

# Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability

Jeremias Berg, Matti Järvisalo, and Brandon Malone

HIIT & Department of Computer Science, University of Helsinki, Finland

**Abstract.** Bayesian network structure learning is the well-known computationally hard problem of finding a directed acyclic graph structure that optimally describes given data. A learned structure can then be used for probabilistic inference. While exact inference in Bayesian networks is in general NP-hard, it is tractable in networks with low treewidth. This provides good motivations for developing algorithms for the NP-hard problem of learning optimal bounded treewidth Bayesian networks (BTW-BNSL). In this work, we develop a novel score-based approach to BTW-BNSL, based on casting BTW-BNSL as weighted partial Maximum satisfiability. We demonstrate empirically that the approach scales notably better than a recent exact dynamic programming algorithm for BTW-BNSL.

**Submitted to LaSh’14 for presentation only.**

**Conference version of this paper appears as**

*Jeremias Berg, Matti Järvisalo, and Brandon Malone. Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability. In Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS 2014), volume 33 of JMLR Workshop and Conference Proceedings, pages 86-95. JMLR, 2014.*

## 1 Introduction

Bayesian networks are an important and widely-used class of probabilistic graphical models for representing joint probability distributions, i.e., probabilistic relationships among a set of variables of interest [1]. A Bayesian network consists of a network structure, represented as an acyclic directed graph (DAG), and the parameters associated with each node (i.e., variable) in the DAG. Most often, a Bayesian network that represents given data well is not known *a priori*, and hence needs to be learned from data. Given a network structure and complete data, determining the parameters of the variables is simple, whereas learning the DAG structure, i.e., the Bayesian network structure learning problem (BNSL), is computationally challenging.

In this work we focus on the BNSL problem within the widely studied score-based framework, in which a score is assigned to each DAG structure, and the goal is to find a best-scoring network. The structure learning problem is NP-complete in general [2], which justified the fact that most early work on BNSL focused on local search algorithms, such as greedy hill climbing in the space of DAGs [3], equivalence classes of DAGs [4], or over variable orderings [5], and local searching over constraint optimization formulations of BNSL [6].

After learning a Bayesian network, the network is typically used for probabilistic inference tasks, such as determining the most likely joint assignments of a set of variables under given evidence. In order to accurately answer such queries, it is important to learn a network that explains the input data well. Throughout the last decade, there has been increasing interest in developing algorithms for *optimally* solving BNSL, and a variety of algorithms which are guaranteed to find a network structure with optimal score have been proposed [7–11].

While exact Bayesian inference is in general NP-hard [12], for *bounded (fixed) treewidth* networks exact inference becomes tractable [13]. This motivates the study of algorithms for the problem of learning optimal bounded treewidth Bayesian networks (BTW-BNSL). Despite the recent progress in practical algorithms for optimally solving BNSL without treewidth constraints, very few practical algorithms have been proposed for learning network structures under restrictions on the treewidth of the networks [14, 15]; the only approach learning optimal bounded treewidth network structures is the recent exact dynamic programming algorithm of [15].

Much like the general BNSL problem, BTW-BNSL is NP-hard [15]: more precisely, BTW-BNSL( $W$ ), the problem of finding an optimal Bayesian network structure of treewidth at most  $W$ , is NP-hard for any *fixed*  $W \geq 2$  (DAGs with  $W = 1$  being trees). Indeed, the restriction on the treewidth of the DAG structures is a *non-trivial* additional constraint over the general BNSL problem, which poses challenges for developing algorithms for BTW-BNSL.

In this work, we develop a novel score-based approach to learning optimal bounded treewidth Bayesian network structures. Our approach is based on casting BTW-BNSL for a given bound  $W$  on the treewidth of the DAG structures of interest as an abstract combinatorial optimization problem. More precisely, we present an intricate encoding of BTW-BNSL as weighted partial Maximum Satisfiability (MaxSAT in short). The encoding ensures that the optimal solutions of the MaxSAT instance encoding an arbitrary instance of BTW-BNSL( $W$ ) correspond to optimal DAG structures wrt a given scoring function. For finding optimal structures using the MaxSAT encoding, we employ a state-of-the-art MaxSAT solver extended to real-valued costs for exactly encoding the local scores. We demonstrate empirically that our approach scales notably better than the recent exact dynamic programming algorithm for BTW-BNSL [15] on standard BNSL benchmarks and for different values of  $W$ . Furthermore, in view of practical efficiency, our approach can benefit from foreseeable future improvements in state-of-the-art MaxSAT solver technology. The approach is applicable under any decomposable scoring function [3], i.e., scoring functions in which the score for an entire network is the sum of the local scores for the chosen parent sets for the individual variables in the network, including e.g. the commonly used scoring functions MDL [16], BD [17, 18], and fNML [19].

## 2 Preliminaries

In order to formally define the problem of learning optimal bounded treewidth Bayesian network structures, we first define necessary concepts related to treewidth and tree-decompositions. We also give necessary background on MaxSAT.

## 2.1 Treewidth

The treewidth of an undirected graph  $G$  is defined in terms of the *tree-decompositions* of  $G$ .

**Definition 1.** A *tree-decomposition* of an undirected graph  $G = (V, E)$  is a tree  $T$  over a set  $\{V_1, \dots, V_m\}$  of nodes, where  $V_i \subseteq V$ , with the following properties.

1.  $\cup_{i=1}^m V_i = V$ .
2. If  $\{u, v\} \in E$ , then  $u, v \in V_i$  for some  $i \in \{1, \dots, m\}$ .
3. For all  $i, j, k \in \{1, \dots, m\}$ , the following holds:  
if  $V_j$  is on the (unique) path from  $V_i$  to  $V_k$  in  $T$ , then  $V_i \cap V_k \subseteq V_j$ .

The width of a tree-decomposition is  $\max_{i=1}^m |V_i| - 1$ .

**Definition 2.** The *treewidth*  $tw(G)$  of an undirected graph  $G = (V, E)$  is the minimum width over all tree-decompositions of  $G$ .

