Turbocharging Treewidth-Bounded Bayesian Network Structure Learning

Vaidyanathan Peruvemba Ramaswamy and Stefan Szeider
Algorithms and Complexity Group, TU Wien, Vienna, Austria
{vaidyanathan,sz}@ac.tuwien.ac.at

Abstract

We present a new approach for learning the structure of a treewidth-bounded Bayesian Network (BN). The key to our approach is applying an exact method (based on MaxSAT) locally, to improve the score of a heuristically computed BN. This approach allows us to scale the power of exact methods—so far only applicable to BNs with several dozens of random variables—to large BNs with several thousands of random variables. Our experiments show that our method improves the score of BNs provided by state-of-the-art heuristic methods, often significantly.

Introduction

Bayesian network structure learning is the notoriously difficult problem of discovering a Bayesian network (BN) that optimally represents a given set of training data (Chickering 2002). Since exact inference on a BN is exponential in the BN's treewidth (Kwisthout, Bodlaender, and van der Gaag 2010), one is particularly interested in learning BNs of bounded treewidth. However, learning a BN of bounded treewidth that optimally fits the data (i.e., with the largest possible score) is, in turn, an NP-hard task (Korhonen and Parviainen 2013). This predicament caused the research on treewidth-bounded BN structure learning to split into two branches:

1. *Heuristic Learning* (see, e.g., Elidan and Gould (2009); Nie, de Campos, and Ji (2015); Scanagatta et al. (2016, 2018); Benjumeda, Bielza, and Larrañaga (2019)), which is scalable to large BNs with thousands of random variables but with a score that can be far from optimal, and
2. *Exact Learning* (see, e.g., Berg, Järvisalo, and Malone (2014); Korhonen and Parviainen (2013); Parviainen, Farahani, and Lagergren (2014)), which learns optimal BNs but is scalable only to a few dozen random variables.

In this paper, we combine heuristic and exact learning and take the best of both worlds.

The basic idea for our approach is to first compute a BN with a heuristic method (the *global solver*), and then to apply an exact method (the *local solver*) to parts of the heuristic solution. The parts are chosen small enough that they allow an optimal solution reasonably quickly with the exact method. Although the basic idea sounds compelling and reasonably simple, its realization requires several conceptual contributions and new results.

For the global solver, any heuristic algorithm for treewidth-bounded BN learning, such as the recent algorithms k-MAX (Scanagatta et al. 2018) or ETL (Benjumeda, Bielza, and Larrañaga 2019). The local solver's task is significantly more complex than treewidth-bounded BN structure learning, as several additional constraints need to be incorporated. Namely, it is not sufficient that the BN computed by the local solver is acyclic. We need *fortified acyclicity constraints* that prevent cycles that run through the other parts of the BN, which have not been changed by the local solver. Similarly, it is not sufficient that the local BN is of bounded treewidth. We need *fortified treewidth constraints* that prevent the local BN from introducing links between a diverse set of nodes that, together with the other parts of the BN, which have not been changed by the local solver, increase the treewidth.

Given these additional requirements, we propose a new local solver BN-SLIM (*SAT-based Local Improvement Method*), which satisfies the fortified constraints. We formulate a fortified version of the treewidth-bounded BN structure learning problem. In Theorem 1, we show that we can express the fortified constraints with certain *virtual arcs* and *virtual edges*. The virtual arcs represent directed paths that run outside the local instance; with these virtual arcs we can ensure fortified acyclicity. The *virtual edges* represent essential parts of a global tree decomposition using which we can ensure bounded treewidth.

The new formulation of the local problem is well-suited to be expressed as a MaxSAT (*Maximum Satisfiability*) problem and hence allows us to harvest the power of state-of-the-art MaxSAT solvers (which received a significant performance gain over the last decade). A distinctive feature of our encoding is that, in contrast to the virtual edges, the virtual arcs are conditional and depend on the local solver's solution.

Results. We implement BN-SLIM and evaluate it empirically on a large set of benchmark data sets, consisting between 64 and 10,000 random variables and for the treewidth bounds 2, 5, and 8. As the global solver, we use the state-of-the-art heuristic algorithms for treewidth-bounded BN learning k-MAX (Scanagatta et al. 2018), and two variants of ETL (Benjumeda, Bielza, and Larrañaga 2019). k-MAX improves over the k-greedy algorithm (Scanagatta et al. 2016), which was the first algorithm for treewidth-bounded structure learning that scaled to thousands of random variables. The more recent algorithm ETL is reported to perform better than k-MAX in many cases (Benjumeda, Bielza, and Larrañaga 2019).

