Supplementary Material - Credal Marginal MAP

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

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

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

Alexander Gray
IBM Research, USA
alexander.gray@ibm.com

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

1 Preliminaries

1.1 Bayesian Networks

A *Bayesian network* (BN) [1] is defined by a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, G \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of variables over multi-valued domains $\mathbf{D} = \{D_1, \ldots, D_n\}$, G is a directed acyclic graph (DAG) over $\mathbf{X}$ as nodes, and $\mathbf{P} = \{P_i\}$ where $P_i = P(X_i|\Pi_i))$ are *conditional probability tables* (CPTs) associated with each variable X_i and Π_i are the parents of X_i in G. A Bayesian network represents a joint probability distribution over $\mathbf{X}$, namely $P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\Pi_i)$.

Let $\mathbf{X}_M = \{X_1, \ldots, X_m\}$ be a subset of $\mathbf{X}$ called MAP variables and $\mathbf{X}_S = \mathbf{X} \setminus \mathbf{X}_M$ be the complement of $\mathbf{X}_M$, called sum variables. The Marginal MAP (MMAP) task seeks an assignment $\mathbf{x}_M^*$ to variables $\mathbf{X}_M$ having maximum probability. This requires access to the marginal distribution over $\mathbf{X}_M$, which is obtained by summing out variables $\mathbf{X}_S$:

$$\mathbf{x}_M^* = \operatorname*{argmax}_{\mathbf{X}_M} \sum_{\mathbf{X}_S} \prod_{i=1}^{n} P(X_i|\Pi_i) \tag{1}$$

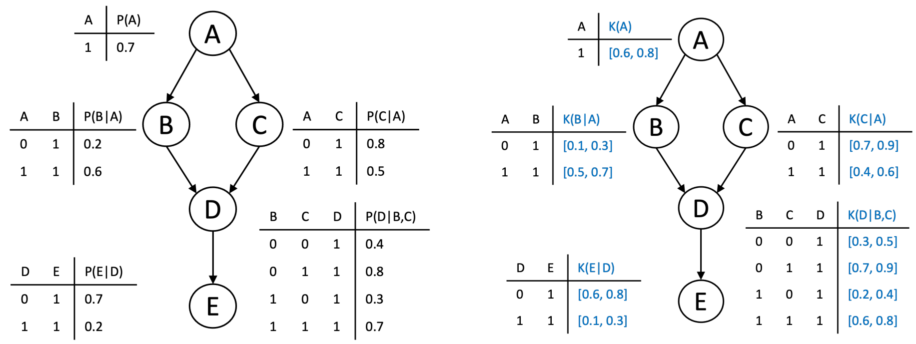
MMAP is a mixed inference task (max-sum) and its complexity is known to be $\mathrm{NP}^{\mathrm{PP}}$-complete [2]. Over the past decades, several algorithmic schemes have been developed for solving MMAP efficiently. We later overview the most relevant exact and approximate algorithms for MMAP.

Given a Bayesian network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, G,$ its *moral graph* G_m is the undirected graph obtained from the directed acyclic graph G by connecting the parents of each variable X_i with undirected edges and removing the direction on the edges in G.

Definition 1 (induced width)**.** *An ordered graph is a pair (G, d) where G is an undirected graph, and $d = (X_1, ..., X_n)$ is an ordering of the nodes. The* width *of a node is the number of the node's neighbors that precede it in the ordering. The* width of an ordering d *is the maximum width over all nodes. The* induced width *of an ordered graph, denoted by w_d^*, is the width of the induced ordered graph obtained as follows: nodes are processed from last to first; when node X_i is processed, all its preceding neighbors are connected. The induced width of a graph, denoted by w^*, is the minimal induced width over all its orderings.*

Marginal MAP requires processing along *constrained elimination orderings* that constrain the sum variables to be processed before the MAP variables. In this case, the induced width obtained for a constrained elimination ordering is called the *constrained induced width*.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).



(a) Bayesian network

(b) Credal network

Figure 1: Examples of Bayesian and credal networks with bi-valued variables.

1.2 Credal Networks

A set of probability distributions for variable X is called a *credal set* and is denoted by $K(X)$ [3]. Similarly, a *conditional credal set* is a set of conditional distributions, obtained by applying Bayes rule to each distribution in a credal set of joint distributions [4]. We consider credal sets that are closed and convex with a finite number of vertices. Two credal sets $K(X|Y = y_1)$ and $K(X|Y = y_2)$, where $y_1 \neq y_2$ are two values in variable Y's domain, are called *separately specified* if there is no constraint on the first set that is based on the properties of the second set.

Two variables X and Y are *strongly independent* when every extreme point of $K(X, Y)$ satisfies standard stochastic independence of X and Y that is $P(X|Y) = P(X)$ and $P(X|Y) = P(Y)$, respectively. Furthermore, *strong conditional independence states* that X and Y are strongly independent conditional on Z when every extreme point of $K(X, Y|Z = z)$ satisfies standard stochastic independence conditional on every value z of Z.

A *credal network* (CN) [5] is defined by a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G\rangle$, where $\mathbf{X} = \{X_1, \dots, X_n\}$ is a set of discrete variables with finite domains $\mathbf{D} = \{D_1, \dots, D_n\}$, G is a directed acyclic graph (DAG) over $\mathbf{X}$ as nodes, and $\mathbf{K} = \{K(X_i|\Pi_i = \pi_{ik})\}$ is a set of separately specified conditional credal sets for each variable X_i and each configuration π_{ik} of its parents Π_i in G. The *strong extension* $K(\mathbf{X})$ 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 [5].

$$K(\mathbf{X}) = CH\{P(\mathbf{X})\ :\ P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\Pi_i, P(X_i|\Pi_i = \pi_{ik}) \text{ is a vertex of } K(X_i|\Pi_i = \pi_{ik})\} \tag{2}$$

Example 1. *Figure 1a shows a simple Bayesian network with 5 bi-valued variables* $\{A, B, C, D, E\}$*. The conditional probability tables are shown next to the nodes. For example, we have that* $P(B = 1|A = 0) = 0.2$*,* $P(B = 0|A = 0) = 0.8$*,* $P(B = 1|A = 1) = 0.6$ *and* $P(B = 0|A = 1) = 0.4$*, respectively. In Figure 1b we show a credal network defined over the same set of variables. In this case, the conditional credal sets associated with the variables are given by closed probability intervals such as, for example,* $0.1 \leq P(B = 1|A = 0) \leq 0.3$, $0.7 \leq P(B = 0|A = 0) \leq 0.9$, $0.5 \leq P(B = 1|A = 1) \leq 0.7$ *and* $0.3 \leq P(B = 0|A = 1) \leq 0.5$*, respectively.*

Unlike in Bayesian networks, in credal networks it is no longer the case that there is a unique marginal distribution corresponding to a MAP assignment. Therefore, we define the following two *Credal Marginal MAP* (CMMAP) tasks:

Definition 2 (maximin)**.** *Let* $\mathcal{C} = \langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G\rangle$ *be a credal network whose variables are partitioned into MAP variables* $\mathbf{X}_M$ *and sum variables* $\mathbf{X}_S = \mathbf{X} \setminus \mathbf{X}_M$*. The* maximin Credal Marginal MAP

