Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability

Jeremias Berg and **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.

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 (Pearl, 1988). 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 (Chickering, 1996), 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 (Heckerman, 1998), equivalence classes of DAGs (Chickering, 2002), or over variable orderings (Teyssier and Koller, 2005), and local searching over constraint optimization formulations of BNSL (Cussens, 2008).

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 (Ott and Miyano, 2003; Koivisto and Sood, 2004; Silander and Myllymäki, 2006; Cussens, 2011; Yuan and Malone, 2013).

While exact Bayesian inference is in general NP-hard (Cooper, 1990), for *bounded (fixed) treewidth* networks exact inference becomes tractable (Lauritzen and Spiegelhalter, 1988). 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 (Elidan and Gould, 2008; Korhonen and Parviainen, 2013); the only approach learning optimal bounded treewidth network structures is the recent exact dynamic programming algorithm of Korhonen and Parviainen

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

(2013).

Much like the general BNSL problem, BTW-BNSL is NP-hard (Korhonen and Parviainen, 2013): 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 (Korhonen and Parviainen, 2013) 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 (Heckerman, 1998), 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 (Lam and Bacchus, 1994), BD (Cooper and Herskovits, 1992; Heckerman et al., 1995), and fNML (Silander et al., 2008).

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 $\{V_1, \ldots, V_m\}$ 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 $i \in \{1, \ldots, m\}$.*
3. *For all $i, j, k \in \{1, \ldots, m\}$, 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 $tw(G)$ (Dechter, 1999; Bodlaender, 2005). 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

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

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

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

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

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

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.

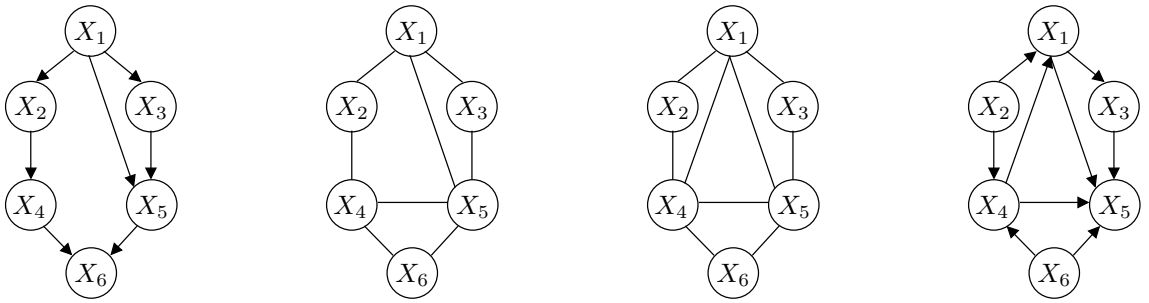

Figure 1: Example: (a) a DAG $G = (X = \{X_1, \ldots, X_6\}, E)$; (b) the moralized graph $\textsc{Moral}(G) = (X, \mathrm{M}(E))$ of G; (c) the triangulation $\Delta(\textsc{Moral}(G), \prec)$ of the moralized graph under the linear ordering $X_6 \prec X_2 \prec X_4 \prec X_1 \prec X_3 \prec X_5$; (d) the ordered graph $\vec{\Delta}(\textsc{Moral}(G), \prec)$.

2.2 Bounded Treewidth Bayesian Network Structure Learning

Given a set $X = \{X_1, \ldots, X_N\}$ 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 (Lauritzen and Spiegelhalter, 1988).

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

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

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 Eq. 1 for the ordered graph $\vec{\Delta}(\textsc{Moral}(G), \prec)$ of the moralization $\textsc{Moral}(G)$ of G under $\prec$. We denote by $\textsc{tw}(W)$ 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 $\textsc{Moral}(G)$, and, for a given linear ordering $\prec$ of the nodes X, (c) the triangulation $\Delta(\textsc{Moral}(G), \prec)$ and (d) the ordered graph $\vec{\Delta}(\textsc{Moral}(G), \prec)$. For this ordering $\prec$, Eq. 1 evaluates to 2, and hence $G \in \textsc{tw}(2)$. In fact, it can be checked that this $\prec$ defines an optimal tree-decomposition of G, minimizing Eq. 2, which implies that $tw(G) = 2$.

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

The BTW-BNSL Problem

Input: A set $X = \{X_1, \ldots, X_N\}$ 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

$$G^* \in \operatorname*{arg\,min}_{G \in \textsc{tw}(W)} \sum_{i=1}^{N} s_i(P_i), \tag{3}$$

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 (Li and Manyà, 2009).

For a Boolean variable x, there are two literals, x and

[1]The problem can equivalently be defined as a maximization problem under non-positive local scores.