We consider about a hundred benchmark data sets based on real-world and synthetic data sets, ranging up to 4000 random variables in our experiments. First we run the global solvers on the data sets, followed by running BN-SLIM to improve the score of the DAG they provided. Our results show that after running BN-SLIM for 5 minutes, 73% of all DAGs could be improved; by extending the time for BN-SLIM to 15 minutes, the improvement extends to 82%. We also notice that, overall, BN-SLIM can improve the lower treewidth DAGs more efficiently.

Since k-MAX is an *anytime* algorithm that can produce better and better solutions over time, we can directly compare the improvements achieved by k-MAX after some initial run with the improvements achieved by BN-SLIM. Our experiments show that after an initial run of k-MAX for 30 minutes, it is highly beneficial to stop k-MAX and hand the torch over to BN-SLIM, as BN-SLIM provides improvements at a significantly higher rate. According to the ΔBIC metric, which was used by Scanagatta et al. (2018) for comparing treewidth-bounded BN structure learning algorithms, the results are "extremely positive" in favor of BN-SLIM over k-MAX in a vast majority of the experiments.

We cannot perform such a direct comparison between ETL and BN-SLIM, since the available implementation of ETL does not support an anytime run, but stops after a certain time. Hence, we let ETL finish, and run BN-SLIM afterwards for 30 minutes. The achieved improvement in terms of the ΔBIC metric is "extremely positive" for 84% of all DAGs computed by two variants of ETL.

Related work. The first SAT-encoding for finding the treewidth of a graph was proposed by Samer and Veith (2009). Lodha, Ordyniak, and Szeider (2016) proposed the first SAT-based local improvement method for branchwidth. Recently, SAT encodings have been proposed for other graph and hypergraph width measures (Fichte et al. 2018; Ganian et al. 2019; Lodha, Ordyniak, and Szeider 2017; Schidler and Szeider 2020). So far, there have been four concrete approaches that use the SLIM framework, one for branchwidth (Lodha, Ordyniak, and Szeider 2016, 2019), one for treewidth (Fichte, Lodha, and Szeider 2017), one for treedepth (Peruvemba Ramaswamy and Szeider 2020) and one for decision trees (Schidler and Szeider 2021). SLIM is a meta-heuristic that, similarly to Large Neighborhood Search (Pisinger and Ropke 2010), tries to improve a current solution by exploring its neighborhood of potentially better solutions. As a distinctive feature, SLIM explores highly structurally constrained neighbouring solutions with a complete method (SAT).

Several exact approaches to treewidth-bounded BN structure learning have been proposed. Korhonen and Parviainen (2013) proposed a dynamic-programming approach, and Parviainen, Farahani, and Lagergren (2014) proposed a Mixed-Integer Programming approach. Berg, Järvisalo, and Malone (2014) proposed a MaxSAT approach by extending the basic Samer-Veith encoding for treewidth. Our approach for BN-SLIM uses a similar general strategy, but we encode acyclicity differently. Moreover, BN-SLIM deals with the fortified constraints in terms of virtual edges and virtual arcs.

Since the exact methods are limited to small domains, Nie, de Campos, and Ji (2015, 2016) suggested heuristic approaches that scale up to hundreds of random variables. The k-greedy algorithm proposed by Scanagatta et al. (2016) at NIPS'16 provided a breakthrough, consistently yielding better DAGs than its competitors and scaling up to several thousand of random variables. As mentioned above, k-MAX (Scanagatta et al. 2018) is a more recent improvement over k-greedy. More recently, Benjumeda, Bielza, and Larrañaga (2019) came up with the ETL algorithms, based on local search within the space of structures called elimination trees. These algorithms perform better than k-MAX and k-greedy in many cases.

Preliminaries

Structure learning. We consider the problem of learning the structure (i.e., the DAG) of a BN from complete data set of N instances $D_1, \dots, D_N$ over a set of n categorical random variables $X_1, \dots, X_n$. The goal is to find a DAG $D = (V, E)$ where V is the set of nodes (one for each random variable) and E is the set of arcs (directed edges). The value of a *score function* determines how well a DAG D fits the data; the DAG D, together with local parameters, forms the BN (Koller and Friedman 2009).

