zeroed out regardless of the vertex choice there. Clamping $E{=}1$ drives the marginal of A toward $A{=}1$: whenever $E{=}1$ forces the child, all surviving vertices in these configurations have $P(A{=}1 \mid \cdot) = 1.0$ for vertex 1 and $P(A{=}0 \mid \cdot) = 1.0$ for vertex 2. Combining across vertex combinations yields $\mu_{f_A \to A} = ([0, 0], [1, 1])$ in iteration 1 — still wide, because B is vacuous.

E.9.3 ITERATION 1: $f_A \to B$ WITH E CLAMPED.

Target is B, $\mathcal{O} = \{A, E\}$. Again $\mathbf{q}_E = [0, 1]$ in both corners, and only parent configurations with $\pi_E = 1$ contribute. Specializing Equation 16 with $\mathbf{q}_A = [0, 0]$ or $[1, 1]$ (vacuous):

$$m(B{=}x) = \sum_{\pi_E=1} \Big(\sum_{x_i} P_{\mathbf{vc}}(A{=}x_i \mid B{=}x, E{=}1) \cdot \mathbf{q}_A(x_i) \Big) \cdot \mathbf{q}_E(1).$$

The inner sum over x_i is exactly $\sum_{x_i} \mathbf{q}_A(x_i) \cdot 1$ (since vertex 1 has $P(A{=}1 \mid \cdot) = 1$ and vertex 2 has $P(A{=}0 \mid \cdot) = 1$ for the two $E{=}1$ rows). The upper corner $\mathbf{q}_A = [1, 1]$ gives $m(B{=}0) = m(B{=}1) = 1$, i.e. $p = [0.5, 0.5]$; the lower corner $\mathbf{q}_A = [0, 0]$ zeros out the marginal. Combined with other vertex combinations, $\mu_{f_A \to B}$ in iteration 1 remains wide.

E.9.4 ITERATION 2: $B \to f_A$ AND RECOMPUTATION.

By Section E.3, E is skipped in the variable-to-factor phase; only B (and A, CD) are updated. B's message to f_A tightens to $\mu_{B \to f_A} = \mu_{f_B \to B} = ([0.80, 0.10], [0.90, 0.20])$. Re-executing $f_A \to B$ with this informative B message and the still-clamped E produces a tighter interval. After message passing converges, the bounds on B follow from the extraction step applied to evidence-clamped messages:

$$\underline{P}(B{=}x \mid E{=}1) = \max_{f \in \{f_B, f_A\}} l_{f \to B}(x), \quad \overline{P}(B{=}x \mid E{=}1) = \min_{f \in \{f_B, f_A\}} u_{f \to B}(x).$$

f_B still contributes its prior bounds $[0.10, 0.20]$ on $P(B{=}1)$, while f_A now carries information about B conditioned on $E{=}1$. On this polytree the two are consistent and the conditional posterior recovers the prior:

$$P(B{=}1 \mid E{=}1) \in [0.10,\ 0.20],$$

reflecting the fact that B and E are marginally independent in the alarm network and the asymmetry of A's CPT does not alter B's marginal under evidence on E alone.

E.10 PROOFS OF THEOREM 3 AND 4

The complexity bound of Theorem 3 follows immediately from the size of the messages passed along the edges of the factor graph.

First, IBP is known to be exact on singly connected structures (ie, trees) Pearl (1988); Dechter (1996). Given that $\mathcal{L}$ is family-realizable there are no cross-family constraints and by Proposition 2 we have that $\mathcal{D}_{\mathrm{SE}} = \mathcal{D}_{\mathrm{LCN}}$. It follows easily that CVE computes exact marginals with respect to $\mathcal{D}_{\mathrm{LCN}}$ if $\mathcal{L}$ is singly connected.

F DETAILED DESCRIPTION: COORDINATE DESCENT WITH VARIABLE ELIMINATION

Our third approach is called Coordinate Descent with Variable Elimination (CDVE) and is described by Algorithm 3. CDVE extends the ApproxLP method Antonucci et al. (2015) that reformulates credal marginal inference as a multilinear program and solves it via coordinate descent.

Specifically, for every non-evidence variable Z, the CDVE runs coordinate descent twice — once with sense $\min$ and once with sense $\max$. Each run starts by initializing all $P(X_i \mid \pi_i)$ at the centroid of their respective credal sets and computing $p^* = \mathrm{VE}(\{P\}, Z, \mathbf{y})$, the full posterior vector on D_Z under these fixed distributions. A *sweep* then visits every (X_i, π_i): for each vertex $v \in \mathrm{ext}\, K(X_i \mid \pi_i)$, it temporarily sets $P(X_i \mid \pi_i) \leftarrow v$, re-evaluates VE, and accepts v (strictly, else rejects) if the target component $p^*(z)$ moves in the sense's direction. Sweeps repeat until one produces no improvement or the cap T is reached. Because each VE call returns a full vector, the two runs together yield componentwise bounds $[\underline{p}(z), \overline{p}(z)]$ for every state $z \in D_Z$ simultaneously. Evidence variables are skipped by a point-mass shortcut. The output is *inner bounds* — coordinate descent explores only a subset of feasible vertex combinations, so the returned interval is contained in the true interval.

Algorithm 3 Coordinate Descend with Variable Elimination (CDVE)

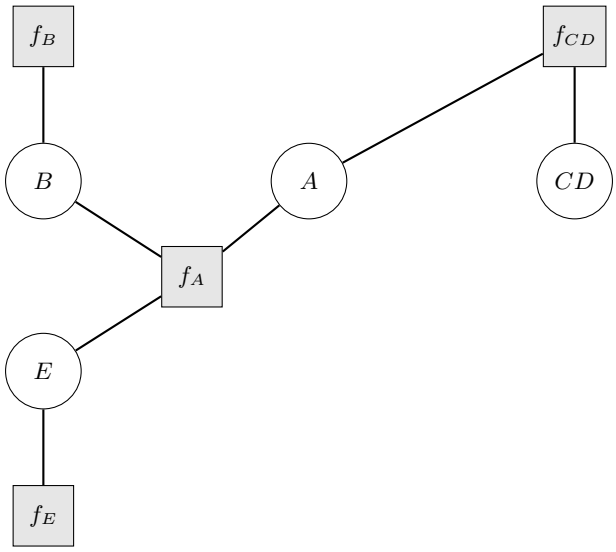
Require: LCN $\mathcal{L}$ with variables $\{X_1, \ldots, X_n\}$, evidence $\mathbf{E} = \mathbf{e}$, iterations T
Ensure: Bounds $[\underline{P}(Z{=}z|\mathbf{e}),\ \overline{P}(Z{=}z|\mathbf{e})]$ for non-evidence variables Z and states z

Build credal network $\mathcal{C}$ using chain graph factorization of $\mathcal{L}$
for each non-evidence variable Z **do**
  $\underline{\mathbf{p}},\ \overline{\mathbf{p}} \leftarrow \mathbf{1},\ \mathbf{0}$ ▷ component-wise over D_Z
  for sense $\in \{\min, \max\}$ **do**
    $P(X_i \mid \pi_i) \leftarrow \mathrm{centroid}(\mathrm{ext}\,\mathbb{K}(X_i \mid \pi_i))$
    $p^* \leftarrow \mathrm{VE}(\{P(X_i|\pi_i)\},\ Z,\ \mathbf{y})$
    for sweep $t = 1, \ldots, T$ **do**
      for each (X_i, π_i) **do**
        $v^\star \leftarrow \arg\,\mathrm{sense}_{v \in \mathrm{ext}\,K(X_i|\pi_i)}\ \mathrm{VE}(\{P\}[P(X_i \mid \pi_i){\leftarrow}v],\ Z,\ \mathbf{y})(z)$
        ▷ z = target state this run is tracking
        if $v^\star$ strictly improves $p^*(z)$ **then** $P(X_i \mid \pi_i) \leftarrow v^\star,\ p^* \leftarrow \mathrm{VE}(\{P\},\ Z,\ \mathbf{y})$
      if sweep made no change **then break**
    $\underline{\mathbf{p}} \leftarrow \min(\underline{\mathbf{p}},\ p^*),\quad \overline{\mathbf{p}} \leftarrow \max(\overline{\mathbf{p}},\ p^*)$
  return $[\underline{p}(z),\ \overline{p}(z)]$ for every $z \in D_Z$

F.1 Detailed Example

We trace CDVE on the example LCN with variables B, E, A, CD and the DAG $B \to A$, $E \to A$, $A \to CD$.

F.1.1 Factor Graph



F.1.2 Credal Factors (Extreme Points)

f_B**:** $K(B)$**.** $v_1 = [0.80, 0.20],\ v_2 = [0.90, 0.10]$.

f_E**:** $K(E)$**.** $v_1 = [0.90, 0.10],\ v_2 = [0.95, 0.05]$.

f_A**:** $K(A \mid B, E)$**.** 4 parent configs, 2 vertices each:

Config	Vtx	$P(A{=}0)$	$P(A{=}1)$
$B{=}0, E{=}0$	v_1	0.9556	0.0444
$B{=}0, E{=}0$	v_2	1.0000	0.0000
$B{=}1, E{=}0$	v_1	0.0000	1.0000
$B{=}1, E{=}0$	v_2	1.0000	0.0000
$B{=}0, E{=}1$	v_1	0.0000	1.0000
$B{=}0, E{=}1$	v_2	1.0000	0.0000
$B{=}1, E{=}1$	v_1	0.0000	1.0000
$B{=}1, E{=}1$	v_2	1.0000	0.0000

f_{CD}: $K(CD \mid A)$. $A{=}0$: 4 vertices; $A{=}1$: 11 vertices (see companion documents for full listing).

F.1.3 Query 1: $P(B{=}1)$, No Evidence

Step 1: Initialization. Each distribution is set to the center (average) of its credal set:

$$P(B) = \tfrac{1}{2}([0.80, 0.20] + [0.90, 0.10]) = [0.85,\ 0.15],$$
$$P(E) = \tfrac{1}{2}([0.90, 0.10] + [0.95, 0.05]) = [0.925,\ 0.075],$$
$$P(A \mid B{=}0, E{=}0) = \tfrac{1}{2}([0.9556, 0.0444] + [1.0, 0.0]) = [0.9778,\ 0.0222].$$

(Other parent configs for A are similarly averaged.)

Step 2: Initial evaluation. Run VE with these fixed distributions:

$$P(B{=}1) = 0.1500 \quad \text{(the center value for } B\text{)}.$$

Step 3: Coordinate descent (sense = $\min$). Iteration 0:

Try variable B, config $()$:

- Set $P(B) = v_1 = [0.80, 0.20]$, run VE: $P(B{=}1) = 0.2000$. Worse than $0.1500 \rightarrow$ reject.
- Set $P(B) = v_2 = [0.90, 0.10]$, run VE: $P(B{=}1) = 0.1000$. Better $\rightarrow$ **accept**. Update incumbent: $p^* = 0.1000$.

Try variable E, config $()$:

- $v_1 = [0.90, 0.10]$: VE gives $P(B{=}1) = 0.1000$. No improvement.
- $v_2 = [0.95, 0.05]$: VE gives $P(B{=}1) = 0.1000$. No improvement.

Since B is a root node, $P(B)$ directly determines $P(B{=}1)$ regardless of other distributions. The algorithm correctly identifies $v_2 = [0.90, 0.10]$ as the minimizer in one iteration.

Other variables (A, CD) have no effect on $P(B{=}1)$ after B's vertex is fixed.

Iteration 1: No improvement found $\rightarrow$ converged.

Result (min): $\underline{P}(B{=}1) = 0.1000$.

Coordinate descent (sense = $\max$). Analogous: selecting $v_1 = [0.80, 0.20]$ gives $\overline{P}(B{=}1) = 0.2000$. Converges in 1 iteration.

Final result:

$$\boxed{P(B{=}1) \in [0.1000,\ 0.2000].}$$

F.1.4 Query 2: $P(A{=}1 \mid B{=}0, E{=}0)$

Initialization. With evidence $B{=}0, E{=}0$, the only relevant parent config for A is $(B{=}0, E{=}0)$. The initial center distribution is:

$$P(A \mid B{=}0, E{=}0) = [0.9778,\ 0.0222].$$

Initial evaluation. VE with fixed distributions and evidence gives $P(A{=}1 \mid B{=}0, E{=}0) = 0.0222$.

Coordinate descent (sense = $\min$**).** *Try A, config* $(B{=}0, E{=}0)$*:*

- $v_1 = [0.9556, 0.0444]$: VE gives $P(A{=}1) = 0.0444$. Worse $\rightarrow$ reject.
- $v_2 = [1.0000, 0.0000]$: VE gives $P(A{=}1) = 0.0000$. Better $\rightarrow$ **accept**.

Converged: $\underline{P}(A{=}1 \mid B{=}0, E{=}0) = 0.0000$.

Coordinate descent (sense = $\max$**).** *Try A, config* $(B{=}0, E{=}0)$*:*

- $v_1 = [0.9556, 0.0444]$: $P(A{=}1) = 0.0444$. Better $\rightarrow$ **accept**.
- $v_2 = [1.0, 0.0]$: $P(A{=}1) = 0.0000$. Worse $\rightarrow$ reject.

Converged: $\overline{P}(A{=}1 \mid B{=}0, E{=}0) = 0.0444$.

Final result:

$$\boxed{P(A{=}1 \mid B{=}0, E{=}0) \in [0.0000,\ 0.0444].}$$

F.2 Proof of Theorem 5

The complexity bound follows easily since the inner computation is done by variable elimination which is exponential in the treewidth w^* of the network Dechter (1996).

G Detailed Description of the CJT Algorithm

CJT proceeds in four steps. It builds a cluster tree over the atoms from the family scopes of the chain graph equation 11; it assigns every sentence and every non-redundant independence assertion to a cluster containing its scope; it writes a single constraint system whose variables are one distribution per cluster; and it reads the query off a cluster containing it, minimizing and maximizing over that system. Algorithm 4 states the procedure; Section H writes the constraint system formally.

The only input the algorithm takes from the chain graph is the collection of family *scopes* $\{\mathrm{ch}_v \cup \mathrm{pa}_v\}_v$, together with the LCN's own sentences and assertions. It never forms the local credal sets $K_{v,\pi}$ of equation 12, and never enumerates their extreme points. This is the structural reason its target is $\mathcal{D}_{\mathrm{LCN}}$ rather than $\mathcal{D}_{\mathrm{SE}}$: neither Gap A nor Gap B can arise from a construction that does not build the objects in which those gaps live.

Throughout, $\mathcal{C}$ denotes the clusters, $A(C) \subseteq \mathbf{V}$ the atoms of cluster C, $S_{CD} = A(C) \cap A(D)$ the separator on a tree edge, and

$$w + 1 = \max_{C \in \mathcal{C}} |A(C)|, \tag{17}$$

so w is the treewidth of the triangulated graph and $w + 1$ the largest cluster measured in atoms. Cluster sizes are always reported in atoms below.

G.1 Constructing the cluster tree

The scopes to be covered are the family scopes, augmented by one clique per *cross-family sentence* — a sentence whose atoms fit inside no single family scope. Augmenting forces such a sentence's atoms to co-occur in a cluster, which is what guarantees it a host in Section G.2.

Independence assertions are deliberately *not* used to augment. They are structural, and a cluster tree with the running-intersection property already enforces them through its separators (Section G.2); adding them as cliques would inflate the width for no gain. A Markov chain, for instance, would collapse from width 2 to a single cluster containing all n atoms.