It is well-known that, for any undirected graph  $G = (V, E)$ , any linear ordering of the nodes  $V$  of  $G$  defines a tree-decomposition of  $G$ , and that there is always an “optimal” linear ordering of  $V$  defining an *optimal* tree-decomposition, i.e., a tree-decomposition of width  $tw(G)$  [20, 21]. Furthermore, without needing to explicitly construct the corresponding optimal tree-decomposition, the treewidth of  $G$  can be determined based on an optimal linear ordering  $\prec$  of  $V$ . A node  $v_i \in V$  is a *predecessor* of  $v_j \in V$  under  $\prec$  if  $i \prec j$  and  $\{v_i, v_j\} \in E$ ;  $v_i$  is a *successor* of  $v_j$  under  $\prec$  if  $j \prec i$  and  $\{v_i, v_j\} \in E$ . Given a linear ordering  $\prec$  of  $V$ , the width of the corresponding tree-decomposition is determined by applying the following *triangulation* procedure on  $G$  under  $\prec$ : For each pair  $v_i, v_j$  of nodes in  $V$ , add the edge  $\{v_i, v_j\}$  to  $E$  if  $v_i$  and  $v_j$  have a common predecessor. Repeat this as long as new edges can be added to  $E$ . We denote the resulting edge-relation by  $\Delta(E, \prec)$ , defining the *triangulation*  $\Delta(G, \prec) = (V, \Delta(E, \prec))$  of  $G$  under  $\prec$ . Orienting the edges of  $\Delta(G, \prec)$  according to  $\prec$  gives the directed edge-relation

$$\Delta(E, \prec) = \{(v_i, v_j) \mid \{v_i, v_j\} \in \Delta(E, \prec), i \prec j\}$$

defining the *ordered graph*  $\Delta(G, \prec) = (V, \Delta(E, \prec))$  of  $G$  under  $\prec$ . Now, the width of the tree-decomposition defined by  $\prec$  is

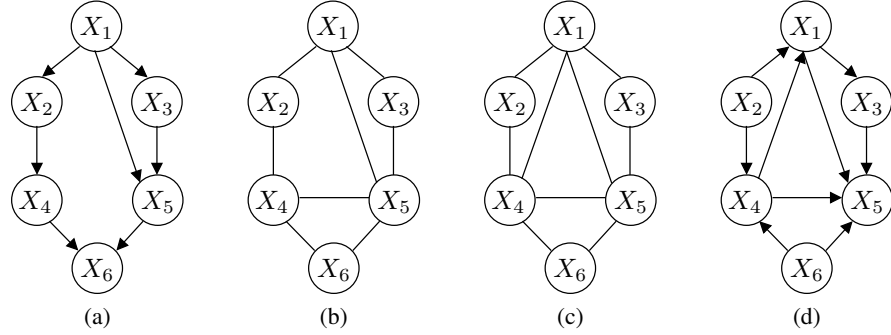
$$\max_{v_i \in V} |\{(v_i, v_j) \in \Delta(E, \prec)\}|, \quad (1)$$

i.e., the maximum number of successors over all nodes in  $\Delta(E, \prec)$ . The treewidth  $tw(G)$  of  $G$  is then

$$\min_{\prec} \max_{v_i \in V} |\{(v_i, v_j) \in \Delta(E, \prec)\}|, \quad (2)$$

over all linear orderings  $\prec$  of the nodes  $V$  of  $G$ .

Before a concrete example of triangulation and ordered graphs, we proceed by defining the treewidth for the DAG structures of Bayesian networks.



**Fig. 1.** Example: (a) a DAG  $G = (X = \{X_1, \dots, X_6\}, E)$ ; (b) the moralized graph  $\text{MORAL}(G) = (X, M(E))$  of  $G$ ; (c) the triangulation  $\Delta(\text{MORAL}(G), \prec)$  of the moralized graph under the linear ordering  $X_6 \prec X_2 \prec X_4 \prec X_1 \prec X_3 \prec X_5$ ; (d) the ordered graph  $\Delta(\text{MORAL}(G), \prec)$ .

## 2.2 Bounded Treewidth Bayesian Network Structure Learning

Given a set  $X = \{X_1, \dots, X_N\}$  of nodes (representing random variables), an element of  $\mathcal{P}_i = 2^{X \setminus \{X_i\}}$  is a *candidate parent set* of  $X_i$ . For a given DAG  $G = (X, E)$ , the parent set of node  $X_i$  is  $\{X_j \mid (X_j, X_i) \in E\}$ , i.e., consists of the parents of  $X_i$  in  $G$ . Picking a single  $P_i \in \mathcal{P}_i$  for each  $X_i$  gives rise to the (not necessarily acyclic) graph in which, for each  $X_i$ , there is an edge  $(X_j, X_i)$  iff  $X_j \in P_i$ .

The treewidth of a Bayesian network structure is defined as the treewidth of the *moralized graph* induced by the DAG structure of the network. This is motivated by the fact that Bayesian inference is tractable in structures whose moralized graph has bounded treewidth, forming the basis for exact join-tree inference algorithms [13].

**Definition 3.** Given a DAG  $G = (X, E)$ , the moralized graph  $\text{MORAL}(G) = (X, M(E))$  induced by  $G$  is an undirected graph defined by the edge relation

$$M(E) = \{ \{X_i, X_j\} \mid (X_i, X_j) \in E \} \cup \{ \{X_i, X_j\} \mid \exists k \text{ s.t. } (X_i, X_k), (X_j, X_k) \in E \}.$$

In words, the moralized graph contains an undirected version of each edge in the DAG, and an edge between every pair of nodes which have a common child in the DAG.

The treewidth of the DAG structure  $G$  of any Bayesian network can be determined by finding a linear ordering  $\prec$  that minimizes Equation 1 for the ordered graph  $\Delta(\text{MORAL}(G), \prec)$  of the moralization  $\text{MORAL}(G)$  of  $G$  under  $\prec$ . We denote by  $\text{TW}(W)$  the class of DAGs having treewidth at most  $W$ .