We assume that the score is *decomposable*, i.e., being constituted by the sum of the individual random variables' scores. Hence we can assume that the score is given in terms of a *score function* f that assigns each node $v \in V$ and each subset $P \subseteq V \setminus \{v\}$ a real number $f_P(v)$, the *score* of P for v. The score of the entire DAG $D = (V, E)$ is then $f(D) := \sum_{v \in V} f(v, P_D(v))$ where $P_D(v) = \{\, u \in V : (u, v) \in E \,\}$ denotes the *parent set* of v in D. This setting accommodates several popular scores like AIC, BDeu, and BIC (Akaike 1974; Heckerman, Geiger, and Chickering 1995; Schwarz 1978). If P and P' are two potential parent sets of a random variable v such that $P \subsetneq P'$ and $f(v, P') \leq f(v, P)$, then we can safely disregard the potential parent set P' of v. Consequently, we can disregard all nonempty potential parent sets of v with a score $\leq f(v, \emptyset)$. Such a restricted score function is a *score function cache*.

Treewidth. Treewidth is a graph invariant that provides a good indication of how costly probabilistic inference on a BN is. Treewidth is defined on undirected graphs and applies to BNs via the *moralized graph* $M(D) = (V, E_M)$ of the DAG $D = (V, E)$ underlying the BN under consideration, where $E_M = \{\, \{u, v\} : (u, v) \in E \,\} \cup \{\, \{u, v\} : (u, w), (v, w) \in E, u \neq v \,\}$.

A *tree decomposition* $\mathcal{T}$ of a graph G is a pair (T, χ), where T is a tree and χ is a function that assigns each tree node t a set $\chi(t)$ of vertices of G such that the following conditions hold:

T1 For every edge e of G there is a tree node t such that $e \subseteq \chi(t)$.

T2 For every vertex v of G, the set of tree nodes t with $v \in \chi(t)$ induces a non-empty subtree of T.

The sets $\chi(t)$ are called *bags* of the decomposition $\mathcal{T}$, and $\chi(t)$ is the bag associated with the tree node t. The *width* of a tree decomposition (T, χ) is the size of a largest bag minus 1. The *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G.

The *treewidth-bounded BN structure learning problem* takes as input a set V of nodes, a decomposable score function f on V, and an integer W, and it asks to compute a DAG $D = (V, E)$ of treewidth $\leq W$, such that $f(D)$ is maximal.

Local Improvement

Consider an instance (V, f, W) of the treewidth-bounded BN structure learning problem, and assume we have computed an *initial solution* $D = (V, E)$ heuristically, together with a tree decomposition $\mathcal{T} = (T, \chi)$ of width $\leq W$ of the moralized graph $M(D)$.

We select a subtree $S \subseteq T$ such that the number of vertices in $V_S := \bigcup_{t \in V(S)} \chi(t)$ is at most some *budget* B. The budget is a parameter that we specify beforehand, such that the subinstance induced by V_S is small enough to be solved optimally by an exact method, which we call the *local solver*. The local solver computes for each $v \in V_S$ a new parent set, optimizing the score of the resulting DAG $D^{\text{new}} = (V, E^{\text{new}})$.

Consider the induced DAG $D_S^{\text{new}} = (V_S, E_S^{\text{new}})$, where $E_S^{\text{new}} = \{\, (u, v) \in E^{\text{new}} : \{u, v\} \subseteq V_S \,\}$. The local solver ensures that the following conditions are met:

C1 D_S^{new} is acyclic.

C2 The moral graph $M(D_S^{\text{new}})$ has treewidth $\leq W$.

We assume that the local solver certifies C2 by producing a tree decomposition $\mathcal{S}^{\text{new}} = (S^{\text{new}}, \chi^{\text{new}})$ of $M(D_S^{\text{new}})$ of width $\leq W$, which can be used by the global solver.

The two conditions stated above are not sufficient to ensure that D^{new} is acyclic and that treewidth of $M(D^{\text{new}})$ remains bounded by W. Acyclicity can be violated by cycles formed by the combination of the new incoming arcs of vertices in S together with old arcs that are kept from D. The treewidth can increase by a number that is linear in $|V_S|$.

