Abductive Reasoning in Logical Credal Networks: Supplementary Material

Radu Marinescu
IBM Research, Ireland
radu.marinescu@ie.ibm.com

Junkyu Lee
IBM Research, USA
junkyu.lee@ibm.com

Debarun Bhattacharjya
IBM Research, USA
debarunb@us.ibm.com

Fabio Cozman
Universidade de São Paulo, Brazil
fgcozman@usp.br

Alexander Gray
Centaur AI Institute, USA
alexander.gray@centaurinstitute.org

Abstract

Logical Credal Networks or LCNs were recently introduced as a powerful probabilistic logic framework for representing and reasoning with imprecise knowledge. Unlike many existing formalisms, LCNs have the ability to represent cycles and allow specifying marginal and conditional probability bounds on logic formulae which may be important in many realistic scenarios. Previous work on LCNs has focused exclusively on marginal inference, namely computing posterior lower and upper probability bounds on a query formula. In this paper, we explore abductive reasoning tasks such as solving MAP and Marginal MAP queries in LCNs given some evidence. We first define formally the MAP and Marginal MAP tasks for LCNs and subsequently show how to solve these tasks exactly using search-based approaches. We then propose several approximate schemes that allow us to scale MAP and Marginal MAP inference to larger problem instances. An extensive empirical evaluation demonstrates the effectiveness of our algorithms on both random LCN instances as well as LCNs derived from more realistic usecases.

1 Introduction

Probabilistic logic which combines probability and logic in a principled manner has emerged over the past decades as a unified representational and reasoning framework capable of dealing effectively with complex real-world applications that require efficient handling of uncertainty and compact representations of domain expert knowledge [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Logical Credal Networks or LCNs [11] were introduced recently as a probabilistic logic designed for representing and reasoning with imprecise knowledge. Unlike many existing probabilistic logics, LCNs have the ability to represent cycles (e.g., feedback loops) as well as allow specifying marginal and conditional probability bounds on logic formulae which may be important in many realistic usecases.

Up until now, the work on LCNs has focused exclusively on marginal inference, namely on computing efficiently posterior lower and upper probability bounds on a query formula. However, *abductive reasoning* tasks such as explaining the evidence observed in an LCN are equally important in many real-world applications. In probabilistic graphical models, these tasks are commonly known as MAP and Marginal MAP (MMAP) inference and have received extensive attention over the past decades

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

[12, 13]. They are typically tackled efficiently with dynamic programming (e.g., variable elimination) or heuristic search (e.g., depth-first branch and bound) based algorithms [13, 14, 15, 16].

Contribution. In this paper, we consider solving MAP and Marginal MAP inference queries in LCNs. Unlike in graphical models, an LCN encodes a set of probability distributions over its interpretations. Therefore, a complete or a partial explanation of the evidence which represents a complete or a partial truth assignment to the LCN's propositions may correspond to more than one distributions. Our work builds on very recent work on Marginal MAP inference for credal networks, a class of probabilistic graphical models that allow reasoning with imprecise probabilities [17]. We introduce formally the MAP and Marginal MAP (MMAP) tasks for LCNs as finding a complete or a partial truth assignment to the LCN's propositions that has maximum *lower* (respectively, *upper*) probability, given some evidence in the LCN. We show how to evaluate such MAP assignments using exact marginal inference for LCNs and, subsequently, propose several search schemes based on depth-first search, limited discrepancy search and simulated annealing for solving these tasks in practice. We then extend a recent message-passing scheme for approximate marginal inference [18] to handle the MAP and MMAP inference in LCNs. In addition, we also adapt the limited discrepancy search and simulated annealing methods to use an approximate evaluation of the MAP assignments during search. We experiment and evaluate our proposed algorithms on several classes of LCNs including random as well as more realistic LCN instances. Our results are quite promising and show that the search methods based on exact evaluation of the MAP assignments are limited to small size problems in practice, while the approximate message-passing scheme and, to some extent, the approximate search-based schemes can scale to much larger problem instances. This is important because it allows us to tackle practical problems involving many thousands of variables. The supplementary material includes additional details and experimental results.

2 Background

We provide next a brief overview of basic concepts about LCNs and marginal inference in these models. Throughout the paper we will use the following notations. Logical propositions are denoted by uppercase letters (e.g., $A, B, C, ...$) while for sets of propositions we use boldfaced uppercase letters (e.g., $\mathbf{A}, \mathbf{B}, \mathbf{C}, ...$). Truth assignments to propositions are denoted by lowercase letters, namely if proposition A holds true then we denote the truth assignment by a, and, alternatively, we use $\neg a$ to denote the fact that A is false. The truth assignments a and $\neg a$ to A are also known as *literals*. Sets of truth assignments (or literals) are denoted by boldfaced lowercase letters (e.g., $\mathbf{a}, \mathbf{b}, \mathbf{c}, ...$).

2.1 Logical Credal Networks

A Logical Credal Network (LCN) [11] is defined by a tuple $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, where $\mathbf{A} = \{A_1, \ldots, A_n\}$ is a set of propositions (or atoms), and $\mathcal{C}$ is a set of probability labeled sentences (or constraints) having the following two forms:

$$\alpha \leq P(\phi) \leq \beta \tag{1}$$
$$\alpha \leq P(\phi|\varphi) \leq \beta \tag{2}$$

Here, ϕ and φ are arbitrary propositional logic formulae[1] involving propositions in $\mathcal{A}$ and logical connectives such as negation, disjunction and conjunction, and $0 \leq \alpha \leq \beta \leq 1$ are lower and upper probability bounds, respectively. An LCN is associated with a *primal graph* defined as follows.

Definition 1 (primal graph)**.** *The* primal graph *of an LCN $\mathcal{L}$ is a directed graph G that contains* formula nodes *associated with the formulae ϕ and φ in $\mathcal{L}$'s sentences and* proposition nodes *associated with the propositions involved in those formulae, respectively. For type (1) sentences, there is a directed edge in G from each proposition node in ϕ to the ϕ. For type (2) sentences, G contains directed edges from each of the proposition nodes in φ to φ, a directed edge from φ to ϕ, and bi-directed edges from ϕ to the proposition nodes in ϕ, respectively. Note that if a formula consists of a single proposition, then G contains a single proposition node for that formula.*

Definition 2 (parents, descendants)**.** *A* parent *of a proposition A is a proposition such that there is a directed path in G from it to A in which all intermediate nodes are formulae. A* descendant *of*

---

[1]The original definition of LCNs allows for relational structures and first-order logic formulae, but their semantics is obtained by grounding on finite domains [11].

2

$$0.05 \le P(B) \le 0.1$$
$$0.3 \le P(S) \le 0.4$$
$$0.1 \le P(B \lor C | S) \le 0.2$$
$$0.6 \le P(D | B \land C) \le 0.7$$
$$0.7 \le P(\neg(X \oplus D) | C) \le 0.8$$

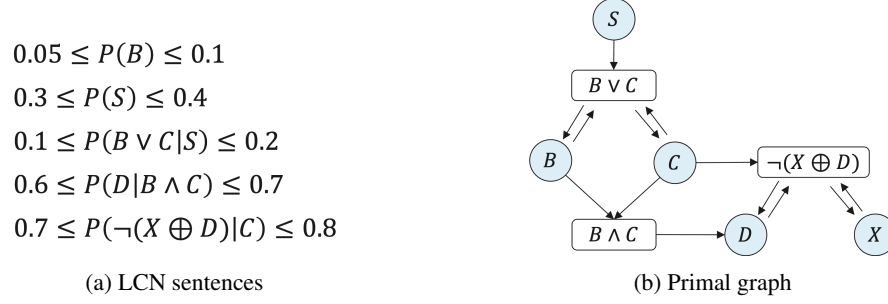
(a) LCN sentences

(b) Primal graph

Figure 1: A simple LCN and its primal graph.

a proposition A is a proposition such that there is a directed path in G from A to it in which no intermediate node is a parent of A.

An LCN is endowed with a local Markov condition where a proposition node A is independent, given its parents, of all proposition nodes that are not A itself nor descendants of A nor parents of A [11]. Therefore, an LCN represents a set of probability distributions over all interpretations of its formulae that satisfy the constraints represented by the type (1) and (2) sentences as well as the constraints induced by the independence relations given by the local Markov property [11].