As an example, Figure 1 illustrates for (a) a given DAG  $G = (X, E)$  (b) the moralized graph  $\text{MORAL}(G)$ , and, for a given linear ordering  $\prec$  of the nodes  $X$ , (c) the triangulation  $\Delta(\text{MORAL}(G), \prec)$  and (d) the ordered graph  $\Delta(\text{MORAL}(G), \prec)$ . For this ordering  $\prec$ , Eq. 1 evaluates to 2, and hence  $G \in \text{TW}(2)$ . In fact, it can be checked that this  $\prec$  defines an optimal tree-decomposition of  $G$ , minimizing Eq. 2, which implies that  $\text{tw}(G) = 2$ .

With these definitions, we can formally state the bounded treewidth Bayesian network structure learning problem (BTW-BNSL) as follows<sup>1</sup>.

*The BTW-BNSL Problem*

**INPUT:** A set  $X = \{X_1, \dots, X_N\}$  of nodes, an integer  $W$ , and for each  $X_i$  a non-negative local score (cost)  $s_i(P_i)$  for each  $P_i \in \mathcal{P}_i$ .

**TASK:** Find a DAG  $G^*$  such that

$$G^* \in \arg \min_{G \in \text{TW}(W)} \sum_{i=1}^N s_i(P_i), \quad (3)$$

where  $P_i$  is the parent set of  $X_i$  in  $G$ .

Note that the  $\mathcal{P}_i$ s can be assumed to contain only parent sets  $P_i$  with  $|P_i| \leq W$ , since the treewidth of any DAG containing a node having more than  $W$  parents is greater than  $W$ . However, the opposite does not hold, i.e., the treewidth of a DAG with at most  $W$  parents for each node can still be greater than  $W$ .

### 2.3 Maximum Satisfiability

We shortly review necessary background on Maximum satisfiability [22].

For a Boolean variable  $x$ , there are two literals,  $x$  and  $\neg x$ . A clause is a disjunction ( $\vee$ , logical OR) of literals. A truth assignment is a function from Boolean variables to  $\{0, 1\}$ . A clause  $C$  is satisfied by a truth assignment  $\tau$  ( $\tau(C) = 1$ ) if  $\tau(x) = 1$  for a literal  $x$  in  $C$ , or  $\tau(x) = 0$  for a literal  $\neg x$  in  $C$ . A set  $F$  of clauses is satisfiable if there is an assignment  $\tau$  satisfying all clauses in  $F$  ( $\tau(F) = 1$ ), and unsatisfiable ( $\tau(F) = 0$  for any assignment  $\tau$ ) otherwise. An instance  $F = (F_h, F_s, c)$  of the *weighted partial MaxSAT* problem consists of two sets of clauses, a set  $F_h$  of *hard* clauses and a set  $F_s$  of *soft* clauses, and a function  $c : F_s \rightarrow \mathbb{R}^+$  that associates a non-negative cost with each of the soft clauses.<sup>2</sup> Any truth assignment  $\tau$  that satisfies  $F_h$  is a *solution* to  $F$ . The *cost* of a solution  $\tau$  to  $F$  is

$$\text{COST}(F, \tau) = \sum_{\substack{C \in F_s: \\ \tau(C)=0}} c(C),$$

i.e., as the sum of the costs of the soft clauses not satisfied by  $\tau$ . A solution  $\tau$  is (globally) *optimal* for  $F$  if  $\text{COST}(F, \tau) \leq \text{COST}(F, \tau')$  holds for any solution  $\tau'$  to  $F$ . The

<sup>1</sup> The problem can equivalently be defined as a maximization problem under non-positive local scores.

<sup>2</sup> Our definition for the function  $c$  is more general than the more standard  $c : F_s \rightarrow \mathbb{N}^+$ , which restricts the costs of soft clauses to be integral. However, in this work we employ a recent MaxSAT solver which allows for assigning real-valued costs to soft clauses.

cost of the optimal solutions of  $F$  is denoted by  $\text{OPT}(F)$ . Given a weighted partial MaxSAT instance  $F$ , the weighted partial MaxSAT problem asks to find an optimal solution to  $F$ . From here on, we refer to weighted partial MaxSAT instances simply as MaxSAT instances.

Due to recent advances in MaxSAT solvers, i.e., algorithms for (optimally) solving MaxSAT, MaxSAT is a viable approach to finding globally optimal solutions to various optimization problems. In general, the MaxSAT-based approach has two steps. First, a MaxSAT encoding of the problem is developed. For any instance  $I$  of the problem, the encoding produces a MaxSAT instance  $F_I$  such that any optimal solution to  $F_I$  can be mapped to an optimal solution of  $I$ . Then, an off-the-shelf MaxSAT solver is used to find an optimal solution to the MaxSAT instance. As SAT solvers continue improving, larger and larger problems can be solved in practice [23].

### 3 BTW-BNSL as MaxSAT

We will now describe an encoding of BTW-BNSL as (weighted partial) MaxSAT.

For the following, we assume an arbitrary input instance of BTW-BNSL, consisting of a set  $X = \{X_1, \dots, X_N\}$  of nodes, a treewidth bound  $W$ , and for each  $X_i$  a non-negative local score (cost)  $s_i(P_i)$  for each  $P_i \in \mathcal{P}_i$  with  $|P_i| \leq W$ . Given  $(X, W, \{s_i\}_{i=1}^N)$ , the encoding will produce the weighted partial MaxSAT instance  $F(X, W, \{s_i\}_{i=1}^N) = (F_h, F_s, c)$  such that any optimal solution to  $F$  corresponds to a DAG  $G^*$  that is an optimal solution the BTW-BNSL instance  $(X, W, \{s_i\}_{i=1}^N)$ , and vice versa.