Given the scopes, an elimination order is computed greedily by *min-fill* on the interaction graph — the graph in which two nodes are adjacent when they co-occur in some scope — eliminating at each step

Algorithm 4 CJT: marginal bounds at treewidth cost

Require: sentences Σ, LMC assertions Λ, family scopes $\mathcal{F}$, evidence e, width budget b
Ensure: $[\underline{P}(a \mid e),\ \overline{P}(a \mid e)]$ for every atom $a \notin \mathrm{atoms}(e)$

$\mathcal{S} \leftarrow \mathcal{F} \cup \{\mathrm{atoms}(s) : s \in \Sigma \text{ cross-family}\}$ ▷ assertions do not augment
$T \leftarrow \textsc{BucketTree}(\mathcal{S}, \textsc{MinFillOrder}(\mathcal{S}))$; $A(C) \leftarrow \bigcup_{v \in C} \mathrm{atoms}(v)$ ▷ RIP holds [H1]
remove subsumed leaves; $S_{CD} \leftarrow A(C) \cap A(D)$ for each edge
for all $s \in \Sigma$ **do** ▷ every sentence must be hosted [H2]
  host s on a cluster C with $\mathrm{atoms}(s) \subseteq A(C)$, if one exists
if $\max_C |A(C)| > b$ **or** some $s \in \Sigma$ was not hosted **then**
  return $\textsc{Direct}(a, e)$ for every atom $a \notin \mathrm{atoms}(e)$ ▷ 2^n, uncertified; Rem. 4
for all $\iota = (X \perp\!\!\!\perp Y \mid S) \in \Lambda$ **do** ▷ three-way triage [H3]
  if S separates X from Y in G^* **then**
    skip ι ▷ entailed by the separators
  else if $X \cup Y \cup S \subseteq A(C)$ for some C **then**
    host ι on C
  else
    drop ι ▷ bound stays outer, may be loose
build $\mathcal{G}$: $q_C \in \Delta(2^{|A(C)|})$, $q_C \restriction S_{CD} = q_D \restriction S_{CD}$, and equation 23–equation 25 on each host ▷ once
for all atoms $a \notin \mathrm{atoms}(e)$ **do**
  if $\{a\} \cup \mathrm{atoms}(e) \subseteq A(C_a)$ for some C_a **then** ▷ objective only [H4]
    $\underline{P} \leftarrow \min_{q \in \mathcal{G}} \rho_a(q)$, $\overline{P} \leftarrow \max_{q \in \mathcal{G}} \rho_a(q)$ ▷ equation 26, equation 27
  else
    $[\underline{P}, \overline{P}] \leftarrow \textsc{Direct}(a, e)$
  clamp $[\underline{P}, \overline{P}]$ to $[0, 1]$

a vertex whose elimination adds fewest fill-in edges. Buckets are then formed along that order, each scope placed in the bucket of the first order variable it contains; a bucket's cluster is the union of its scopes, and its parent is the first later order variable appearing in the cluster after the bucket variable is removed. This is standard bucket elimination, so **the resulting tree has the running-intersection property by construction**, discharging hypothesis H1 of Theorem 8 without further argument.

Each cluster's atom set $A(C)$ is the union of the atoms of its nodes; a compound node contributes all atoms of its chain component at once, so a cluster may be wider in atoms than in nodes. Finally, any *leaf* whose atom set is contained in its parent's is removed, repeatedly: such a leaf is redundant, since the parent already carries that joint, and removing it is safe because a leaf has no children to reattach. Only subsumed leaves are removed, so the tree is not claimed to be minimal — a non-maximal cluster may survive in the interior.

G.2 Hosting and the assertion triage

Say a constraint is *hosted* on a cluster C when $\mathrm{scope}(c) \subseteq A(C)$; by Lemma 1 its value is then a function of q_C alone. Hosting each constraint on a single cluster suffices: separator consistency propagates its effect to the rest of the tree, as the reconstruction argument of Theorem 8 makes precise.

Every sentence is hosted if it can be. A sentence with no host cannot be represented exactly on this tree, and the algorithm falls back to the direct formulation (Section G.3).

Independence assertions admit a three-way triage, and this is the algorithm's most consequential step:

1. **Separator-entailed ⇒ omitted.** If the conditioning set S separates X from Y in the chordal graph G^* (Definition 13), every separator-consistent family of cluster marginals already satisfies $(X \perp\!\!\!\perp Y \mid S)$, so imposing it would be redundant. This is what keeps a chain at width 2.
2. **Hosted ⇒ imposed** as the bilinear equalities equation 9 on the host cluster.
3. **Neither ⇒ dropped.**

Case 3 is the algorithm's only source of looseness, and it is exactly the failure of hypothesis H3. The bound remains valid — the projection half of Theorem 8 requires neither H2 nor H3 — but it can be strictly loose; Remark **??** exhibits an instance where it is.

G.3 SOLVING

The constraint system of Step 3 mentions no query, and every atom lies in some cluster because each family scope is itself a clique. One system therefore serves all marginals: per atom, only the objective changes, and two optimizations are solved. This is the sense in which the construction is *query-independent*.

Global solution. The system is nonconvex: the assertion equalities equation 25 are bilinear, and under evidence so is the ratio equality equation 27. Exactness in Theorem 8 refers to the global optimum, so the bounds reported here were computed with a spatial branch-and-bound solver at gap tolerance 0.

Remark 3 (A local solver yields an inner bound)**.** *Solved instead by a local method from a finite set of starting points — keeping the smallest minimum and largest maximum found — the procedure returns a value no more extreme than the true optimum in each direction. The resulting interval is therefore contained in the exact one: an* inner *bound in the sense of Definition 6, and not a guarantee. Global solution is what makes Theorem 8 applicable.*

Fallback. Two conditions divert a query to the direct 2^n formulation: some cluster exceeding the width budget b, and a sentence or query having no host. The first is global to the instance; the second may be local to one atom, arising when that atom and the evidence share no cluster, i.e. when H4 fails for it alone.

Remark 4 (The fallback is exact in formulation, not in certificate)**.** *The fallback optimizes the true 2^n program over $\mathcal{D}_{\mathrm{LCN}}$, but by a local method, so its output carries no global guarantee and can be over-tight. Conversely, on instances where the 2^n formulation is solvable but its global solver fails to certify some endpoint, the cluster program of Section H is often the more reliable route to a bound, even at n small enough for the joint to be written down.*

A failed optimization is reported as the corresponding vacuous endpoint (0 for a minimum, 1 for a maximum): uninformative, never unsound. If every marginal returns vacuous, the instance or the evidence is likely inconsistent.

G.4 COMPLEXITY

Variables. One per cluster state, so by equation 17

$$\sum_{C \in \mathcal{C}} 2^{|A(C)|} \;=\; O\big(n\, 2^{w+1}\big), \tag{18}$$

since $|\mathcal{C}| = O(n)$, plus one auxiliary scalar per atom under evidence. Against 2^n for the direct formulation, this is the algorithm's reason for being: the cost is exponential in the treewidth, not in the number of atoms.

Constraints. There are $|\mathcal{C}|$ normalization equalities; $\sum_{(C,D),\, S_{CD} \neq \varnothing} 2^{|S_{CD}|} = O(n2^w)$ separator equalities; two rows per hosted sentence; and, per hosted assertion $(X \perp\!\!\!\perp Y \mid S)$,

$$2^{|Y|} \cdot \max\big(1, 2^{|S|}\big) \quad \text{equalities.} \tag{19}$$

Remark 5 (Which rows are nonconvex, and why the width bound survives)**.** *(a) Only the assertion equalities equation 25 and, under evidence, the ratio equality equation 27 are nonconvex. The cleared Type-2 rows equation 24 are* linear *in q, since ℓ and u are constants. Consequently an instance with no evidence and no hosted assertion yields a* linear *program — which is exactly the situation in Section H.1, despite that instance being loopy. (b) The count equation 19 appears unbounded in $|Y|$, but a* hosted *assertion satisfies $X \cup Y \cup S \subseteq A(C)$, so it contributes at most 2^{w+1} rows and equation 18 is unaffected. An assertion with a wide Y is precisely one that the triage of Section G.2 either discharges structurally or drops.*

Per atom the cost is two optimizations over the shared system, hence $2|\mathbf{V}|$ in total.

H THE NONLINEAR PROGRAM

Fix a cluster C and index the $2^{|A(C)|}$ assignments of $A(C)$ by i, writing ω_i^C for the i-th. For a formula φ with $\mathrm{atoms}(\varphi) \subseteq A(C)$ let $A_\varphi^{(C)} \in \{0,1\}^{2^{|A(C)|}}$ be its indicator vector, $A_\varphi^{(C)}[i] = \mathbf{1}[\omega_i^C \models \varphi]$, and abbreviate $\langle v, q_C\rangle = \sum_i v_i q_{C,i}$; thus $\langle A_\varphi^{(C)}, q_C\rangle = P(\varphi)$ whenever q_C is the marginal of a joint on $A(C)$.

Variables and normalization. One distribution per cluster,

$$q_C \in [0,1]^{2^{|A(C)|}}, \qquad \sum_{i=0}^{2^{|A(C)|}-1} q_{C,i} = 1 \qquad \forall C \in \mathcal{C}. \tag{20}$$

There are no separator variables, no per-family conditional variables and no global joint variable; the tree structure enters only through the equalities below.

Marginalization. Let $\pi_{C\to S}$ map a state of C to its restriction to $S \subseteq A(C)$, and define the $0/1$ aggregation matrix

$$M_{r,j}^{C\to S} = \mathbf{1}\big[\pi_{C\to S}(j) = r\big], \qquad \text{so} \qquad \big(M^{C\to S} q_C\big)[r] = (q_C \restriction S)(r). \tag{21}$$

Separator consistency. For every edge with a non-empty separator, the two endpoints must agree on it:

$$\sum_{j:\,\pi_{C\to S}(j)=r} q_{C,j} \;=\; \sum_{k:\,\pi_{D\to S}(k)=r} q_{D,k} \qquad \forall (C,D)\in E,\ \forall r \in \{0,\dots,2^{|S_{CD}|}-1\}, \tag{22}$$

equivalently $q_C \restriction S_{CD} = q_D \restriction S_{CD}$: linear equalities, $2^{|S_{CD}|}$ per edge. No messages are computed; the consistency conditions are imposed simultaneously with all others, which is what distinguishes this formulation from message passing.

Sentences. A Type-1 sentence hosted on $C_h \supseteq \mathrm{atoms}(\varphi)$ contributes

$$\ell \;\le\; \big\langle A_\varphi^{(C_h)},\, q_{C_h}\big\rangle \;\le\; u, \tag{23}$$

and a Type-2 sentence contributes the cleared pair

$$\ell\,\big\langle A_\psi^{(C_h)}, q_{C_h}\big\rangle \;\le\; \big\langle A_{\varphi\wedge\psi}^{(C_h)}, q_{C_h}\big\rangle \;\le\; u\,\big\langle A_\psi^{(C_h)}, q_{C_h}\big\rangle. \tag{24}$$

No division appears, and consistently with Remark 1 the rows equation 24 are vacuously satisfied when $P(\psi) = 0$: a Type-2 sentence constrains nothing on a null conditioning event.

Independence assertions. For a hosted $(X \perp\!\!\!\perp Y \mid S)$ with $X = \{x\}$, one equality per configuration $(\mathbf{y}, \mathbf{s})$. With $S \neq \varnothing$, writing the four indicators for $\{x{=}1, Y{=}\mathbf{y}, S{=}\mathbf{s}\}$, $\{S{=}\mathbf{s}\}$, $\{x{=}1, S{=}\mathbf{s}\}$, $\{Y{=}\mathbf{y}, S{=}\mathbf{s}\}$ as A_a, A_b, A_c, A_d:

$$\langle A_a, q_{C_h}\rangle\,\langle A_b, q_{C_h}\rangle - \langle A_c, q_{C_h}\rangle\,\langle A_d, q_{C_h}\rangle = 0, \tag{25}$$

which is equation 9 verbatim; and with $S = \varnothing$ the marginal form $\langle A_a, q\rangle - \langle A_b, q\rangle\langle A_c, q\rangle = 0$ for $P(x,\mathbf{y}) = P(x)P(\mathbf{y})$. These rows are bilinear, hence nonconvex, and are the reason a global solver is required.

Two features of this encoding matter. First, only the $x{=}1$ slice is needed: with $(\mathbf{y},\mathbf{s})$ fixed a binary X has one degree of freedom, so the $x{=}0$ equality follows. Second, the configurations $\mathbf{y}$ must range over *whole* assignments of Y. Replacing $(X \perp\!\!\!\perp Y \mid S)$ by the pairwise family $\{(X \perp\!\!\!\perp t \mid S)\}_{t\in Y}$ is strictly weaker for $|Y| \ge 2$ — pairwise independence does not imply joint independence — and under-constrains the program, yielding spuriously loose bounds.

Objective. Let C_a be a cluster containing the query atom a and all evidence atoms (hypothesis H4). Without evidence the read-off is linear,

$$\rho_a(q) = \left\langle A^{(C_a)}_{a=1},\, q_{C_a} \right\rangle = P(a{=}1). \tag{26}$$

With evidence e the ratio is reformulated with one auxiliary $t \in [0,1]$,

$$\rho_a(q) = t, \qquad t \cdot \left\langle A^{(C_a)}_{e}, q_{C_a} \right\rangle = \left\langle A^{(C_a)}_{a=1} \cdot A^{(C_a)}_{e},\, q_{C_a} \right\rangle, \tag{27}$$

so $t = P(a{=}1 \mid e)$ whenever $P(e) > 0$. Since t is unconstrained where $P(e) = 0$, a global solver must be prevented from exploiting that degeneracy, and a denominator floor

$$\left\langle A^{(C_a)}_{e}, q_{C_a} \right\rangle \;\geq\; \varepsilon \tag{28}$$

is imposed for a small $\varepsilon > 0$. Remark 6 shows this floor is a soundness hypothesis, not a numerical convenience.

The program. Collecting equation 20–equation 27,

$$\begin{array}{rlll} \underline{P}(a),\, \overline{P}(a) \;=\; \min / \max & \rho_a(q) & & \\ \text{s.t.} & \sum_i q_{C,i} = 1, \quad q_{C,i} \in [0,1] & \forall C \in \mathcal{C} & \\ & q_C \restriction S_{CD} = q_D \restriction S_{CD} & \forall (C,D) \in E,\; S_{CD} \neq \varnothing & \\ & \text{equation 23, equation 24} & \forall \text{ hosted sentence} & \\ & \text{equation 25} & \forall \text{ hosted assertion.} & \end{array} \tag{29}$$

H.1 A WORKED EXAMPLE

We work `examples/ktree_fr_k2_n4_example.lcn` completely. It is a k-tree with $k = 2$ on four atoms — *loopy* after moralization, so the example is not a degenerate singly-connected case — yet small enough that every variable and every constraint can be written down.

H.1.1 THE INSTANCE

$$\begin{aligned} s_0 &: \quad 0.084382 \leq P(x_1) \leq 0.684382 \\ s_1 &: \quad 0.01 \leq P(\neg x_2 \mid \neg x_1) \leq 0.572656 \\ s_2 &: \quad 0.092785 \leq P(x_3 \mid x_1 \wedge x_2) \leq 0.692785 \\ s_3 &: \quad 0.068242 \leq P(x_0 \mid x_1 \wedge x_3) \leq 0.668242 \end{aligned}$$

The chain-graph families, in construction order, are

$$x_1 \mid \varnothing, \qquad x_2 \mid x_1, \qquad x_3 \mid x_1, x_2, \qquad x_0 \mid x_1, x_3,$$

and the Local Markov Condition yields exactly one assertion,

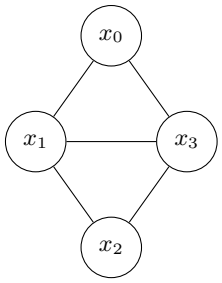
$$(x_0 \perp\!\!\!\perp x_2 \mid x_1, x_3).$$

H.1.2 THE JUNCTION TREE

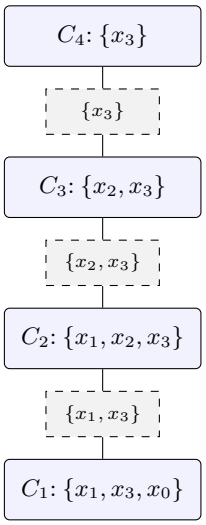
Min-fill elimination produces four clusters of at most three atoms, arranged in a path (Figure 4). In the convention equation 17 this is $w + 1 = 3$, i.e. treewidth $w = 2 = k$, as expected for a k-tree:

$$\underbrace{\{x_3\}}_{C_4} - \underbrace{\{x_2, x_3\}}_{C_3} - \underbrace{\{x_1, x_2, x_3\}}_{C_2} - \underbrace{\{x_1, x_3, x_0\}}_{C_1},$$

with separators $S_{12} = \{x_1, x_3\}$, $S_{23} = \{x_2, x_3\}$, $S_{34} = \{x_3\}$. Hosting assigns s_0, s_3 to C_1 and s_1, s_2 to C_2.



(a) moral graph (loopy)



(b) cluster tree; separators dashed, on the edges

Figure 4: The example instance. Moralization makes $\{x_1, x_2, x_3\}$ and $\{x_0, x_1, x_3\}$ triangles, so the graph is loopy; the cluster tree nonetheless has width 2. The separator $S_{12} = \{x_1, x_3\}$ equals the conditioning set of the sole assertion $(x_0 \perp\!\!\!\perp x_2 \mid x_1, x_3)$, which is therefore separator-entailed and never imposed.

H.1.3 Separator-entailment discharges the assertion

The chordal graph G^* of Definition 13 — two atoms adjacent iff they co-occur in a cluster — has edge set

$$x_0x_1, \quad x_0x_3, \quad x_1x_2, \quad x_1x_3, \quad x_2x_3,$$

and in particular $x_0x_2 \notin G^*$. Deleting $S = \{x_1, x_3\}$ therefore disconnects x_0 from x_2: the only paths between them run through x_1 or x_3. So S separates x_0 from x_2 in G^*, and the sole assertion $(x_0 \perp\!\!\!\perp x_2 \mid x_1, x_3)$ is separator-entailed — case 1 of the triage in Section G.2. It is *not* imposed, and the constraint count below shows zero assertion rows. By Remark 5(a) the program is consequently a *linear* program, even though the instance is loopy.