Example 1. *Figure 1 describes a simple LCN whose sentences shown in Figure 1a state that: bronchitis (B) is more likely than smoking (S); smoking may cause cancer (C) or bronchitis; dyspnea (D) is a common symptom for cancer and bronchitis; in case of cancer we have either a positive X-ray result (X) and dyspnea, or a negative X-ray and no dyspnea. Figure 1b shows the primal graph where the formula and proposition nodes are displayed as rectangles and shaded circles, respectively.*

2.2 Marginal Inference in Logical Credal Networks

Given an LCN $\mathcal{L}$ with n propositions, the *marginal inference* task is to compute lower and upper bounds on the posterior probability $P(\psi)$ of a query formula ψ, which we denote by $\underline{P}(\psi)$ and $\overline{P}(\psi)$, respectively. This is achieved by solving a non-linear program defined by a set of non-negative real-valued variables representing the probabilities of $\mathcal{L}$'s interpretations, a set of linear constraints derived from $\mathcal{L}$'s sentences, a set of non-linear constraints corresponding to the independence assumptions given by the local Markov condition, and a linear objective function encoding the query $P(\psi)$ which is minimized and maximized to yield the desired bounds (Equation 8).

$$\sum_{i=1}^{m} p_i = 1 \tag{3}$$

$$p_i \ge 0, \forall i = 1, \dots, m \tag{4}$$

$$\alpha \le \vec{I}_\phi \odot \vec{p} \le \beta \tag{5}$$

$$\alpha \cdot \vec{I}_\varphi \odot \vec{p} \le \vec{I}_{\phi \land \varphi} \odot \vec{p} \le \beta \cdot \vec{I}_\varphi \odot \vec{p} \tag{6}$$

$$(\vec{I}_a \odot \vec{p}) \cdot (\vec{I}_b \odot \vec{p}) - (\vec{I}_c \odot \vec{p}) \cdot (\vec{I}_d \cdot \vec{p}) = 0 \tag{7}$$

$$\text{minimize/maximize } \vec{I}_\psi \odot \vec{p} \tag{8}$$

More specifically, let $\vec{p} = (p_1, \dots, p_m)$ be the vector representing the probabilities of the $m = 2^n$ interpretations and let $\vec{I}_\phi = (a_1^\phi, \dots, a_m^\phi)$ be a binary vector, called an *indicator vector*, such that a_i^ϕ is 1 if formula ϕ is true in the i-th interpretation and 0 otherwise. Since the probability of a formula ϕ is the sum of the probabilities of the interpretations in which ϕ is true, we can write $P(\phi)$ as $\vec{I}_\phi \odot \vec{p}$ where $\odot$ is the dot-product of two vectors. Therefore, Equations (3) and (4) ensure that $\vec{p}$ is a valid probability distribution, Equations (5) and (6) encode the type (1) and (2) sentences in $\mathcal{L}$ while Equation 7 encodes the conditional independencies of the form $P(X_j|\mathbf{S}_j, \mathbf{T}_j) = P(X_j|\mathbf{S}_j)$, where X_j is a proposition, $\mathbf{S}_j = \{S_{j1}, \dots, S_{jk}\}$ and $\mathbf{T}_j = \{T_{j1}, \dots, T_{jl}\}$ are X_j's parents and

3

non-descendants in the primal graph of $\mathcal{L}$, $\vec{I}_\phi$ and $\vec{I}_{\phi\wedge\varphi}$ are the indicator vectors for formulae ϕ and $\phi \wedge \varphi$ involved in $\mathcal{L}$'s sentences, and $\vec{I}_a$, $\vec{I}_b$, $\vec{I}_c$ and $\vec{I}_d$ are the indicator vectors corresponding to the formulae $a = (x_j \wedge s_{j1} \wedge \cdots \wedge s_{jk} \wedge t_{j1} \wedge \cdots \wedge t_{jl})$, $b = (s_{j1} \wedge \cdots \wedge s_{jk})$, $c = (x_j \wedge s_{j1} \wedge \cdots \wedge s_{jk})$, and $d = (s_{j1} \wedge \cdots \wedge s_{jk} \wedge t_{j1} \wedge \cdots \wedge t_{jl})$, respectively (see also [11] for more details).

3 MAP and Marginal MAP Inference in LCNs

Maximum A Posteriori (MAP) and Marginal MAP (MMAP) inference are well known abductive reasoning tasks in probabilistic graphical models such as Bayesian networks and Markov networks [12, 13, 14, 15, 16]. Specifically, the MAP task calls for finding a complete assignment to all variables having maximum probability, given the evidence. Marginal MAP generalizes MAP and looks for a partial variable assignment that has maximum marginal probability, given the evidence. MAP and MMAP inference tasks appear in many real-world applications such as diagnosis, abduction and explanation and are typically tackled with dynamic programming (e.g., variable elimination) or heuristic search (e.g., depth-first branch and bound) based algorithms [13, 14, 15, 16].

In this section, we present our novel approach for solving the MAP and Marginal MAP inference tasks in Logical Credal Networks. Unlike in graphical models, a (partial) variable assignment (or interpretation) in an LCN may correspond to more than one distribution. Therefore, we begin by defining formally two MAP and MMAP inference tasks for LCNs, called *maximin MAP* (resp. *maximin MMAP*) and *maximax MAP* (resp. *maximax MMAP*). Subsequently, we develop several exact and approximation schemes for solving these tasks efficiently in practice.

3.1 The MAP and Marginal MAP Tasks in LCNs

Let $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ be an LCN with n propositions and let $\mathbf{E} = \{E_1, \ldots, E_k\} \subseteq \mathbf{A}$ be subset of k propositions, called *evidence*, for which the truth values $\mathbf{e} = \{e_1, \ldots, e_k\}$ are known. Let $\mathbf{Y} = \{Y_1, \ldots, Y_m\} \subseteq \mathbf{A} \setminus \mathbf{E}$ be a subset of m propositions called *MAP propositions*. A truth assignment to $\mathbf{Y}$ is is called a *MAP assignment* and is denoted by $\mathbf{y} = \{y_1, \ldots, y_m\}$, respectively. If $\mathbf{Y} = \mathbf{A} \setminus \mathbf{E}$ (i.e., $m = n - k$) then we have a MAP task, otherwise we have a MMAP task (i.e., $m < n - k$). The *maximin* and *maximax* MAP/MMAP tasks are defined as follows:

Definition 3 (maximin). *Given an LCN $\mathcal{L}$ with n propositions, evidence $\mathbf{e}$, and MAP propositions $\mathbf{Y}$, the* maximin *MAP (or* maximin *MMAP if $m < n - k$) task is finding a truth assignment $\mathbf{y}^*$ to $\mathbf{Y}$ having maximum* lower *probability, given evidence $\mathbf{e}$, namely:*

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \Omega(\mathbf{Y})} \underline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}}) \tag{9}$$

where $\Omega(\mathbf{Y})$ is the set of all truth assignments to the MAP propositions, and $\psi_{\mathbf{y} \wedge \mathbf{e}} = y_1 \wedge \cdots \wedge y_m \wedge e_1 \wedge \cdots \wedge e_k$ is the conjunction of the literals in $\mathbf{y}$ and $\mathbf{e}$, respectively.

Definition 4 (maximax). *Given an LCN $\mathcal{L}$ with n propositions, evidence $\mathbf{e}$, and MAP propositions $\mathbf{Y}$, the* maximax *MAP (or* maximax *MMAP if $m < n - k$) task is finding a truth assignment $\mathbf{y}^*$ to $\mathbf{Y}$ having maximum* upper *probability, given evidence $\mathbf{e}$, namely:*

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \Omega(\mathbf{Y})} \overline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}}) \tag{10}$$

where $\Omega(\mathbf{Y})$ is the set of all truth assignments to the MAP propositions, and $\psi_{\mathbf{y} \wedge \mathbf{e}} = y_1 \wedge \cdots \wedge y_m \wedge e_1 \wedge \cdots \wedge e_k$ is the conjunction of the literals in $\mathbf{y}$ and $\mathbf{e}$, respectively.

3.2 Exact Evaluation of a MAP Assignment

Clearly, computing the lower and upper probabilities $\underline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$ and $\overline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$ of a MAP assignment $\mathbf{y}$ given evidence $\mathbf{e}$ can be done easily by minimizing and, respectively maximizing the non-linear program defined by Equations (3)–(8), where the query formula is the conjunction of positive or negative literals in $\mathbf{y}$ and $\mathbf{e}$, namely $\psi_{\mathbf{y} \wedge \mathbf{e}} = y_1 \wedge \cdots \wedge y_m \wedge e_1 \wedge \cdots \wedge e_k$. Therefore, evaluating a MAP assignment in case of both MAP and Marginal MAP inference in LCNs is quite difficult as it involves solving a maginal inference problem for LCNs which is know to be NP-hard [11]. This is in contrast with graphical models where, at least for MAP inference, the evaluation of a MAP assignment is linear in the number of variables [13].