#### 3.1 Overview

In order to exactly represent the BTW-BNSL instance as a weighted partial MaxSAT instance, we will encode the following constraints:

1. For each  $X_i$ , exactly one parent set  $P_i \in \mathcal{P}_i$  is chosen.
2. The graph  $G^*$ , corresponding to the choice of a parent set  $P_i$  for each  $i$ , is acyclic.
3. The moralized graph  $\text{MORAL}(G^*)$  of  $G^*$  has treewidth  $tw(\text{MORAL}(G^*)) \leq W$ .
4. The graph  $G^*$  is an optimal solution of the BTW-BNSL instance, that is,  $G^* \in \arg \min_{G \in \text{TW}(W)} \sum_{i=1}^N s_i(P_i)$ .

Constraints 1 and 2 together enforce that any choice of a single parent set  $P_i$  for each variable  $X_i$  corresponds to a DAG  $G^*$ . Constraint 3 is the most intricate one, and enforces that  $G^*$  has treewidth at most  $W$ . Constraint 4 represents the objective function (Eq. 3) of BTW-BNSL.

The main variables used in the encoding are summarized in Table 1.

- The variables  $P_i^S$  represent for each node  $X_i$  the chosen parent set  $S \in \mathcal{P}_i$ .
- The variables  $M_{ij}$  represent the edges in the moralized graph  $\text{MORAL}(G^*)$  of  $G^*$ .
- The variables  $ord_{ij}$  represent a linear ordering  $ord$  of the nodes of  $G^*$ .
- The variables  $O_{ij}$  represent the successors  $X_j$  of node  $X_i$  in the ordered graph of  $\text{MORAL}(G^*)$  under  $ord$ .

**Table 1.** The main variables used in the MaxSAT encoding of the BTW-BNSL problem.

| Boolean variables | Interpretation                                                                                                                                                            | Indices                                         |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| $P_i^S$           | represent the parent set of each node in $G^*$ :<br>$P_i^S = 1$ iff $S$ is the parent set of node $X_i$ in $G^*$                                                          | for all $i = 1..N$<br>and $S \in \mathcal{P}_i$ |
| $M_{ij}$          | represent the moralized graph of $G^*$ :<br>$M_{ij} = 1$ iff $\text{MORAL}(G^*)$ contains the edge $\{X_i, X_j\}$                                                         | for all $i, j = 1..N$<br>such that $i < j$      |
| $ord_{ij}$        | represent a linear ordering $ord$ of the nodes of $G^*$ :<br>$ord_{ij} = 1$ iff node $X_i$ is a predecessor of node $X_j$<br>in the linear ordering                       | for all $i, j = 1..N$<br>such that $i < j$      |
| $O_{ij}$          | represent the ordered graph $\Delta(\text{MORAL}(G^*), ord)$ :<br>$O_{ij} = 1$ iff the ordered graph of $\text{MORAL}(G^*)$ under $ord$<br>contains the edge $(X_i, X_j)$ | for all $i, j = 1..N$<br>such that $i \neq j$   |

### 3.2 Details

We will now detail the MaxSAT encoding of Constraints 1–4, i.e., our MaxSAT encoding of BTW-BNSL. For clarity, we present the various parts of the encoding using propositional logic, instead of directly presenting the corresponding individual clauses.

*1: Enforcing Exactly One Parent Set.* For each node  $X_i$ , exactly one parent set from  $\mathcal{P}_i$  must be chosen. This is enforced by introducing for each node  $X_i$  the cardinality constraint

$$\sum_{S \in \mathcal{P}_i} P_i^S = 1. \quad (4)$$

Many different ways of representing such special types of cardinality constraints, often called *exactly-one* constraints, as (hard) clauses have been proposed in the literature. Here we use the so-called *improved sequential counter encoding* for representing Eq.(4) as a set of hard clauses; for details on the improved sequential counted encoding, see [24].

*2: Enforcing Acyclicity.* For ruling out cyclic graphs, i.e., for ensuring that any solution to the MaxSAT encoding corresponds to a DAG, we apply the idea of associating a unique, pair-wise different *level number* from  $\{1, \dots, N\}$  with each node  $X_i$ , and enforce that, given that a parent set  $S \in \mathcal{P}_i$  is chosen for  $X_i$ , the level number of  $X_i$  is greater than the level number of each  $X_j \in S$ .<sup>3</sup>

We use a binary encoding of the level numbers of the nodes. For each node  $X_i$ ,  $\log_2 N$  Boolean variables  $b_i^1, \dots, b_i^{\log_2 N}$  form the binary representation  $b_i^{\log_2 N} \dots b_i^1$  of the level number of  $X_i$ . For a compact encoding, we also use auxiliary variables  $EQ_{ij}^k$  and  $GT_{ij}^k$ , with the interpretations that  $EQ_{ij}^k = 1$  iff  $b_i^k = b_j^k$ , and  $GT_{ij}^k = 1$  iff

<sup>3</sup> Variations of the same idea have been applied for enforcing acyclicity with linear integer constraints in different contexts, under e.g. the terms *level rankings* [25] and *generation numbers* [26].

$b_k^i = 1$ ,  $b_k^j = 0$ , and  $EQ_{ij}^{k'} = 1$  for all  $k' > k$  (i.e., the  $k$ th bit is the most significant bit in which the level numbers of  $X_i$  and  $X_j$  differ, and the level number of  $X_i$  is greater than that of  $X_j$ ). Using these variables, the unique level numbers for the nodes are enforced as follows.

The fact that each node gets a different level number from  $\{1, \dots, N\}$  is enforced by stating that for each pair of distinct nodes  $X_i, X_j$ , the level number of  $X_i$  is different from that of  $X_j$ . This is enforced by

$$\bigvee_{k=1}^{\log_2 N} \neg EQ_{ij}^k, \quad (5)$$

i.e., there is a bit-position  $k$  in which the binary representations of the level numbers of  $X_i$  and  $X_j$  differ.