Algorithm 1 Variable Elimination for Credal Marginal MAP

1: **procedure** CVE($\mathcal{C}, \mathbf{X}_M, \mathbf{X}_S$)
2: initialize $\Gamma \leftarrow \emptyset$
3: **for all** variable $X_i \in \mathbf{X}$ **do**
4: $\phi = \{p : p \in ext(K(X_i|\Pi_i))\}$
5: update $\Gamma = \Gamma \cup \{\phi\}$
6: create constrained elimination ordering o
7: **for all** variable $X_i \in o$ **do**
8: $\Gamma_{X_i} = \{\phi : \phi \in \Gamma, X_i \in vars(\phi)\}$
9: update $\Gamma = \Gamma \setminus \Gamma_{X_i}$
10: **for all** variable $X_i \in o$ **do**
11: **if** $X_i \in \mathbf{X}_S$ **then**
12: $\psi = \max\left(\sum_{X_i} \prod\{\phi \in \Gamma_{X_i}\}\right)$
13: **else**
14: $\psi = \max\left(\max_{X_i} \prod\{\phi \in \Gamma_{X_i}\}\right)$
15: let $Y \in vars(\phi)$ be the closest to X_i
16: update $\Gamma_Y = \Gamma_Y \cup \{\psi\}$
17: initialize $\mathbf{x}_M^* = \emptyset$
18: **for all** variable $X_i \in reversed(o)$ **do**
19: **if** $X_i \in \mathbf{X}_M$ **then**
20: $x_i^* = \mathrm{argmax}_{X_i} \prod\{\phi(\mathbf{x}_M^*) \in \Gamma_{X_i}\}$
21: $\mathbf{x}_M^* = \mathbf{x}_M^* \cup \{X_i = x_i^*\}$
22: **return** $\mathbf{x}_M^*$

task is finding the assignment $\mathbf{x}_M^*$ *to* $\mathbf{X}_M$ *with maximum lower marginal probability, namely:*

$$\mathbf{x}_M^* = \operatorname*{argmax}_{\mathbf{X}_M} \min_{P(\mathbf{X}) \in K(\mathbf{X})} \sum_{\mathbf{X}_S} \prod_{i=1}^{n} P(X_i|\Pi_i) \tag{3}$$

Definition 3 (maximax)**.** *Let* $\mathcal{C} = \langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G \rangle$ *be a credal network whose variables are partitioned into MAP variables* $\mathbf{X}_M$ *and sum variables* $\mathbf{X}_S = \mathbf{X} \setminus \mathbf{X}_M$*. The* maximax Credal Marginal MAP *task is finding the assignment* $\mathbf{x}_M^*$ *to* $\mathbf{X}_M$ *with maximum upper marginal probability, namely:*

$$\mathbf{x}_M^* = \operatorname*{argmax}_{\mathbf{X}_M} \max_{P(\mathbf{X}) \in K(\mathbf{X})} \sum_{\mathbf{X}_S} \prod_{i=1}^{n} P(X_i|\Pi_i) \tag{4}$$

It can be shown that solving CMMAP is $\mathrm{NP}^{\mathrm{PP}}$-hard [6]. CMMAP is applicable to probabilistic diagnosis [7], counterfactual analysis [8] or uncertainty quantification in machine learning [9].

2 Exact Credal MMAP

2.1 Variable Elimination

Algorithm 1 describes our variable elimination procedure for CMMAP which extends the exact method developed previously for marginal inference tasks [10] and operates on *potentials*.

Definition 4 (potential)**.** *Given a set of variables* $\mathbf{Y}$*, a* potential $\phi(\mathbf{Y})$ *is a set of non-negative real-valued functions* $p(\mathbf{Y})$ *on* $\mathbf{Y}$*. The product of two potentials* $\phi(\mathbf{y})$ *and* $\psi(\mathbf{Z})$ *is defined by* $\phi(\mathbf{Y}) \cdot \psi(\mathbf{Z}) = \{p \cdot q : p \in \phi, q \in \psi\}$*. The sum-marginal* $\sum_{\mathbf{Z}} \phi(\mathbf{Y})$ *and the max-marginal* $\max_{\mathbf{Z}} \phi(\mathbf{Y})$ *of a potential* $\phi(\mathbf{Y})$ *with respect to a subset of variables* $\mathbf{Z} \subseteq \mathbf{Y}$ *are defined by* $\sum_{\mathbf{Z}} \phi(\mathbf{Y}) = \{\sum_{\mathbf{Z}} p(\mathbf{Y}) : p \in \phi\}$ *and* $\max_{\mathbf{Z}} \phi(\mathbf{Y}) = \{\max_{\mathbf{Z}} p(\mathbf{Y}) : p \in \phi\}$*, respectively.*

Since the multiplication operator may grow the size of potentials dramatically, we introduce an additional pruning operation that can reduces the cardinality of a potential. Specifically, the operator $\max \phi(\mathbf{Y})$ returns the set of non-zero maximal elements of $\phi(\mathbf{Y})$, under the partial order $\geq$ defined component wise as $p \geq q$ iff $\forall \mathbf{y}_k \in \mathbf{D}_{\mathbf{Y}}, p(\mathbf{y}_k) \geq q(\mathbf{y}_k)$, where $\mathbf{D}_{\mathbf{Y}}$ is the cartesian product of the domains of the variables in $\mathbf{Y}$: $\max \phi(\mathbf{Y}) = \{p \in \phi(\mathbf{Y}) : \nexists q \in \phi, q \geq p\}$.

Given a credal network $\mathcal{C} = \langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G \rangle$ as input together with a partitioning of its variables into disjoint subsets $\mathbf{X}_M$ (as MAP variables) and $\mathbf{X}_S$ (as sum variables) algorithm CVE transforms each conditional credal set $K(X_i|\Pi_i)$ into a corresponding potential that contains the set of all conditional probability distributions in the strong extension of $K(X_i|\Pi_i)$ (lines 3–5). Subsequently, given an ordering o of the variables in which all the MAP variables come after the summation variables, the potentials are partitioned into buckets. A bucket is associated with a single variable X_i and contains all of the un-allocated potentials that have X_i as an argument (lines 6–9). The algorithm then processes each bucket, from first to last, by multiplying all potentials in the current bucket

Algorithm 2 Depth-First Search Credal Marginal MAP

procedure DFS($\mathcal{C}$, $\mathbf{X}_M \subseteq \mathbf{X}$)
  initialize $\mathbf{x}_M^* \leftarrow \emptyset$, $best \leftarrow -\infty$
  SEARCH($\emptyset$)
  return $\mathbf{x}_M^*$
procedure SEARCH($\mathbf{x}$, $\mathbf{X}_M$)
  if $size(\mathbf{x}) == size(\mathbf{X}_M)$ **then**
    $score(\mathbf{x}) \leftarrow CVE^{+}(\mathbf{x})$
    if $score(\mathbf{x}) > best$ **then**
      $\mathbf{x}_M^* \leftarrow \mathbf{x}$, $best \leftarrow score(\mathbf{x})$
  else
    select uninstantiated variable $X_i \in \mathbf{X}_M$
    for all values x_i of X_i **do**
      $\mathbf{x} \leftarrow \mathbf{x} \cup \{X_i = x_i\}$
      SEARCH($\mathbf{x}$)

and eliminating the bucket's variable (by summation for sum variables, and by maximization for MAP variables), 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 (lines 10–16). Following the top-down elimination phase, a bottom-up pass over the MAP buckets, from the last to the first MAP variable in the ordering, assembles the solution $\mathbf{x}_M^*$ by selecting the value x_i^* of variable X_i that maximizes the combination of potentials in its bucket, conditioned on the already assigned MAP variables in the ordering (lines 18–21). Note that the bucket X_i's combined potential may contain more than one components. In this case, we choose the value x_i^* that maximizes the largest number of components in that potential (breaking ties arbitrarily). Clearly, we have the following complexity result:

Theorem 1 (complexity)**.** *Given a credal network $\mathcal{C}$, the complexity of algorithm CVE is time and space $O(n \cdot C \cdot k^{w_o^*})$, where n is the number of variables, k bounds the domain sizes, w_o^* is the induced width of the constrained elimination order o and C bounds the cardinality of the potentials.*

Proof. Let $o = (X_1, ..., X_n)$ be a constrained ordering of the variables such that all sum variables appear before the MAP variables. Based on previous work on bucket elimination [11], the scope size of the intermediate potentials generated during variable elimination is bounded by the induced width of the constrained elimination ordering (i.e., the constrained induced width) denoted by w_o^*. Let ψ_{X_i} be the potential generated by eliminating variable X_i (either by summation or by maximization). Clearly, the size of each component p_j of ψ_{X_i} is bounded exponentially by w_o^*, namely $O(k^{w_o^*})$ where k is the maximum domain size. Since the cardinality of the potentials is bounded by C, it follows easily that eliminating variable X_i is time and space bounded by $O(C \cdot k^{w_o^*})$. Moreover, since there are at most n elimination steps, we have that CVE is bounded time and space by $O(n \cdot C \cdot k^{w_o^*})$. $\square$