4

Algorithm 1 Depth-First Search for MAP and Marginal MAP Inference in LCNs

---

1: **procedure** DFS($\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, $\mathbf{E} = \mathbf{e}$, $\mathbf{Y}$)
2:    initialize $\mathbf{y}^* \leftarrow \emptyset$, $best \leftarrow -\infty$
3:    SEARCH($\emptyset$, $\mathbf{Y}$)
4:    **return** $\mathbf{y}^*$
5: **procedure** SEARCH($\mathbf{y}$, $\mathbf{Y}$)
6:    **if** $size(\mathbf{y}) == size(\mathbf{Y})$ **then**
7:      **if** maximin **then**
8:        $score(\mathbf{y}) \leftarrow \underline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$
9:      **else**

10:        $score(\mathbf{y}) \leftarrow \overline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$
11:      **if** $score(\mathbf{y}) > best$ **then**
12:        $\mathbf{y}^* \leftarrow \mathbf{y}$, $best \leftarrow score(\mathbf{y})$
13:    **else**
14:      select unassigned proposition $Y_i \in \mathbf{Y}$
15:      **for all** values $y \in \{y_i, \neg y_i\}$ **do**
16:        $\mathbf{y} \leftarrow \mathbf{y} \cup \{Y_i = y\}$
17:        SEARCH($\mathbf{y}$, $\mathbf{Y}$)

Algorithm 2 Limited Discrepancy Search for MAP and Marginal MAP Inference in LCNs

---

1: **procedure** LDS($\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, $\mathbf{E} = \mathbf{e}$, $\mathbf{Y}$, δ)
2:    initialize $\mathbf{y}_0$ randomly and let $\mathbf{y}^* \leftarrow \mathbf{y}_0$
3:    $best \leftarrow score(\mathbf{y}^*)$
4:    **for all** $\theta = 1 \ldots \delta$ **do**
5:      SEARCH($\mathbf{y}^*$, $\mathbf{Y}$, θ, 1)
6:    **return** $\mathbf{y}^*$, $best$
7: **procedure** SEARCH($\mathbf{y}$, $\mathbf{Y}$, θ, i)
8:    **if** $\theta == 0$ or $i > |\mathbf{Y}|$ **then**
9:      **if** maximin **then**
10:        $score(\mathbf{y}) \leftarrow \underline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$
11:      **else**

12:        $score(\mathbf{y}) \leftarrow \overline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$
13:      **if** $score(\mathbf{y}) > best$ **then**
14:        $\mathbf{y}^* \leftarrow \mathbf{y}$, $best \leftarrow score(\mathbf{y})$
15:    **else**
16:      **for all** values $y \in \{y_i, \neg y_i\}$ **do**
17:        **if** $\mathbf{y}[i] == y$ **then**
18:          $\mathbf{z} \leftarrow$ SEARCH($\mathbf{y}$, $\mathbf{Y}$, $i+1$, θ)
19:        **else**
20:          $\mathbf{y}' \leftarrow \mathbf{y}$; $\mathbf{y}'[i] \leftarrow y$
21:          $\mathbf{z} \leftarrow$ SEARCH($\mathbf{y}'$, $\mathbf{Y}$, $i+1$, $\theta-1$)
22:      **return** $\mathbf{z}$

Example 2. *For illustration, consider the LCN example from Figure 1 and assume that we have evidence* $\mathbf{E} = \{x, \neg s\}$, *namely a patient that has a positive X-ray result ($X = x$) is not smoking ($S = \neg s$). The MAP propositions in this case are* $\mathbf{Y} = \{B, C, D\}$. *For example, the MAP assignment* $\mathbf{y} = (b, \neg c, \neg d)$ *corresponds to the query formula* $\psi = b \wedge \neg c \wedge \neg d \wedge x \wedge \neg s$, *while its lower and upper probabilities are 9.9e-09 and 0.1, respectively.*

3.3 Search Algorithms Using Exact MAP Assignment Evaluations

We present next several search-based schemes for solving the MAP and Marginal MAP tasks in LCNs. These methods employ different search strategies for exploring the search space defined by the MAP propositions while evaluating exactly each complete or partial MAP assignment.

Depth-First Search. Our first approach for solving the MAP and Marginal MAP tasks, called DFS, is described by Algorithm 1. It takes as input an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, evidence $\mathbf{E} = \mathbf{e}$ and a set of MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$ and outputs the optimal MAP assignment $\mathbf{y}^*$. The method conducts a *depth-first search* over the space of partial assignments to the MAP propositions, and, for each complete MAP assignment $\mathbf{y}$ computes its score as the exact lower probability $\underline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$ (respectively, upper probability $\overline{P}(\psi_{\mathbf{y} \wedge \mathbf{e}})$), given the evidence $\mathbf{e}$. This way, the optimal solution $\mathbf{y}^*$ corresponds to the MAP assignment (or configuration) with the highest score. It is easy to see that if $\mathbf{Y} = \mathbf{A} \setminus \mathbf{E}$ then $\mathbf{y}^*$ is the solution of a MAP task, otherwise, it is the solution of a Marginal MAP task. Based on previous work [11], the time complexity of evaluating each MAP assignment can be bounded by $O(2^{2^n})$, where 2^n is the number of $\mathcal{L}$'s interpretations. Therefore, we have that:

Theorem 1 (complexity). *Given an LCN* $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ *with* n *propositions, evidence* $\mathbf{E} = \mathbf{e}$ *and MAP propositions* $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$, *algorithm DFS is sound and complete. The time and space complexity of the algorithm is* $O(2^{m+2^n})$ *and* $O(2^n)$, *respectively, where* m *is the number of MAP propositions.*

5

Algorithm 3 Simulated Annealing for MAP and Marginal MAP Inference in LCNs

---

1: **procedure** SA($\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle, \mathbf{E} = \mathbf{e}, \mathbf{Y}$)
2:   initialize $\mathbf{y}_0$ randomly and let $\mathbf{y}^* \leftarrow \mathbf{y}_0$
3:   $best \leftarrow score(\mathbf{y}^*)$
4:   **for all** iterations $i = 1 \ldots N$ **do**
5:     set $\mathbf{y} \leftarrow \mathbf{y}^*, T \leftarrow T_{init}$
6:     **for all** flips $j = 1 \ldots M$ **do**
7:      let $\mathcal{N}$ be $\mathbf{y}$'s neighbors
8:      select random neighbor $\mathbf{y}' \in \mathcal{N}$
9:      $\Delta \leftarrow \log score(\mathbf{y}') - \log score(\mathbf{y})$
10:      **if** $\Delta > 0$ **then**
11:       $\mathbf{y} \leftarrow \mathbf{y}'$
12:      **else**
13:       sample randomly $p \in (0, 1)$
14:       **if** $p < e^{\frac{\Delta}{T}}$ **then**
15:        $\mathbf{y} \leftarrow \mathbf{y}'$
16:      **if** $score(\mathbf{y}) > best$ **then**
17:       $\mathbf{y}^* \leftarrow \mathbf{y}, best \leftarrow score(\mathbf{y})$
18:     $T \leftarrow T * \sigma$
19:   **return** $\mathbf{y}^*$

Proof. Soundness and completeness of algorithm DFS follow easily since it enumerates all 2^m MAP assignments and the evaluation of each assignment is exact. The size of the MAP search space is bounded by $O(2^m)$. The evaluation of a MAP assignment involves solving a non-linear constraint program associated with the input LCN and therefore its time complexity can be bounded by $O(2^{2^n})$ because the LCN has 2^n interpretations. Therefore the time complexity of algorithm DFS is $O(2^{m+2^n})$. The space complexity of the MAP evaluation is linear in the number of interpretations and therefore is bounded by $O(2^n)$ which also bounds the space complexity of algorithm DFS. $\qquad\square$

Example 3. *Consider again the LCN from Figure 1 with evidence $\mathbf{E} = \{x, \neg s\}$. In this case, the exact maximin MAP assignment found by algorithm DFS is $\mathbf{y}^* = \{\neg b, c, d\}$ with value 9.99e-09, while the exact maximax MAP assignment is $\mathbf{y}^* = \{\neg b, \neg c, d\}$ with value 0.7, respectively.*

Limited Discrepancy Search. Our second approach for MAP and Marginal MAP inference in LCNs, called LDS, is described by Algorithm 2 and uses Limited Discrepancy Search (LDS) [19, 20] to explore the search space of partial MAP assignments. LDS is a depth-first search strategy that searches for new solutions by iteratively increasing the number of *discrepancy* values, where a discrepancy value indicates the maximum number of allowed variable-value assignment changes to an initial solution [19]. Therefore, algorithm LDS takes as input an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, evidence $\mathbf{E} = \mathbf{e}$, MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \backslash \mathbf{E}$ and maximum discrepancy value δ and outputs the best possible solution $\mathbf{y}^*$ that can be obtained with discrepancy δ. Specifically, LDS starts with a discrepancy value θ of 1 and conducts an iterative search that allows to change the truth values of at most θ propositions in the initial solution $\mathbf{y}_0$ (initialized randomly). It then increments θ while keeping track of the current best solution $\mathbf{y}^*$ until θ exceeds the maximum discrepancy value δ. Function SEARCH performs the actual exploration of the MAP search space limited by discrepancy θ. When SEARCH selects a truth value $y \in \{y_i, \neg y_i\}$ that is different from the one corresponding to proposition $Y_i \in \mathbf{Y}$ at position i in the assignment $\mathbf{y}$, it decrements θ to reduce the number of changes allowed to the remaining MAP propositions. Otherwise, the truth value for proposition Y_i remains unchanged and, therefore, the θ value is preserved. When SEARCH has checked all MAP propositions in $\mathbf{y}$ or consumed the discrepancy budget, it computes the score of the corresponding MAP configuration $\mathbf{y}$ and updates the best solution found so far if $score(\mathbf{y})$ is currently the best one. It is easy to see that if the maximum discrepancy value δ is equal to the number of MAP propositions then LDS is equivalent to DFS. Therefore, we can show that:

Theorem 2 (complexity). *Given an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ with n propositions, evidence $\mathbf{E} = \mathbf{e}$ and MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$, algorithm LDS is sound and complete. The time and space complexity of the algorithm is $O(2^{m+2^n})$ and $O(n)$, respectively, where m is the number of MAP propositions.*

Proof. Soundness and completeness of algorithm LDS follow easily since LDS with discrepancy value equal to the number of MAP propositions m enumerates all 2^m MAP assignments and the evaluation of each assignment is exact. We have that the size of the MAP search space is bounded by $O(2^m)$. The evaluation of a MAP assignment involves solving a non-linear constraint program associated with the input LCN and therefore its time complexity can be bounded by $O(2^{2^n})$ because the LCN has 2^n interpretations. Therefore the time complexity of algorithm LDS is $O(2^{m+2^n})$. The

6

(a) Depth-First Search   (b) Limited Discrepancy Search ($\delta = 1$)

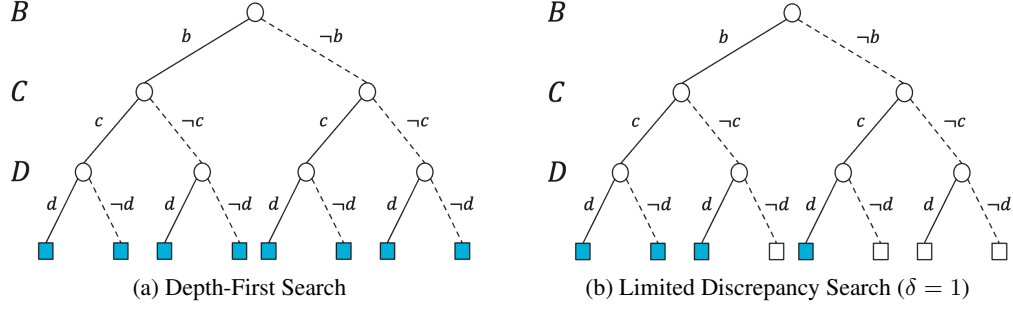



Figure 2: Search spaces traversed by algorithms DFS and LDS for the LCN from Figure 1.

space complexity of the MAP evaluation is linear in the number of interpretations and therefore is bounded by $O(2^n)$ which also bounds the space complexity of algorithm LDS. $\square$

Example 4. *Figures 2a and 2b show the search space explored by algorithms* DFS *and* LDS *with maximum discrepancy $\delta = 1$ for the LCN given in Figure 1 with MAP propositions $\{B, C, D\}$ and evidence $\mathbf{E} = \{x, \neg s\}$, respectively. In this case,* LDS *starts the initial assignment $\mathbf{y}_0 = \{b, c, d\}$. The solutions corresponding to the empty leaf nodes in Figure 2b are not visited by algorithm* LDS.

Simulated Annealing.   The third approach for solving MAP and Marginal MAP tasks in LCNs is described by Algorithm 3 and employs a form of stochastic local search known as Simulated Annealing (SA) [21] to explore the search space defined by the MAP propositions. More specifically, our algorithm denoted by SA starts from an initial guess $\mathbf{y}$ as a truth assignment to the MAP propositions $\mathbf{Y}$, and iteratively tries to improve it by moving to a better neighbor $\mathbf{y}'$ that has a higher score. A *neighbor* $\mathbf{y}'$ of configuration $\mathbf{y}$ is defined as a new configuration $\mathbf{y}'$ which results from changing the truth value of a single proposition Y in $\mathbf{Y}$. For example, the neighbors of $\mathbf{y} = \{\neg b, c, \neg d\}$ for the LCN from Figure 1 with MAP propositions $\mathbf{Y} = \{B, C, D\}$ are $\{b, c, \neg d\}$, $\{\neg b, \neg c, \neg d\}$ and $\{\neg b, c, d\}$, respectively. As before, computing the score of a neighbor $\mathbf{y}'$ requires evaluating the lower probability (respectively, the upper probability) of the corresponding MAP assignment given the evidence. At each step during the search, algorithm SA considers some neighboring state $\mathbf{y}'$ of the current state $\mathbf{y}$, and probabilistically decide between moving to state $\mathbf{y}'$ or staying in the current state. The probability of making the transition from $\mathbf{y}$ to $\mathbf{y}'$ is specified by an acceptance probability function $P(\mathbf{y}', \mathbf{y}, T)$ that depends on the scores of the two states as well as a global time-varying parameter T called *temperature*. We chose $P(\mathbf{y}', \mathbf{y}, T) = e^{\frac{\Delta}{T}}$, where $\Delta = \log score(\mathbf{y}') - \log score(\mathbf{y})$. The temperature is decreased during search using a cooling schedule $\sigma < 1$. Furthermore, algorithm SA can be executed for a number of N iterations with a maximum of M flips allowed per iteration, thus implementing a random restarts strategy.

Theorem 3 (complexity). *Given an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ with n propositions, evidence $\mathbf{E} = \mathbf{e}$ and MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$, the time and space complexity of algorithm* SA *is $O(N \cdot M \cdot 2^{2^n})$ and $O(2^n)$, respectively, where N is the number of iterations and M is the number of flips per iterations.*

Proof. Clearly, the number of MAP assignments visited by algorithm SA is bounded by $O(N \cdot M)$ where N is the number of iterations and M is the maximum number of flips allowed per iteration. The evaluation of a MAP assignment involves solving a non-linear constraint program associated with the input LCN and therefore its time complexity can be bounded by $O(2^{2^n})$ because the LCN has 2^n interpretations. Therefore the time complexity of algorithm SA is $O(N \cdot M \cdot 2^{2^n})$. The space complexity of the MAP evaluation is linear in the number of interpretations and therefore is bounded by $O(2^n)$ which also bounds the space complexity of algorithm SA. $\square$

Finally, since simulated annealing is known to converge to the optimal solution if the temperature decays slowly enough (i.e., exponential decay) [21], it follows that algorithm SA is also sound and complete, very much like the other two methods DFS and LDS, respectively.

7

Algorithm 4 Approximate MAP and Marginal MAP Inference in LCNs

---

1: **procedure** AMAP($\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, $\mathbf{E} = \mathbf{e}$, $\mathbf{Y}$)
2:    Create factor graph $\mathcal{F}$ of $\mathcal{L}$
3:    **for all** evidence propositions $E_j \in \mathbf{E}$ **do**
4:      Select factor node $f \in \mathcal{F}$ s.t. $E_j \in sc(f)$
5:      **if** $e_j \in \mathbf{e}$ **then** add $P(e_j) = 1.0$ to f
6:      **else** add $P(\neg e_j) = 1.0$ to f
7:    **for all** edges $(A, f) \in \mathcal{F}$ **do**
8:      Init message $[l_{A \to f}, u_{A \to f}] = [0, 1]$
9:      Init message $[l_{f \to A}, u_{f \to A}] = [0, 1]$
10:    **repeat**
11:      ▷ Update the variable-to-factor messages
12:      **for all** edges $(A, f) \in \mathcal{F}$ **do**
13:        $l = \max_{f' \in N(A) \backslash \{f\}} l_{f' \to A}$
14:        $u = \min_{f' \in N(A) \backslash \{f\}} u_{f' \to A}$
15:        Update $[l_{A \to f}, u_{A \to f}] = [l, u]$
16:      ▷ Update the factor-to-variable messages
17:      **for all** edges $(A, f) \in \mathcal{F}$ **do**

18:        $l = \min P(a)$ subject to (19)–(24)
19:        $u = \max P(a)$ subject to (19)–(24)
20:        Update $[l_{f \to A}, u_{f \to A}] = [l, u]$
21:    **until** convergence
22:    **for all** MAP propositions $Y \in \mathbf{Y}$ **do**
23:     **if** `maximin` **then**
24:      $\underline{P}(y) = \max_{f \in N(Y)} l_{f \to Y}$
25:      $\underline{P}(\neg y) = 1 - \underline{P}(y)$
26:      **if** $\underline{P}(y) > \underline{P}(\neg y)$ **then** $\mathbf{y}^* \leftarrow \mathbf{y}^* \cup \{y\}$
27:      **else** $\mathbf{y}^* \leftarrow \mathbf{y}^* \cup \{\neg y\}$
28:     **else**
29:      $\overline{P}(y) = \min_{f \in N(Y)} u_{f \to Y}$
30:      $\overline{P}(\neg y) = 1 - \overline{P}(y)$
31:      **if** $\overline{P}(y) > \overline{P}(\neg y)$ **then** $\mathbf{y}^* \leftarrow \mathbf{y}^* \cup \{y\}$
32:      **else** $\mathbf{y}^* \leftarrow \mathbf{y}^* \cup \{\neg y\}$
33:    **return** $\mathbf{y}^*$

3.4 Approximate MAP and Marginal MAP Inference

The main drawback of the inference schemes presented in the previous section is the exact evaluation of the MAP assignments. The latter is known to be exponentially bounded by the number of interpretation of the input LCN [22, 11] which limits the scalability of these methods to relatively small LCNs [11]. Therefore, in order to tackle larger LCNs, we extend a recent message-passing approximation scheme for marginal inference in LCNs [18] to solve the `maximin` and `maximax` MAP/MMAP tasks. Subsequently, we also adapt the limited discrepancy search and simulated annealing methods to use an approximate evaluation of the MAP assignments during search.