Hence, we need additional side conditions, which we will formulate using the following additional concepts.

Let us call a vertex $v \in V_S$ a *boundary vertex* if there exists a tree node $t \in V(T) \setminus V(S)$ such that $v \in \chi(t)$, i.e., it occurs in some bag outside S. We call the other vertices in V_S *internal vertices*, and the vertices in $V \setminus V_S$ *external vertices*. Further, we call two boundary vertices v, v' *adjacent* if there exists a tree node $t \in V(T) \setminus V(S)$ such that $v, v' \in \chi(t)$, i.e., both vertices occur together in some bag outside S. It is easy to see that any pair of adjacent boundary vertices occur together in a bag of S as well.

For any two adjacent boundary vertices v, v', we call $\{v, v'\}$ a *virtual edge*. Let E_{virt} be the set of all virtual edges. These virtual edges form a clique and serve a similar purpose as the marker cliques used in other work (Fichte, Lodha, and Szeider 2017). The *extended moral graph* $M_{\text{ext}} = (V_S, E_{\text{ext}})$ is obtained from $M(D_S^{\text{new}})$ by adding all virtual edges.

For any two adjacent boundary vertices v, v', we call (v', v) a *virtual arc*, if D^{new} contains a directed path from v' to v, where all the vertices on the path, except for v' and v, are external. Let $E_{\text{virt}}^{\rightarrow}$ be the set of all virtual arcs.

We can now formulate the side conditions.

C3 $\mathcal{S}^{\text{new}}$ is a tree decomposition of the extended moral graph M_{ext}.

C4 For each $v \in V_S$, if $P_{D^{\text{new}}}(v)$ contains external vertices, then there is some $t \in V(T) \setminus V(S)$ such that $P_{D^{\text{new}}}(v) \cup \{v\} \subseteq \chi(t)$.

C5 The digraph $(V_S, E_S^{\text{new}} \cup E_{\text{virt}}^{\rightarrow})$ is acyclic.

We note that condition C4 implies that in D^{new}, all parents of an internal vertex are internal.

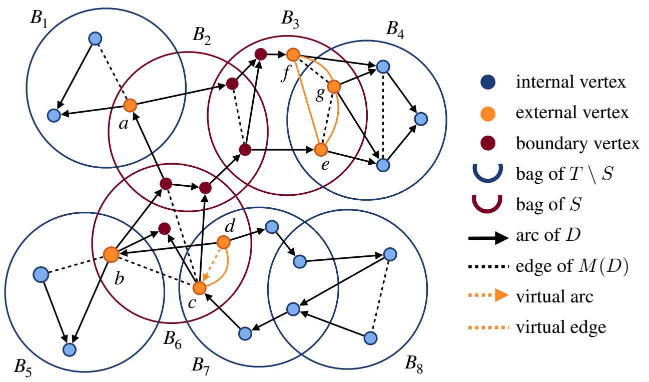

Figure 1: Illustration for Theorem 1. The large circles $B_1, \ldots, B_8$ represent the bags of T, where B_2, B_3, B_6 belong to S. The boundary vertices are $a, \ldots, g$, where c, d are adjacent, and e, f, g are mutually adjacent. Since there is a directed path from d to c using external vertices from the bags B_7 and B_8, there is a virtual arc from d to c.

Theorem 1. *If all the conditions C1–C5 are satisfied, then D^{new} is acyclic, the treewidth of $M(D^{\text{new}})$ is at most W, and the score of D^{new} is at least the score of D.*

Proof. We define a new tree decomposition $\mathcal{T}^{\text{new}} = (T^{\text{new}}, \chi^{\text{new}})$ of $M(D^{\text{new}})$ as follows. Let $T_1, \ldots, T_r$ be the connected components of $T \setminus V(S)$, i.e., the T_i's are the subtrees of T that we get when deleting the subtree S. Let $V_i = \bigcup_{t \in V(T_i)} \chi(t)$, $1 \leq i \leq r$, and observe that each external vertex x belongs to exactly one of the sets $V_1, \ldots, V_r$. Let $B_i = V_S \cap V_i$, $1 \leq i \leq r$, be the set of boundary vertices in V_i. We observe that all the vertices in B_i are mutually adjacent boundary vertices and occur together in a bag $\chi(s_i)$ of $s_i \in V(S)$ and in a bag $\chi(t_i)$, for $t_i \in V(T_i)$, as we can take s_i and t_i to be the two neighboring tree nodes of T with $s_i \in V(S)$ and $t_i \in V(T_i)$. We also observe that each B_i forms a clique in the extended moral graph M_{ext}.