This is the mechanism of Theorem 8 in its clearest form: the independence is enforced by the tree's *topology* — through Lemma 3's reconstruction, which produces a joint factorizing over the chordal graph and hence satisfying its global Markov property — rather than by any algebraic row.

H.1.4 The program in full

Variables (22). $q_{C_1} \in \Delta(8)$ over (x_1, x_3, x_0); $q_{C_2} \in \Delta(8)$ over (x_1, x_2, x_3); $q_{C_3} \in \Delta(4)$ over (x_2, x_3); $q_{C_4} \in \Delta(2)$ over (x_3). Total $8 + 8 + 4 + 2 = 22$.

Simplex (4). $\sum_i q_{C_k,i} = 1$ for $k = 1, \ldots, 4$.

Separator consistency ($4 + 4 + 2 = 10$). Writing $q_C[\,\cdot\,]$ for a cell of cluster C, edge $C_1 - C_2$ over $\{x_1, x_3\}$ gives four equalities,

$$q_{C_1}[x_1{=}0, x_3{=}0, x_0{=}0] + q_{C_1}[x_1{=}0, x_3{=}0, x_0{=}1] = q_{C_2}[x_1{=}0, x_2{=}0, x_3{=}0] + q_{C_2}[x_1{=}0, x_2{=}1, x_3{=}0],$$
$$q_{C_1}[x_1{=}0, x_3{=}1, x_0{=}0] + q_{C_1}[x_1{=}0, x_3{=}1, x_0{=}1] = q_{C_2}[x_1{=}0, x_2{=}0, x_3{=}1] + q_{C_2}[x_1{=}0, x_2{=}1, x_3{=}1],$$
$$q_{C_1}[x_1{=}1, x_3{=}0, x_0{=}0] + q_{C_1}[x_1{=}1, x_3{=}0, x_0{=}1] = q_{C_2}[x_1{=}1, x_2{=}0, x_3{=}0] + q_{C_2}[x_1{=}1, x_2{=}1, x_3{=}0],$$
$$q_{C_1}[x_1{=}1, x_3{=}1, x_0{=}0] + q_{C_1}[x_1{=}1, x_3{=}1, x_0{=}1] = q_{C_2}[x_1{=}1, x_2{=}0, x_3{=}1] + q_{C_2}[x_1{=}1, x_2{=}1, x_3{=}1];$$

edge $C_2 - C_3$ over $\{x_2, x_3\}$ gives four more,

$$q_{C_2}[x_1{=}0, x_2{=}\beta, x_3{=}\gamma] + q_{C_2}[x_1{=}1, x_2{=}\beta, x_3{=}\gamma] = q_{C_3}[x_2{=}\beta, x_3{=}\gamma], \qquad \beta, \gamma \in \{0, 1\};$$

and edge $C_3 - C_4$ over $\{x_3\}$ gives two,

$$q_{C_3}[x_2{=}0, x_3{=}\gamma] + q_{C_3}[x_2{=}1, x_3{=}\gamma] = q_{C_4}[x_3{=}\gamma], \qquad \gamma \in \{0, 1\}.$$

Sentence rows (8). On C_1, from s_0 (Type-1) and s_3 (Type-2, cleared per equation 24):

$$0.084382 \le \left\langle A^{(C_1)}_{x_1}, q_{C_1} \right\rangle \le 0.684382,$$
$$0.068242 \left\langle A^{(C_1)}_{x_1 \wedge x_3}, q_{C_1} \right\rangle \le \left\langle A^{(C_1)}_{x_0 \wedge x_1 \wedge x_3}, q_{C_1} \right\rangle \le 0.668242 \left\langle A^{(C_1)}_{x_1 \wedge x_3}, q_{C_1} \right\rangle.$$

On C_2, from s_1 and s_2 (both Type-2):

$$0.01 \left\langle A^{(C_2)}_{\neg x_1}, q_{C_2} \right\rangle \le \left\langle A^{(C_2)}_{\neg x_2 \wedge \neg x_1}, q_{C_2} \right\rangle \le 0.572656 \left\langle A^{(C_2)}_{\neg x_1}, q_{C_2} \right\rangle,$$
$$0.092785 \left\langle A^{(C_2)}_{x_1 \wedge x_2}, q_{C_2} \right\rangle \le \left\langle A^{(C_2)}_{x_3 \wedge x_1 \wedge x_2}, q_{C_2} \right\rangle \le 0.692785 \left\langle A^{(C_2)}_{x_1 \wedge x_2}, q_{C_2} \right\rangle.$$

Independence rows (0). None, by separator-entailment as argued above.

Totals. 22 variables and 22 constraints (4 simplex, 10 separator, 8 sentence, 0 LMC), against a joint formulation's $2^4 = 16$ variables. The example is deliberately small enough that the treewidth formulation is not yet a saving; the saving is asymptotic, $O(n2^{w+1})$ against 2^n, and Table **??** shows the same machinery on two five-atom instances.

H.1.5 Results

Solving equation 29 for each atom with SCIP (global, gap 0) and comparing against the independent 2^n global engine:

engine	$P(x_0)$	$P(x_1)$	$P(x_2)$	$P(x_3)$
direct 2^n formulation, global	$[0, 1]$	$[0.084382, 0.684382]$	$[0.134877, 0.996844]$	$[0, 1]$
CJT (SCIP)	$[0, 1]$	$[0.084382, 0.684382]$	$[0.134877, 0.996844]$	$[0, 1]$

The two solvers agree on every atom to every digit reported, with all eight of the reference optimizations certified at optimality gap 0. CJT reports maximum cluster 3 atoms, and a total time of about 1.8 s.

The vacuous $[0, 1]$ on x_0 and x_3 is genuine rather than a solver artifact, and the independent oracle confirms it: no sentence constrains $P(x_3)$ except through conditioning events that can themselves be driven to probability zero, at which point equation 24 becomes vacuous (Remark 1) and frees the atom. Setting $P(x_1 \wedge x_3) = 0$ likewise releases s_3 and hence x_0. Both endpoints are thus attained by feasible witnesses.

The separator marginals attained at an optimum make the mechanism concrete. Table 3 lists them at the maximizer of $P(x_1)$. Both endpoints of every edge agree on these values by equation 22; this is the calibrated family of cluster marginals that Lemma 3 reassembles into a joint of $\mathcal{D}_{\mathrm{LCN}}$ attaining the bound.

Table 3: Separator marginals $q_C \restriction S$ at the maximizer of $P(x_1)$, listed over the assignments of S. Both clusters incident to an edge realize the same vector, by equation 22.

edge	separator S	$q \restriction S$ by assignment of S			
$C_1 - C_2$	$\{x_1, x_3\}$	00: 0.3125	01: 0.0032	10: 0.6844	11: 0.0000
$C_2 - C_3$	$\{x_2, x_3\}$	00: 0.6844	01: 0.0032	10: 0.3125	11: 0.0000
$C_3 - C_4$	$\{x_3\}$	0: 0.9968	1: 0.0032		

H.2 Exactness

This section states and proves the guarantees. The development is closed: it uses only Lemma 1 from Section A and Lemma 3 below, and in particular needs no notion of realizability, no property of the strong extension, and no result about any other engine.

Definition 13 (Cluster-feasible set, hosting, separator-entailment, read-off)**.** *Let T be a clique tree over clusters $\mathcal{C}$, built by the standard moralize → triangulate → clique-tree pipeline, and let G^* be the chordal graph whose maximal cliques are the clusters (two atoms adjacent iff they co-occur in a cluster).*

- *The* cluster-feasible set *is*
$$\mathcal{G} = \big\{(q_C)_{C\in\mathcal{C}} \ : q_C \in \Delta(2^{|C|}) \text{ for each } C;\ q_C \restriction S = q_D \restriction S \text{ for every edge } (C,D)\in T,\ S = C\cap D;\ \text{every hosted constraint holds on its host cluster}\big\}.$$
- *A constraint* c *is* hosted *if some cluster* $C_h \supseteq \mathrm{scope}(c)$ *exists; its row is then imposed on* q_{C_h}*'s marginal.*
- *An LMC assertion* $(X \perp\!\!\!\perp Y \mid S)$ *is* separator-entailed *if* S *separates* X *from* Y *in* G^*.
- *For a query atom* a *and evidence* e *lying together in a cluster* C_a*, the* read-off *is* $\rho_a(q) = \sum_{i:a=1} q_{C_a}[i]$ *(no evidence), or* $\rho_a(q) = \big(\sum_{i:a=1,e} q_{C_a}[i]\big)/\big(\sum_{i:e} q_{C_a}[i]\big)$ *(with evidence).*
- $\pi(p) = (p \restriction C)_{C\in\mathcal{C}}$ *is the* projection *of a joint onto the clusters.*

The reconstruction direction needs the following lemma, whose usual statement contains two defects we must repair: it is normally asserted with a *uniqueness* claim that is false in the presence of zero separator cells, and the division in the product formula is left unaddressed.

Lemma 3 (RIP reconstruction, with zeros)**.** *Let* T *be a clique tree with the running-intersection property over clusters* $\mathcal{C}$*, and let* $(q_C) \in \prod_C \Delta(2^{|C|})$ *be separator-consistent:* $q_C \restriction S = q_D \restriction S$ *for every edge* (C,D) *with* $S = C\cap D$*. Then there exists a joint distribution* $p \in \Delta(2^{\mathbf{V}})$ *with* $p \restriction C = q_C$ *for every* $C \in \mathcal{C}$*, and* p *factorizes over the cliques of* G^**. Such* p *is unique if every separator marginal is strictly positive, but* not *in general.*

Proof. Induction on $|\mathcal{C}|$. If $|\mathcal{C}| = 1$ take $p = q_C$.

Let $|\mathcal{C}| \geq 2$ and pick a leaf cluster C of T with neighbour D and separator $S = C\cap D$. Let $T' = T - C$; by the running-intersection property $C \setminus S$ meets no cluster of T', and $(q_{C'})_{C'\in T'}$ is still separator-consistent, so the inductive hypothesis gives a joint p' over $\mathbf{V}' = \bigcup_{C'\in T'} C'$ with $p' \restriction C' = q_{C'}$ for all $C' \in T'$. Note $p' \restriction S = (p' \restriction D) \restriction S = q_D \restriction S = q_C \restriction S =: q_S$, using separator consistency.

Define a conditional kernel on the fresh atoms $C \setminus S$: for each assignment s of S,
$$\kappa(\cdot \mid s) = \begin{cases} q_C(\cdot, s)/q_S(s), & q_S(s) > 0, \\ \delta_{c_0}, & q_S(s) = 0, \end{cases}$$
where c_0 is any fixed assignment of $C \setminus S$. This is well defined: when $q_S(s) > 0$ the right-hand side is non-negative and sums over $C \setminus S$ to $q_S(s)/q_S(s) = 1$, so it is a distribution. Now set $p(x) = p'(x_{\mathbf{V}'}) \cdot \kappa(x_{C\setminus S} \mid x_S)$.

p is a distribution: it is non-negative, and summing over $x_{C\setminus S}$ first gives p', which sums to 1.

The arbitrary choice is irrelevant. On $\{x : q_S(x_S) = 0\}$ we have $p'(x_{\mathbf{V}'}) = 0$, because $p' \restriction S = q_S$ assigns zero mass to that event and $p' \geq 0$. Hence p vanishes there regardless of κ, so p does not depend on c_0. (This is also the precise sense in which the familiar quotient formula $p = \prod_C q_C / \prod_{(C,D)} q_S$ is legitimate: a zero separator cell forces every contributing cluster cell to zero, since $q_S = q_C \restriction S$ is a sum of non-negative q_C entries, so numerator and denominator vanish together and the term is assigned 0.)

Cluster marginals. For $C' \in T'$: summing p over $x_{C\setminus S}$ gives p', so $p \restriction C' = p' \restriction C' = q_{C'}$. For C itself: $p \restriction C(c,s) = \big(\sum \text{ of } p \text{ over } \mathbf{V}' \setminus S \text{ at fixed } x_S = s\big) \cdot \kappa(c \mid s) = p' \restriction S(s)\, \kappa(c \mid s) = q_S(s)\kappa(c \mid s)$, which equals $q_C(c,s)$ when $q_S(s) > 0$ and equals $0 = q_C(c,s)$ when $q_S(s) = 0$. So $p \restriction C = q_C$.

Factorization. Unwinding the induction, p is a product of one non-negative factor per cluster (namely q_C or the kernel κ derived from it, each a function of the atoms of that cluster only), hence factorizes over the cliques of G^*.

Non-uniqueness. If some separator cell $q_S(s) = 0$, then $\kappa(\cdot \mid s)$ was chosen arbitrarily; any other choice yields a joint with the same cluster marginals, since all such joints vanish on that event. Hence p is not unique. When all separator marginals are positive the kernels are forced and the induction determines p uniquely. □

Theorem 8 (CJT exactness)**.** *Suppose* (H1) T *has the running-intersection property (true by construction);* (H2) *every LCN sentence is hosted;* (H3) *every LMC assertion is either hosted or separator-entailed in* G^**;* (H4) *the query atom and all evidence atoms lie together in some cluster. Then for every such atom* a*,*

$$\min_{q\in\mathcal{G}} \rho_a(q) = \min_{p\in\mathcal{D}_{\mathrm{LCN}}} \rho_a(\pi(p)) = \underline{P}(a), \qquad \max_{q\in\mathcal{G}} \rho_a(q) = \max_{p\in\mathcal{D}_{\mathrm{LCN}}} \rho_a(\pi(p)) = \overline{P}(a),$$

and likewise for a conditional query $P(a \mid e)$ *subject to Remark 6. Moreover the* projection *half holds without* (H2)*/*(H3)*, so* CJT *is always a valid* outer *bound.*

Proof. First note the read-off identity: for any joint p, $\rho_a(\pi(p)) = P_p(a)$ (resp. $P_p(a \mid e)$), because ρ_a sums the cells of $\pi(p)_{C_a} = p \restriction C_a$ in which a (and e) hold, which by definition of a marginal equals $P_p(a)$ (resp. the ratio). So both sides optimize the *same* functional.

Projection $\mathcal{D}_{\mathrm{LCN}} \to \mathcal{G}$ *(needs only H1, H4).* Let $p \in \mathcal{D}_{\mathrm{LCN}}$ and set $q_C := p \restriction C$, i.e. $q = \pi(p)$. Each q_C is a distribution (a marginal of one). Each separator equality holds because both $q_C \restriction S$ and $q_D \restriction S$ equal $p \restriction S$. Each hosted constraint c holds: by Lemma 1 its value depends on p only through $p \restriction \mathrm{scope}(c)$, and since $\mathrm{scope}(c) \subseteq C_h$ we have $q_{C_h} \restriction \mathrm{scope}(c) = p \restriction \mathrm{scope}(c)$, so c holds on the host exactly because it holds for p. Hence $q \in \mathcal{G}$, with $\rho_a(q) = P_p(a)$. Therefore every value achievable over $\mathcal{D}_{\mathrm{LCN}}$ is achievable over $\mathcal{G}$, giving $\min_{\mathcal{G}} \rho_a \le \min_{\mathcal{D}_{\mathrm{LCN}}}$ and $\max_{\mathcal{G}} \rho_a \ge \max_{\mathcal{D}_{\mathrm{LCN}}}$: the cluster interval *contains* the true interval. This is the outer-bound claim, and it used neither H2 nor H3.

Reconstruction $\mathcal{G} \to \mathcal{D}_{\mathrm{LCN}}$ *(needs H2, H3).* Let $q \in \mathcal{G}$. By H1 and Lemma 3 there is a joint p with $p \restriction C = q_C$ for every cluster, factorizing over the cliques of G^*. We check $p \in \mathcal{D}_{\mathrm{LCN}}$.

Sub-scope consistency: if $A \subseteq C$ for some cluster C, then $p \restriction A = (p \restriction C) \restriction A = q_C \restriction A$, since marginalizing in stages agrees with marginalizing at once.

Hosted constraints (H2 and the hosted part of H3): let c be hosted on C_h. By the previous paragraph $p \restriction \mathrm{scope}(c) = q_{C_h} \restriction \mathrm{scope}(c)$, and q satisfies c on C_h by definition of $\mathcal{G}$; by Lemma 1 c therefore holds for p.