Algorithm 4 describes our message-passing based approximation scheme for MAP and Marginal MAP inference in LCNs. The method, which we call AMAP, takes as input an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$, the evidence $\mathbf{E} = \mathbf{e}$ and the MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$, and outputs the most likely truth values assignment $\mathbf{y}^*$ to $\mathbf{Y}$ after propagating messages along the edges of a *factor graph* associated with the LCN, until convergence. The *factor graph* $\mathcal{F}$ of $\mathcal{L}$ is a bipartite graph with *proposition nodes* and *factor nodes*, respectively. A proposition node is labeled by a proposition A in $\mathbf{A}$, while a *factor node* denoted by f represents one or more sentences in $\mathcal{C}$ that involve the same set of propositions. A factor node is connected to a proposition node if they share the same proposition [18]. Before the message propagation phase, each evidence proposition E_j is converted into a constraint $P(e_j) = 1.0$ or $P(\neg e_j) = 1$ depending on its observed truth value e_j or $\neg e_j$ and placed into a factor node f whose scope $sc(f)$ contains E_j.

$$\sum_{i=1}^{R} p_i = 1 \tag{11}$$

$$p_i \geq 0, \forall i = 1, \dots, R \tag{12}$$

$$\alpha \leq \vec{I}_\phi \cdot \vec{p} \leq \beta \tag{13}$$

$$\alpha \vec{I}_\varphi \odot \vec{p} \leq \vec{I}_{\phi \wedge \varphi} \odot \vec{p} \leq \beta \vec{I}_\varphi \odot \vec{p} \tag{14}$$

$$l_{A' \to f} \leq \vec{I}_{a'} \odot \vec{p} \leq u_{A' \to f}, \forall A' \in N(f) \tag{15}$$

$$\vec{I}_{a' \wedge a''} \odot \vec{p} = (\vec{I}_{a'} \odot \vec{p}) \cdot (\vec{I}_{a''} \odot \vec{p}), \forall A' \neq A'' \in N(f) \tag{16}$$

$$\text{minimize/maximize } \vec{I}_a \odot \vec{p} \tag{17}$$

The message sent by a proposition node A to a neighboring factor node f is an interval $[l_{A \to f}, u_{A \to f}]$ defined by $l_{A \to f} = \max_{f' \in N(A) \backslash \{f\}} l_{f' \to A}$ and $u_{A \to f} = \min_{f' \in N(A) \backslash \{f\}} u_{f' \to A}$, where $0 \leq$

8

$0.05 \le P(B) \le 0.1$	$P(W_1	\neg B) = 1$
$0.3 \le P(S) \le 0.4$	$P(W_1	B) = 0$
$0.1 \le P(B \vee C|S) \le 0.2$	$P(W_2|W_1 \wedge C) = 1$
$0.6 \le P(D|B \wedge C) \le 0.7$	$P(W_2|W_1 \wedge \neg C) = 0$
$0.7 \le P(\neg(X \oplus D)|C) \le 0.8$	$P(W_2|\neg W_1 \wedge \neg C) = 0$
	$P(W_2	\neg W_1 \wedge C) = 0$

(a) LCN sentences (augmented)   (b) Primal graph (augmented)

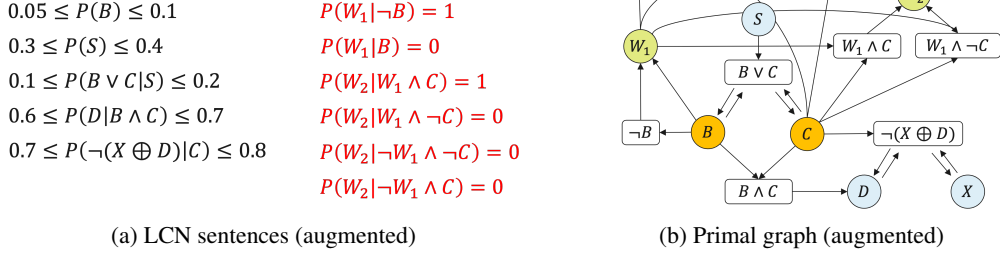



Figure 3: An augmented LCN and its primal graph.

$l_{A \to f} \le u_{A \to f} \le 1$, $[l_{f' \to A}, u_{f' \to A}]$ is the message sent by a neighboring factor node $f' \in N(A) \setminus \{f\}$, other than f, to A, and $N(\cdot)$ denotes the neighbors of a node in the factor graph. The message sent by a factor node f to a neighboring proposition node A is also an interval $[l_{f \to A}, u_{f \to A}]$ obtained by solving a local non-linear constraint program whose objective function encoding $P(a)$ is minimized and, respectively, maximized (Eq. 17) subject to linear constraints encoding f's sentences (Eqs. 11-14), linear constraints ensuring that, for each proposition node other than A that is connected to f, its marginal probability is within the bounds given by the corresponding proposition-to-factor messages (Eq. 15), and non-linear constraints encoding the assumption that f's propositions (other than A) are independent of each other (Eq. 16). Furthermore, a factor node f is assumed to involve at most r propositions and therefore there are $R = 2^r$ interpretations to f's sentences. Upon convergence, the maximin MAP assignment $\mathbf{y}^*$ can be obtained as follows: for each MAP proposition $Y \in \mathbf{Y}$ we compute the tightest lower probability bound $\underline{P}(y)$ by maximizing the lower bound of all incoming factor-to-proposition messages to Y, and, subsequently, select y as the most likely value assignment to Y if $\underline{P}(y) > \underline{P}(\neg y)$ and $\neg y$ otherwise. The maximax MAP assignment is obtained in a similar manner but the maximization is done using the upper probability bounds $\overline{P}(y)$ and $\overline{P}(\neg y)$, respectively.

Theorem 4 (complexity). *Given an LCN $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ with n propositions, evidence $\mathbf{E} = \mathbf{e}$ and MAP propositions $\mathbf{Y} \subseteq \mathbf{A} \setminus \mathbf{E}$, the time and space complexity of algorithm AMAP is $O(N \cdot M \cdot 2^{2^r})$ and $O(2^r)$, where N is the number of iterations, M bounds the number of factor-to-node messages per iteration and r bounds the number of propositions in the factor nodes, respectively.*

Proof. Clearly, the complexity of algorithm AMAP is dominated by the complexity of the factor-to-node messages which involve solving a local non-linear constraint program defined over 2^r interpretations, where r bounds the number of propositions in the factor nodes of the factor graph. Therefore, it follows that the time complexity of algorithm AMAP is bounded by $O(N \cdot M \cdot 2^{2^r})$ where N is the number of iterations and M is the number of factor-to-node messages. $\square$

3.5 Approximate Evaluation of a MAP Assignment

Estimating the lower and upper probabilities of a MAP assignment $\mathbf{y}$ can be done by approximate marginal inference on an *augmented* LCN as follows. Let $\mathcal{L} = \langle \mathbf{A}, \mathcal{C} \rangle$ be the input LCN and let $\mathbf{y} = (y_1, \ldots, y_m)$ be a MAP assignment to propositions $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$ (for simplicity, we include the evidence $\mathbf{e}$ in $\mathbf{y}$). The *augmented* LCN $\mathcal{L}' = \langle \mathbf{A}', \mathcal{C}' \rangle$ is constructed by adding a set of auxiliary propositions $\mathbf{W} = \{W_1, \ldots, W_m\}$, one for each MAP proposition, and additional constraints of the following two forms: $P(W_1|Y_1)$ and $P(W_j|W_{j-1} \wedge Y_j)$, for all $2 \le j \le m$, such that $P(w_1|y_1) = 1$, $P(w_1|\neg y_1) = 0$, $P(w_j|w_{j-1} \wedge y_j) = 1$, $P(w_j|w_{j-1} \wedge \neg y_j) = 0$, $P(w_j|\neg w_{j-1} \wedge y_j) = 0$ and $P(w_j|\neg w_{j-1} \wedge \neg y_j) = 0$, respectively. Then, we can estimate $\underline{P}(\psi_{\mathbf{y}})$ and $\overline{P}(\psi_{\mathbf{y}})$, where $\psi_{\mathbf{y}} = y_1 \wedge \cdots \wedge y_m$, by computing the posterior marginals $\underline{P}(w_m)$ and $\overline{P}(w_m)$ in the augmented LCN $\mathcal{L}'$ using the method from [18].

