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

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

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

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

$$\min_{\prec} \max_{v_i \in V} |\{(v_i, v_j) \in \Delta(E, \prec)\}|,$$  \hspace{1cm} (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.
The problem can equivalently be defined as a maximization problem under non-positive local scores.

The treewidth of the DAG structure of the network can be determined by finding a linear order (necessarily acyclic) graph in which, for each \( X \in P \), an element of \( E \) assigns a single \( \delta \) to \( X \) and \( x \) in \( P \) such that \( x \) and \( \delta \) form a minimal cut of the moralized graph with respect to \( x \) and \( \delta \).

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 \( \text{MORAL}(G) = (X, M(E)) \) 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 Eq. 1 for the ordered graph \( \Delta(\text{MORAL}(G), \prec) \) of the moralization \( \text{MORAL}(G) \) of \( G \) under \( \prec \). We denote by \( \text{tw}(G) \) 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 \( \Delta(\text{MORAL}(G), \prec) \).

**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 \( P_i \in P_i \).

**Task:** Find a DAG \( G^* \) such that

\[
G^* \in \arg \min_{G \in \text{tw}(W)} \sum_{i=1}^{N} s_i(P_i),
\]

where \( P_i \) is the parent set of \( X_i \) in \( G \).

Note that the \( P_i \)'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 briefly review necessary background on Maximum satisfiability (Li and Manyà, 2009).

For a Boolean variable \( x \), there are two literals, \( x \) and \( \neg x \).
Learning Bounded Treewidth Bayesian Networks via MaxSAT

¬x. A clause is a disjunction (∨, 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 if τ(C) = 1 if τ(x) = 1 for a literal x in C, or τ(x) = 0 for a literal ¬x in C. A set F of clauses is satisfiable if there is an assignment τ satisfying all clauses in F (τ(F) = 1), and unsatisfiable (τ(F) = 0 for any assignment τ) otherwise. An instance F = (Fh, Fs, c) of the weighted partial MaxSAT problem consists of two sets of clauses, a set Fh of hard clauses and a set Fs of soft clauses, and a function c : Fs → R+ that associates a non-negative cost with each of the soft clauses. Any truth assignment τ that satisfies Fh is a solution to F. The cost of a solution τ to F is

\[ \text{cost}(F, \tau) = \sum_{C \in Fs: \tau(C) = 0} c(C), \]

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 \( \text{cost}(F, \tau) \leq \text{cost}(F, \tau') \) holds for any solution τ' to F. The 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 (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.

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. \( G^* \) is an optimal solution of the BTW-BNSL instance, i.e., \( G^* \in \arg \min_{G \in \text{tw}(W)} \sum_{i=1}^N s_i(P_i) \).

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 \( ord \).

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

\[ \sum_{S \in \mathcal{P}_i} P_i^S = 1. \]
Table 1: The main variables used in the MaxSAT encoding of the BTW-BNSL problem.

<table>
<thead>
<tr>
<th>Boolean variables</th>
<th>Interpretation</th>
<th>Indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P^S_i$</td>
<td>represent the parent set of each node in $G^<em>$: $P^S_i = 1$ iff $S$ is the parent set of node $X_i$ in $G^</em>$</td>
<td>for all $i = 1..N$ and $S \in \mathcal{P}_i$</td>
</tr>
<tr>
<td>$M_{ij}$</td>
<td>represent the moralized graph of $G^<em>$: $M_{ij} = 1$ iff $\text{MORAL}(G^</em>)$ contains the edge ${X_i, X_j}$</td>
<td>for all $i, j = 1..N$ such that $i &lt; j$</td>
</tr>
<tr>
<td>$\text{ord}_{ij}$</td>
<td>represent a linear ordering $ord$ of the nodes of $G^*$: $\text{ord}_{ij} = 1$ if node $X_i$ is a predecessor of node $X_j$ in the linear ordering</td>
<td>for all $i, j = 1..N$ such that $i &lt; j$</td>
</tr>
<tr>
<td>$O_{ij}$</td>
<td>represent the ordered graph $\Delta(\text{MORAL}(G^<em>), \text{ord})$: $O_{ij} = 1$ if the ordered graph of $\text{MORAL}(G^</em>)$ under $\text{ord}$ contains the edge $(X_i, X_j)$</td>
<td>for all $i, j = 1..N$ such that $i \neq j$</td>
</tr>
</tbody>
</table>

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 $\text{Eq.(4)}$ 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 $b_1^{i\log_2 N} \ldots b_1^{i\log_2 N}$ form the binary representation $b_1^{i\log_2 N} \ldots b_1^{i\log_2 N}$ 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 $b^k_i = b^k_j$ 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 $k$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 $\{1, \ldots, N\}$ 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

$$
\bigwedge_{k=1}^{\log_2 N} \neg EQ_{ij}^k,
$$

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 $X_j$:

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

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

$$
GT_{ij}^k \iff b^k_i \wedge \neg b^k_j \wedge \bigwedge_{k'=k+1}^{\log_2 N} EQ_{ij}^{k'}, \quad (7)
$$

$$
EQ_{ij}^k \iff (b^k_i \leftrightarrow b^k_j). \quad (8)
$$

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 $tw(\text{MORAL}(G^*)) \leq \mathcal{W}$.