Recall that by assumption, the local solver provides a tree decomposition $\mathcal{S}^{\text{new}} = (S^{\text{new}}, \chi^{\text{new}})$ of $D_S^{\text{new}} = (V_S, E_S^{\text{new}})$ of width $\leq W$. Additionally, by condition C3, $\mathcal{S}^{\text{new}}$ is also a tree decomposition of M_{ext}, and hence, by a basic property of tree decompositions (see, e.g., Bodlaender and Möhring (1993, Lem. 3.1)), there must exist a bag $\chi^{\text{new}}(s_i^*)$,

$s_i^* \in V(S^{\text{new}})$, with $B_i \subseteq \chi^{\text{new}}(s_i^*)$. Hence we can define T^{new} as the tree we get by connecting the disjoint trees $S^{\text{new}}, T_1, \ldots, T_r$ with the edges $\{s_i^*, t_i\}$, $1 \leq i \leq r$. We extend χ^{new} from $V(S^{\text{new}})$ to $V(T^{\text{new}})$ by setting $\chi^{\text{new}}(t) = \chi(t)$ for $t \in \bigcup_{i=1}^{r} V(T_i)$.

Claim 1: $\mathcal{T}^{\text{new}} = (T^{\text{new}}, \chi^{\text{new}})$ is a tree decomposition of $M(D^{\text{new}})$ of width $\leq W$. To prove the claim, we show that $\mathcal{T}^{\text{new}}$ satisfies the conditions T1 and T2.

Condition T1. There are two reasons for an edge $\{u, v\}$ to belong to $M(D^{\text{new}})$: first, because of an arc $(u, v) \in E^{\text{new}}$ and second, because of two arcs $(u, w), (v, w) \in E^{\text{new}}$. *First case:* $(u, v) \in E^{\text{new}}$. If u and v are both external, then $\{u, v\} \subseteq \chi(t) = \chi^{\text{new}}(t)$ for some $t \in V(T^{\text{new}}) \setminus V(S^{\text{new}}) = V(T) \setminus V(S)$. If neither u nor v is external, then $(u, v) \in E_S^{\text{new}}$, and since S^{new} is a tree decomposition of D_S^{new}, $\{u, v\} \subseteq \chi^{\text{new}}(s)$ for some $s \in V(S^{\text{new}})$. If v is external but u isn't, then the arc (u, v) was already present in D, as the parents of external vertices didn't change. Hence, since $\mathcal{T}$ is a tree decomposition of $M(D)$, it follows that $\{u, v\} \subseteq \chi(t) = \chi^{\text{new}}(t)$ for some $t \in V(T^{\text{new}}) \setminus V(S^{\text{new}}) = V(T^{\text{new}}) \setminus V(S)$. If u is external but v isn't, it follows from C4 that $\{u, v\} \subseteq \chi(t) = \chi^{\text{new}}(t)$ for some $t \in V(T^{\text{new}}) \setminus V(S^{\text{new}}) = V(T^{\text{new}}) \setminus V(S)$. *Second case:* $(u, w), (v, w) \in E^{\text{new}}$. If $u, v, w \in V_S$, then $\{u, v\} \in E(M(D_S^{\text{new}}))$, and so $\{u, v\} \subseteq \chi^{\text{new}}(s)$ for some $s \in V(S^{\text{new}})$, since $\mathcal{S}^{\text{new}}$ is a tree decomposition of $M(D_S^{\text{new}})$. If $w \in V_S$ but $u \notin V_S$ or $v \notin V_S$, then C4 implies that $\{u, v\} \subseteq \chi(t) = \chi^{\text{new}}(t)$ for some $t \in V(T^{\text{new}}) \setminus V(S^{\text{new}}) = V(T^{\text{new}}) \setminus V(S)$. If $w \notin V_S$, then u, v are two adjacent boundary vertices, hence $\{u, v\}$ is a virtual edge which, by C3, means $\{u, v\} \subseteq \chi^{\text{new}}(s)$ for some $s \in V(S^{\text{new}})$. We conclude that T1 holds.