Example 5. *Figure 3 shows the augmented LCN obtained from the example LCN given in Figure 1 for the MAP assignment $\mathbf{y} = (\neg b, c)$, where B and C are the MAP propositions. The auxiliary propositions W_1 and W_2 and the corresponding auxiliary constraints are shown in red in Figure 3a whereas the augmented primal graph is given in Figure 3b, respectively.*

9

3.6 Search Algorithms Based on Approximate MAP Evaluations

The main assumption behind algorithm AMAP is that all MAP propositions are independent of each other. Therefore, the solution returned by AMAP to a MAP or a MMAP task most likely corresponds to a local optima. In order to obtain a potentially better solution to the given task, we can employ an approximate search scheme based on either limited discrepancy search or simulated annealing. Specifically, our approximate LDS and SA based algorithms denoted by ALDS and ASA can be obtained from Algorithms 2 and 3 by replacing the $score(\mathbf{y})$ function with the approximate MAP evaluation scheme described in the previous section. In addition, we can enable algorithms ALDS and ASA to start the search either from a random MAP assignment or from the solution found by algorithm AMAP. Finally, the complexity of algorithms ALDS and ASA at each step during search is dominated by the complexity of approximate marginal inference which can be bounded by $O(2^{2^r})$ where r bounds the number of propositions in a factor node in the corresponding factor graph [18].

4 Experiments

In this section, we evaluate empirically the proposed exact and approximate schemes for MAP and MMAP inference in LCNs. All competing algorithms were implemented[2] in Python 3.10 and used the `ipopt` 3.14 solver [23] with default settings to handle the non-linear constraint programs. We ran all experiments on a 3.0GHz Intel Core processor with 128GB of RAM.

4.1 Random LCNs

We generated three classes of random LCNs with n propositions $\{X_1, \ldots X_n\}$ and sentences of the following types: (a) $l \leq P(x_i) \leq u$, (b) $l \leq P(x_i|x_j) \leq u$, $x_i \neq x_j$ and (c) $l \leq P(x_i|X_j \wedge X_k) \leq u$, $X_i \neq X_j \neq X_k$, such that the corresponding primal graph is a `polytree`, a `dag` or a `random` graph. The type (c) sentences were generated for all truth values of propositions X_j and X_k, namely $P(x_i|x_j)$, $P(x_i|\neg x_j)$, $P(x_i|x_j \wedge x_k)$, $P(x_i|x_j \wedge \neg x_k)$, $P(x_i|\neg x_j \wedge x_k)$ and $P(x_i|\neg x_j \wedge \neg x_k)$, respectively. The probability bounds l and u were selected uniformly at random between 0 and 1 such that $u - l \leq 0.6$, and we ensured that all problem instances with $n \leq 10$ were consistent.

Tables 1 and 2 summarize the results obtained for `maximax` MAP queries on the random LCNs. For each problem class we consider both smaller ($5 \leq n \leq 10$) – i.e., Table 1 – and larger ($30 \leq n \leq 70$) scale instances – i.e., Table 2, respectively. We report the average CPU time in seconds and number of problem instance solved (out of 10) for each problem size. A '-' indicates that the respective algorithm exceeded the 2 hour time limit. The maximum discrepancy value use by algorithms LDS and ALDS was set to $\delta = 3$, while algorithms SA and ASA used up to 30 flips over a single iteration. Note that for the larger instance (Table 2 we also ran ALDS with the smallest discrepancy value $\delta = 1$.

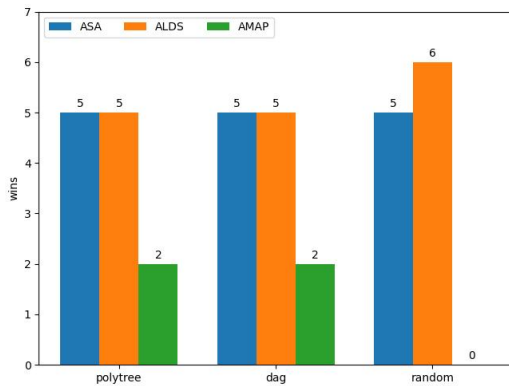



Figure 4: Number of wins for LCNs with $n = 10$.

We can see that the algorithms using exact MAP assignment evaluations (i.e., DFS, LDS and SA) are limited to small scale problem instances with up to 8 propositions and they run out of time on the larger instances. This is caused by the prohibitively large computational overhead incurred during search for solving exactly the non-linear constraint program that encodes each of the MAP assignments. In contrast, the approximate search algorithms ALDS and, especially ASA, can scale to much larger problem instances due to the less expensive approximate MAP assignment evaluations. AMAP is the best performing algorithm in terms of running time and number of problems solved for all reported problem sizes. However, since the solution found by AMAP is only a local maxima, in Figure 4 we report on the solution quality found by algorithms AMAP, ALDS and ASA on LCN instances of size 10. Specifically, we show the number of wins as the number of times (out of 10)

[2]The open-source implementation of LCNs is available at: `https://github.com/IBM/LCN`

10

Table 1: Results for maximax MAP tasks obtained on smaller `polytree`, `dag`, and `random` LCNs. Average CPU time in seconds $\pm$ standard deviation and number of problem instances solved. Time limit is 1 hour.

size	exact MAP evaluation			approximate MAP evaluation		
n	DFS	LDS(3)	SA	AMAP	ALDS(3)	ASA
polytree						
5	15.30±9.37 (10)	26.07±19.26 (10)	20.18±7.66 (10)	2.87±0.55 (10)	174.17±19.81 (10)	188.27±21.25 (10)
6	71.01±27.06 (10)	77.69±40.11 (10)	62.95±40.64 (10)	4.33±1.33 (10)	386.44±44.37 (10)	309.40±56.69 (10)
7	514.16±251.38 (10)	658.02±315.34 (10)	167.24±113.35 (10)	5.85±1.32 (10)	678.41±140.58 (10)	365.50±125.82 (10)
8	3246.28±456.58 (4)	3072.18±360.02 (4)	1199.51±453.31 (10)	8.05±1.59 (10)	1054.53±306.54 (10)	518.18±185.77 (10)
9	- (0)	- (0)	- (0)	10.36±3.76 (10)	1597.66±489.80 (10)	723.42±189.83 (10)
10	- (0)	- (0)	- (0)	11.81±3.66 (10)	2273.16±734.23 (10)	813.30±218.12 (10)
dag						
5	21.09± 17.10 (10)	15.66±15.64 (10)	24.04±16.23 (10)	5.54±1.34 (10)	163.02±14.45 (10)	156.34±30.72 (10)
6	61.98± 44.64 (10)	52.44±38.64 (10)	51.27±36.24 (10)	9.79±2.01 (10)	352.28±29.45 (10)	287.60±37.53 (10)
7	340.10±179.97 (10)	341.34±203.77 (10)	165.45±119.55 (10)	10.44±1.44 (10)	642.25±99.85 (10)	354.35±90.33 (10)
8	1633.38±567.88 (8)	1958.16±752.57 (9)	633.77±303.56 (10)	13.05±3.19 (10)	1339.71±295.81 (10)	571.55±121.61 (10)
9	- (0)	- (0)	2231.88±358.74 (3)	15.13±5.07 (10)	2060.69±425.99 (10)	714.80±151.70 (10)
10	- (0)	- (0)	- (0)	15.55±6.15 (10)	2903.05±831.48 (10)	944.17±287.20 (10)
random						
5	19.51±13.81 (10)	17.56±9.08 (10)	20.37±15.95 (10)	5.26±0.64 (10)	152.99±11.59 (10)	143.60±19.22 (10)
6	119.13±69.94 (10)	99.14±39.21 (10)	60.13±44.33 (10)	6.93±1.46 (10)	322.49±38.44 (10)	217.25±46.39 (10)
7	659.83±377.79 (10)	584.53±333.61 (10)	196.01±124.34 (10)	8.04±2.30 (10)	583.56±99.44 (10)	290.28±115.01 (10)
8	3152.57±0.00 (1)	3209.54±591.11 (5)	1226.88±810.58 (10)	10.29±3.48 (10)	954.46±220.67 (10)	444.17±169.52 (10)
9	- (0)	- (0)	1524.48±919.19 (3)	11.19±3.59 (10)	1480.52±343.05 (10)	600.84±155.34 (10)
10	- (0)	- (0)	- (0)	12.21±4.87 (10)	2150.27±520.21 (10)	717.75±164.64 (10)