Separator-entailed LMC (the other part of H3): let $(X \perp\!\!\!\perp Y \mid S)$ be separator-entailed, so S separates X from Y in G^*. Since p factorizes over the cliques of G^*, it satisfies the *global Markov property* of G^*: factorization implies global Markov for any undirected graph, and crucially this direction requires *no* positivity assumption. Global Markov states that if S separates A from B in G^* then $A \perp\!\!\!\perp B \mid S$ under p; applying it to the connected components separated by S and then restricting to the subsets X and Y of those components (conditional independence is preserved under taking subsets, by the decomposition property of conditional independence) gives $X \perp\!\!\!\perp Y \mid S$ under p, i.e. equation 9 holds. Note this assertion was never imposed on $\mathcal{G}$: it holds automatically for the reconstructed p.

Hence $p \in \mathcal{D}_{\mathrm{LCN}}$, and $\rho_a(q) = \rho_a(\pi(p)) = P_p(a)$ because $\pi(p) = q$ on the host cluster. So every value achievable over $\mathcal{G}$ is achievable over $\mathcal{D}_{\mathrm{LCN}}$, giving the reverse inequalities. Combined with the projection half, the optima coincide. For a conditional query, by H4 the read-off is a functional of the single cluster C_a and both maps preserve q_{C_a}, so the same argument applies verbatim. □

Remark 6 (The denominator floor is a soundness hypothesis)**.** *The conditional read-off is linearized by the auxiliary equality equation 27 together with the denominator floor equation 28. The floor is an extra constraint on* $\mathcal{G}$ *that* $\mathcal{D}_{\mathrm{LCN}}$ *does not have, so the projection half of Theorem 8 — and with it the outer-bound guarantee — holds only for joints with* $P(e) \ge \varepsilon$*. If* $\min_{p\in\mathcal{D}_{\mathrm{LCN}}} P(e) < \varepsilon$*, the returned conditional interval can be* unsound*. This is a soundness caveat, not merely a tightness one, and it is the price of admitting conditional queries into a globally solved formulation.*

I Detailed Experimental Evaluation

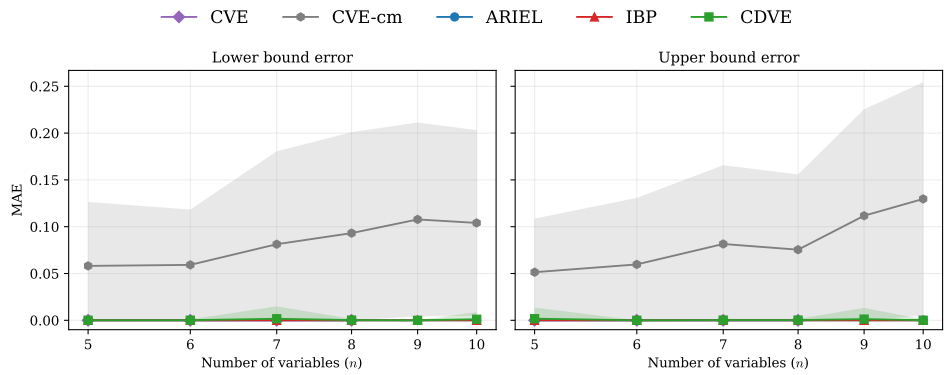
This appendix reports every measurement behind Section 5: all four random families at both scales, and the complete per-instance results for the two realistic suites. Throughout, the reference is CredalJT and the budget is 3600 seconds and 30 GB per run.

I.1 COVERAGE AND EXCLUSIONS

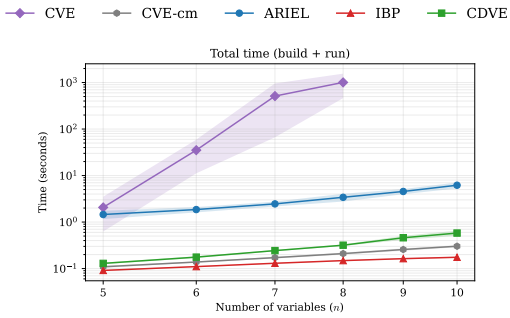
Not every algorithm completes on every instance, and reading the figures without this in mind would overstate coverage, since a curve simply stops where its algorithm ran out of budget. Over the 480 random instances, ARIEL and IBP complete essentially all of them, and CredalJT's `ipopt` backend completes all of the instances it was run on. The exceptions are concentrated in the CVE family. Plain CVE, without clustering, is the least scalable: it solves $35/60$ of `tree_small`, $30/60$ of `polytree_small` and $24/60$ of `dag_small`, but *none* of `ktree_small` (21 timeouts and 39 errors) and none of `polytree_large`; on the other three large families it was not run at all. It is therefore absent from most plots, and where present it covers only the smallest sizes. Clustering restores coverage at small scale — CVE$_{cm}$ and CVE$_{cp}$ solve all 60 instances of every small family — but degrades on the large loopy ones, where CVE$_{cm}$ reaches only $14/60$ on `ktree_large` and $30/60$ on `dag_large`. Exact inference over the 2^n joint is viable only at the very smallest sizes ($15/60$ on `tree_small`, $13/60$ on `polytree_small`, $2/10$ on the Bayesian-network instances, none of the Supreme Court ones), which is why it is not used as the reference. Three bookkeeping gaps are worth recording: the `ktree_large` CDVE log holds 58 records rather than 60, the Supreme Court exact-inference log holds 12 rather than 16, and the Bayesian-network CVE log holds 7 rather than 10; the corresponding instances are simply missing rather than failed, and are excluded from the aggregates.

I.2 RANDOM LCNS

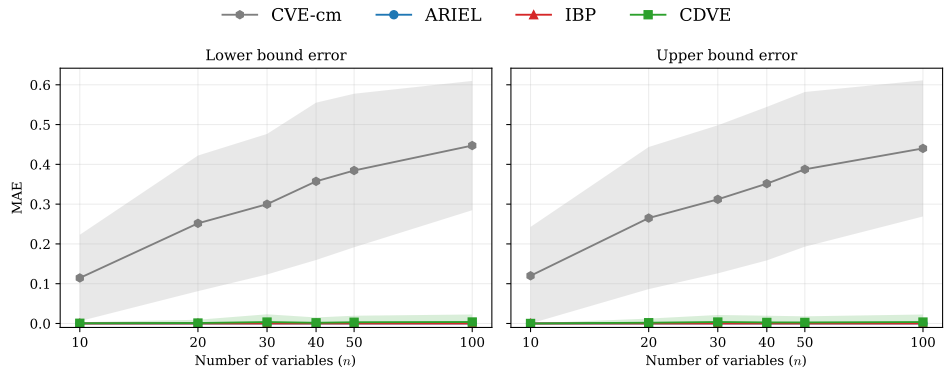
Figures 5–8 give the error and timing curves for all four families at both scales. The tables that follow them give the underlying per-size aggregates; for each family and scale the first table reports means and the second the corresponding standard deviations over the ten instances at each size.



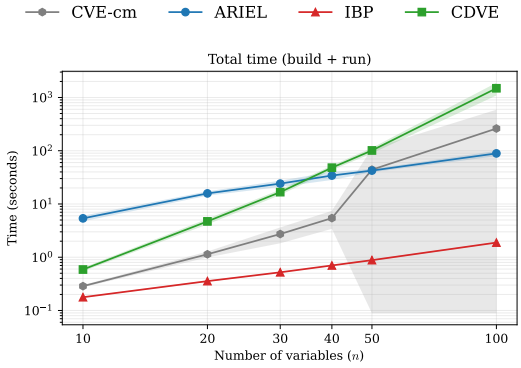
(a) small, error



(b) small, time

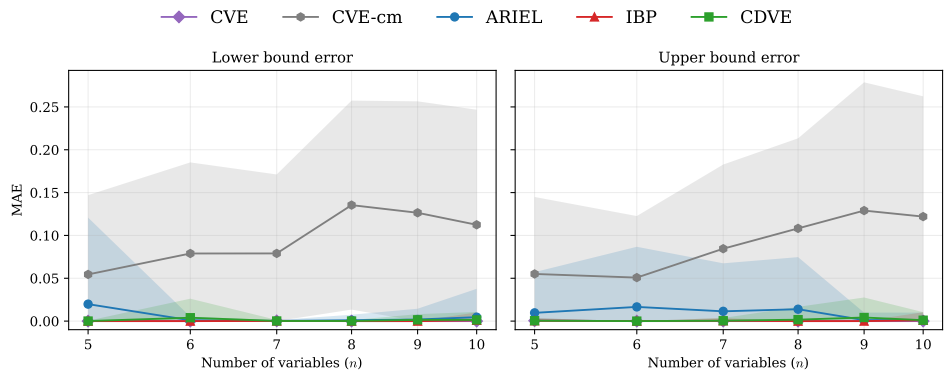


(c) large, error

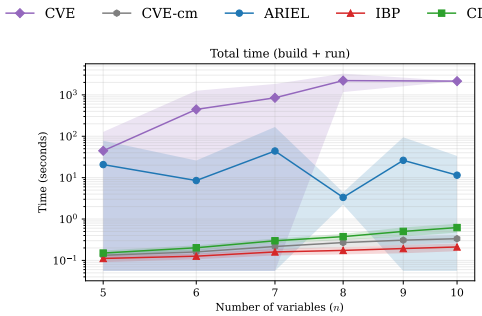


(d) large, time

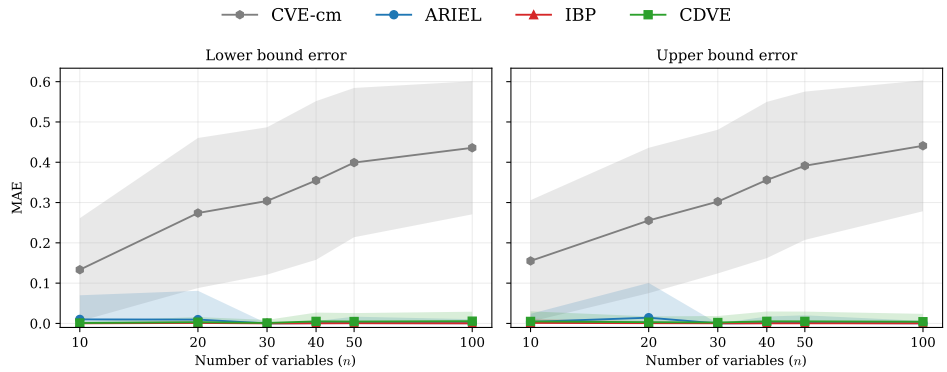
Figure 5: Random `tree` LCNs. Singly connected and fully realizable, so the credal-network relaxation is tight: ARIEL and IBP are indistinguishable from the reference.



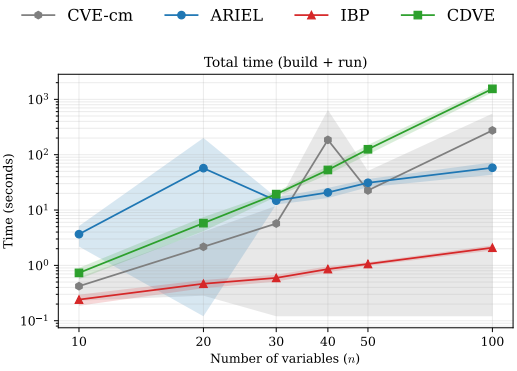
(a) small, error



(b) small, time

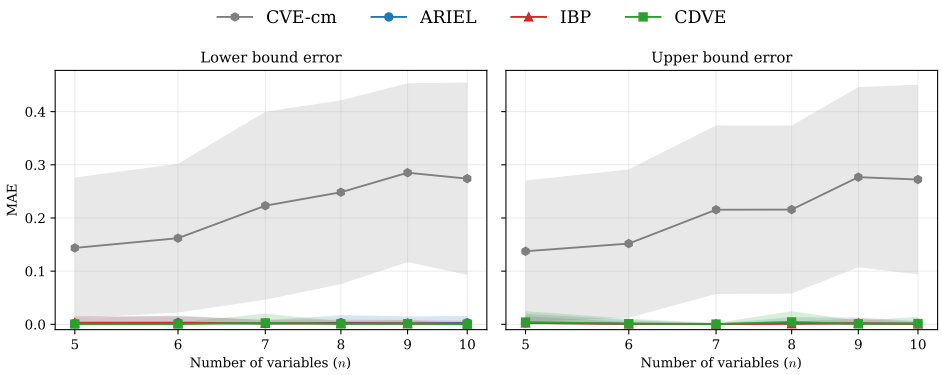


(c) large, error

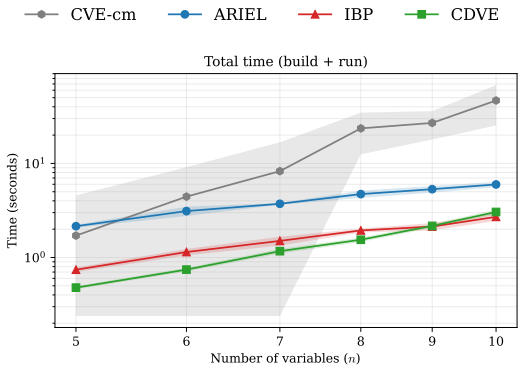


(d) large, time

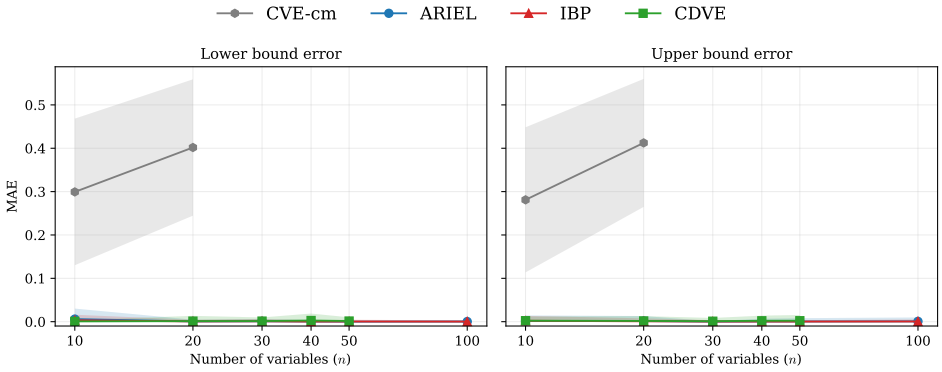
Figure 6: Random `polytree` LCNs. Also singly connected; the residual ARIEL error at the small scale comes from instances where the reference itself is the tighter engine.



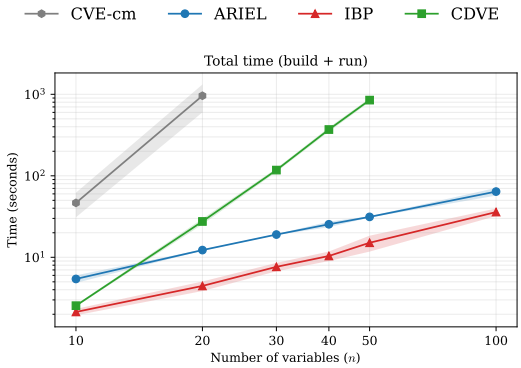
(a) small, error



(b) small, time

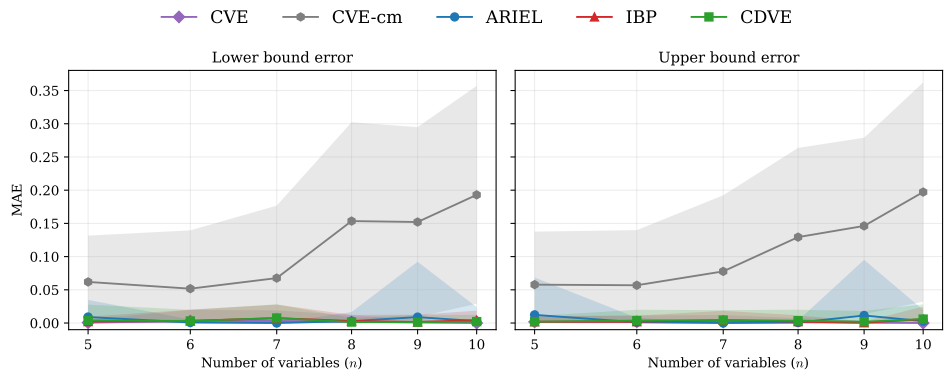


(c) large, error

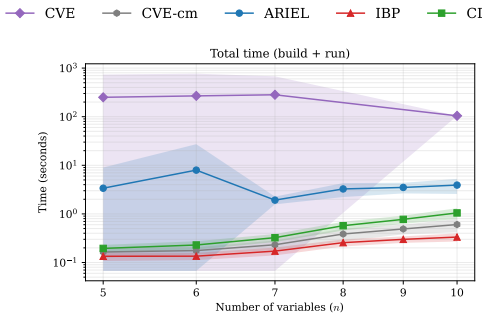


(d) large, time

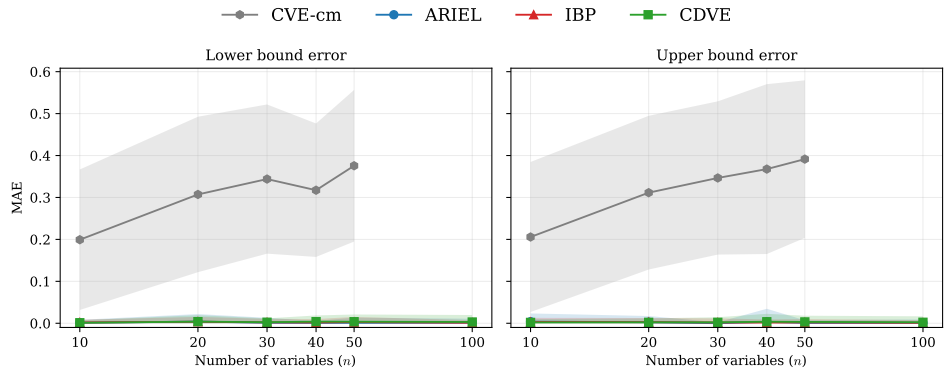
Figure 7: Random `ktree` LCNs (k=3). Loopy despite bounded treewidth, so all credal-network methods give outer bounds only; plain CVE solves none of these instances and CVE$_{cm}$ stops after n=20.