Furthermore, if parent set  $S \in \mathcal{P}_i \setminus \{\emptyset\}$  is chosen for node  $X_i$ , then for each  $X_j \in S$ , there is a bit position  $k$  which is the most significant bit in which the level numbers of  $X_i$  and  $X_j$  differ, and the level number of  $X_i$  is greater than that of  $X_j$ :

$$P_i^S \rightarrow \bigvee_{k=1}^{\log_2 N} GT_{ij}^k \quad \text{for all } j \text{ s.t. } X_j \in S. \quad (6)$$

The semantics of the variables  $GT_{ij}^k$  and  $EQ_{ij}^k$  are encoded as

$$GT_{ij}^k \leftrightarrow b_i^k \wedge \neg b_j^k \wedge \bigwedge_{k'=k+1}^{\log_2 N} EQ_{ij}^{k'}, \quad (7)$$

$$EQ_{ij}^k \leftrightarrow (b_i^k \leftrightarrow b_j^k). \quad (8)$$

While Eqs. 5–8 together with Eq. 4 ensure that any solution corresponds to a DAG, we also include a single additional *redundant* clause, stating the fact that a DAG has at least one root node, i.e., a node  $X_i$  with the empty parent set  $\emptyset$ :

$$\bigvee_{i=1}^N P_i^\emptyset. \quad (9)$$

While this clause is redundant in that it does not change the set of solutions, it turned out that in practice adding this clause speeds up MaxSAT solving.

**3: Enforcing the Treewidth Bound.** The most intricate part of the MaxSAT encoding deals with mapping parent sets to the moralized graph of a DAG  $G^*$  corresponding to the parent sets, and then enforcing that the moralized graph  $\text{MORAL}(G^*)$  of  $G^*$  has treewidth  $tw(\text{MORAL}(G^*)) \leq W$ .

(i) *From Parent Sets to the Moralized Graph.* We directly connect the choices of parent sets, represented by the  $P_i^S$  variables, with the edges in the corresponding moralized graph, represented by the variables  $M_{ij}$ . The encoding follows closely the definition of moralized graphs (Def. 3). Eq. 10 enforces that, if a particular parent set  $S \in \mathcal{P}_i$  is

chosen, then in the moralized graph there is (i) an edge between  $X_i$  and each  $X_j \in S$ , and (ii) an edge between each pair of distinct nodes  $X_j, X_k \in S$ .

$$P_i^S \rightarrow \bigwedge_{X_j \in S} M_{ij} \wedge \bigwedge_{X_j, X_k \in S} M_{jk}. \quad (10)$$

The opposite direction is encoded as Eq. 11: if there is an edge in the moralized graph between nodes  $X_i$  and  $X_j$ , it must hold that: (i)  $X_j$  is in the parent set of  $X_i$ , (ii)  $X_i$  is in the parent set of  $X_j$ , or (iii) both  $X_i$  and  $X_j$  are in the parent set of some  $X_k \in X \setminus \{X_i, X_j\}$ .

$$M_{ij} \rightarrow \bigvee_{S: X_j \in S} P_i^S \vee \bigvee_{S: X_i \in S} P_j^S \vee \bigvee_{\substack{X_k \in X \setminus \{X_i, X_j\} \\ S: X_i, X_j \in S}} P_k^S \quad (11)$$

Notice that, with this encoding, we do not need to introduce explicit Boolean variables for explicitly representing the actual edges of the DAG corresponding to the choice of parent sets.

(ii) *Encoding Linear Orderings.* For enforcing the treewidth bound on the moralized graphs, we follow—with minor modifications—a SAT encoding of treewidth in undirected graphs presented in [24]. Following [24], we do not encode the construction of a tree-decomposition of  $\text{MORAL}(G^*)$  explicitly. Instead, our encoding enforces the condition that for any  $G^*$ , there needs to be a linear ordering  $ord$  of  $X$  under which the maximum number of successors over all nodes in the ordered graph of  $\text{MORAL}(G^*)$  is at most  $W$ .

The choice of a linear ordering of  $X$  is represented by the  $ord_{ij}$  variables. For notational convenience, let

$$ord_{ij}^* = \begin{cases} ord_{ij} & \text{if } i < j \\ -ord_{ji} & \text{else} \end{cases}.$$

Transitivity of linear orderings is enforced in the encoding by stating for all distinct  $i, j, k = 1..N$

$$ord_{ij}^* \wedge ord_{jk}^* \rightarrow ord_{ik}^*. \quad (12)$$

(iii) *Bounding Treewidth via Triangulation.* We recall that the treewidth of the tree-decomposition corresponding to a linear ordering  $\prec$  is  $\max_{v_i \in V} |\{\{v_i, v_j\} \in E : i \prec j\}|$ , where  $E$  is the edge-relation of the triangulated moralized graph; and that the variable  $O_{ij}$  represents the fact that the ordered graph of  $\text{MORAL}(G^*)$  under the linear ordering  $\prec$  (represented by the  $ord_{ij}$  variables) contains the edge  $(X_i, X_j)$ . It follows that enforcing the cardinality constraint

$$\sum_{j \neq i} O_{ij} \leq W \quad (13)$$

for each  $i = 1..N$  is equivalent to the requirement  $\max_{v_i \in V} |\{\{v_i, v_j\} \in E \mid i \prec j\}| \leq W$ . Again, different ways of representing such general cardinality constraints

as clauses have been proposed in the literature. Since here the interesting cases are when  $W$  takes values greater than one, we use a compact encoding based on so-called *cardinality networks* [27, 28] for representing the constraints as hard clauses.

What remains is the definition of the  $O_{ij}$  variables, i.e., encoding of the ordered graph induced by a linear ordering.