$\neg x$. 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 τ ($\tau(C) = 1$) 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 τ satisfying all clauses in F ($\tau(F) = 1$), and unsatisfiable ($\tau(F) = 0$ for any assignment τ) 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 $c : F_s \to \mathbb{R}^+$ that associates a non-negative cost with each of the soft clauses.[2] Any truth assignment τ that satisfies F_h is a *solution* to F. The *cost* of a solution τ to F is

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

i.e., as the sum of the costs of the soft clauses not satisfied by τ. A solution τ is (globally) *optimal* for F if $\textsc{cost}(F, \tau) \leq \textsc{cost}(F, \tau')$ holds for any solution τ' to F. The cost of the optimal solutions of F is denoted by $\textsc{opt}(F)$. 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 (Järvisalo et al., 2012).

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 $X = \{X_1, \ldots, X_N\}$ 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 $(X, W, \{s_i\}_{i=1}^N)$, our encoding will produce a 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.

[2]Our definition for the function c is more general than the more standard $c : F_s \to \mathbb{N}^+$, 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.

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 $\textsc{Moral}(G^*)$ of G^* has treewidth $tw(\textsc{Moral}(G^*)) \leq W$.
4. G^* is an optimal solution of the BTW-BNSL instance, i.e., $G^* \in \arg\min_{G \in \textsc{Tw}(W)} \sum_{i=1}^N s_i(P_i)$.

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 $\textsc{Moral}(G^*)$ 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 $\textsc{Moral}(G^*)$ under ord.

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

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

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

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 (Samer and Veith, 2009).

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 $\{1, \ldots, N\}$ 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 $X_j \in S$.[3]

We use a binary encoding of the level numbers of the nodes. For each node X_i, $\log_2 N$ Boolean variables $b_i^1, \ldots, b_i^{\log_2 N}$ form the binary representation $b_i^{\log_2 N} \ldots b_i^1$ 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 $b_k^i = 1$, $b_k^j = 0$, and $EQ_{ij}^{k'} = 1$ for all $k' > k$ (i.e., the kth 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 $\{1, \ldots, N\}$ 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

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

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

[3]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* (Niemelä, 2008) and *generation numbers* (Cussens et al., 2013).

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:

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

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

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

$$EQ_{ij}^k \leftrightarrow \left(b_i^k \leftrightarrow b_j^k\right). \tag{8}$$

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 $\emptyset$:

$$\bigvee_{i=1}^{N} P_i^{\emptyset}. \tag{9}$$

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 $\textsc{Moral}(G^*)$ of G^* has treewidth $tw(\textsc{Moral}(G^*)) \leq W$.

(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 $X_j, X_k \in S$.

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

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 $X_k \in X \setminus \{X_i, X_j\}$.

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

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 (Samer and Veith, 2009). Following Samer and Veith (2009), we do not encode the construction of a tree-decomposition of $\textsc{Moral}(G^*)$ 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 $\textsc{Moral}(G^*)$ is at most W.

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

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

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

$$ord_{ij}^* \wedge ord_{jk}^* \rightarrow ord_{ik}^*. \tag{12}$$

(iii) Bounding Treewidth via Triangulation. 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 $\textsc{Moral}(G^*)$ 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

$$\sum_{j \neq i} O_{ij} \leq W \tag{13}$$

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* (Asín et al., 2011; Abío et al., 2013) 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

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

–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

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

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*:

$$ord_{ij}^* \rightarrow \neg O_{ji}. \tag{16}$$

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

$$(\neg P_i^S) \tag{17}$$

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

$$c((\neg P_i^S)) = s_i(S). \tag{18}$$

3.3 Summary of the Encoding

Assume an arbitrary instance $(X, W, \{s_i\}_{i=1}^N)$ of BTW-BNSL, consisting of a set $X = \{X_1, \ldots, X_N\}$ 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 τ 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_τ the DAG corresponding to this choice S of a parent set for each node X_i.

Theorem 1 *For any solution τ to $F(X, W, \{s_i\}_{i=1}^N) = (F_h, F_s, c)$, let G_τ be the DAG corresponding to τ. It holds that τ is an optimal solution to $F(X, W, \{s_i\}_{i=1}^N)$ if and only if G_τ 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_τ is a DAG. Eqs. 10–11 ensure that the M_{ij} variables with $\tau(M_{ij})$ correspond exactly to the moralization of G_τ. 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_τ 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_τ 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. □

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 (Davies and Bacchus, 2013)[4]. 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/`.

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 (Lam and Bacchus, 1994) for computing the local scores of parent sets from the datasets. Furthermore, we included as benchmarks the two datasets (Adult, Housing) made available by Korhonen and Parviainen (2013) 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.

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

Cussens (2008) formulated BNSL *without* treewidth restrictions as MaxSAT. 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 pro-

[4]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 (Korhonen and Parviainen, 2013) 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.

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

vides *provably optimal* solutions, and use the actual (non-integer) local scores without rounding.