2.2 Depth-First Search

An alternative approach to solving CMMAP exactly, described by procedure DFS in Algorithm 2, is to conduct a depth-first search over the space of partial assignments to the MAP variables, and, for each complete MAP assignment $\mathbf{x}_M$ compute its score as the exact upper probability $\overline{P}(\mathbf{x}_M)$. This way, the optimal solution $\mathbf{x}_M^*$ corresponds to the configuration with the highest score. Evaluating $\overline{P}(\mathbf{x}_M)$ can be done by using a simple modification of the CVE algorithm described in the previous section. Specifically, given a complete assignment $\mathbf{x}_M$ to the MAP variables, the modified CVE, denoted by CVE$^+$, computes an *unconstrained* elimination ordering of all the variables regardless of whether they are MAP or summation variables. Then, for each MAP variable X_i and corresponding value $x_i \in \mathbf{x}_M$, CVE$^+$ adds to the bucket of X_i a deterministic potential $\phi(X_i) = \{\delta_{x_i}\}$, where δ_{x_i} returns one if $X_i = x_i$ and zero otherwise. Finally, CVE$^+$ eliminates all variables by summation and obtains the desired upper probability bound after processing the bucket of the last variable in the ordering. The following complexity result holds:

Theorem 2 (complexity)**.** *Given a credal network $\mathcal{C}$, the complexity of the depth-first search algorithm is time $O(n \cdot C \cdot k^{m+w_u^*})$ and space $O(n \cdot C \cdot k^{w_u^*})$, where n is the number of variables, m is the number of MAP variable, k is the maximum domain size, w_u^* bounds the induced width of the unconstrained elimination ordering u and C bounds the cardinality of the potentials.*

Proof. The size of the search space defined by the MAP variables is bounded by $O(k^m)$ where m is the number of MAP variables and k is the maximum domain size. The evaluation of each complete

Algorithm 3 Mini-Buckets for Credal Marginal MAP

1: **procedure** CMBE($\mathcal{C}$, $\mathbf{X}_M$, $\mathbf{X}_S$, i-bound)
2: create constrained elimination ordering o
3: initialize buckets Γ_{X_i} as in Algorithm 1
4: **for all** variable $X_i \in o$ **do**
5: create mini-buckets $\{Q_1, \ldots, Q_l\}$ of Γ_{X_i}
6: **for all** mini-bucket Q_j, $j \in \{1, \ldots, l\}$ **do**
7: **if** $X_i \in \mathbf{X}_S$ **then**
8: $\psi = \max \sum_{X_i} \prod \{\phi \in Q_j\}$
9: **else**
10: $\psi = \max \max_{X_i} \prod \{\phi \in Q_j\}$
11: let $Y \in vars(\phi)$ be the closest to X_i
12: update $\Gamma_Y = \Gamma_Y \cup \{\psi\}$
13: generate $\mathbf{x}_M^*$ as in Algorithm 1
14: **return** $\mathbf{x}_M^*$

MAP assignment involves solving exactly a summation subproblem defined over n variables (m of them already instantiated and acting as evidence, and $(n-m)$ uninstantiated representing the sum variables). Let w_u^* be the induced width of an unconstrained variable ordering. Clearly, the complexity of evaluating a MAP assignment is bounded time and space by $O(n \cdot C \cdot k^{w_u^*})$, where C bounds the cardinality of the potentials involved in the summation subproblem. Therefore, it follows that the complexity of the DFS algorithm is time $O(n \cdot C \cdot k^{m+w_u^*})$ and space $O(n \cdot C \cdot k^{w_u^*})$. $\square$

3 Approximate Credal Marginal MAP

Solving the CMMAP task exactly is computationally hard and does not scale to large problems. Therefore, in this section, we present several schemes to approximate CMMAP using the mini-bucket partitioning as well as stochastic local search combined with approximate credal marginal inference.

3.1 Mini-Buckets Approximation

The first approximation method is described by Algorithm 3 and adapts the mini-bucket partitioning scheme developed for graphical models [12] to the CMMAP task. Specifically, algorithm CMBE(i) is parameterized by an i-bound i and works by partitioning large buckets into smaller subsets, called *mini-buckets*, each containing at most i distinct variables (line 5). The mini-buckets are processed separately, as follows: MAP mini-buckets (in $\mathbf{X}_M$) are eliminated by maximization, while variables in $\mathbf{X}_S$ are eliminated by summation. In practice, however, for variables in $\mathbf{X}_S$, one (arbitrarily selected) is eliminated by summation, while the rest of the mini-buckets are processed by maximization. Clearly, CMBE(i) outputs an upper bound on the optimal maximax CMMAP value from Equation 4.

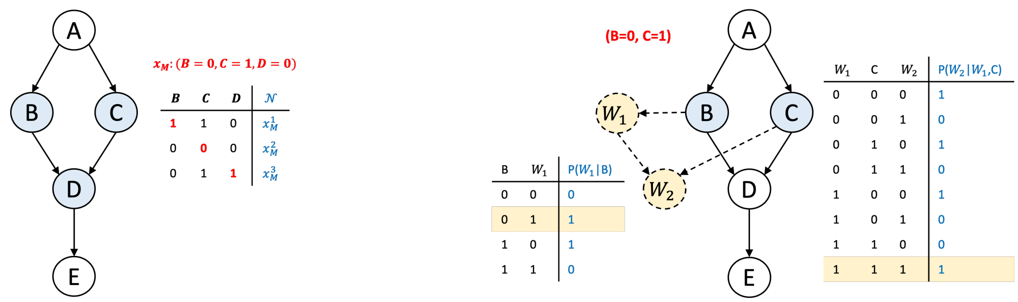
Theorem 3 (complexity). *Given a credal network $\mathcal{C}$, the complexity of algorithm CMBE(i) is time and space $O(n \cdot C \cdot k^i)$, where n is the number of variables, k is the maximum domain size, i is the mini-bucket i-bound and C bounds the cardinality of the potentials.*

Proof. Based on previous work on mini-bucket approximation [12], the scope size of each intermediate potential is bounded by the i-bound i. Therefore, the complexity bound of each intermediate step is bounded time and space by $O(C \cdot k^i)$, where k is the maximum domain size and C bounds the cardinality of the potential. Since there are at most n elimination steps, the time and space complexity of algorithm CMBE(i) is $O(n \cdot C \cdot k^i)$. $\square$

3.2 Local Search

The basic idea behind any local search scheme is to start from an initial guess $\mathbf{x}_M$ as the solution (i.e., a complete assignment to the MAP variables), and iteratively try to improve the solution by moving to a better neighbor $\mathbf{x}'_M$ such that $\overline{P}(\mathbf{x}'_M) > \overline{P}(\mathbf{x}_M)$ (or, alternatively, $\underline{P}(\mathbf{x}'_M) > \underline{P}(\mathbf{x}_M)$ for maximin). A *neighbor* of instantiation $\mathbf{x}_M$ is defined as the instantiation which results from changing the value of a single variable X in $\mathbf{x}_M$. For example, if variables have bi-valued domains, then we have $|\mathbf{X}_M|$ neighbors in this case. Figure 2a shows the neighbors of $\mathbf{x}_M : (B=0, C=1, D=0)$ for the credal network from Figure 1b.

In order to perform local search efficiently, we need to compute the scores for all the neighbors efficiently. Therefore, computing the score of a neighbor, $\overline{P}(\mathbf{x}'_M)$, requires estimating the upper



(a) Neighbors of $(B = 0, C = 1, D = 0)$. (b) Augmented CN for $(B = 0, C = 1)$.