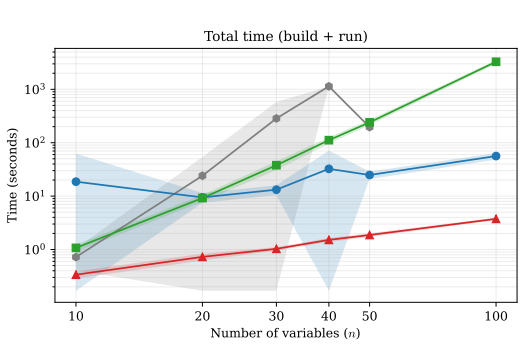
(a) small, error



(b) small, time



(c) large, error



(d) large, time

Figure 8: Random `dag` LCNs (moralized treewidth ≤ 3, at most two parents). The least structured family; CDVE's runtime grows fastest here, exceeding 600 seconds at n=100.

Table 4: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
tree_fr	5	CDVE	0.0001	0.0017	0.08	0.04	0.13	0.31	1.00
tree_fr	5	ARIEL	0.0000	0.0000	0.00	1.45	1.45	0.31	–
tree_fr	5	CVE	0.0000	0.0000	0.10	1.97	2.07	0.31	1.00
tree_fr	5	CVE_{cm}	0.0581	0.0514	0.09	0.02	0.11	0.31	1.00
tree_fr	5	IBP	0.0000	0.0000	0.09	0.00	0.09	0.31	1.00
tree_fr	6	CDVE	0.0001	0.0000	0.10	0.08	0.18	0.28	1.00
tree_fr	6	ARIEL	0.0000	0.0000	0.00	1.86	1.86	0.28	–
tree_fr	6	CVE	0.0000	0.0000	0.12	35.01	35.13	0.28	1.00
tree_fr	6	CVE_{cm}	0.0592	0.0597	0.10	0.04	0.14	0.28	1.00
tree_fr	6	IBP	0.0000	0.0000	0.11	0.00	0.11	0.28	1.00
tree_fr	7	CDVE	0.0016	0.0002	0.12	0.12	0.24	0.33	1.00
tree_fr	7	ARIEL	0.0000	0.0000	0.00	2.46	2.46	0.33	–
tree_fr	7	CVE	0.0000	0.0000	0.12	511.13	511.25	0.33	1.00
tree_fr	7	CVE_{cm}	0.0813	0.0815	0.12	0.05	0.17	0.33	1.00
tree_fr	7	IBP	0.0000	0.0000	0.13	0.00	0.13	0.33	1.00
tree_fr	8	CDVE	0.0002	0.0002	0.14	0.18	0.32	0.39	1.00
tree_fr	8	ARIEL	0.0000	0.0000	0.00	3.41	3.41	0.39	–
tree_fr	8	CVE	0.0000	0.0000	0.14	1006.64	1006.78	0.39	1.00
tree_fr	8	CVE_{cm}	0.0931	0.0754	0.14	0.07	0.21	0.39	1.00
tree_fr	8	IBP	0.0000	0.0000	0.14	0.01	0.15	0.39	1.00
tree_fr	9	CDVE	0.0000	0.0013	0.15	0.31	0.46	0.44	1.00
tree_fr	9	ARIEL	0.0000	0.0000	0.00	4.55	4.55	0.44	–
tree_fr	9	CVE_{cm}	0.1078	0.1117	0.16	0.10	0.26	0.44	1.00
tree_fr	9	IBP	0.0000	0.0000	0.16	0.01	0.16	0.44	1.00
tree_fr	10	CDVE	0.0011	0.0001	0.17	0.41	0.58	0.50	1.00
tree_fr	10	ARIEL	0.0000	0.0000	0.00	6.19	6.19	0.50	–
tree_fr	10	CVE_{cm}	0.1040	0.1297	0.18	0.12	0.30	0.50	1.00
tree_fr	10	IBP	0.0000	0.0000	0.17	0.01	0.17	0.50	1.00

Table 5: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
tree_fr	5	CDVE	0.0005	0.0119	0.08	0.04	0.13	0.31	1.00
tree_fr	5	ARIEL	0.0000	0.0000	0.00	1.45	1.45	0.31	–
tree_fr	5	CVE	0.0000	0.0000	0.10	1.97	2.07	0.31	1.00
tree_fr	5	CVE_{cm}	0.0683	0.0572	0.09	0.02	0.11	0.31	1.00
tree_fr	5	IBP	0.0000	0.0000	0.09	0.00	0.09	0.31	1.00
tree_fr	6	CDVE	0.0009	0.0000	0.10	0.08	0.18	0.28	1.00
tree_fr	6	ARIEL	0.0000	0.0000	0.00	1.86	1.86	0.28	–
tree_fr	6	CVE	0.0000	0.0000	0.12	35.01	35.13	0.28	1.00
tree_fr	6	CVE_{cm}	0.0591	0.0711	0.10	0.04	0.14	0.28	1.00
tree_fr	6	IBP	0.0000	0.0000	0.11	0.00	0.11	0.28	1.00
tree_fr	7	CDVE	0.0133	0.0014	0.12	0.12	0.24	0.33	1.00
tree_fr	7	ARIEL	0.0000	0.0000	0.00	2.46	2.46	0.33	–
tree_fr	7	CVE	0.0000	0.0000	0.12	511.13	511.25	0.33	1.00
tree_fr	7	CVE_{cm}	0.0993	0.0842	0.12	0.05	0.17	0.33	1.00
tree_fr	7	IBP	0.0000	0.0000	0.13	0.00	0.13	0.33	1.00
tree_fr	8	CDVE	0.0013	0.0014	0.14	0.18	0.32	0.39	1.00
tree_fr	8	ARIEL	0.0000	0.0000	0.00	3.41	3.41	0.39	–
tree_fr	8	CVE	0.0000	0.0000	0.14	1006.64	1006.78	0.39	1.00
tree_fr	8	CVE_{cm}	0.1078	0.0804	0.14	0.07	0.21	0.39	1.00
tree_fr	8	IBP	0.0000	0.0000	0.14	0.01	0.15	0.39	1.00
tree_fr	9	CDVE	0.0000	0.0120	0.15	0.31	0.46	0.44	1.00
tree_fr	9	ARIEL	0.0000	0.0000	0.00	4.55	4.55	0.44	–
tree_fr	9	CVE_{cm}	0.1035	0.1138	0.16	0.10	0.26	0.44	1.00
tree_fr	9	IBP	0.0000	0.0000	0.16	0.01	0.16	0.44	1.00
tree_fr	10	CDVE	0.0076	0.0009	0.17	0.41	0.58	0.50	1.00
tree_fr	10	ARIEL	0.0000	0.0000	0.00	6.19	6.19	0.50	–
tree_fr	10	CVE_{cm}	0.0992	0.1246	0.18	0.12	0.30	0.50	1.00
tree_fr	10	IBP	0.0000	0.0000	0.17	0.01	0.17	0.50	1.00

Table 6: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
tree_fr	10	CDVE	0.0005	0.0000	0.18	0.41	0.59	0.64	1.00
tree_fr	10	ARIEL	0.0000	0.0000	0.00	5.37	5.37	0.64	–
tree_fr	10	CVE_{cm}	0.1145	0.1199	0.18	0.11	0.29	0.64	1.00
tree_fr	10	IBP	0.0000	0.0000	0.17	0.01	0.18	0.64	1.00
tree_fr	20	CDVE	0.0011	0.0019	0.35	4.36	4.71	1.43	1.00
tree_fr	20	ARIEL	0.0000	0.0000	0.00	15.80	15.80	1.43	–
tree_fr	20	CVE_{cm}	0.2517	0.2650	0.37	0.77	1.14	1.43	1.00
tree_fr	20	IBP	0.0000	0.0000	0.34	0.02	0.36	1.43	1.00
tree_fr	30	CDVE	0.0036	0.0037	0.50	16.18	16.68	2.58	1.00
tree_fr	30	ARIEL	0.0000	0.0000	0.00	24.13	24.13	2.58	–
tree_fr	30	CVE_{cm}	0.2998	0.3120	0.55	2.18	2.73	2.58	1.00
tree_fr	30	IBP	0.0000	0.0000	0.49	0.03	0.52	2.58	1.00
tree_fr	40	CDVE	0.0017	0.0026	0.66	47.39	48.04	3.90	1.00
tree_fr	40	ARIEL	0.0000	0.0000	0.00	34.21	34.21	3.90	–
tree_fr	40	CVE_{cm}	0.3573	0.3515	0.70	4.72	5.43	3.90	1.00
tree_fr	40	IBP	0.0000	0.0000	0.66	0.04	0.70	3.90	1.00
tree_fr	50	CDVE	0.0031	0.0026	0.83	100.39	101.22	5.95	1.00
tree_fr	50	ARIEL	0.0000	0.0000	0.00	42.41	42.41	5.95	–
tree_fr	50	CVE_{cm}	0.3845	0.3875	0.96	43.09	44.05	5.95	1.00
tree_fr	50	IBP	0.0000	0.0000	0.83	0.05	0.88	5.95	1.00
tree_fr	100	CDVE	0.0037	0.0036	1.67	1487.61	1489.29	16.24	1.00
tree_fr	100	ARIEL	0.0000	0.0000	0.00	89.09	89.09	16.24	–
tree_fr	100	CVE_{cm}	0.4473	0.4401	1.94	259.37	261.31	16.24	1.00
tree_fr	100	IBP	0.0000	0.0000	1.76	0.12	1.88	16.24	1.00

Table 7: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
tree_fr	10	CDVE	0.0026	0.0003	0.18	0.41	0.59	0.64	1.00
tree_fr	10	ARIEL	0.0000	0.0000	0.00	5.37	5.37	0.64	–
tree_fr	10	CVE_{cm}	0.1085	0.1227	0.18	0.11	0.29	0.64	1.00
tree_fr	10	IBP	0.0000	0.0000	0.17	0.01	0.18	0.64	1.00
tree_fr	20	CDVE	0.0083	0.0104	0.35	4.36	4.71	1.43	1.00
tree_fr	20	ARIEL	0.0000	0.0000	0.00	15.80	15.80	1.43	–
tree_fr	20	CVE_{cm}	0.1704	0.1786	0.37	0.77	1.14	1.43	1.00
tree_fr	20	IBP	0.0000	0.0000	0.34	0.02	0.36	1.43	1.00
tree_fr	30	CDVE	0.0195	0.0177	0.50	16.18	16.68	2.58	1.00
tree_fr	30	ARIEL	0.0000	0.0000	0.00	24.13	24.13	2.58	–
tree_fr	30	CVE_{cm}	0.1766	0.1860	0.55	2.18	2.73	2.58	1.00
tree_fr	30	IBP	0.0000	0.0000	0.49	0.03	0.52	2.58	1.00
tree_fr	40	CDVE	0.0134	0.0169	0.66	47.39	48.04	3.90	1.00
tree_fr	40	ARIEL	0.0000	0.0000	0.00	34.21	34.21	3.90	–
tree_fr	40	CVE_{cm}	0.1977	0.1929	0.70	4.72	5.43	3.90	1.00
tree_fr	40	IBP	0.0000	0.0000	0.66	0.04	0.70	3.90	1.00
tree_fr	50	CDVE	0.0164	0.0154	0.83	100.39	101.22	5.95	1.00
tree_fr	50	ARIEL	0.0000	0.0000	0.00	42.41	42.41	5.95	–
tree_fr	50	CVE_{cm}	0.1929	0.1944	0.96	43.09	44.05	5.95	1.00
tree_fr	50	IBP	0.0000	0.0000	0.83	0.05	0.88	5.95	1.00
tree_fr	100	CDVE	0.0189	0.0190	1.67	1487.61	1489.29	16.24	1.00
tree_fr	100	ARIEL	0.0000	0.0000	0.00	89.09	89.09	16.24	–
tree_fr	100	CVE_{cm}	0.1625	0.1711	1.94	259.37	261.31	16.24	1.00
tree_fr	100	IBP	0.0000	0.0000	1.76	0.12	1.88	16.24	1.00

Table 8: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
polytree_fr	5	CDVE	0.0001	0.0005	0.10	0.05	0.15	0.39	1.50
polytree_fr	5	ARIEL	0.0198	0.0096	0.00	20.68	20.68	0.39	–
polytree_fr	5	CVE	0.0000	0.0005	0.11	44.41	44.52	0.39	1.50
polytree_fr	5	CVE_{cm}	0.0545	0.0550	0.11	0.03	0.13	0.39	1.50
polytree_fr	5	IBP	0.0000	0.0005	0.11	0.01	0.11	0.39	1.50
polytree_fr	6	CDVE	0.0039	0.0000	0.12	0.08	0.20	0.44	1.40
polytree_fr	6	ARIEL	0.0010	0.0166	0.00	8.51	8.51	0.44	–
polytree_fr	6	CVE	0.0000	0.0000	0.14	445.58	445.71	0.44	1.40
polytree_fr	6	CVE_{cm}	0.0789	0.0508	0.12	0.04	0.16	0.44	1.40
polytree_fr	6	IBP	0.0000	0.0000	0.12	0.01	0.13	0.44	1.40
polytree_fr	7	CDVE	0.0002	0.0004	0.15	0.15	0.30	0.73	1.70
polytree_fr	7	ARIEL	0.0002	0.0114	0.00	44.04	44.04	0.73	–
polytree_fr	7	CVE	0.0003	0.0000	0.13	849.79	849.92	0.73	1.40
polytree_fr	7	CVE_{cm}	0.0790	0.0845	0.15	0.06	0.22	0.73	1.70
polytree_fr	7	IBP	0.0002	0.0004	0.15	0.01	0.16	0.73	1.70
polytree_fr	8	CDVE	0.0001	0.0017	0.16	0.22	0.38	0.91	1.40
polytree_fr	8	ARIEL	0.0010	0.0139	0.00	3.33	3.33	0.91	–
polytree_fr	8	CVE	0.0000	0.0000	0.15	2213.21	2213.37	0.91	1.00
polytree_fr	8	CVE_{cm}	0.1354	0.1082	0.18	0.09	0.27	0.91	1.40
polytree_fr	8	IBP	0.0000	0.0000	0.17	0.01	0.17	0.91	1.40
polytree_fr	9	CDVE	0.0015	0.0042	0.18	0.32	0.50	0.99	1.30
polytree_fr	9	ARIEL	0.0019	0.0012	0.00	26.20	26.20	0.99	–
polytree_fr	9	CVE_{cm}	0.1264	0.1290	0.19	0.12	0.31	0.99	1.30
polytree_fr	9	IBP	0.0000	0.0000	0.18	0.01	0.19	0.99	1.30
polytree_fr	10	CDVE	0.0014	0.0011	0.19	0.43	0.62	1.11	1.30
polytree_fr	10	ARIEL	0.0046	0.0011	0.00	11.47	11.47	1.11	–
polytree_fr	10	CVE	0.0000	0.0000	0.16	2149.12	2149.28	1.11	1.00
polytree_fr	10	CVE_{cm}	0.1124	0.1220	0.20	0.14	0.33	1.11	1.30
polytree_fr	10	IBP	0.0010	0.0010	0.20	0.01	0.21	1.11	1.30