–If the moralized graph contains an edge  $\{X_i, X_j\}$ , then the triangulation of the moralized graph also contains the edge  $\{X_i, X_j\}$ , and hence the ordered graph contains either the edge  $(X_i, X_j)$  or the edge  $(X_j, X_i)$ . This is enforced by

$$M_{ij} \rightarrow (O_{ij} \vee O_{ji}) \quad \text{for all } i < j. \quad (14)$$

–If nodes  $X_i$  and  $X_j$  have a common predecessor in the moralized graph, then the triangulation of the moralized graph contains the edge  $\{X_i, X_j\}$ , and hence the ordered graph contains either the edge  $(X_i, X_j)$  or the edge  $(X_j, X_i)$ . This is enforced for all distinct  $i, j, k = 1..N$  by

$$(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji}). \quad (15)$$

Finally, in both Eqs. 14 and 15, the choice of which of the edges  $(X_i, X_j)$  or  $(X_j, X_i)$  occur in the ordered graph depends on the linear ordering  $ord$ . Essentially,  $O_{ij}$  must be consistent with  $ord_{ij}$  in that, if  $i$  comes before  $j$  in  $ord$ , then the edge  $(X_j, X_i)$  does not occur in the ordered graph under  $ord$ :

$$ord_{ij}^* \rightarrow \neg O_{ji}. \quad (16)$$

**4: Encoding the Objective Function.** We encode the BTW-BNSL objective function (Eq. 3) using soft clauses. Accordingly, choosing a specific parent set  $S \in \mathcal{P}_i$  for node  $X_i$  should incur a cost equal to the local score  $s_i(S)$ . Thus, we introduce for each  $X_i$  and each  $S \in \mathcal{P}_i$  the soft clause

$$(\neg P_i^S) \quad (17)$$

and associate the local score  $s_i(S)$  as the weight of this soft clause by defining

$$c((\neg P_i^S)) = s_i(S). \quad (18)$$

### 3.3 Summary of the Encoding

Assume an arbitrary instance  $(X, W, \{s_i\}_{i=1}^N)$  of BTW-BNSL, consisting of a set  $X = \{X_1, \dots, X_N\}$  of nodes, a treewidth bound  $W$ , and for each  $X_i$  a non-negative local score (cost)  $s_i(P_i)$  for each  $P_i \in \mathcal{P}_i$  with  $|P_i| \leq W$ . The weighted partial MaxSAT instance  $F(X, W, \{s_i\}_{i=1}^N) = (F_h, F_s, c)$  consists of the hard clauses corresponding to Eqs. 4–16 and the soft clauses corresponding to Eq. 17 with weights assigned according to Eq. 18.

Given an arbitrary solution  $\tau$  to  $F(X, W, \{s_i\}_{i=1}^N)$ , the choice of the parent set  $S$  for each node  $X_i$  is given by the Boolean variable  $P_i^S$  for which  $\tau(P_i^S) = 1$ . We denote by  $G_\tau$  the DAG corresponding to this choice  $S$  of a parent set for each node  $X_i$ .

**Theorem 1.** *For any solution  $\tau$  to  $F(X, W, \{s_i\}_{i=1}^N) = (F_h, F_s, c)$ , let  $G_\tau$  be the DAG corresponding to  $\tau$ . It holds that  $\tau$  is an optimal solution to  $F(X, W, \{s_i\}_{i=1}^N)$  if and only if  $G_\tau$  is an optimal solution the BTW-BNSL instance  $(X, W, \{s_i\}_{i=1}^N)$ .*

*Proof. (sketch)* Eq. 4 ensures that for each node  $X_i$ ,  $\tau(P_i^S) = 1$  for exactly one parent set  $S \in \mathcal{P}_i$ , i.e., a single parent set for  $X_i$  is chosen. Eqs. 5–8 ensure that  $G_\tau$  is a DAG. Eqs. 10–11 ensure that the  $M_{ij}$  variables with  $\tau(M_{ij})$  correspond exactly to the moralization of  $G_\tau$ . Eq. 12 ensures that any assignment to the  $ord_{ij}$  variables corresponds to the linear ordering  $ord$  over  $X$  for which  $i$  comes before  $j$  iff  $\tau(ord_{ij}) = 1$ . Eqs. 14–15 encode exactly the conditions for an edge to be present in the triangulation of  $G_\tau$  under  $ord$ , and Eq. 16 enforces the edge-directions of the triangulation according to  $ord$ , corresponding exactly to the ordered graph (consisting of the edges  $(X_i, X_j)$  for which  $\tau(O_{ij}) = 1$ ) of  $G_\tau$  under  $ord$ . Eq. 13 is satisfied iff there is a linear ordering  $ord$ , i.e., an assignment over the variables  $ord_{ij}$ , such that the maximum number of successors in the ordered graph represented by the  $O_{ij}$  variables is at most  $W$ . Finally, Eqs. 17–18 encode exactly the objective function of BTW-BNSL.  $\square$

## 4 Experiments

We present results on the efficiency of optimally solving the BTW-BNSL problem via our MaxSAT encoding using a state-of-the-art MaxSAT solver. As the MaxSAT solver we used MaxHS [29]<sup>4</sup>. For comparing to the recent exact approach to BTW-BNSL based on dynamic programming, we used the `best-w-tree` implementation available from the authors at <http://www.cs.helsinki.fi/u/jazkorho/aistats-2013/>.

### 4.1 Setup

The experiments were performed on a cluster of 2.8-GHz Intel Xeon quad core machines with 32-GB memory and Ubuntu Linux 10.04. A timeout of 8 h (28 800 seconds) and a memory limit of 30 GB were enforced on the solvers on the individual benchmark instances.

As benchmark data, we used a set of well-known UCI dataset with 9–29 variables. We used the MDL scoring function [16] for computing the local scores of parent sets from the datasets. Furthermore, we included as benchmarks the two datasets (Adult, Housing) made available by [15] with pre-computed local scores, giving a total of 10 datasets. As treewidth bounds, we used the values  $W = 2, 3, 4$ , resulting in a total of 30 benchmark instances. We pruned candidate parent sets using the following well-known pruning rule that maintains the set of optimal solutions: Given two parent sets  $S, S' \in \mathcal{P}_i$ , if  $S' \subset S$  and  $s_i(S') \leq s_i(S)$ , then  $S$  can be pruned away from consideration. We observed that applying this pruning rule had a positive effect on the running times of both the MaxSAT solver and the dynamic programming approach. The pruning of a particular candidate parent set  $S \in \mathcal{P}_i$  is reflected in the MaxSAT encoding by the fact that the corresponding Boolean variable  $P_i^S$  is not introduced.