Figure 2: Examples of neighbors and an augmented credal network for given MAP assignments.

probability of the evidence represented by $\mathbf{x}'_M$. This can be done efficiently using any of the approximation schemes developed for marginal inference in credal networks such as L2U [13] or ApproxLP [14]. However, since these schemes were originally designed to compute the marginal lower and upper probabilities of a query variable $Z = z$ conditioned on evidence $\mathbf{Y} = \mathbf{y}$, we use the following transformation of the credal network to evaluate the probability of evidence $P(\mathbf{Y} = \mathbf{y})$.

Let $\mathcal{C} = \langle \mathbf{X}, \mathbf{D}, \mathbf{K}, G \rangle$ be a credal network and let $\mathbf{Y} = \mathbf{y}$ be the evidence set, where $\mathbf{Y} = \{Y_1, \dots, Y_k\}$ and $\mathbf{y} = (y_1, \dots, y_k)$, respectively. We construct an *augmented* credal network $\mathcal{C}' = \langle \mathbf{X}', \mathbf{D}', \mathbf{K}', G' \rangle$ by adding set of bi-valued variables $\mathbf{W} = \{W_1, \dots, W_k\}$ and deterministic conditional probability tables $P(W_1|Y_1)$ and $P(W_j|W_{j-1}, Y_j)$, for all $2 \leq j \leq k$, such that $P(W_1 = 1|Y_1 = y_1) = 1$, $P(W_1 = 1|Y_1 \neq y_1) = 0$, $P(W_j = 1|W_{j-1} = 1, Y_j = y_j) = 1$, and $P(W_j = 1|W_{j-1}, Y_j \neq y_j) = 0$, respectively. It is easy to see that computing $\underline{P}(\mathbf{Y} = \mathbf{y})$ and $\overline{P}(\mathbf{Y} = \mathbf{y})$ in $\mathcal{C}$ is equivalent to computing the posterior marginals $\underline{P}(W_k = 1)$ and $\overline{P}(W_k = 1)$ in the augmented network $\mathcal{C}'$, respectively. For illustration, Figure 2b shows the augmented credal network corresponding to the assignment $(B = 0, C = 1)$ in the network from Figure 1b.

Stochastic Hill Climbing Our first method is based on Stochastic Hill Climbing and is described by procedure SHC in Algorithm 4. More specifically, SHC proceeds by repeatedly either changing the state of the variable that creates the maximum score change (line 13), or changing a variable at random (lines 9 and 15). The quality of the solution returned by the method depends to a large extent on which part of the search space it is given to explore. Therefore, our scheme restarts the search from a different initial solution which is initialized uniformly at random (lines 3-4).

Taboo Search Our second procedure denoted by TS in Algorithm 4 implements a Taboo Search approach for credal Marginal MAP. Specifically, Taboo search is similar to stochastic hill climbing except that the next neighbor of the current state is chosen as the best neighbor that hasn't been visited recently. A taboo list maintains a portion of the previously visited states so that at the next step a unique point is selected. Our TS algorithm implements a random restarts scheme.

Simulated Annealing Procedure SA in Algorithm 4 describes our Simulated Annealing based scheme for credal Marginal MAP. The basic principle behind this approach is to consider some neighboring state $\mathbf{x}'_M$ of the current state $\mathbf{x}_M$, and probabilistically decides between moving to state $\mathbf{x}'_M$ or staying in the current state. The probability of making the transition from $\mathbf{x}_M$ to $\mathbf{x}'_M$ is specified by an acceptance probability function $P(\mathbf{x}'_M, \mathbf{x}_M, T)$ that depends on the scores of the two states as well as a global time-varying parameter T called *temperature*. We chose $P(\mathbf{x}'_M, \mathbf{x}_M, T) = e^{\frac{\Delta}{T}}$, where $\Delta = \log \overline{P}(\mathbf{x}'_M) - \log \overline{P}(\mathbf{x}_M)$. At each iteration, the temperature is decreased using a cooling schedule σ (e.g., $\sigma = 0.9$). Like SHC and TS, algorithm SA implements a random restarts strategy.

Theorem 4 (complexity)**.** *Given a credal network $\mathcal{C}$, the complexity of algorithms SHC, TS and SA is time $O(N \cdot M \cdot P)$ and space $O(n)$, where n is the number of variables, N is the number of iterations, M is the maximum number of flips allowed per iteration, and P bounds the complexity of approximating the probability of evidence in $\mathcal{C}$.*

Algorithm 4 Local Search for Credal Marginal MAP

procedure SHC($\mathcal{C}$, $\mathbf{X}_M \subseteq \mathbf{X}$, p_{flip})
  initialize $\mathbf{x}_M^* \leftarrow \emptyset$, $best \leftarrow -\infty$
  for all iterations $i = 1 \ldots N$ **do**
    initialize $\mathbf{x}_M$ randomly
    for all flips $j = 1 \ldots M$ **do**
      sample randomly $p \in (0, 1)$
      let $\mathcal{N}$ be $\mathbf{x}_M$'s neighbors
      if ($p \leq p_{flip}$) **then**
        select random neighbor $\mathbf{x}'_M \in \mathcal{N}$
      else
        for all neighbor $\mathbf{x}'_M \in \mathcal{N}$ **do**
          compute $score(\mathbf{x}'_M)$
        select highest score neighbor $\mathbf{x}''_M \in \mathcal{N}$
        if $score(\mathbf{x}''_M) \leq score(\mathbf{x}_M)$ **then**
          select random neighbor $\mathbf{x}'_M \in \mathcal{N}$
        else
          select $\mathbf{x}'_M \leftarrow \mathbf{x}''_M$
      if $score(\mathbf{x}'_M) > best$ **then**
        $\mathbf{x}_M^* \leftarrow \mathbf{x}'_M$
        $best \leftarrow score(\mathbf{x}'_M)$
      $\mathbf{x}_M \leftarrow \mathbf{x}'_M$
  return $\mathbf{x}_M^*$
procedure TS($\mathcal{C}$, $\mathbf{X}_M \subseteq \mathbf{X}$)
  initialize $\mathbf{x}_M = \emptyset$, $best \leftarrow -\infty$
  for all iterations $i = 1 \ldots N$ **do**
    initialize $\mathbf{x}_M$ randomly
    $\mathcal{T} \leftarrow \emptyset$
    for all flips $j = 1 \ldots M$ **do**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_M\}$
      let $\mathcal{N}$ be $\mathbf{x}_M$'s neighbors
      initialize $\mathbf{x}''_M \leftarrow \emptyset$, $b \leftarrow -\infty$
      for all neighbor $\mathbf{x}'_M \in \mathcal{N}$ **do**
        if $\mathbf{x}'_M \notin \mathcal{T}$ and $score(\mathbf{x}'_M) > b$ **then**
          $\mathbf{x}''_M \leftarrow \mathbf{x}'_M$
          $b \leftarrow score(\mathbf{x}'_M)$
      if $\mathbf{x}''_M = \emptyset$ **then**
        select random neighbor $\mathbf{x}'_M \in \mathcal{N}$
      else
        $\mathbf{x}'_M \leftarrow \mathbf{x}''_M$
      if $score(\mathbf{x}'_M) > best$ **then**
        $\mathbf{x}^*_M \leftarrow \mathbf{x}'_M$
        $best \leftarrow score(\mathbf{x}'_M)$
      $\mathbf{x}_M \leftarrow \mathbf{x}'_M$
      if $size(\mathcal{T}) \geq S$ **then**
        prune $\mathcal{T}$ until $size(\mathcal{T}) < S$
  return $\mathbf{x}^*_M$
procedure SA($\mathcal{C}$, $\mathbf{X}_M \subseteq \mathbf{X}$, $temp$, σ)
  initialize $\mathbf{x}^*_M$ randomly
  $best \leftarrow score(\mathbf{x}^*_M)$
  for all iterations $i = 1 \ldots N$ **do**
    set $\mathbf{x}_M \leftarrow \mathbf{x}^*_M$, $T \leftarrow temp$
    for all flips $j = 1 \ldots M$ **do**
      let $\mathcal{N}$ be $\mathbf{x}_M$'s neighbors
      select random neighbor $\mathbf{x}'_M \in \mathcal{N}$
      $\Delta \leftarrow \log score(\mathbf{x}'_M) - \log score(\mathbf{x}_M)$
      if $\Delta > 0$ **then**
        $\mathbf{x}_M \leftarrow \mathbf{x}'_M$
      else
        sample randomly $p \in (0, 1)$
        if $p < e^{\frac{\Delta}{T}}$ **then**
          $\mathbf{x}_M \leftarrow \mathbf{x}'_M$
      if $score(\mathbf{x}_M) > best$ **then**
        $\mathbf{x}^*_M \leftarrow \mathbf{x}_M$
        $best \leftarrow score(\mathbf{x}_M)$
      $T \leftarrow T * \sigma$
  return $\mathbf{x}^*_M$