Korhonen and Parviainen (2013) proposed an exact algorithm for BTW-BNSL based on dynamic programming. 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.

Elidan and Gould (2008) proposed a greedy search strategy for learning Bayesian networks under treewidth constraints. 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.

Ordyniak and Szeider (2013) consider the problem of learning and optimal network structure given a super-structure of bounded treewidth, and show 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 Jaakkola et al. (2010); Studený et al. (2010); Cussens (2011); Bartlett and Cussens (2013).

Finally, algorithms for learning undirected graphical models, especially, classes of Markov networks (Malvestuto, 1991; Bach and Jordan, 2001; Karger and Srebro, 2001; Srebro, 2003; Narasimhan and Bilmes, 2004; Chechetka and Guestrin, 2007; Gogate et al., 2010; Szántai and Kovács, 2012; Kumar and Bach, 2013) 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

Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A parametric approach for smaller and better encodings of cardinality constraints. In *Proc. CP*, volume 8124 of *LNCS*, pages 80–96. Springer, 2013.

Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

Francis Bach and Michael Jordan. Thin junction trees. In *Proc. NIPS*, pages 569–576. MIT Press, 2001.

Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proc. UAI*, pages 182–191. UAUI Press, 2013.

Hans L. Bodlaender. Discovering treewidth. In *Proc. SOFSEM*, volume 3381 of *LNCS*, pages 1–16. Springer, 2005.

Anton Chechetka and Carlos Guestrin. Efficient principled learning of thin junction trees. In *Proc. NIPS*, pages 273–280. MIT Press, 2007.

David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

David Maxwell Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393 – 405, 1990.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

James Cussens. Bayesian network learning by compiling to weighted MAX-SAT. In *Proc. UAI*, pages 105–112. AUAI Press, 2008.

James Cussens. Bayesian network learning with cutting planes. In *Proc. UAI*, pages 153–160. AUAI Press, 2011.

James Cussens, Mark Bartlett, Elinor M. Jones, and Nuala A. Sheehan. Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology*, 37(1):69–83, 2013.

Jessica Davies and Fahiem Bacchus. Exploiting the power of MIP solvers in Maxsat. In *Proc. SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013.

Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2): 41–85, 1999.

Gal Elidan and Stephen Gould. Learning bounded treewidth bayesian networks. *Journal of Machine Learning Research*, 9:2699–2731, 2008.

Vibhav Gogate, William Webb, and Pedro Domingos. Learning efficient Markov networks. In *Proc. NIPS*, pages 748–756. MIT Press, 2010.

David Heckerman. A tutorial on learning with Bayesian networks. In *Learning in Graphical Models*, volume 89 of *NATO ASI Series*, pages 301–354. Springer, 1998.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proc. AISTATS*, 2010.

Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, 33(1):89–92, 2012.

David Karger and Nathan Srebro. Learning Markov networks: maximum bounded tree-width graphs. In *Proc. SODA*, pages 392–401. SIAM, 2001.

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, pages 549–573, 2004.

Janne H. Korhonen and Pekka Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Proc. AISTATS*, pages 370–378, 2013.

K. S. Sesh Kumar and Francis Bach. Convex relaxations for learning bounded-treewidth decomposable graphs. In *Proc. ICML*, pages 525–533, 2013.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 19, pages 613–631. IOS Press, 2009.

Francesco M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5), 1991.

Mukund Narasimhan and Jeff Bilmes. PAC-learning bounded tree-width graphical models. In *Proc. UAI*, pages 410–417. AUAI Press, 2004.

Ilkka Niemelä. Stable models and difference logic. *Annals of Mathematics and Artificial Intelligence*, 53 (1-4):313–329, 2008.

Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact Bayesian network structure learning. *Journal of Artificial Intelligence Research*, 46:263–302, 2013.

Sascha Ott and Satoru Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

Marko Samer and Helmut Veith. Encoding treewidth into SAT. In *Proc. SAT*, volume 5584 of *LNCS*, pages 45–50. Springer, 2009.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proc. UAI*, pages 445–452. AUAI Press, 2006.

Tomi Silander, Teemu Roos, Petri Kontkanen, and Petri Myllymaki. Factorized normalized maximum likelihood criterion for learning Bayesian network structures. In *Proc. PGM*, pages 257–272, 2008.

Nathan Srebro. Maximum likelihood bounded treewidth Markov networks. *Artificial Intelligence*, 143 (1):123 – 138, 2003.

Milan Studený, Jirí Vomlel, and Raymond Hemmecke. A geometric view on learning bayesian network structures. *International Journal of Approximate Reasoning*, 51(5):573–586, 2010.

Tamás Szántai and Edith Kovács. Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. *Annals of Operations Research*, 193(1), 2012.

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proc. UAI*, pages 584–590. AUAI Press, 2005.

Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48: 23–65, 2013.