(i) From Parent Sets to the Moralized Graph. We directly connect the choices of parent sets, represented by the $P^S_i$ 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
Learning Bounded Treewidth Bayesian Networks via MaxSAT

\( X_j \in S, \) and (ii) an edge between each pair of distinct nodes \( X_j, X_k \in S. \)

\[
P^S_i \rightarrow \bigwedge_{X_j \in S} M_{ij} \land \bigwedge_{X_j, X_k \in S} M_{jk}. \quad (10)
\]

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) \( X_i \) is in the parent set of \( X_j, \) (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 \( X_k \in X \setminus \{X_i, X_j\}. \)

\[
M_{ij} \rightarrow \bigvee_{S : X_j \in S} P^S_i \lor \bigvee_{S : X_i \in S} P^S_j \lor \bigvee_{X_k \in X \setminus \{X_i, X_j\}} \bigwedge_{S : X_i, X_j \in S} P^S_k \quad (11)
\]

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 (Samer and Veith, 2009). Following Samer and Veith (2009), we do not encode the construction of a tree-decomposition of 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 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 \( i, j, k = 1..N \)

\[
ord^*_{ij} \land ord^*_{jk} \rightarrow ord^*_{ik}. \quad (12)
\]

(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 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 \( \max_{v_i \in V} |\{v_i, v_j \in E : i \prec j\}| \leq W. \) 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; Abió 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

\[
O_{ij} \lor O_{ji} \quad \text{for all } i < j. \quad (14)
\]

-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

\[
O_{ki} \land O_{kj} \rightarrow O_{ij}. \quad (15)
\]

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 \( ord: \)

\[
ord^*_{ij} \rightarrow \neg \ord_{ji}. \quad (16)
\]

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

\[
(\neg P^S_i) \rightarrow s_i(S). \quad (17)
\]

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

\[
\mathcal{C}(\neg P^S_i) = s_i(S). \quad (18)
\]

3.3 Summary of the Encoding

Assume an arbitrary instance \( (X, W, \{s_i\}_{i=1}^N) \) of BTW-BNSL, consisting of a set \( X = \{X_1, \ldots, X_N\} \) 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 $\tau$ 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 $F(X,W,\{s_i\}_{i=1}^N)$, 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 $X_i$.

**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 (Davies and Bacchus, 2013). 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 (“mem”) 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-
Learning Bounded Treewidth Bayesian Networks via MaxSAT

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$N$</th>
<th>treewidth $\leq 2$</th>
<th>treewidth $\leq 3$</th>
<th>treewidth $\leq 4$</th>
<th>#fails</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MS (s)</td>
<td>DP (s)</td>
<td>MS (s)</td>
<td>DP (s)</td>
</tr>
<tr>
<td>Abalone</td>
<td>9</td>
<td>64</td>
<td>7</td>
<td>166</td>
<td>57</td>
</tr>
<tr>
<td>Housing</td>
<td>14</td>
<td>2326</td>
<td>6927</td>
<td>2329</td>
<td>339</td>
</tr>
<tr>
<td>Wine</td>
<td>14</td>
<td>27</td>
<td>6924</td>
<td>22</td>
<td>28900</td>
</tr>
<tr>
<td>Adult</td>
<td>15</td>
<td>998</td>
<td>&gt; 28800</td>
<td>625</td>
<td>&gt; 28800</td>
</tr>
<tr>
<td>Voting</td>
<td>17</td>
<td>22909</td>
<td>&gt; 28800</td>
<td>2.419</td>
<td>mo</td>
</tr>
<tr>
<td>Zoo</td>
<td>17</td>
<td>410</td>
<td>&gt; 28800</td>
<td>412</td>
<td>mo</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>20</td>
<td>315</td>
<td>mo</td>
<td>100</td>
<td>mo</td>
</tr>
<tr>
<td>Heart</td>
<td>23</td>
<td>1198</td>
<td>mo</td>
<td>2186</td>
<td>mo</td>
</tr>
<tr>
<td>Horse</td>
<td>28</td>
<td>192</td>
<td>mo</td>
<td>&gt; 28800</td>
<td>mo</td>
</tr>
<tr>
<td>Flag</td>
<td>29</td>
<td>418</td>
<td>mo</td>
<td>1148</td>
<td>mo</td>
</tr>
<tr>
<td>#fails</td>
<td></td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

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 superstructure of bounded treewidth, and show that this problem is fixed parameter tractable in the treewidth of the super-structure. The treewidth of the superstructure 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 Guerstrin, 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