Proof. Assuming for example the L2U based approximation of probability of evidence in the input credal network $\mathcal{C}$ [13], its complexity can be bounded time by $O(t \cdot n \cdot e \cdot 2^p)$ and space by $O(n \cdot e)$, where n is the number of nodes in G, e is the number of edges in G, p bounds the size of a node's family in G. Denoting by P the time complexity of L2U, then it follows that the time complexity of local search algorithms is $O(N \cdot M \cdot P)$ where N is the number of iterations and M is the maximum number of flips per iteration. The space complexity is clearly linear and bounded by $O(n)$. □

4 Additional Experiments

We evaluate the proposed algorithms for CMMAP on random credal networks and credal networks derived from real-world applications. All competing algorithms were implemented in C++ and the experiments were run on a 32-core machine with 128GB of RAM running Ubuntu Linux 20.04.

We consider the two exact algorithms denoted by CVE and DFS, as well as the four approximation schemes denoted by SHC, TS, SA and CMBE(i), respectively. The local search algorithms used $N = 10$ iterations and $M = 10,000$ maximum flips per iteration, and they all used the approximate L2U algorithm with 10 iterations [13] to evaluate the MAP assignments during search. Furthermore, for SHC we set the flip probability p_{flip} to 0.2, TS used a taboo list of size 100, while for SA we set the initial temperature and cooling schedule to $T_{init} = 100$ and $\sigma = 0.9$, respectively. For CMBE(i) we set the i-bound i to 2 and used the same L2U algorithm to evaluate the solution found. All competing algorithms were allocated a 1 hour time limit and 8GB of memory per problem instance.

In all our experiments, we report the CPU time in seconds, the number of problems solved within the time/memory limit and the number of times an algorithm converged to the best possible solution. The latter is called the number of *wins* and is meant to be a measure of solution quality for the respective algorithm. We also record the number of variables (n), the number (or percentage) of MAP variables (Q) and the constrained induced widths (w^*). The best performance points are highlighted.

4.1 Random Credal Networks

For our purpose, we generated random credal networks, m-by-m grid networks as well as k-tree networks. Specifically, for the random networks, we varied the number of variables $n \in \{100, 150, 200\}$, for grids, we choose $m \in \{10, 14, 16\}$, and for k-trees we selected $k = 2$ and the number of variables $n \in \{100, 150, 200\}$, respectively. In all cases, the maximum domain size was set to 2 and the conditional credal sets were generated uniformly at random as probability intervals such that the difference between the lower and upper probability bounds was at most 0.3.

First, we note that the exact algorithms CVE and DFS could only solve very small problems with up to 10 variables and 5 MAP variables. The main reason for the poor performance of these algorithms is the extremely large size of the intermediate potentials generated during the variable elimination procedure which causes the algorithms to run out of memory or time on larger problems. Therefore, we omit their evaluation hereafter.

Table 1 summarizes the results obtained on random, grid and k-tree networks. Each data point represents an average over 100 random problem instances generated for each problem size (n) and number of MAP variables (Q), respectively. Next to the running time we show the number of instances solved within the time/memory limit (if the number is omitted then all instances were solved). We can see that in terms of running time, CMBE(2) performs best on the grid networks. This is because the intermediate potentials generated during elimination are relatively small size and therefore are processed quickly. However, the algorithm is not able converge to good quality solutions compared with its competitors. The picture is reversed on the random and k-tree networks where CMBE(2) is the worst performing algorithm both in terms of running time and solution quality. In this case, the relatively large intermediate potentials cause the algorithm to exceed the time and memory limits on many problem instances and thus impact negatively its performance.

The local search algorithms SHC, TS and SA yield the best performance in terms of solution quality with all three algorithms almost always converging to the best possible solutions on these problem instances. In terms of running time, SA is the fastest algorithm achieving almost one order of magnitude speedup over its competitors, especially for larger numbers of MAP variables (e.g., k-trees with $n = 100$ variables and $Q = 60$ MAP variables). Algorithms SHC and TS have comparable running times (with SHC being slightly slower than TS) but they are significantly slower than SA. This likely caused by the significantly larger computational overhead required for evaluating the scores of all the neighbors of the current state, especially when there are many MAP variables.

Table 2 summarizes the results obtained on random credal networks for the maximin CMMAP case. We can see that the resutls display a similar pattern with those for the maximax case from Table 1.

4.2 Real-World Credal Networks

Table 5 shows the results obtained on a set of credal networks derived from 22 real-world Bayesian networks[1] by converting the probability values in the CPTs into probability intervals such that the difference between the corresponding lower and upper probability bounds was at most 0.3. Furthermore, since the local search algorithms rely on the L2U approximation to evaluate the MAP configurations, we restricted the domains of the multi-valued variables to the first two values in the domain while shrinking and re-normalizing the corresponding CPTs. For each network we selected uniformly at random $Q = 50\%$ of the variables to act as MAP variable and generated 10 random instances. The number of variables for these networks is recorded in Table 3. As before, we indicate next to the average running times the number of instances solved by the respective algorithms within

[1]Available at https://www.bnlearn.com/bnrepository/

n	#Q	w^*	SHC time (#)	W	TS time (#)	W	SA time (#)	W	CMBE(2) time (#)	W
random										
100	20	25	32.69±7.99	100	23.08±2.30	100	6.47±2.30	100	225.28±407.06 (70)	1
	40	37	163.05±38.99	100	79.11±14.12	100	14.78±5.38	100	327.67±612.07 (43)	0
	60	23	421.93±117.26	100	185.41±38.82	100	29.99±9.52	100	224.93±299.38 (7)	0
150	30	39	254.32±31.36	100	141.03±15.60	100	24.08±3.86	100	294.48±587.48	0
	60	57	1143.78±127.75	100	531.45±58.89	100	70.06±8.69	100	555.98±827.29	0
	90	66	2811.47±336.31	100	1259.79±117.77	100	139.78±15.68	75	925.41±912.05	0
200	50	58	1044.79±116.91	100	490.38±50.70	100	72.09±8.29	100	276.98±392.39	0
	100	86	3496.77±181.34	32	2143.79±211.97	100	211.47±25.25	14	927.31±0.00	0
	150	69	3601.67±2.14	16	3550.99±79.44	17	339.34±38.06	72	-	0
grid										
100	20	25	31.35±9.94	100	22.77±7.77	100	4.66±1.65	100	0.07±0.05	2
	40	37	155.34±45.51	100	79.83±25.55	100	10.51±3.18	100	3.85±24.27	0
	60	23	358.81±93.73	100	168.79±50.13	100	19.18±6.35	100	28.76±244.41	0
144	30	36	219.49±23.99	100	121.86±11.73	100	21.02±1.98	100	0.34±0.85	0
	60	53	878.63±104.22	100	426.70±49.57	100	54.77±5.59	100	1.03±1.84	0
	90	26	2109.47±197.10	100	958.13±108.68	100	102.93±10.20	73	27.79±225.99	0
196	50	55	817.52±94.16	100	382.46±45.32	100	58.13±6.37	100	0.68±0.90	0
	100	56	3045.54±436.60	94	1453.39±155.86	100	147.11±10.73	13	51.01±368.96	0
	150	22	3601.25±1.95	23	3011.47±410.05	93	190.26±14.20	3	41.27±145.47	2
ktree										
100	20	25	68.25±22.23	100	44.25±14.10	100	10.48±3.67	100	221.18±526.97	0
	40	37	307.91±91.63	100	151.97±50.63	100	23.19±8.36	100	163.59±396.07	0
	60	23	650.72±170.18	100	306.26±94.47	100	40.19±13.59	100	-	0
150	30	28	443.33±48.49	100	245.71±20.98	100	44.58±4.17	100	492.55±932.29 (26)	0
	60	47	1647.29±223.33	100	724.01±102.54	100	106.71±12.41	100	14.68±0.00 (1)	0
	90	51	2917.01±531.54 (84)	84	1541.91±208.37	100	192.76±18.32	82	- (0)	0
200	50	45	1306.43±238.31	100	660.59±91.77	100	108.36±13.82	100	1199.83±1029.86	0
	100	64	3376.59±121.62 (54)	54	1917.95±388.87	100	266.24±23.09	21	- (0)	0
	1500	48	3602.98±4.05 (4)	4	3334.49±229.35 (59)	59	344.96±28.40	38	- (0)	0