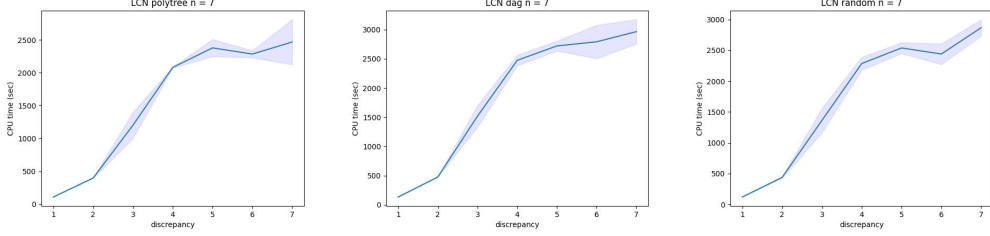



Figure 5: Average CPU time in seconds and standard deviation vs discrepancy δ for ALDS(δ) on maximax MAP tasks. Time limit 1 hour.

each algorithm found the best solution. In this case, algorithms ALDS and ASA were initialized with the MAP assignment found by AMAP. We can see that almost always the search-based approaches ALDS and ASA are able to find better solutions than AMAP. This is important in practice, especially on larger scale problems where we can use AMAP to find a MAP solution quickly and, subsequently, refine that solution using a search-based algorithm like ALDS or ASA if the time budget wasn't exceeded. Finally, in Figures 6 and 5 we show the impact of the maximum discrepancy value δ on the running time of algorithms LDS(δ) and ALDS(δ), respectively. It is easy to see that as the discrepancy value δ increases, the search space explored by both LDS(δ) and ALDS(δ) becomes larger and, therefore, their corresponding running times increase as well.

4.2 Real-world LCNs

We experimented with a set of more realistic LCNs which were first introduced in [18]. These LCNs were derived from real-world Bayesian networks [24] and contain up to 10 propositions as well as up to 24 sentences of the form $l \leq P(x_i) \leq u$ and $l \leq P(x_i|\pi_i) \leq u$, respectively, where x_i is the positive literal of proposition X_i and $\pi_i = y_{i1} \wedge \cdots \wedge y_{ik}$ is the conjunction of the positive or negative literals corresponding to a particular configuration of the parents $\{Y_{i1}, \ldots Y_{ik}\}$ of X_i in the Bayesian network. The specification of these LCNs is given below. Table 5 reports the results obtained on 10 LCN instances for the `maximax` MMAP task with 4 MAP propositions and no evidence. As before, algorithms DFS, LDS(3) and SA which rely on exact evaluations of the MAP assignments during search can only solve the smallest problem instances within the 2 hour time limit. In contrast, algorithms ALDS(3) and ASA solve all problem instances due to a much reduced overhead associated with the approximate MAP assignment evaluations. In this case, the search spaces explored by ALDS(3) and ASA are approximately the same in size and therefore the corresponding running times are comparable. AMAP is the fastest algorithm in this case as well.

11

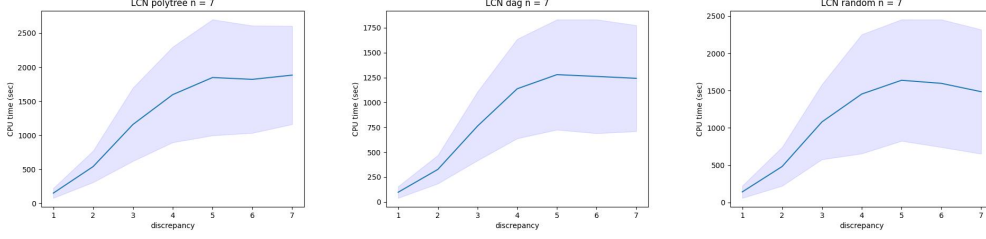



Figure 6: Average CPU time in seconds and standard deviation vs discrepancy δ for LDS(δ) on maximax MAP tasks. Time limit 1 hour.

Table 2: Results for maximax MAP tasks obtained on larger `polytree`, `dag`, and `random` LCNs. Average CPU time in seconds $\pm$ standard deviation and number of problem instances solved. Time limit is 2 hours.

size		approximate MAP evaluation		
n	AMAP	ALDS(1)	ALDS(3)	ASA
		polytree		
30	31.55±5.19 (10)	2707.27±199.78 (10)	- (0)	3091.74±113.82 (10)
50	52.30±11.85 (10)	7527.53±765.31 (6)	- (0)	5324.71±197.07 (10)
70	79.28±16.01 (10)	- (0)	- (0)	7279.56±423.03 (10)
		dag		
30	49.94±9.65 (10)	3219.12±213.77 (10)	- (0)	3593.71±241.37 (10)
50	89.13±11.67 (10)	- (0)	- (0)	5639.90±815.94 (10)
70	132.34±19.03 (10)	- (0)	- (0)	6093.28±496.45 (10)
		random		
30	40.54±7.45 (10)	2984.91±167.25 (10)	- (0)	3335.14±211.39 (10)
50	76.83±14.37 (10)	7114.88±9.17 (2)	- (0)	5276.93±746.95 (10)
70	105.70±21.61 (10)	- (0)	- (0)	6059.57±946.44 (7)

Table 3: Results for maximax MAP tasks obtained on realistic LCNs. CPU time in seconds. Time limit is 2 hours.

LCN (n)	exact MAP evaluation			approximate MAP evaluation		
	DFS	LDS(3)	SA	AMAP	ALDS(3)	ASA
Toy	1.87	1.82	1.29	1.19	133.55	129.62
Earth	15.41	13.37	3.93	1.56	308.67	258.01
Cancer	45.97	31.14	10.87	3.87	333.12	348.28
Asia	-	4102.55	945.12	6.40	1906.52	1853.78
Credit	-	-	-	7.12	6558.84	-
Engine	-	-	1909.28	8.13	3312.54	3521.72
Suicide	-	-	-	8.01	4707.68	6325.47
Tank	-	-	-	10.74	5190.90	-
Alarm	-	-	-	7.83	4855.23	6806.79
Hepatitis	-	-	-	8.22	5237.19	332.17

Finally, Tables 6, 3 and 4 report similar results for `maximin` MMAP task with 4 MAP propositions, as well as `maximax` and `maximin` MAP tasks on the same set of real-world LCN instances.

4.3 Application to Factuality in Language Models

Factuality in large language models (LLMs) refer to their capacity to generate content aligned with factual information, drawing from sources such as dictionaries, Wikipedia, textbooks, and reliable repositories. For our purpose, we consider an application of MMAP inference for LCNs to assess the factuality of the output A generated by an LLM in response to a user query Q (e.g., the summary of a larger document). More specifically, let A be the LLM's output to the user query Q and let C be a trusted external source of information such as Wikipedia that may contain contradicting factual information. The goal is to compute the *factuality score* of the response A,

12

Table 4: Results for maximin MAP tasks obtained on realistic LCNs. CPU time in seconds. Time limit is 2 hours.

LCN (n)	exact MAP evaluation			approximate MAP evaluation		
	DFS	LDS(3)	SA	AMAP	ALDS(3)	ASA
Toy	1.82	1.77	1.57	0.83	109.13	107.31
Earth	15.90	11.79	3.20	1.42	256.62	252.32
Cancer	40.66	27.93	8.50	3.05	266.98	190.74
Asia	-	3333.16	987.76	3.73	1682.79	2046.00
Credit	-	-	-	5.21	5662.80	-
Engine	-	-	5286.17	7.19	2832.00	3303.15
Suicide	-	-	-	6.50	4078.62	6090.35
Tank	-	-	-	9.13	4442.92	5371.48
Alarm	-	-	-	5.22	4210.59	6722.45
Hepatitis	-	-	-	7.62	4494.66	5056.21

Table 5: Results for maximax MMAP tasks with 4 MAP propositions obtained on realistic LCNs. CPU time in seconds. Time limit is 2 hours.

Network	exact MAP evaluation			approximate MAP evaluation		
	DFS	LDS(3)	SA	AMAP	ALDS(3)	ASA
Toy	2.20	3.18	1.85	0.85	134.83	141.17
Earth	9.19	7.67	2.75	1.28	150.99	162.35
Cancer	16.34	14.09	8.52	2.64	157.92	159.66
Asia	811.82	800.18	312.10	4.07	187.44	201.76
Credit	-	6719.30	2976.55	5.09	204.77	222.52
Engine	4786.12	4502.34	2033.77	6.57	212.61	235.70
Suicide	-	-	-	5.99	220.31	203.68
Tank	-	-	-	8.04	263.65	281.73
Alarm	-	-	-	4.28	216.19	186.67
Hepatitis	-	-	-	8.22	260.38	250.45

Table 6: Results for maximin MMAP tasks with 4 MAP propositions obtained on realistic LCNs. CPU time in seconds. Time limit is 2 hours.