<sup>4</sup> The developers of MaxHS provided a version which allows for assigning real-values as costs on soft clauses.

**Table 2.** Running times in seconds of our MaxSAT-based approach (**MS**) and the dynamic programming (**DP**) approach [15] for different UCI datasets and treewidth bounds  $W = 2, 3, 4$ . Explanations: “mo” denotes a memory out;  $N$  denotes the number of variables (nodes); **#fails** denotes the number of times the memory or time limit was exceeded.

| Dataset        | $N$ | treewidth $\leq 2$ |          | treewidth $\leq 3$ |           | treewidth $\leq 4$ |        | #fails   |    |
|----------------|-----|--------------------|----------|--------------------|-----------|--------------------|--------|----------|----|
|                |     | MS (s)             | DP (s)   | MS (s)             | DP (s)    | MS (s)             | DP (s) | MS       | DP |
| Abalone        | 9   | 64                 | <b>7</b> | 166                | <b>57</b> | <b>215</b>         | 536    | 0        | 0  |
| Housing        | 14  | <b>2 226</b>       | 6 927    | <b>2 329</b>       | > 28 800  | <b>2 991</b>       | mo     | <b>0</b> | 2  |
| Wine           | 14  | <b>27</b>          | 6 924    | <b>22</b>          | > 28 800  | <b>171</b>         | mo     | <b>0</b> | 2  |
| Adult          | 15  | <b>998</b>         | > 28 800 | <b>1 623</b>       | > 28 800  | <b>1 782</b>       | mo     | <b>0</b> | 3  |
| Voting         | 17  | <b>22 909</b>      | > 28 800 | <b>26 419</b>      | mo        | > 28 800           | mo     | <b>1</b> | 3  |
| Zoo            | 17  | <b>410</b>         | > 28 800 | <b>412</b>         | mo        | <b>105</b>         | mo     | <b>0</b> | 3  |
| Hepatitis      | 20  | <b>315</b>         | mo       | <b>100</b>         | mo        | <b>1 164</b>       | mo     | <b>0</b> | 3  |
| Heart          | 23  | <b>1 198</b>       | mo       | <b>2 186</b>       | mo        | <b>41</b>          | mo     | <b>0</b> | 3  |
| Horse          | 28  | <b>192</b>         | mo       | > 28 800           | mo        | <b>544</b>         | mo     | <b>1</b> | 3  |
| Flag           | 29  | <b>1 418</b>       | mo       | <b>11 148</b>      | mo        | <b>1 356</b>       | mo     | <b>0</b> | 3  |
| <b>#fails:</b> |     | <b>0</b>           | 7        | <b>1</b>           | 9         | <b>1</b>           | 9      | <b>2</b> | 25 |

## 4.2 Results

are presented in Table 2 under treewidth bounds  $W = 2, 3, 4$ . For each bound, the best running time to find an optimal solution is highlighted in boldface.

We observe that the dynamic programming approach (**DP**) is competitive with our MaxSAT-approach only for the smallest dataset with 9 variables. Apart from the multiple timeouts (“> 28 800”), we observe that **DP** most often runs out of memory (“mo”) on the datasets with more variables, especially for treewidth bounds greater than 2; memoryouts can be considered more critical than timeouts since they imply that the algorithm cannot give a solution however much time it is given. In contrast, the MaxSAT-approach (**MS**) timeouts on only two instances, and, especially, does not suffer from memouts. For a clear 2/3 majority of the instances, **MS** produces an optimal solution within half-an-hour; and for half of the instances within around 10 minutes.

## 5 Related Work

In [6] a MaxSAT formulation of BNSL *without* treewidth restrictions was studied. Our encoding is more involved: we enforce a strict treewidth bound, and apply a more intricate encoding of the acyclicity constraint. Cussens used at-the-time state-of-the-art *local search* MaxSAT solvers, and was hence unable to find optimal networks, and also used integer-rounded local scores for candidate parent sets; in contrast we use a current state-of-the-art *complete* MaxSAT solver which provides *provably optimal* solutions, and use the actual (non-integer) local scores without rounding.

In [15] an exact algorithm for BTW-BNSL based on dynamic programming was proposed. Their algorithm is also to our best knowledge the only approach for learning guaranteed-optimal bounded treewidth Bayesian network structures. We provide in this

paper an empirical comparison: our MaxSAT-based approach scales both to larger numbers of variables and larger treewidth bounds than the dynamic programming approach.

A greedy search strategy for learning Bayesian networks under treewidth constraints was proposed in [14]. Their algorithm relies on a search operator which is guaranteed to increase the treewidth of the current solution by at most one. Their approximation algorithm is polynomial-time in the number of variables and treewidth. However, due to the local search strategy, no bounds on the quality of the learned network can be guaranteed.

In [30] the problem of learning and optimal network structure given a super-structure of bounded treewidth is considered, and it was shown that this problem is fixed parameter tractable in the treewidth of the super-structure. The treewidth of the super-structure does not, in general, bound the treewidth of the network, and hence does not ensure efficient exact inference after learning the network.

Integer-linear programming (ILP) provides another constrained optimization approach to BNSL, as studied by [31, 32, 10, 33].

Finally, algorithms for learning undirected graphical models, especially, classes of Markov networks [34–42] which enable fast inference by e.g., bounding the treewidth of the underlying tree-decompositions (often referred to as *junction trees*) have been developed. To our understanding, none of these algorithms guarantee to learn globally optimal structures.

## 6 Conclusions

Exact inference in low-treewidth Bayesian networks is tractable, which motivates the development of practical approaches to learning bounded treewidth networks. However, few practical algorithms have been proposed for learning networks under treewidth constraints. In this paper, we presented an approach to learning bounded treewidth Bayesian network structures that is guaranteed to provide optimal structures. Our approach is based on encoding the structure learning problem as weighted partial Maximum satisfiability, and then using a state-of-the-art MaxSAT solver for solving the resulting MaxSAT instances, i.e., for finding optimal bounded treewidth Bayesian network structures. We showed that our non-trivial MaxSAT encoding results in notably better performance compared to an implementation of a recently proposed dynamic programming algorithm for optimal bounded treewidth Bayesian network structure learning.