Condition T2. Let $v \in V$. If v is external, then there is exactly one $i \in \{1, \ldots, r\}$, such that $v \in V_i = \bigcup_{t \in V(T_i)} \chi(t)$. Since we do not change the tree decomposition of T_i, condition T2 carries over from $\mathcal{T}$ to $\mathcal{T}^{\text{new}}$. Similarly, if v is internal, then v does not appear in any bag $\chi^{\text{new}}(t)$ for $t \in V(T^{\text{new}}) \setminus V(S^{\text{new}})$, hence condition T2 carries over from $\mathcal{S}^{\text{new}}$ to $\mathcal{T}^{\text{new}}$. It remains to consider the case where v is a boundary vertex. The tree nodes $t \in V(S^{\text{new}})$ with $v \in \chi(t)$ are connected, because $\mathcal{S}^{\text{new}}$ satisfies T2, and for $B_i : v \in B_i$, the tree nodes $t \in V(T_i)$ for which $v \in \chi(t)$ are connected, since $\mathcal{T}$ satisfies T2. By construction of T^{new}, if $v \in B_i$, then there are neighboring tree nodes $s_i^* \in V(S^{\text{new}})$ and $t_i \in V(T_i)$ with $v \in \chi^{\text{new}}(s_i^*) \cap \chi^{\text{new}}(t_i)$. Hence all the tree nodes $t \in V(T^{\text{new}})$ with $v \in \chi(t)$ are connected, and T2 also holds for boundary vertices.

To conclude the proof of the claim, it remains to observe the width of $\mathcal{T}^{\text{new}}$ cannot exceed the widths of $\mathcal{T}$ or $\mathcal{S}^{\text{new}}$, hence the width of $\mathcal{T}^{\text{new}}$ is at most W.

Claim 2: D^{new} is acyclic. To prove the claim, suppose to the contrary that D contains a directed cycle $C = (V(C), E(C))$. The cycle cannot lie entirely in D_S^{new}, nor can it lie entirely in $D^{\text{new}} - V_S = D - V_S$, because D_S^{new} and D are acyclic. Hence, C contains at least one arc from $V_S \times (V \setminus V_S)$ and at least one arc from $(V \setminus V_S) \times V_S$. Let $(v_j, x_j) \in E(C) \cap (V_S \times (V \setminus V_S))$ and $(x'_j, v'_j) \in E(C) \cap ((V \setminus V_S) \times V_S)$, for $0 \leq j \leq p$, be these arcs, such that they appear on C in the order (v'_0, x'_0), (x_0, v_0), $(v'_1, x'_1), \ldots, (v'_p, x'_p)$, (x_p, v_p). It is possible that $x'_j = x_j$ or $v_j = v'_{j+1}$. We observe that the vertices on the path from x'_j to x_j on C all belong to some $V_i = \bigcup_{t \in V(T_i)} \chi(t)$. Hence v_j and v'_j are adjacent boundary vertices, and $E_{\text{virt}}^{\rightarrow}$ contains all the arcs (v'_j, v_j), $1 \leq j \leq p$. However, the cycle C contains also the paths from v_j to $v'_{j+1 \pmod p}$, for $1 \leq j \leq p$, which only run through vertices in V_S. These paths, together with the virtual arcs (v'_j, v_j) form a cycle C' which lies in $(V_S, E_S^{\text{new}} \cup E_{\text{virt}}^{\rightarrow})$. This contradicts C5 which requires that this digraph be acyclic. Hence the claim holds.

Claim 3: The score of D^{new} is at least the score of D. We observe that by taking $D^{\text{new}} = D$ we have a solution that satisfies all the required conditions and maintains the score. $\square$

Implementing the Local Improvement

In this section, we first discuss how the set S representing the subinstance is constructed. Then we provide a detailed explanation of the MaxSAT encoding that is responsible for solving the subinstance.