Table 1: Results on random, grid and k-tree credal networks. Mean CPU times in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximax* CMMAP. Time limit 1 hour, memory limit 8GB of RAM.

the time and memory limits. We can see again that CMBE(2) is competitive only on the easiest instances (e.g., child, mildew) while SA yields the best performance in terms of both running time and solution quality on the majority of the problem instances. In summary, the relatively large potentials hinder CMBE's performance, while the computational overhead incurred during the evaluation of relatively large neighborhoods of the current state slows down significantly SHC and TS compared with SA.

Table 7 reports the results obtained on the 22 networks with $Q = 50\%$ MAP variables for the maximin CMMAP case. Similarly, Tables 4 and 6 summarize the results obtained on the real-world networks with $Q = 25\%$ MAP variables for the maximax and the maximin CMMAP cases, respectively. We can see that the results show the same pattern as those from Table 5.

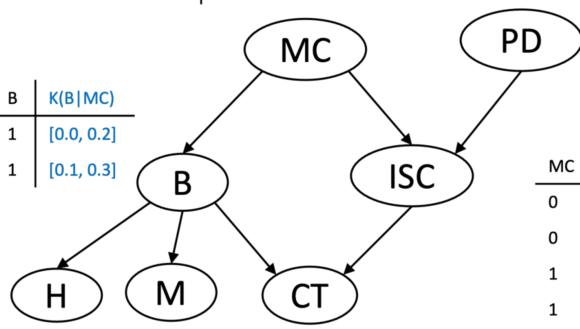
4.3 Applications

Figure 3 shows the credal network for the brain tumour diagnosis use case derived from the Bayesian network described in [15]. The variables are: MC - metastatic cancer, PD - Paget disease, B - brain tumour, ISC - increased serum calcium, H - headaches, M - memory loss, CT - scan result.

Considering the query variables B and ISC, the exact solution for both maximax and maximin CMMAP is $(B = 0, ISC = 0)$ (obtained by both the CVE and DFS algorithms). In this case, the maximax and maximin scores are 0.837 and 0.42, respectively. Algorithms SHC, TS and SA also find the optimal configuration $(B = 0, ISC = 0)$ which is evaluated by L2U to 0.8316 for maximax CMMAP and to 0.37296 for maximin CMMAP, respectively.

Figure 4 shows the credal network for the intelligence report analysis described in [10]. The variables are: As - assassination, C - coup/revolt, R - regime change, D - decision to invade, At - attack, B - build-up, P - propaganda, I - invasion.

Considering the query variables D, At and I, the exact solution for both maximax and maximin CMMAP is $(D = 0, At = 0, I = 0)$ and is obtained by both algorithms CVE and DFS. The corresponding scores are in this case 0.765 and 0.458806, respectively. The approximation schemes



MC	K(MC)
1	[0.1, 0.3]

PD	K(PD)
1	[0.0, 0.2]

MC	B	K(B|MC)
0	1	[0.0, 0.2]
1	1	[0.1, 0.3]

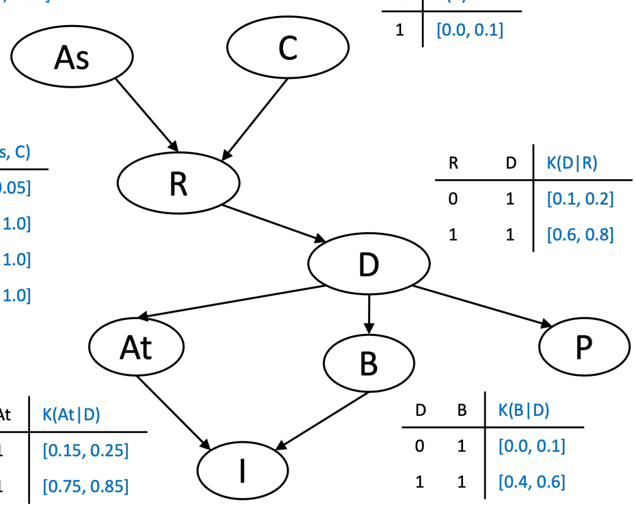
MC	PD	ISC	K(ISC|MC,PD)
0	0	1	[0.1, 0.3]
0	1	1	[0.6, 0.8]
1	0	1	[0.7, 0.9]
1	1	1	[0.8, 1.0]

B	H	K(H|B)
0	1	[0.4, 0.6]
1	1	[0.6, 0.8]

B	M	K(M|B)
0	1	[0.2, 0.4]
1	1	[0.4, 0.6]

B	ISC	CT	K(CT|B,ISC)
0	0	1	[0.0, 0.1]
0	1	1	[0.0, 0.2]
1	0	1	[0.7, 0.9]
1	1	1	[0.8, 1.0]

Figure 3: Credal network for the Brain Tumour diagnosis.



As	K(As)
1	[0.0, 0.05]

C	K(C)
1	[0.0, 0.1]

As	C	R	K(R|As, C)
0	0	1	[0.0, 0.05]
0	1	1	[0.95, 1.0]
1	0	1	[0.95, 1.0]
1	1	1	[0.95, 1.0]

R	D	K(D|R)
0	1	[0.1, 0.2]
1	1	[0.6, 0.8]

D	P	K(P|D)
0	1	[0.4, 0.6]
1	1	[0.9, 1.0]

D	At	K(At|D)
0	1	[0.15, 0.25]
1	1	[0.75, 0.85]

D	B	K(B|D)
0	1	[0.0, 0.1]
1	1	[0.4, 0.6]

At	B	I	K(I|At, B)
0	0	1	[0.0, 0.1]
0	1	1	[0.1, 0.2]
1	0	1	[0.05, 0.1]
1	1	1	[0.8, 1.0]

Figure 4: Credal network for the Attack intelligence report.