**Acknowledgements** Work supported by Academy of Finland (COIN Centre of Excellence in Computational Inference Research, grant #251170) and Finnish Funding Agency for Technology and Innovation (project D2I). The authors thank Jessica Davies for providing the MaxHS version used in the experiments.

## References

1. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc. (1988)

2. Chickering, D.M.: Learning Bayesian networks is NP-complete. In: *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag (1996) 121–130
3. Heckerman, D.: A tutorial on learning with Bayesian networks. In: *Learning in Graphical Models*. Volume 89 of NATO ASI Series. Springer (1998) 301–354
4. Chickering, D.M.: Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research* **2** (2002) 445–498
5. Teyssier, M., Koller, D.: Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In: *Proc. UAI, AUAI Press* (2005) 584–590
6. Cussens, J.: Bayesian network learning by compiling to weighted MAX-SAT. In: *Proc. UAI, AUAI Press* (2008) 105–112
7. Ott, S., Miyano, S.: Finding optimal gene networks using biological constraints. *Genome Informatics* **14** (2003) 124–133
8. Koivisto, M., Sood, K.: Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research* (2004) 549–573
9. Silander, T., Myllymäki, P.: A simple approach for finding the globally optimal Bayesian network structure. In: *Proc. UAI, AUAI Press* (2006) 445–452
10. Cussens, J.: Bayesian network learning with cutting planes. In: *Proc. UAI, AUAI Press* (2011) 153–160
11. Yuan, C., Malone, B.: Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research* **48** (2013) 23–65
12. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* **42**(2-3) (1990) 393 – 405
13. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)* **50**(2) (1988) 157–224
14. Elidan, G., Gould, S.: Learning bounded treewidth bayesian networks. *Journal of Machine Learning Research* **9** (2008) 2699–2731
15. Korhonen, J.H., Parviainen, P.: Exact learning of bounded tree-width Bayesian networks. In: *Proc. AISTATS*. (2013) 370–378
16. Lam, W., Bacchus, F.: Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence* **10** (1994) 269–293
17. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* **9** (1992) 309–347
18. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* **20** (1995) 197–243
19. Silander, T., Roos, T., Kontkanen, P., Myllymaki, P.: Factorized normalized maximum likelihood criterion for learning Bayesian network structures. In: *Proc. PGM*. (2008) 257–272
20. Dechter, R.: Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* **113**(1-2) (1999) 41–85
21. Bodlaender, H.L.: Discovering treewidth. In: *Proc. SOFSEM*. Volume 3381 of LNCS., Springer (2005) 1–16
22. Li, C.M., Manyà, F.: MaxSAT, hard and soft constraints. In: *Handbook of Satisfiability*. Volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press (2009) 613–631
23. Jarvisalo, M., Le Berre, D., Roussel, O., Simon, L.: The international SAT solver competitions. *AI Magazine* **33**(1) (2012) 89–92
24. Samer, M., Veith, H.: Encoding treewidth into SAT. In: *Proc. SAT*. Volume 5584 of LNCS. Springer (2009) 45–50
25. Niemelä, I.: Stable models and difference logic. *Annals of Mathematics and Artificial Intelligence* **53**(1-4) (2008) 313–329

26. Cussens, J., Bartlett, M., Jones, E.M., Sheehan, N.A.: Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology* **37**(1) (2013) 69–83
27. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks: a theoretical and empirical study. *Constraints* **16**(2) (2011) 195–221
28. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: A parametric approach for smaller and better encodings of cardinality constraints. In: *Proc. CP. Volume 8124 of LNCS.*, Springer (2013) 80–96
29. Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in Maxsat. In: *Proc. SAT. Volume 7962 of LNCS.*, Springer (2013) 166–181
30. Ordyniak, S., Szeider, S.: Parameterized complexity results for exact Bayesian network structure learning. *Journal of Artificial Intelligence Research* **46** (2013) 263–302
31. Jaakkola, T., Sontag, D., Globerson, A., Meila, M.: Learning Bayesian network structure using LP relaxations. In: *Proc. AISTATS.* (2010)
32. Studený, M., Vomlel, J., Hemmecke, R.: A geometric view on learning bayesian network structures. *International Journal of Approximate Reasoning* **51**(5) (2010) 573–586
33. Bartlett, M., Cussens, J.: Advances in Bayesian network learning using integer programming. In: *Proc. UAI, UAI Press* (2013) 182–191
34. Malvestuto, F.M.: Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man, and Cybernetics* **21**(5) (1991)
35. Bach, F., Jordan, M.: Thin junction trees. In: *Proc. NIPS, MIT Press* (2001) 569–576
36. Karger, D., Srebro, N.: Learning Markov networks: maximum bounded tree-width graphs. In: *Proc. SODA, SIAM* (2001) 392–401
37. Srebro, N.: Maximum likelihood bounded tree-width Markov networks. *Artificial Intelligence* **143**(1) (2003) 123 – 138
38. Narasimhan, M., Bilmes, J.: PAC-learning bounded tree-width graphical models. In: *Proc. UAI, AUAI Press* (2004) 410–417
39. Checheta, A., Guestrin, C.: Efficient principled learning of thin junction trees. In: *Proc. NIPS, MIT Press* (2007) 273–280
40. Gogate, V., Webb, W., Domingos, P.: Learning efficient Markov networks. In: *Proc. NIPS, MIT Press* (2010) 748–756
41. Szántai, T., Kovács, E.: Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. *Annals of Operations Research* **193**(1) (2012)
42. Kumar, K.S.S., Bach, F.: Convex relaxations for learning bounded-treewidth decomposable graphs. In: *Proc. ICML.* (2013) 525–533