Constructing the subinstance. For this section, we follow the same notation as used in the previous section. To construct the subinstance, we initialize the subtree S with a tree node r picked at random from $V(T)$. We then expand S by performing a bread-first search from $V(S)$ and adding a new tree node to S as long as the size of V_S does not exceed the budget. Next, we compute E_{virt} for the chosen S. Finally, we prune the parent sets of each vertex so as to only retain those parent sets which satisfy conditions C3 and C4. This can be done by first checking, for each parent set, if the required tree node t is present $V(T) \setminus V(S)$, and if it does, we record the set of virtual arcs that are imposed by this parent set as long as none of the virtual arcs are self-loops. For each $v \in S$ and $P \in \mathcal{P}_v$, we denote by $A_{\text{virt}}^{\rightarrow}(v, P)$ the set of imposed virtual arcs when v has the parent set P in D^{new}. We denote by $\mathcal{P}_v$, the collection of parent sets of node v that remain after this pruning process. Notice that, under this pruning, all remaining parent sets $P \in \mathcal{P}_v$ satisfy C4. Also note that, since $E_{\text{virt}}^{\rightarrow}$ is conditional on the chosen parent sets, it cannot be precomputed.

Further, since we intend to solve the subinstance using a MaxSAT encoding, we need to ensure that the score of each parent set is non-negative. Recall that $\mathcal{P}_v$ only contains those non-empty parent sets whose score is at least that of the empty parent set. Thus, we may assume that the empty parent set has the lowest score among all the parents of a certain vertex. Consequently, we can adjust the score function by setting $f'_P(v) = f_P(v) - f_\emptyset(v)$ for $v \in S$ and $P \in \mathcal{P}_v$, which implies that $f'_P(v) \geq 0$ for all $v \in S$ and $P \in \mathcal{P}_v$.

MaxSAT encoding. We now describe the weighted partial MaxSAT instance that encodes conditions C1–C5. We build on top of the SAT encoding proposed by Samer and Veith (2009). The only difference in our case is that there are no explicit edges and hence we do not require the corresponding clauses. Instead, the edges of the moralized graph are dependent on and decided by other variables that govern the DAG structure. For convenience, let n denote the size of the subinstance, i.e., $n := |S|$. A part of the encoding is based on the elimination ordering of a tree decomposition (see, e.g., Samer and Veith (2009, Sec. 2)).

The main variables used in our encoding are

- variables par_v^P represent for each node $v \in S$ the chosen parent set P,
- $n(n-1)/2$ variables $\mathrm{acyc}_{u,v}$ represent the topological ordering of D_S^{new},
- $n(n-1)/2$ variables $\mathrm{ord}_{u,v}$ represent the elimination ordering of the tree decomposition,
- n^2 variables $\mathrm{arc}_{u,v}$ represent the arcs in the moralized graph M_{ext}, along with the *fill-in edges* (see Samer and Veith (2009)).

Since $\mathrm{acyc}_{u,v}$ and $\mathrm{ord}_{u,v}$ represent linear orderings, we enforce transitivity of these variables by means of the clauses

$$\left.\begin{array}{r}(\mathrm{acyc}^*_{u,v} \wedge \mathrm{acyc}^*_{v,w}) \rightarrow \mathrm{acyc}^*_{u,w}\\ (\mathrm{ord}^*_{u,v} \wedge \mathrm{ord}^*_{v,w}) \rightarrow \mathrm{ord}^*_{u,w}\end{array}\right\} \text{ for distinct } u,v,w \in S.$$

To prevent self-loops in the moralized graph, we add the clauses

$$\neg \mathrm{arc}_{v,v} \quad \text{for } v \in S.$$

For each node $v \in S$, and parent set $P \in \mathcal{P}_v$, the variable par_v^P is true if and only if P is the parent set of v. Since each node must have exactly one parent set, we introduce the cardinality constraint $\sum_{P \in \mathcal{P}_v} \mathrm{par}_v^P = 1$ for $v \in S$.

Next, for each node v, parent set P, and $u \in P$, if P is the parent set of v then u must precede v in the topological ordering. Hence we add the clause

$$\mathrm{par}_v^P \rightarrow \mathrm{acyc}_{u,v} \quad \text{for } v \in S, P \in \mathcal{P}_v, \text{ and } u \in P.$$

Similarly, for each node v, parent set P, and $u \in P$, if P is the parent set of v then we must add an arc in the moralized graph respecting the elimination ordering between u and v, as follows:

$$\left.\begin{array}{r}(\mathrm{par}_v^P \wedge \mathrm{ord}_{u,v}) \rightarrow \mathrm{arc}_{u,v}\\ (\mathrm{par}_v^P \wedge \mathrm{ord}_{v,u}) \rightarrow \mathrm{arc}_{v,u}\end{array}\right\} \begin{array}{c}\text{for } v \in S, P \in \mathcal{P}_v,\\ \text{and } u \in P.\end{array}$$