n	#Q	w^*	SHC time (#)	W	TS time (#)	W	SA time (#)	W	CMBE(2) time (#)	W
					random					
100	20	25	76.16±7.25	100	52.16±5.16	100	10.04±1.96	100	265.59±669.38 (67)	2
	40	37	361.50±30.47	100	186.75±13.02	100	25.01±3.62	100	389.16±668.72 (34)	0
	60	44	797.28±98.43	100	404.23±38.40	100	48.46±5.32	100	191.01±151.74 (3)	0
150	30	39	275.49±26.51	100	149.96±9.69	100	23.37±3.03	99	226.29±390.17 (48)	0
	60	57	1131.33±466.66 (97)	95	542.28±102.92	100	67.86±13.44	97	603.33±1012.22 (11)	0
	90	66	2630.73±504.23 (97)	97	1166.62±191.76	100	141.99±19.33	82	- (0)	0
200	50	58	1020.76±294.29 (97)	96	495.72±70.67	100	66.53±14.69	96	541.06±823.06 (29)	0
	100	86	3440.58±233.34 (39)	39	2045.59±514.98 (99)	99	202.11±65.25	18	609.13±522.39 (3)	0
	150	69	3605.31±6.27 (18)	17	3520.87±132.30 (26)	26	316.75±95.36	68	- (0)	0
					grid					
100	20	25	57.56±10.27	100	42.28±6.38	100	7.98±1.74	100	0.07±0.07	0
	40	37	324.16±504.02	100	132.12±32.66	100	20.09±4.06	100	0.12±0.09	0
	60	23	667.17±649.35	97	262.16±69.02	100	34.36±9.20	95	35.47±348.60	0
144	30	36	243.26±484.61 (99)	98	102.52±33.82	99	17.76±3.46	98	0.13±0.15	2
	60	53	751.01±583.37 (93)	93	326.01±95.12	100	44.03±11.96	94	0.34±0.80	0
	90	26	1583.69±576.19 (81)	80	670.62±159.36	100	75.69±28.81	62	1.71±6.50	0
196	50	55	576.72±183.48	100	302.31±77.39	100	46.77±7.52	100	0.30±0.20	0
	100	56	2494.47±596.88 (90)	90	959.20±198.18	100	118.59±29.84	13	6.21±54.31	0
	150	22	3602.12±2.70 (22)	19	2486.36±708.78	96	134.59±48.13	2	22.25±99.44 (99)	2
					ktree					
100	20	18	159.04±118.30	100	92.46±8.66	100	20.99±3.80	99	89.58±202.59 (54)	2
	40	31	822.01±820.36 (99)	97	315.42±74.03	100	43.17±9.75	96	13.92±12.96 (7)	0
	60	32	1329.12±848.12 (94)	87	637.93±150.93	100	68.69±26.24	87	- (0)	0
150	30	28	500.28±314.29	100	254.92±24.43	100	43.02±6.80	99	161.23±427.67 (20)	0
	60	47	1677.08±757.66 (88)	86	783.84±143.94	100	97.89±32.43	90	7.37±0.00 (1)	0
	90	51	2683.21±624.66 (52)	51	1487.89±285.65	100	152.45±81.03	60	- (0)	0
200	50	45	1396.06±370.51 (94)	94	727.15±82.52	100	103.90±22.73	96	1158.34±564.21 (2)	0
	100	64	3466.46±129.44 (40)	38	2075.95±487.57 (99)	99	227.81±101.48	17	- (0)	0
	1500	48	3606.52±5.78 (11)	11	3467.11±152.83 (43)	43	237.44±156.28	57	- (0)	0

Table 2: Results on random, grid and k-tree credal networks. Mean CPU times in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximin* CMMAP. Time limit 1 hour, memory limit 8GB of RAM.

SHC, TS and SA also find the same optimal CMMAP configuration $(D = 0, At = 0, I = 0)$ which is evaluated by L2U to 0.69651 for maximax CMMAP and to 0.305486 for maximin CMMAP, respectively.

We note that in both cases, the constrained induced width is 2 and therefore CMBE(2) coincides with the exact CVE. Therefore, all our approximation schemes found the optimal solutions.

References

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

[2] James Park. MAP complexity results and approximation methods. In *Uncertainty in Artificial Intelligence (UAI)*, pages 388–396, 2002.

[3] Isaac Levi. *The Enterprise of Knowledge*. MIT Press, 1980.

[4] Peter Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, UK, 1991.

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

[6] Cassio Campos and Fabio Cozman. The inferential complexity of Bayesian and credal networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1313–1318, 2005.

[7] Gregory Cooper. Nestor: A computer-based medical diagnosis aid that integrates causal and probabilistic knowledge. Technical report, Computer Science department, Stanford University, Palo-Alto, California, 1984.

[8] Marco Zaffalon, Alessandro Antonucci, and Rafael Cabañas. Structural causal models are (solvable by) credal networks. In *European Workshop on Probabilistic Graphical Models*, 2020.

problem	n
alarm	37
child	20
link	724
insurance	27
hepar2	70
pathfinder	109
hailfinder	56
largefam	1997
mastermind1	1220
mastermind2	2288
mastermind3	3692
mildew	35
munin	1041
pedigree1	334
pedigree7	1068
pedigree9	1118
win95pts	76
xandes	223
xdiabetes	413
zbarley	48
zpigs	441
zwater	32

Table 3: Number of variables for the real-world networks.

problem	w^*	SHC		TS		SA		CMBE	
		time (#)	W	time (#)	W	time (#)	W	time (#)	W
alarm	8	3.92±0.71	10	4.43±0.69	10	1.81±0.24	10	294.42±36.55	0
child	5	0.91±0.08	10	1.34±0.18	10	0.37±0.04	10	0.02±0.01	0
link	173	3612.23±13.19 (5)	5	3622.01±6.16 (3)	2	827.79±43.95	5	- (0)	0
insurance	8	3.98±0.38	10	4.57±0.49	10	3.32±0.26	10	17.84	0
hepar2	16	428.06±24.73	10	277.24±21.11	10	82.73±	10	- (0)	0
pathfinder	18	911.28±99.63	10	21.91±1.21	10	60.70±10.01	10	- (0)	0
hailfinder	12	27.43±3.92	10	14.20±5.01	10	3.99±1.49	10	268.53±44.19	1
largefam	313	- (0)	0	- (0)	0	2406.27±269.09	10	- (0)	0
mastermind1	297	- (0)	0	- (0)	0	3078.89±365.56	5	228.11±258.14 (5)	5
mastermind2	559	- (0)	0	- (0)	0	3603.02±0.43 (2)	2	- (0)	0
mastermind3	908	- (0)	0	- (0)	0	3601.56±0.0 (1)	1	- (0)	0
mildew	8	3.96±0.26	10	4.38±0.43	10	2.29±0.16	10	0.30±0.41	3
munin	203	3600.00±0.0 (1)	0	3600.00±0.0 (4)	4	483.73±21.68	4	37.55pm9.12 (3)	2
pedigree1	77	3598.53±3.67 (5)	5	1593.69±229.87	10	224.76±34.80	10	867.15±281.98	0
pedigree7	191	-	0	-	0	1085±121.73	2	433.92±1006.64 (8)	8
pedigree9	203	3600.86±0.0 (1)	0	- (0)	0	1104.05±102.45	0	33.72±36.79 (9)	9
win95pts	18	2774.38±413.86	10	1621.45±204.85	10	445.95±82.33	10	- (0)	0
xandes	54	3608.36±4.41	10	3605.52±4.20	10	2906.51±888.74	0	- (0)	0
xdiabetes	85	2820.95±392.70	10	1077.20±217.63	10	104.12±33.10	0	137.63±133.95 (2)	0
zbarley	12	27.26±4.91	10	19.06±4.56	10	9.02±1.44	10	384.42±129.99	1
zpigs	89	3240.72±340.71	10	1160±133.09	10	105.87±15.47	0	115.44±0.0 (1)	0
zwater	12	18.11±6.01	10	21.83±3.87	10	10.70±3.08	10	- (0)	0

Table 4: Results on real-world credal networks with $Q = 25\%$ MAP variables. Mean CPU time in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximax* CMMAP. Time limit 1 hour, 8GB of RAM.

[9] Eyke Hüllermeier, Sébastien Destercke, and Mohammad Hossein Shaker. Quantification of credal uncertainty in machine learning: A critical analysis and empirical comparison. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 548–557. PMLR, 01–05 Aug 2022.