Network	exact MAP evaluation			approximate MAP evaluation		
	DFS	LDS(3)	SA	AMAP	ALDS(3)	ASA
Toy	1.82	1.73	1.41	0.68	87.60	101.56
Earth	6.44	8.48	4.29	1.05	101.21	108.92
Cancer	30.91	17.66	6.87	2.21	105.98	91.40
Asia	781.24	503.52	394.01	2.89	128.18	136.44
Credit	6661.35	5961.40	3295.50	4.33	137.96	146.98
Engine	3847.36	3619.16	2048.40	6.24	156.46	155.63
Suicide	-	-	-	5.87	149.65	159.45
Tank	-	-	-	7.06	172.14	173.48
Alarm	-	-	-	4.95	157.05	146.08
Hepatitis	-	-	-	7.99	176.87	178.51

denoted by $f_C(A)$, in the context of the external information C. In the following, we assume that the response A can be decomposed into a set of n *atoms* (or *atomic facts*) $A = \{A_1, \ldots, A_n\}$ (e.g., a straightforward way to decompose A is to split it into sentences) and for each atom A_i up to k relevant passages $\{C_{i1}, \ldots, C_{ik}\}$ called *contexts* can be retrieved from C. Therefore, we have a total of $m = n \times k$ contexts. Subsequently, we consider textual relationships between two atoms $r(A_i, A_j)$, an atom and a context $r(A_i, C_{ij})$, and two contexts $r(C_{ij}, C_{pq})$, respectively, where $r(\cdot) \in \{\text{entail}, \text{contradict}, \text{neutral}\}$. We define an LCN $\mathcal{L}$ with $n + m$ propositions, one for each atom and context, and two types of sentences corresponding to the entailment and contradiction relationships only, as follows: $l \leq P(Y|X) \leq u$ if X entails Y (i.e., $r(X, Y) = \text{entail}$), and $l \leq P(\neg Y|X) \leq u$ if X contradicts Y (i.e., $r(X, Y) = \text{contradict}$), respectively, where X and Y are the propositions corresponding to the atoms and/or contexts involved in the textual relationships $r(X, Y)$. The probability bounds l and u can be estimated from the NLI scores associated with the

13

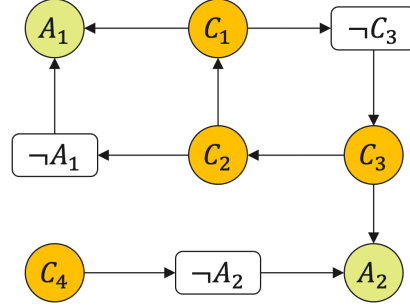
$$0.6 \le P(A_1|C_1) \le 0.8$$
$$0.8 \le P(\neg A_1|C_2) \le 0.9$$
$$0.5 \le P(C_1|C_2) \le 0.6$$
$$0.8 \le P(C_2|C_3) \le 0.9$$
$$0.8 \le P(A_2|C_3) \le 0.9$$
$$0.5 \le P(\neg A_2|C_4) \le 0.6$$
$$0.3 \le P(\neg C_3|C_1) \le 0.5$$

(a) LCN sentences   (b) Primal graph

Figure 7: An example of factuality encoded as an LCN.

Table 7: Results for factuality LCNs. Average CPU time in seconds $\pm$ standard deviation and number of problem instances solved. Time limit is 2 hours.

size	exact MAP evaluation				approx MAP evaluation	
$n, k = 2$	DFS	LDS(2)	SA	AMAP	ALDS(2)	ASA
2	56.95±24.42 (10)	57.37±21.80 (10)	60.09±24.06 (10)	0.31±0.05 (10)	5.25±0.95 (10)	4.13±0.42 (10)
4	-	-	-	0.98±0.12 (10)	80.07±36.83 (10)	54.15±16.40 (10)
6	-	-	-	1.97±0.21 (10)	453.88±190.06 (10)	219.57±69.27 (10)
10	-	-	-	7.33±4.18 (10)	2713.90±334.66 (10)	928.28±249.32 (10)
20	-	-	-	28.42±14.34 (10)	-	3809.23±335.94 (10)
50	-	-	-	379.18±99.10 (10)	-	-
100	-	-	-	1807.10±352.79 (10)	-	-

textual relationships between the respective atoms and contexts (in practice, we may want to run multiple NLI models to obtain reliable probability estimates for entailment and contradiction). In this case, the factuality score $f_C(A)$ can be calculated as the proportion of true atoms from the MAP assignment obtained by solving a `maximax` MMAP task over the LCN encoding where the MAP propositions are the propositions corresponding to the response's atoms.

In Figure 7 we show a simple example with 2 atoms $\{A_1, A_2\}$ and 4 contexts $\{C_1, C_2, C_3, C_4\}$ such that contexts C_1 and C_2 are relevant to A_1, while contexts C_3 and C_4 are relevant to A_2, respectively. The LCN sentences from Figure 7a encode 4 entailment and 3 contradiction relationships between atoms and contexts. The primal graph of the LCN is given in Figure 7b. In this case, the MAP propositions of the `maximax` MMAP task are $\{A_1, A_2\}$ and the corresponding MAP assignment is $(A_1, \neg A_2)$ meaning that the first atom is factually correct while the second atom is not. Therefore, the factuality score is calculated as 0.5.

Table 7 summarizes the results obtained on randomly generated factuality LCNs. More specifically, for each reported problem size $n \in \{2, 4, 6, 10, 20, 50, 100\}$, we generated 10 random instances with n atoms and $k = 2$ contexts per atom such that 10% of all possible pairwise relationships between atoms and contexts were selected to be either *entailment* or *contradiction* with probability 0.5 while the remaining relationships were labeled as *neutral* and thus ignored. The lower and upper probability bounds l and u in the corresponding LCN sentences were also generated randomly between 0 and 1 such that $u - l \le 0.6$. In this case, the maximum discrepancy values was set to 2 and simulated annealing was allowed a single iteration and 30 flips. We see again that algorithms DFS, LDS(2) and SA can only solve the smallest instances due to large computational overhead associated with exact evaluation of the MAP assignments. In contrast, algorithms ALDS(2) and ASA which rely on less expensive approximate evaluations of the MAP assignments can scale to larger problems with up to 20 atoms. Algorithm AMAP outperforms its competitors and solves all problem instances.

5 Conclusions

The paper considers solving abductive reasoning tasks such as generating MAP and Marginal MAP (MMAP) explanations in Logical Credal Networks (LCNs), a recently introduced probabilistic logic framework for reasoning with imprecise knowledge. Since an LCN encodes a set of distributions over its interpretations, a complete or partial explanation of the evidence (i.e., a MAP assignment)

14

may correspond to more than one distribution. Therefore, we define the maximin/maximax MAP and MMAP tasks for LCNs as finding complete or partial MAP assignments that have maximum lower/upper probability given the evidence. We propose several search algorithms that combine depth-first search, limited-discrepancy search or simulated annealing with exact evaluations of the MAP assignments using marginal inference for LCNs. We also develop an approximate message-passing scheme as well as extend limited discrepancy search and simulated annealing to use an approximate evaluation of the MAP assignments during search. Our experiments with random LCNs and LCNs derived from realistic usecases demonstrate conclusively that the search methods based on exact evaluations of the MAP assignments are limited to small size problems, while the approximation schemes can scale to much larger problems. For future work we plan to investigate more advanced depth-first branch-and-bound and best-first search techniques. However, these kinds of methods require developing novel heuristic bounding schemes to guide the search more effectively [16].

Acknowledgements

Fabio Cozman thanks CNPq (grant 305753/2022-3) and the Center for AI at Universidade de São Paulo, funded by FAPESP (grant 2019/07665-4) and IBM.

References

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

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

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

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

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

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

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

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

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

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

[11] 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.

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

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

[14] Radu Marinescu and Rina Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

[15] Radu Marinescu and Rina Dechter. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1492–1524, 2009.

15

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

[17] Radu Marinescu, Debarun Bhattacharjya, Junkyu Lee, Alexander Gray, and Fabio Cozman. Credal marginal map. In *37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[18] 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.

[19] William Harvey and Matthew Ginsberg. Limited discrepancy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 607–613, 1995.

[20] Richard Korf. Improved limited discrepancy search. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 286–291, 1996.

[21] Scott Kirkpatrick, Daniel Gelatt, and Mario Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[22] P. Pardalos and S. Vavasis. Quadratic programming with one negative eigenvalue is (strongly) np-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.

[23] 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.

[24] 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.

16