Table 9: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
polytree_fr	5	CDVE	0.0004	0.0029	0.10	0.05	0.15	0.39	1.50
polytree_fr	5	ARIEL	0.1010	0.0475	0.00	20.68	20.68	0.39	–
polytree_fr	5	CVE	0.0001	0.0029	0.11	44.41	44.52	0.39	1.50
polytree_fr	5	CVE_{cm}	0.0925	0.0898	0.11	0.03	0.13	0.39	1.50
polytree_fr	5	IBP	0.0001	0.0029	0.11	0.01	0.11	0.39	1.50
polytree_fr	6	CDVE	0.0222	0.0000	0.12	0.08	0.20	0.44	1.40
polytree_fr	6	ARIEL	0.0042	0.0702	0.00	8.51	8.51	0.44	–
polytree_fr	6	CVE	0.0000	0.0000	0.14	445.58	445.71	0.44	1.40
polytree_fr	6	CVE_{cm}	0.1063	0.0718	0.12	0.04	0.16	0.44	1.40
polytree_fr	6	IBP	0.0000	0.0000	0.12	0.01	0.13	0.44	1.40
polytree_fr	7	CDVE	0.0012	0.0032	0.15	0.15	0.30	0.73	1.70
polytree_fr	7	ARIEL	0.0009	0.0560	0.00	44.04	44.04	0.73	–
polytree_fr	7	CVE	0.0017	0.0001	0.13	849.79	849.92	0.73	1.40
polytree_fr	7	CVE_{cm}	0.0922	0.0982	0.15	0.06	0.22	0.73	1.70
polytree_fr	7	IBP	0.0012	0.0032	0.15	0.01	0.16	0.73	1.70
polytree_fr	8	CDVE	0.0011	0.0150	0.16	0.22	0.38	0.91	1.40
polytree_fr	8	ARIEL	0.0064	0.0608	0.00	3.33	3.33	0.91	–
polytree_fr	8	CVE	0.0000	0.0000	0.15	2213.21	2213.37	0.91	1.00
polytree_fr	8	CVE_{cm}	0.1220	0.1051	0.18	0.09	0.27	0.91	1.40
polytree_fr	8	IBP	0.0000	0.0000	0.17	0.01	0.17	0.91	1.40
polytree_fr	9	CDVE	0.0067	0.0234	0.18	0.32	0.50	0.99	1.30
polytree_fr	9	ARIEL	0.0126	0.0088	0.00	26.20	26.20	0.99	–
polytree_fr	9	CVE_{cm}	0.1301	0.1497	0.19	0.12	0.31	0.99	1.30
polytree_fr	9	IBP	0.0000	0.0001	0.18	0.01	0.19	0.99	1.30
polytree_fr	10	CDVE	0.0093	0.0094	0.19	0.43	0.62	1.11	1.30
polytree_fr	10	ARIEL	0.0331	0.0088	0.00	11.47	11.47	1.11	–
polytree_fr	10	CVE	0.0000	0.0000	0.16	2149.12	2149.28	1.11	1.00
polytree_fr	10	CVE_{cm}	0.1343	0.1404	0.20	0.14	0.33	1.11	1.30
polytree_fr	10	IBP	0.0092	0.0094	0.20	0.01	0.21	1.11	1.30

Table 10: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
polytree_fr	10	CDVE	0.0010	0.0050	0.23	0.51	0.73	13.77	1.50
polytree_fr	10	ARIEL	0.0100	0.0038	0.00	3.66	3.66	13.77	–
polytree_fr	10	CVE_{cm}	0.1333	0.1551	0.22	0.20	0.42	13.77	1.50
polytree_fr	10	IBP	0.0003	0.0013	0.23	0.01	0.24	13.77	1.50
polytree_fr	20	CDVE	0.0031	0.0031	0.45	5.35	5.80	3.21	1.60
polytree_fr	20	ARIEL	0.0095	0.0143	0.00	57.07	57.07	3.21	–
polytree_fr	20	CVE_{cm}	0.2740	0.2555	0.44	1.73	2.17	3.21	1.60
polytree_fr	20	IBP	0.0012	0.0006	0.43	0.03	0.47	3.21	1.60
polytree_fr	30	CDVE	0.0012	0.0023	0.54	18.84	19.38	5.22	1.10
polytree_fr	30	ARIEL	0.0000	0.0000	0.00	14.78	14.78	5.22	–
polytree_fr	30	CVE_{cm}	0.3039	0.3023	0.58	5.12	5.71	5.22	1.10
polytree_fr	30	IBP	0.0001	0.0003	0.56	0.04	0.59	5.22	1.10
polytree_fr	40	CDVE	0.0051	0.0052	0.76	52.22	52.98	12.10	1.40
polytree_fr	40	ARIEL	0.0007	0.0009	0.00	20.76	20.76	12.10	–
polytree_fr	40	CVE_{cm}	0.3548	0.3560	0.78	184.60	185.37	12.10	1.40
polytree_fr	40	IBP	0.0003	0.0002	0.80	0.06	0.86	12.10	1.40
polytree_fr	50	CDVE	0.0047	0.0051	1.10	124.13	125.23	38.34	1.50
polytree_fr	50	ARIEL	0.0008	0.0012	0.00	31.07	31.07	38.34	–
polytree_fr	50	CVE_{cm}	0.3992	0.3913	0.93	21.67	22.60	38.34	1.50
polytree_fr	50	IBP	0.0005	0.0003	0.99	0.07	1.06	38.34	1.50
polytree_fr	100	CDVE	0.0056	0.0043	1.95	1545.82	1547.77	68.64	1.40
polytree_fr	100	ARIEL	0.0005	0.0003	0.00	58.08	58.08	68.64	–
polytree_fr	100	CVE_{cm}	0.4359	0.4408	1.86	272.65	274.51	68.64	1.33
polytree_fr	100	IBP	0.0002	0.0002	1.93	0.15	2.08	68.64	1.40

Table 11: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
polytree_fr	10	CDVE	0.0041	0.0251	0.23	0.51	0.73	13.77	1.50
polytree_fr	10	ARIEL	0.0598	0.0200	0.00	3.66	3.66	13.77	–
polytree_fr	10	CVE_{cm}	0.1278	0.1507	0.22	0.20	0.42	13.77	1.50
polytree_fr	10	IBP	0.0013	0.0105	0.23	0.01	0.24	13.77	1.50
polytree_fr	20	CDVE	0.0127	0.0143	0.45	5.35	5.80	3.21	1.60
polytree_fr	20	ARIEL	0.0714	0.0862	0.00	57.07	57.07	3.21	–
polytree_fr	20	CVE_{cm}	0.1862	0.1804	0.44	1.73	2.17	3.21	1.60
polytree_fr	20	IBP	0.0060	0.0043	0.43	0.03	0.47	3.21	1.60
polytree_fr	30	CDVE	0.0076	0.0161	0.54	18.84	19.38	5.22	1.10
polytree_fr	30	ARIEL	0.0000	0.0000	0.00	14.78	14.78	5.22	–
polytree_fr	30	CVE_{cm}	0.1829	0.1783	0.58	5.12	5.71	5.22	1.10
polytree_fr	30	IBP	0.0014	0.0027	0.56	0.04	0.59	5.22	1.10
polytree_fr	40	CDVE	0.0218	0.0244	0.76	52.22	52.98	12.10	1.40
polytree_fr	40	ARIEL	0.0052	0.0162	0.00	20.76	20.76	12.10	–
polytree_fr	40	CVE_{cm}	0.1969	0.1938	0.78	184.60	185.37	12.10	1.40
polytree_fr	40	IBP	0.0019	0.0019	0.80	0.06	0.86	12.10	1.40
polytree_fr	50	CDVE	0.0210	0.0242	1.10	124.13	125.23	38.34	1.50
polytree_fr	50	ARIEL	0.0160	0.0190	0.00	31.07	31.07	38.34	–
polytree_fr	50	CVE_{cm}	0.1852	0.1840	0.93	21.67	22.60	38.34	1.50
polytree_fr	50	IBP	0.0049	0.0022	0.99	0.07	1.06	38.34	1.50
polytree_fr	100	CDVE	0.0235	0.0195	1.95	1545.82	1547.77	68.64	1.40
polytree_fr	100	ARIEL	0.0100	0.0060	0.00	58.08	58.08	68.64	–
polytree_fr	100	CVE_{cm}	0.1651	0.1624	1.86	272.65	274.51	68.64	1.33
polytree_fr	100	IBP	0.0025	0.0025	1.93	0.15	2.08	68.64	1.40

Table 12: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
ktree_fr_k3	5	CDVE	0.0006	0.0042	0.33	0.14	0.48	0.29	3.00
ktree_fr_k3	5	ARIEL	0.0015	0.0028	0.00	2.15	2.15	0.29	–
ktree_fr_k3	5	CVE_{cm}	0.1437	0.1374	0.33	1.38	1.71	0.29	3.00
ktree_fr_k3	5	IBP	0.0030	0.0027	0.33	0.41	0.74	0.29	3.00
ktree_fr_k3	6	CDVE	0.0006	0.0012	0.45	0.29	0.74	0.36	3.00
ktree_fr_k3	6	ARIEL	0.0029	0.0011	0.00	3.11	3.11	0.36	–
ktree_fr_k3	6	CVE_{cm}	0.1620	0.1519	0.45	3.99	4.43	0.36	3.00
ktree_fr_k3	6	IBP	0.0030	0.0008	0.45	0.69	1.14	0.36	3.00
ktree_fr_k3	7	CDVE	0.0027	0.0005	0.57	0.59	1.17	0.42	3.00
ktree_fr_k3	7	ARIEL	0.0015	0.0002	0.00	3.72	3.72	0.42	–
ktree_fr_k3	7	CVE_{cm}	0.2231	0.2155	0.60	7.68	8.27	0.42	3.00
ktree_fr_k3	7	IBP	0.0018	0.0005	0.58	0.92	1.50	0.42	3.00
ktree_fr_k3	8	CDVE	0.0008	0.0044	0.64	0.90	1.54	0.51	3.00
ktree_fr_k3	8	ARIEL	0.0028	0.0020	0.00	4.72	4.72	0.51	–
ktree_fr_k3	8	CVE_{cm}	0.2484	0.2158	0.69	22.88	23.57	0.51	3.00
ktree_fr_k3	8	IBP	0.0018	0.0011	0.71	1.23	1.94	0.51	3.00
ktree_fr_k3	9	CDVE	0.0011	0.0009	0.78	1.39	2.17	0.60	3.00
ktree_fr_k3	9	ARIEL	0.0019	0.0017	0.00	5.32	5.32	0.60	–
ktree_fr_k3	9	CVE_{cm}	0.2851	0.2768	0.82	26.13	26.95	0.60	3.00
ktree_fr_k3	9	IBP	0.0015	0.0020	0.76	1.38	2.13	0.60	3.00
ktree_fr_k3	10	CDVE	0.0002	0.0015	0.95	2.10	3.05	0.68	3.00
ktree_fr_k3	10	ARIEL	0.0027	0.0011	0.00	5.98	5.98	0.68	–
ktree_fr_k3	10	CVE_{cm}	0.2740	0.2723	0.96	45.69	46.65	0.68	3.00
ktree_fr_k3	10	IBP	0.0013	0.0009	0.97	1.74	2.70	0.68	3.00

Table 13: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
ktree_fr_k3	5	CDVE	0.0021	0.0206	0.33	0.14	0.48	0.29	3.00
ktree_fr_k3	5	ARIEL	0.0105	0.0171	0.00	2.15	2.15	0.29	–
ktree_fr_k3	5	CVE_{cm}	0.1324	0.1330	0.33	1.38	1.71	0.29	3.00
ktree_fr_k3	5	IBP	0.0125	0.0122	0.33	0.41	0.74	0.29	3.00
ktree_fr_k3	6	CDVE	0.0034	0.0094	0.45	0.29	0.74	0.36	3.00
ktree_fr_k3	6	ARIEL	0.0137	0.0071	0.00	3.11	3.11	0.36	–
ktree_fr_k3	6	CVE_{cm}	0.1399	0.1394	0.45	3.99	4.43	0.36	3.00
ktree_fr_k3	6	IBP	0.0119	0.0037	0.45	0.69	1.14	0.36	3.00
ktree_fr_k3	7	CDVE	0.0174	0.0022	0.57	0.59	1.17	0.42	3.00
ktree_fr_k3	7	ARIEL	0.0063	0.0013	0.00	3.72	3.72	0.42	–
ktree_fr_k3	7	CVE_{cm}	0.1767	0.1585	0.60	7.68	8.27	0.42	3.00
ktree_fr_k3	7	IBP	0.0069	0.0027	0.58	0.92	1.50	0.42	3.00
ktree_fr_k3	8	CDVE	0.0050	0.0202	0.64	0.90	1.54	0.51	3.00
ktree_fr_k3	8	ARIEL	0.0144	0.0114	0.00	4.72	4.72	0.51	–
ktree_fr_k3	8	CVE_{cm}	0.1728	0.1580	0.69	22.88	23.57	0.51	3.00
ktree_fr_k3	8	IBP	0.0071	0.0041	0.71	1.23	1.94	0.51	3.00
ktree_fr_k3	9	CDVE	0.0067	0.0057	0.78	1.39	2.17	0.60	3.00
ktree_fr_k3	9	ARIEL	0.0131	0.0110	0.00	5.32	5.32	0.60	–
ktree_fr_k3	9	CVE_{cm}	0.1681	0.1694	0.82	26.13	26.95	0.60	3.00
ktree_fr_k3	9	IBP	0.0068	0.0089	0.76	1.38	2.13	0.60	3.00
ktree_fr_k3	10	CDVE	0.0012	0.0127	0.95	2.10	3.05	0.68	3.00
ktree_fr_k3	10	ARIEL	0.0129	0.0052	0.00	5.98	5.98	0.68	–
ktree_fr_k3	10	CVE_{cm}	0.1808	0.1783	0.96	45.69	46.65	0.68	3.00
ktree_fr_k3	10	IBP	0.0054	0.0039	0.97	1.74	2.70	0.68	3.00

Table 14: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
ktree_fr_k3	10	CDVE	0.0007	0.0023	0.40	2.14	2.55	0.65	3.00
ktree_fr_k3	10	ARIEL	0.0060	0.0017	0.00	5.42	5.42	0.65	–
ktree_fr_k3	10	CVE_{cm}	0.2993	0.2810	0.40	46.11	46.51	0.65	3.00
ktree_fr_k3	10	IBP	0.0032	0.0021	0.40	1.74	2.13	0.65	3.00
ktree_fr_k3	20	CDVE	0.0013	0.0017	0.85	26.67	27.52	1.57	3.00
ktree_fr_k3	20	ARIEL	0.0006	0.0014	0.00	12.26	12.26	1.57	–
ktree_fr_k3	20	CVE_{cm}	0.4017	0.4124	0.81	959.08	959.88	1.57	3.00
ktree_fr_k3	20	IBP	0.0007	0.0009	0.83	3.62	4.45	1.57	3.00
ktree_fr_k3	30	CDVE	0.0013	0.0011	1.28	116.25	117.53	2.91	3.00
ktree_fr_k3	30	ARIEL	0.0009	0.0001	0.00	19.01	19.01	2.91	–
ktree_fr_k3	30	IBP	0.0009	0.0004	1.26	6.38	7.64	2.91	3.00
ktree_fr_k3	40	CDVE	0.0027	0.0020	1.71	367.85	369.55	4.80	3.00
ktree_fr_k3	40	ARIEL	0.0003	0.0004	0.00	25.41	25.41	4.80	–
ktree_fr_k3	40	IBP	0.0003	0.0003	1.68	8.70	10.38	4.80	3.00
ktree_fr_k3	50	CDVE	0.0015	0.0022	3.68	845.10	848.78	6.89	3.00
ktree_fr_k3	50	ARIEL	0.0001	0.0004	0.00	31.32	31.32	6.89	–
ktree_fr_k3	50	IBP	0.0004	0.0003	3.64	11.44	15.08	6.89	3.00
ktree_fr_k3	100	ARIEL	0.0003	0.0007	0.00	64.03	64.03	22.71	–
ktree_fr_k3	100	IBP	0.0002	0.0005	10.55	25.34	35.89	22.71	3.00

Table 15: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
ktree_fr_k3	10	CDVE	0.0031	0.0095	0.40	2.14	2.55	0.65	3.00
ktree_fr_k3	10	ARIEL	0.0242	0.0128	0.00	5.42	5.42	0.65	–
ktree_fr_k3	10	CVE_{cm}	0.1690	0.1673	0.40	46.11	46.51	0.65	3.00
ktree_fr_k3	10	IBP	0.0128	0.0113	0.40	1.74	2.13	0.65	3.00
ktree_fr_k3	20	CDVE	0.0122	0.0111	0.85	26.67	27.52	1.57	3.00
ktree_fr_k3	20	ARIEL	0.0041	0.0116	0.00	12.26	12.26	1.57	–
ktree_fr_k3	20	CVE_{cm}	0.1572	0.1476	0.81	959.08	959.88	1.57	3.00
ktree_fr_k3	20	IBP	0.0038	0.0068	0.83	3.62	4.45	1.57	3.00
ktree_fr_k3	30	CDVE	0.0085	0.0075	1.28	116.25	117.53	2.91	3.00
ktree_fr_k3	30	ARIEL	0.0055	0.0011	0.00	19.01	19.01	2.91	–
ktree_fr_k3	30	IBP	0.0055	0.0032	1.26	6.38	7.64	2.91	3.00
ktree_fr_k3	40	CDVE	0.0161	0.0119	1.71	367.85	369.55	4.80	3.00
ktree_fr_k3	40	ARIEL	0.0034	0.0065	0.00	25.41	25.41	4.80	–
ktree_fr_k3	40	IBP	0.0029	0.0036	1.68	8.70	10.38	4.80	3.00
ktree_fr_k3	50	CDVE	0.0087	0.0133	3.68	845.10	848.78	6.89	3.00
ktree_fr_k3	50	ARIEL	0.0010	0.0077	0.00	31.32	31.32	6.89	–
ktree_fr_k3	50	IBP	0.0042	0.0030	3.64	11.44	15.08	6.89	3.00
ktree_fr_k3	100	ARIEL	0.0037	0.0090	0.00	64.03	64.03	22.71	–
ktree_fr_k3	100	IBP	0.0024	0.0048	10.55	25.34	35.89	22.71	3.00