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

[11] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

problem	w^*	SHC time (#)	W	TS time (#)	W	SA time (#)	W	CMBE time (#)	W
alarm	12	27.32±3.33	10	21.31±1.95	10	4.89±0.56	10	324±174.99	0
child	7	3.51±0.59	10	3.69±0.46	10	1.19±0.09	10	0.64±1.83	0
link	239	3655±17.76	2	3628.15±1.52	1	1300.14±79.99	8	-	0
insurance	12	64.22±7.58	10	45.77±5.28	10	17.11±0.99	10	97.16±230.11	0
hepar2	25	1734.11±102.70	10	833.82±37.12	10	163.15±22.01	10	- (0)	0
pathfinder	33	2509.89±493.46	10	79.78±13.88	10	93.98±13.92	10	-	0
hailfinder	14	126.60±7.79	10	72.73±7.34	10	12.53±1.42	10	531.52±110.19	0
largefam	402	-	0	-	0	2903.55±195.07	10	-	0
mastermind1	389	-	0	-	0	3600.85	10	-	0
mastermind2	726	-	0	-	0	3617.17±4.22	5	-	0
mastermind3	1193	-	0	-	0	3650.54±20.35	3	-	0
mildew	12	22.49±4.52	10	15.22±3.54	10	3.15±0.82	10	0.16±0.43	0
munin	175	3615.45±15.74	4	3639.57±12.33	3	652.72±25.59	5	-	0
pedigree1	74	3603.87±3.26	7	3609.44±6.66	8	410.61±14.27	0	1269.45±478.10	0
pedigree7	147	3620.74±19.31	1	1689.89±102.59	1	1689.89±102.59	9	-	0
pedigree9	175	-	0	-	0	1719.92±59.88	6	1128.50±993.15	4
win95pts	28	3612.38±7.69	6	3610.94±4.83	6	821.20±145.74	10	-	0
xandes	75	3619.00±11.04	6	3611±8.25	6	3565±103.55	3	-	0
xdiabetes	75	3605.08±4.22	9	3603.76±2.92	10	175.76±18.44	0	182.38±338.30	0
zbarley	18	140.93±6.68	10	82.36±2.71	10	18.47±8.86	10	863.64±0.00	0
zpigs	105	3606.83±4.61	4	3603.8±3.59	5	234.84±13.58	5	100.47±8.08	0
zwater	16	207.07±7.39	10	126.87±8.77	10	44.68±2.43	10	-	0

Table 5: Results on real-world credal networks with $Q = 50\%$ MAP variables. Mean CPU time in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximax* CMMAP. Time limit 1 hour, 8GB of RAM.

problem	w^*	SHC time (#)	W	TS time (#)	W	SA time (#)	W	CMBE time (#)	W
alarm	8	2.57±0.09	10	2.81±0.11	10	1.51±0.22	10	321.21±55.30	0
child	5	0.55±0.03	10	0.77±0.03	10	0.38±0.03	10	0.01±0.01	0
link	173	3601.58±0.89 (2)	2	3613.40±0.00 (1)	1	49.87±8.41	10	- (0)	0
insurance	8	2.07±0.07	10	2.32±0.08	10	2.53±0.58	10	1.77±3.76	0
hepar2	16	1399.11±1190.89	10	395.48±150.11	10	28.23±29.11	4	- (0)	0
pathfinder	18	1111.08±77.54	10	26.79±4.70	10	90.09±3.17	10	- (0)	0
hailfinder	12	19.74±5.64	10	17.92±3.99	10	5.52±0.95	10	598.36±110.52	2
largefam	313	- (0)	0	- (0)	0	318.52±38.98	10	- (0)	0
mastermind1	297	- (0)	0	3600.27 (1)	1	287.21±34.75	10	565.28±915.66 (5)	5
mastermind2	559	- (0)	0	- (0)	0	2962.85±281.46	10	- (0)	0
mastermind3	908	- (0)	0	- (0)	0	3613.59±0.0 (1)	1	- (0)	0
mildew	8	2.42±0.56	10	2.73±0.45	10	1.86±0.20	10	0.19±0.12	5
munin	203	3601.32±0.46 (2)	1	3613.88±0.0 (1)	1	50.04±2.09	7	11.88pm10.99 (3)	3
pedigree1	77	3601.19±0.72 (3)	0	3586.05±244.33 (5)	1	13.42±0.86	0	747.01±236.58	9
pedigree7	191	- (0)	0	- (0)	0	120.42±6.92	3	361.48±799.46 (7)	7
pedigree9	203	- (0)	0	3600.57±0.0 (1)	0	126.79±7.42	0	19.46±17.30	10
win95pts	18	3600.28±0.0 (1)	1	3606.87±5.20 (2)	2	28.29±2.08	10	- (0)	0
xandes	54	3601.63±0.20 (2)	2	3601.28±0.62 (3)	3	1094.33±1640.46	6	- (0)	0
xdiabetes	85	3600.65±0.58 (5)	2	2102.27±361.71	9	37.71±57.13	4	170.86±166.47 (2)	1
zbarley	12	43.30±21.26	10	21.22±4.13	10	8.58±2.57	9	734.30±195.93 (8)	1
zpigs	89	3601.49±0.38 (5)	4	2173.38±299.06	10	40.51±61.89	2	92.51±0.0 (1)	0
zwater	12	18.79±2.54	10	22.18±2.03	10	18.59±2.69	10	- (0)	0

Table 6: Results on real-world credal networks with $Q = 25\%$ MAP variables. Mean CPU time in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximin* CMMAP. Time limit 1 hour, 8GB of RAM.

[12] Rina Dechter and Irina Rish. Mini-buckets: A general scheme of approximating inference. *Journal of ACM*, 50(2):107–153, 2003.

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

[14] Alessandro Antonucci, Yi Sun, Cassio P 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.

[15] Johan Kwisthout. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52(1):1452–1469, 2011.

problem	w^*	SHC		TS		SA		CMBE	
		time (#)	W	time (#)	W	time (#)	W	time (#)	W
alarm	12	157.61±280.73	10	19.48±6.49	10	2.58±1.32	7	328.28±94.58	0
child	7	2.01±0.04	10	2.31±0.06	10	1.01±0.20	10	0.65±1.77	0
link	239	- (0)	0	- (0)	0	143.09±8.89	10	- (0)	0
insurance	12	53.64±9.72	10	37.64±8.86	10	15.41±1.40	10	12.22±23.19 (9)	0
hepar2	25	3601.21±0.79 (8)	7	1888.79±155.99	9	27.31±40.52	6	- (0)	0
pathfinder	33	2909.51±369.03	10	106.33±10.73	10	134.98±14.99	10	- (0)	0
hailfinder	14	212.33±277.89	10	70.94±10.48	10	12.39±1.06	10	650.11±120.27	0
largefam	402	- (0)	0	- (0)	0	490.31±24.04	10	- (0)	0
mastermind1	389	- (0)	0	- (0)	0	812.42±79.77	10	- (0)	0
mastermind2	726	- (0)	0	- (0)	0	3606.25±0.0 (1)	1	- (0)	0
mastermind3	1193	- (0)	0	- (0)	0	3601.45±0.0 (1)	1	- (0)	0
mildew	12	107.96±95.62	10	19.91±2.89	10	5.02±0.40	10	0.18±0.18	0
munin	174	3601.93±0.0 (1)	1	3601.37±0.0 (1)	1	99.98±2.35	10	- (0)	0
pedigree1	74	3602±0.0 (1)	0	3610.61±8.77 (2)	1	29.93±0.68	2	1064.17±244.33 (8)	8
pedigree7	147	3601.96±0.0 (1)	1	- (0)	0	211.73±14.19	10	- (0)	0
pedigree9	175	- (0)	0	- (0)	0	219.91±11.82	8	150.66±116.54 (2)	2
win95pts	28	3601.27±1.21 (2)	2	3604.71±2.53 (2)	2	52.42±1.29	10	- (0)	0
xandes	75	- (0)	0	- (0)	0	51.96±0.78	10	- (0)	0
xdiabetes	75	3602.83±2.69 (2)	1	3604.15±2.41 (4)	2	22.98±0.42	2	55.42±102.97 (6)	6
zbarley	18	1855.64±1543.11	10	91.88±40.42	10	9.29±7.54	5	1387.73±0.0 (1)	0
zpigs	105	3601.30±1.29 (2)	1	3601.81±2.00	10	25.96±0.48	8	35.02±5.45 (2)	2
zwater	16	199.83±11.44	10	113.82±9.49	10	39.97±2.99	10	- (0)	0

Table 7: Results on real-world credal networks with $Q = 50\%$ MAP variables. Mean CPU time in seconds with standard deviations, number of instances solved (#) and number of wins (W) for *maximin* CMMAP. Time limit 1 hour, 8GB of RAM.