Table 16: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
dag_fr_tw3	5	CDVE	0.0039	0.0020	0.13	0.06	0.20	0.42	2.00
dag_fr_tw3	5	ARIEL	0.0090	0.0123	0.00	3.38	3.38	0.42	–
dag_fr_tw3	5	CVE	0.0006	0.0017	0.13	250.50	250.64	0.42	2.00
dag_fr_tw3	5	CVE_{cm}	0.0619	0.0578	0.13	0.03	0.16	0.42	2.00
dag_fr_tw3	5	IBP	0.0021	0.0020	0.13	0.01	0.13	0.42	2.00
dag_fr_tw3	6	CDVE	0.0031	0.0037	0.14	0.09	0.23	0.49	1.90
dag_fr_tw3	6	ARIEL	0.0008	0.0012	0.00	7.96	7.96	0.49	–
dag_fr_tw3	6	CVE	0.0031	0.0019	0.15	267.68	267.84	0.49	1.90
dag_fr_tw3	6	CVE_{cm}	0.0517	0.0569	0.14	0.04	0.18	0.49	1.90
dag_fr_tw3	6	IBP	0.0033	0.0020	0.13	0.01	0.14	0.49	1.90
dag_fr_tw3	7	CDVE	0.0076	0.0045	0.17	0.15	0.32	0.79	1.80
dag_fr_tw3	7	ARIEL	0.0000	0.0000	0.00	1.92	1.92	0.79	–
dag_fr_tw3	7	CVE	0.0031	0.0020	0.13	282.36	282.49	0.79	1.50
dag_fr_tw3	7	CVE_{cm}	0.0677	0.0778	0.17	0.07	0.23	0.79	1.80
dag_fr_tw3	7	IBP	0.0073	0.0041	0.16	0.01	0.17	0.79	1.80
dag_fr_tw3	8	CDVE	0.0015	0.0035	0.24	0.33	0.57	1.10	2.20
dag_fr_tw3	8	ARIEL	0.0029	0.0007	0.00	3.26	3.26	1.10	–
dag_fr_tw3	8	CVE_{cm}	0.1535	0.1293	0.24	0.14	0.39	1.10	2.20
dag_fr_tw3	8	IBP	0.0032	0.0021	0.24	0.02	0.26	1.10	2.20
dag_fr_tw3	9	CDVE	0.0013	0.0019	0.28	0.49	0.77	1.23	2.20
dag_fr_tw3	9	ARIEL	0.0089	0.0114	0.00	3.50	3.50	1.23	–
dag_fr_tw3	9	CVE_{cm}	0.1521	0.1462	0.27	0.22	0.49	1.23	2.20
dag_fr_tw3	9	IBP	0.0016	0.0004	0.28	0.02	0.30	1.23	2.20
dag_fr_tw3	10	CDVE	0.0021	0.0060	0.31	0.73	1.05	1.62	2.20
dag_fr_tw3	10	ARIEL	0.0029	0.0029	0.00	3.92	3.92	1.62	–
dag_fr_tw3	10	CVE	0.0000	0.0000	0.20	104.02	104.21	1.62	2.00
dag_fr_tw3	10	CVE_{cm}	0.1930	0.1972	0.30	0.30	0.60	1.62	2.20
dag_fr_tw3	10	IBP	0.0044	0.0058	0.31	0.03	0.33	1.62	2.20

Table 17: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
dag_fr_tw3	5	CDVE	0.0238	0.0085	0.13	0.06	0.20	0.42	2.00
dag_fr_tw3	5	ARIEL	0.0258	0.0563	0.00	3.38	3.38	0.42	–
dag_fr_tw3	5	CVE	0.0041	0.0085	0.13	250.50	250.64	0.42	2.00
dag_fr_tw3	5	CVE_{cm}	0.0697	0.0798	0.13	0.03	0.16	0.42	2.00
dag_fr_tw3	5	IBP	0.0074	0.0083	0.13	0.01	0.13	0.42	2.00
dag_fr_tw3	6	CDVE	0.0168	0.0163	0.14	0.09	0.23	0.49	1.90
dag_fr_tw3	6	ARIEL	0.0047	0.0067	0.00	7.96	7.96	0.49	–
dag_fr_tw3	6	CVE	0.0168	0.0094	0.15	267.68	267.84	0.49	1.90
dag_fr_tw3	6	CVE_{cm}	0.0878	0.0830	0.14	0.04	0.18	0.49	1.90
dag_fr_tw3	6	IBP	0.0168	0.0092	0.13	0.01	0.14	0.49	1.90
dag_fr_tw3	7	CDVE	0.0205	0.0150	0.17	0.15	0.32	0.79	1.80
dag_fr_tw3	7	ARIEL	0.0000	0.0001	0.00	1.92	1.92	0.79	–
dag_fr_tw3	7	CVE	0.0160	0.0102	0.13	282.36	282.49	0.79	1.50
dag_fr_tw3	7	CVE_{cm}	0.1090	0.1145	0.17	0.07	0.23	0.79	1.80
dag_fr_tw3	7	IBP	0.0206	0.0143	0.16	0.01	0.17	0.79	1.80
dag_fr_tw3	8	CDVE	0.0061	0.0165	0.24	0.33	0.57	1.10	2.20
dag_fr_tw3	8	ARIEL	0.0136	0.0031	0.00	3.26	3.26	1.10	–
dag_fr_tw3	8	CVE_{cm}	0.1487	0.1343	0.24	0.14	0.39	1.10	2.20
dag_fr_tw3	8	IBP	0.0091	0.0096	0.24	0.02	0.26	1.10	2.20
dag_fr_tw3	9	CDVE	0.0109	0.0163	0.28	0.49	0.77	1.23	2.20
dag_fr_tw3	9	ARIEL	0.0834	0.0838	0.00	3.50	3.50	1.23	–
dag_fr_tw3	9	CVE_{cm}	0.1428	0.1327	0.27	0.22	0.49	1.23	2.20
dag_fr_tw3	9	IBP	0.0110	0.0018	0.28	0.02	0.30	1.23	2.20
dag_fr_tw3	10	CDVE	0.0101	0.0223	0.31	0.73	1.05	1.62	2.20
dag_fr_tw3	10	ARIEL	0.0211	0.0170	0.00	3.92	3.92	1.62	–
dag_fr_tw3	10	CVE	0.0000	0.0000	0.20	104.02	104.21	1.62	2.00
dag_fr_tw3	10	CVE_{cm}	0.1640	0.1650	0.30	0.30	0.60	1.62	2.20
dag_fr_tw3	10	IBP	0.0142	0.0190	0.31	0.03	0.33	1.62	2.20

Table 18: Mean error metrics vs. CJT reference

Type	n	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
dag_fr_tw3	10	CDVE	0.0012	0.0020	0.32	0.76	1.07	1.53	2.40
dag_fr_tw3	10	ARIEL	0.0013	0.0038	0.00	18.64	18.64	1.53	–
dag_fr_tw3	10	CVE_{cm}	0.1992	0.2057	0.32	0.41	0.72	1.53	2.40
dag_fr_tw3	10	IBP	0.0019	0.0030	0.31	0.03	0.34	1.53	2.40
dag_fr_tw3	20	CDVE	0.0039	0.0023	0.67	8.53	9.20	4.17	2.60
dag_fr_tw3	20	ARIEL	0.0040	0.0026	0.00	9.46	9.46	4.17	–
dag_fr_tw3	20	CVE_{cm}	0.3072	0.3114	0.65	23.43	24.08	4.17	2.60
dag_fr_tw3	20	IBP	0.0037	0.0026	0.66	0.07	0.73	4.17	2.60
dag_fr_tw3	30	CDVE	0.0025	0.0023	1.03	36.88	37.90	10.25	2.80
dag_fr_tw3	30	ARIEL	0.0013	0.0001	0.00	13.20	13.20	10.25	–
dag_fr_tw3	30	CVE_{cm}	0.3439	0.3465	0.95	284.01	284.96	10.25	2.75
dag_fr_tw3	30	IBP	0.0021	0.0018	0.94	0.09	1.03	10.25	2.80
dag_fr_tw3	40	CDVE	0.0039	0.0036	1.43	110.19	111.62	14.22	3.00
dag_fr_tw3	40	ARIEL	0.0009	0.0034	0.00	32.43	32.43	14.22	–
dag_fr_tw3	40	CVE_{cm}	0.3173	0.3677	1.09	1135.19	1136.29	14.22	3.00
dag_fr_tw3	40	IBP	0.0019	0.0018	1.35	0.17	1.51	14.22	3.00
dag_fr_tw3	50	CDVE	0.0037	0.0032	1.65	238.85	240.50	23.21	3.00
dag_fr_tw3	50	ARIEL	0.0011	0.0008	0.00	24.86	24.86	23.21	–
dag_fr_tw3	50	CVE_{cm}	0.3756	0.3915	1.54	194.40	195.94	23.21	3.00
dag_fr_tw3	50	IBP	0.0027	0.0020	1.67	0.20	1.87	23.21	3.00
dag_fr_tw3	100	CDVE	0.0030	0.0024	3.00	3278.99	3281.98	80.21	3.00
dag_fr_tw3	100	ARIEL	0.0013	0.0010	0.00	56.21	56.21	80.21	–
dag_fr_tw3	100	IBP	0.0013	0.0012	3.34	0.40	3.75	80.21	3.00

Table 19: Standard deviation of error metrics vs. CJT reference

Type	n	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
dag_fr_tw3	10	CDVE	0.0060	0.0079	0.32	0.76	1.07	1.53	2.40
dag_fr_tw3	10	ARIEL	0.0070	0.0196	0.00	18.64	18.64	1.53	–
dag_fr_tw3	10	CVE_{cm}	0.1675	0.1789	0.32	0.41	0.72	1.53	2.40
dag_fr_tw3	10	IBP	0.0066	0.0099	0.31	0.03	0.34	1.53	2.40
dag_fr_tw3	20	CDVE	0.0155	0.0093	0.67	8.53	9.20	4.17	2.60
dag_fr_tw3	20	ARIEL	0.0180	0.0147	0.00	9.46	9.46	4.17	–
dag_fr_tw3	20	CVE_{cm}	0.1853	0.1833	0.65	23.43	24.08	4.17	2.60
dag_fr_tw3	20	IBP	0.0115	0.0099	0.66	0.07	0.73	4.17	2.60
dag_fr_tw3	30	CDVE	0.0092	0.0122	1.03	36.88	37.90	10.25	2.80
dag_fr_tw3	30	ARIEL	0.0118	0.0022	0.00	13.20	13.20	10.25	–
dag_fr_tw3	30	CVE_{cm}	0.1779	0.1827	0.95	284.01	284.96	10.25	2.75
dag_fr_tw3	30	IBP	0.0085	0.0100	0.94	0.09	1.03	10.25	2.80
dag_fr_tw3	40	CDVE	0.0148	0.0189	1.43	110.19	111.62	14.22	3.00
dag_fr_tw3	40	ARIEL	0.0046	0.0312	0.00	32.43	32.43	14.22	–
dag_fr_tw3	40	CVE_{cm}	0.1591	0.2025	1.09	1135.19	1136.29	14.22	3.00
dag_fr_tw3	40	IBP	0.0081	0.0073	1.35	0.17	1.51	14.22	3.00
dag_fr_tw3	50	CDVE	0.0173	0.0148	1.65	238.85	240.50	23.21	3.00
dag_fr_tw3	50	ARIEL	0.0122	0.0063	0.00	24.86	24.86	23.21	–
dag_fr_tw3	50	CVE_{cm}	0.1807	0.1878	1.54	194.40	195.94	23.21	3.00
dag_fr_tw3	50	IBP	0.0124	0.0080	1.67	0.20	1.87	23.21	3.00
dag_fr_tw3	100	CDVE	0.0167	0.0142	3.00	3278.99	3281.98	80.21	3.00
dag_fr_tw3	100	ARIEL	0.0096	0.0083	0.00	56.21	56.21	80.21	–
dag_fr_tw3	100	IBP	0.0062	0.0064	3.34	0.40	3.75	80.21	3.00

I.3 REALISTIC LCNS

The two tables below give the complete per-instance results for the ten Bayesian-network-derived LCNs and the sixteen Supreme Court chain-graph LCNs, including the nine dense `supreme_court` instances omitted from Table 1. On those nine, only ARIEL and the CredalJT reference complete: with n=10 atoms but 345–1009 sentences, the size of the constraint system rather than the treewidth is the binding limit, and IBP, CDVE and all CVE variants exhaust the budget. One instance in the suite, `benchmark_scaling_directions_n24_2`, is a repaired variant of the original benchmark: in the chain graph the four reviewers of a stage are mutually independent, so the probability of their conjunction is forced to the product of their marginals, which the original's `component` sentences contradicted; the three incompatible sentences were dropped and the repaired instance is consistent, verified by a junction-tree feasibility check.

Table 20: Mean error metrics vs. CJT reference

Instance	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
alarm_consistent.lcn	CDVE	0.1008	0.0146	0.10	0.59	0.69	0.72	2.00
alarm_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.92	3.92	0.72	–
alarm_consistent.lcn	CVE_{cm}	0.0866	0.0670	0.10	0.14	0.24	0.72	2.00
alarm_consistent.lcn	IBP	0.1008	0.0149	0.10	0.03	0.13	0.72	2.00
asia_consistent.lcn	CDVE	0.0033	0.0073	0.08	0.24	0.32	0.45	2.00
asia_consistent.lcn	ARIEL	0.0000	0.0000	0.00	2.86	2.86	0.45	–
asia_consistent.lcn	CVE_{cm}	0.0323	0.0291	0.08	0.08	0.16	0.45	2.00
asia_consistent.lcn	IBP	0.0035	0.0063	0.08	0.01	0.10	0.45	2.00
cancer_consistent.lcn	CDVE	0.0653	0.0272	0.06	0.06	0.12	0.37	2.00
cancer_consistent.lcn	ARIEL	0.0013	0.0139	0.00	2.10	2.10	0.37	–
cancer_consistent.lcn	CVE	0.0653	0.0272	0.06	42.46	42.52	0.37	2.00
cancer_consistent.lcn	CVE_{cm}	0.0491	0.0267	0.06	0.03	0.09	0.37	2.00
cancer_consistent.lcn	IBP	0.0639	0.0392	0.06	0.01	0.07	0.37	2.00
credit_consistent.lcn	CDVE	0.2570	0.0000	0.11	0.58	0.69	1.15	3.00
credit_consistent.lcn	ARIEL	0.0496	0.0000	0.00	3.18	3.18	1.15	–
credit_consistent.lcn	CVE_{cm}	0.1079	0.1650	0.11	1.62	1.73	1.15	3.00
credit_consistent.lcn	IBP	0.2570	0.0000	0.11	0.13	0.24	1.15	3.00
earthquake_consistent.lcn	CDVE	0.0041	0.0000	0.05	0.05	0.10	0.53	2.00
earthquake_consistent.lcn	ARIEL	0.0002	0.0000	0.00	1.88	1.88	0.53	–
earthquake_consistent.lcn	CVE	0.0041	0.0000	0.05	7.63	7.68	0.53	2.00
earthquake_consistent.lcn	CVE_{cm}	0.0019	0.0036	0.05	0.03	0.08	0.53	2.00
earthquake_consistent.lcn	IBP	0.0041	0.0000	0.05	0.01	0.06	0.53	2.00
engine_consistent.lcn	CDVE	0.0591	0.0104	0.10	0.43	0.53	0.53	2.00
engine_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.25	3.25	0.53	–
engine_consistent.lcn	CVE_{cm}	0.1234	0.0839	0.10	0.32	0.42	0.53	2.00
engine_consistent.lcn	IBP	0.0591	0.0104	0.10	0.01	0.12	0.53	2.00
hepatitis_consistent.lcn	CDVE	0.0703	0.0851	0.24	0.90	1.15	0.56	3.00
hepatitis_consistent.lcn	ARIEL	0.0084	0.0075	0.00	3.91	3.91	0.56	–
hepatitis_consistent.lcn	CVE_{cm}	0.1137	0.1838	0.24	16.29	16.53	0.56	3.00
hepatitis_consistent.lcn	IBP	0.0686	0.0797	0.24	0.50	0.75	0.56	3.00
suicide_consistent.lcn	CDVE	0.0849	0.0302	0.10	0.52	0.62	0.87	2.00
suicide_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.74	3.74	0.87	–
suicide_consistent.lcn	CVE_{cm}	0.0970	0.0513	0.10	0.13	0.23	0.87	2.00
suicide_consistent.lcn	IBP	0.0849	0.0302	0.10	0.02	0.12	0.87	2.00
tank_consistent.lcn	CDVE	0.1149	0.0751	0.30	0.87	1.18	0.77	3.00
tank_consistent.lcn	ARIEL	0.0048	0.0034	0.00	5.02	5.02	0.77	–
tank_consistent.lcn	CVE_{cm}	0.0653	0.0802	0.31	0.19	0.50	0.77	3.00
tank_consistent.lcn	IBP	0.1163	0.0739	0.30	0.02	0.32	0.77	3.00
toy_consistent.lcn	CDVE	0.0891	0.0745	0.09	0.03	0.12	0.16	2.00
toy_consistent.lcn	ARIEL	0.0000	0.0000	0.00	1.60	1.60	0.16	–
toy_consistent.lcn	CVE_{cm}	0.0742	0.0294	0.09	0.01	0.10	0.16	2.00
toy_consistent.lcn	IBP	0.0891	0.0745	0.09	0.01	0.09	0.16	2.00

Table 21: Standard deviation of error metrics vs. CJT reference

Instance	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
alarm_consistent.lcn	CDVE	0.1476	0.0203	0.10	0.59	0.69	0.72	2.00
alarm_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.92	3.92	0.72	–
alarm_consistent.lcn	CVE_{cm}	0.1279	0.0854	0.10	0.14	0.24	0.72	2.00
alarm_consistent.lcn	IBP	0.1476	0.0208	0.10	0.03	0.13	0.72	2.00
asia_consistent.lcn	CDVE	0.0043	0.0172	0.08	0.24	0.32	0.45	2.00
asia_consistent.lcn	ARIEL	0.0000	0.0000	0.00	2.86	2.86	0.45	–
asia_consistent.lcn	CVE_{cm}	0.0308	0.0359	0.08	0.08	0.16	0.45	2.00
asia_consistent.lcn	IBP	0.0047	0.0144	0.08	0.01	0.10	0.45	2.00
cancer_consistent.lcn	CDVE	0.0374	0.0320	0.06	0.06	0.12	0.37	2.00
cancer_consistent.lcn	ARIEL	0.0026	0.0278	0.00	2.10	2.10	0.37	–
cancer_consistent.lcn	CVE	0.0374	0.0320	0.06	42.46	42.52	0.37	2.00
cancer_consistent.lcn	CVE_{cm}	0.0397	0.0233	0.06	0.03	0.09	0.37	2.00
cancer_consistent.lcn	IBP	0.0369	0.0323	0.06	0.01	0.07	0.37	2.00
credit_consistent.lcn	CDVE	0.2149	0.0000	0.11	0.58	0.69	1.15	3.00
credit_consistent.lcn	ARIEL	0.0414	0.0000	0.00	3.18	3.18	1.15	–
credit_consistent.lcn	CVE_{cm}	0.1401	0.1170	0.11	1.62	1.73	1.15	3.00
credit_consistent.lcn	IBP	0.2149	0.0000	0.11	0.13	0.24	1.15	3.00
earthquake_consistent.lcn	CDVE	0.0035	0.0000	0.05	0.05	0.10	0.53	2.00
earthquake_consistent.lcn	ARIEL	0.0003	0.0000	0.00	1.88	1.88	0.53	–
earthquake_consistent.lcn	CVE	0.0035	0.0000	0.05	7.63	7.68	0.53	2.00
earthquake_consistent.lcn	CVE_{cm}	0.0029	0.0043	0.05	0.03	0.08	0.53	2.00
earthquake_consistent.lcn	IBP	0.0035	0.0000	0.05	0.01	0.06	0.53	2.00
engine_consistent.lcn	CDVE	0.1131	0.0131	0.10	0.43	0.53	0.53	2.00
engine_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.25	3.25	0.53	–
engine_consistent.lcn	CVE_{cm}	0.1318	0.0677	0.10	0.32	0.42	0.53	2.00
engine_consistent.lcn	IBP	0.1131	0.0131	0.10	0.01	0.12	0.53	2.00
hepatitis_consistent.lcn	CDVE	0.1232	0.1429	0.24	0.90	1.15	0.56	3.00
hepatitis_consistent.lcn	ARIEL	0.0252	0.0167	0.00	3.91	3.91	0.56	–
hepatitis_consistent.lcn	CVE_{cm}	0.1580	0.1630	0.24	16.29	16.53	0.56	3.00
hepatitis_consistent.lcn	IBP	0.1236	0.1445	0.24	0.50	0.75	0.56	3.00
suicide_consistent.lcn	CDVE	0.1160	0.0430	0.10	0.52	0.62	0.87	2.00
suicide_consistent.lcn	ARIEL	0.0000	0.0000	0.00	3.74	3.74	0.87	–
suicide_consistent.lcn	CVE_{cm}	0.1147	0.0637	0.10	0.13	0.23	0.87	2.00
suicide_consistent.lcn	IBP	0.1160	0.0430	0.10	0.02	0.12	0.87	2.00
tank_consistent.lcn	CDVE	0.1337	0.1096	0.30	0.87	1.18	0.77	3.00
tank_consistent.lcn	ARIEL	0.0144	0.0102	0.00	5.02	5.02	0.77	–
tank_consistent.lcn	CVE_{cm}	0.0659	0.0555	0.31	0.19	0.50	0.77	3.00
tank_consistent.lcn	IBP	0.1351	0.1089	0.30	0.02	0.32	0.77	3.00
toy_consistent.lcn	CDVE	0.1363	0.0751	0.09	0.03	0.12	0.16	2.00
toy_consistent.lcn	ARIEL	0.0000	0.0000	0.00	1.60	1.60	0.16	–
toy_consistent.lcn	CVE_{cm}	0.1190	0.0345	0.09	0.01	0.10	0.16	2.00
toy_consistent.lcn	IBP	0.1363	0.0751	0.09	0.01	0.09	0.16	2.00

Table 22: Mean error metrics vs. CJT reference

Instance	Algorithm	MAE_{lb}	MAE_{ub}	Build	Run	Total	Ref	IW
benchmark_scaling_directions_n24_2.lcn	CDVE	0.1215	0.4370	1.78	22.80	24.57	1258.24	4.00
benchmark_scaling_directions_n24_2.lcn	ARIEL	0.0000	0.0000	0.00	13.42	13.42	1258.24	–
benchmark_scaling_directions_n24_2.lcn	CVE_{cm}	0.1123	0.1749	1.86	2206.67	2208.54	1258.24	4.00
benchmark_scaling_directions_n24_2.lcn	IBP	0.1215	0.4370	1.91	685.00	686.91	1258.24	4.00
benchmark_scaling_directions_n32_1.lcn	CDVE	0.1148	0.2247	0.57	19.98	20.55	2.42	1.00
benchmark_scaling_directions_n32_1.lcn	ARIEL	0.0000	0.0000	0.00	19.80	19.80	2.42	–
benchmark_scaling_directions_n32_1.lcn	CVE_{cm}	0.0643	0.0716	0.59	8.59	9.18	2.42	1.00
benchmark_scaling_directions_n32_1.lcn	IBP	0.1148	0.2247	0.64	0.05	0.69	2.42	1.00
maintenance_scaling_n12_3.lcn	CDVE	0.1287	0.5013	0.29	0.94	1.23	0.66	2.00
maintenance_scaling_n12_3.lcn	ARIEL	0.0284	0.0000	0.00	8.13	8.13	0.66	–
maintenance_scaling_n12_3.lcn	CVE_{cm}	0.0793	0.2270	0.31	0.25	0.57	0.66	2.00
maintenance_scaling_n12_3.lcn	IBP	0.1286	0.4926	0.31	0.04	0.35	0.66	2.00
maintenance_scaling_n6_1.lcn	CDVE	0.1029	0.3972	0.17	0.10	0.27	0.28	2.00
maintenance_scaling_n6_1.lcn	ARIEL	0.0000	0.0000	0.00	4.16	4.16	0.28	–
maintenance_scaling_n6_1.lcn	CVE	0.1029	0.4017	0.14	789.77	789.91	0.28	2.00
maintenance_scaling_n6_1.lcn	CVE_{cm}	0.1021	0.2685	0.14	0.05	0.19	0.28	2.00
maintenance_scaling_n6_1.lcn	IBP	0.1029	0.4017	0.15	0.01	0.16	0.28	2.00
maintenance_scaling_n9_2.lcn	CDVE	0.1447	0.4748	0.21	0.48	0.69	0.44	2.00
maintenance_scaling_n9_2.lcn	ARIEL	0.0398	0.0000	0.00	6.09	6.09	0.44	–
maintenance_scaling_n9_2.lcn	CVE_{cm}	0.1095	0.2437	0.20	0.12	0.32	0.44	2.00
maintenance_scaling_n9_2.lcn	IBP	0.1447	0.4662	0.23	0.02	0.25	0.44	2.00
paper_scaling_directions_n35_2.lcn	CDVE	0.1506	0.4467	7.59	190.48	198.07	1342.14	5.00
paper_scaling_directions_n35_2.lcn	ARIEL	0.0064	0.0047	0.00	36.58	36.58	1342.14	–
paper_scaling_directions_n40_1.lcn	CDVE	0.1288	0.2631	0.70	50.55	51.25	3.27	1.00
paper_scaling_directions_n40_1.lcn	ARIEL	0.0000	0.0000	0.00	34.72	34.72	3.27	–
paper_scaling_directions_n40_1.lcn	CVE_{cm}	0.0696	0.0566	0.74	658.10	658.84	3.27	1.00
paper_scaling_directions_n40_1.lcn	IBP	0.1288	0.2631	0.82	0.09	0.91	3.27	1.00
supreme_court_judicial_power_credal_source_comparison_n10_1.lcn	ARIEL	0.0532	0.1234	0.00	115.72	115.72	348.07	–
supreme_court_judicial_power_credal_source_comparison_n10_2.lcn	ARIEL	0.0321	0.1107	0.00	76.72	76.72	337.23	–
supreme_court_judicial_power_credal_source_comparison_n10_3.lcn	ARIEL	0.0259	0.0972	0.00	527.92	527.92	426.93	–
supreme_court_judicial_power_dual_source_credal_n10_1.lcn	ARIEL	0.0259	0.0972	0.00	523.64	523.64	435.71	–
supreme_court_judicial_power_release_comparison_n10_1.lcn	ARIEL	0.0317	0.0860	0.00	68.03	68.03	346.18	–
supreme_court_judicial_power_release_comparison_n10_2.lcn	ARIEL	0.0532	0.1234	0.00	117.31	117.31	337.95	–
supreme_court_judicial_power_release_comparison_n10_3.lcn	ARIEL	0.0259	0.0972	0.00	562.73	562.73	426.77	–
supreme_court_judicial_power_source_variants_n10_1.lcn	ARIEL	0.0317	0.0860	0.00	71.64	71.64	338.65	–
supreme_court_judicial_power_source_variants_n10_2.lcn	ARIEL	0.0176	0.0913	0.00	70.58	70.58	357.82	–

Table 23: Standard deviation of error metrics vs. CJT reference

Instance	Algorithm	SD_{lb}	SD_{ub}	Build	Run	Total	Ref	IW
benchmark_scaling_directions_n24_2.lcn	CDVE	0.0648	0.1928	1.78	22.80	24.57	1258.24	4.00
benchmark_scaling_directions_n24_2.lcn	ARIEL	0.0000	0.0000	0.00	13.42	13.42	1258.24	–
benchmark_scaling_directions_n24_2.lcn	CVE_{cm}	0.0780	0.1686	1.86	2206.67	2208.54	1258.24	4.00
benchmark_scaling_directions_n24_2.lcn	IBP	0.0648	0.1928	1.91	685.00	686.91	1258.24	4.00
benchmark_scaling_directions_n32_1.lcn	CDVE	0.0657	0.1463	0.57	19.98	20.55	2.42	1.00
benchmark_scaling_directions_n32_1.lcn	ARIEL	0.0000	0.0000	0.00	19.80	19.80	2.42	–
benchmark_scaling_directions_n32_1.lcn	CVE_{cm}	0.0430	0.0738	0.59	8.59	9.18	2.42	1.00
benchmark_scaling_directions_n32_1.lcn	IBP	0.0657	0.1463	0.64	0.05	0.69	2.42	1.00
maintenance_scaling_n12_3.lcn	CDVE	0.0795	0.2363	0.29	0.94	1.23	0.66	2.00
maintenance_scaling_n12_3.lcn	ARIEL	0.0514	0.0000	0.00	8.13	8.13	0.66	–
maintenance_scaling_n12_3.lcn	CVE_{cm}	0.0491	0.1572	0.31	0.25	0.57	0.66	2.00
maintenance_scaling_n12_3.lcn	IBP	0.0794	0.2313	0.31	0.04	0.35	0.66	2.00
maintenance_scaling_n6_1.lcn	CDVE	0.0729	0.2197	0.17	0.10	0.27	0.28	2.00
maintenance_scaling_n6_1.lcn	ARIEL	0.0000	0.0000	0.00	4.16	4.16	0.28	–
maintenance_scaling_n6_1.lcn	CVE	0.0729	0.2232	0.14	789.77	789.91	0.28	2.00
maintenance_scaling_n6_1.lcn	CVE_{cm}	0.0710	0.1599	0.14	0.05	0.19	0.28	2.00
maintenance_scaling_n6_1.lcn	IBP	0.0729	0.2232	0.15	0.01	0.16	0.28	2.00
maintenance_scaling_n9_2.lcn	CDVE	0.0953	0.2482	0.21	0.48	0.69	0.44	2.00
maintenance_scaling_n9_2.lcn	ARIEL	0.0496	0.0000	0.00	6.09	6.09	0.44	–
maintenance_scaling_n9_2.lcn	CVE_{cm}	0.0699	0.1858	0.20	0.12	0.32	0.44	2.00
maintenance_scaling_n9_2.lcn	IBP	0.0952	0.2428	0.23	0.02	0.25	0.44	2.00
paper_scaling_directions_n35_2.lcn	CDVE	0.0682	0.1408	7.59	190.48	198.07	1342.14	5.00
paper_scaling_directions_n35_2.lcn	ARIEL	0.0250	0.0276	0.00	36.58	36.58	1342.14	–
paper_scaling_directions_n40_1.lcn	CDVE	0.0656	0.1334	0.70	50.55	51.25	3.27	1.00
paper_scaling_directions_n40_1.lcn	ARIEL	0.0000	0.0000	0.00	34.72	34.72	3.27	–
paper_scaling_directions_n40_1.lcn	CVE_{cm}	0.0672	0.0365	0.74	658.10	658.84	3.27	1.00
paper_scaling_directions_n40_1.lcn	IBP	0.0656	0.1334	0.82	0.09	0.91	3.27	1.00
supreme_court_judicial_power_credal_source_comparison_n10_1.lcn	ARIEL	0.0620	0.2451	0.00	115.72	115.72	348.07	–
supreme_court_judicial_power_credal_source_comparison_n10_2.lcn	ARIEL	0.0407	0.2477	0.00	76.72	76.72	337.23	–
supreme_court_judicial_power_credal_source_comparison_n10_3.lcn	ARIEL	0.0414	0.2514	0.00	527.92	527.92	426.93	–
supreme_court_judicial_power_dual_source_credal_n10_1.lcn	ARIEL	0.0414	0.2514	0.00	523.64	523.64	435.71	–
supreme_court_judicial_power_release_comparison_n10_1.lcn	ARIEL	0.0624	0.2550	0.00	68.03	68.03	346.18	–
supreme_court_judicial_power_release_comparison_n10_2.lcn	ARIEL	0.0620	0.2451	0.00	117.31	117.31	337.95	–
supreme_court_judicial_power_release_comparison_n10_3.lcn	ARIEL	0.0414	0.2514	0.00	562.73	562.73	426.77	–
supreme_court_judicial_power_source_variants_n10_1.lcn	ARIEL	0.0624	0.2550	0.00	71.64	71.64	338.65	–
supreme_court_judicial_power_source_variants_n10_2.lcn	ARIEL	0.0442	0.2536	0.00	70.58	70.58	357.82	–