Next, we encode the moralization by adding an arc between every pair of parents of a node, using the following clauses

$$\left.\begin{array}{r}(\mathrm{par}_v^P \wedge \mathrm{ord}_{u,w}) \rightarrow \mathrm{arc}_{u,w}\\ (\mathrm{par}_v^P \wedge \mathrm{ord}_{w,u}) \rightarrow \mathrm{arc}_{w,u}\end{array}\right\} \begin{array}{c}\text{for } v \in S, P \in \mathcal{P}_v,\\ \text{and } u,w \in P.\end{array}$$

Now, we encode the fill-in edges, with the following clauses

$$\left.\begin{array}{r}(\mathrm{arc}_{u,v} \wedge \mathrm{arc}_{u,w} \wedge \mathrm{ord}_{v,w}) \rightarrow \mathrm{arc}_{v,w}\\ (\mathrm{arc}_{u,v} \wedge \mathrm{arc}_{u,w} \wedge \mathrm{ord}_{w,v}) \rightarrow \mathrm{arc}_{w,v}\end{array}\right\} \text{ for } u,v,w \in S.$$

Lastly, to bound the treewidth, we add a cardinality constraint on the number of outgoing arcs for each node as follows

$$\sum_{w \in S, w \neq v} \mathrm{arc}_{v,w} \leq W \quad \text{for } v \in S.$$

To complete the basic encoding, for every node $v \in S$, and every parent set $P \in \mathcal{P}_v$ we add a soft clause weighted by the score of the parent set as follows

$$(\mathrm{par}_v^P) : \text{weight } f'_P(v) \quad \text{for } v \in S, P \in \mathcal{P}_v.$$

To speed up the solving, we encode that for every pair of nodes, at most one of the arcs between them can exist. We add the following redundant clauses

$$\neg \mathrm{arc}_{u,v} \vee \neg \mathrm{arc}_{v,u} \quad \text{for } u,v \in S.$$

Now, we describe the additional clauses required to satisfy the fortified constraints, and thus conditions C3 and C5. For every virtual edge $\{u,v\} \in E_{\mathrm{virt}}$, we introduce a forced arc depending on the elimination ordering using the following pair of clauses

$$\mathrm{ord}^*_{u,v} \rightarrow \mathrm{arc}_{u,v} \wedge \mathrm{ord}^*_{v,u} \rightarrow \mathrm{arc}_{v,u} \quad \text{for } \{u,v\} \in E_{\mathrm{virt}}.$$

This takes care of the fortified treewidth constraints, satisfying C3 and ensuring that the edge $\{u,v\} \subseteq \chi(s)$ for some $s \in V(S)$. Finally, we add the clauses that encode the forced arcs $E^{\rightarrow}_{\mathrm{virt}}$. For each $v \in S$, $P \in \mathcal{P}_v$, and $(u,v) \in A^{\rightarrow}_{\mathrm{virt}}(v,P)$, we add the clause

$$\mathrm{par}_v^P \rightarrow \mathrm{acyc}^*_{u,v},$$

which forces the virtual arc (u,v) if P is the parent set of v in D^{new}, thereby handling the fortified acyclicity constraints and ensuring that C5 is satisfied.

This concludes the definition of the MaxSAT instance, to which we will refer as $\Phi_{D,f}(S)$. We refer to the weight of a satisfying assignment τ of $\Phi_{D,f}(S)$ as the sum of the weights of all the soft clauses satisfied by τ. Let $\alpha(S) := \sum_{v \in S} f_\emptyset(v)$. To each satisfying assignment τ of $\Phi_{D,f}(S)$ we can associate for each $v \in V$ the corresponding parent set, which in turn determines a directed graph D^{new}. Due to Theorem 1, the treewidth of $M(D^{\mathrm{new}})$ is bounded by W, and D^{new} is acyclic. By construction of $\Phi_{D,f}(S)$, the weight of τ equals $\sum_{v \in S} f'_P(v) = f(D_S^{\mathrm{new}}) - \alpha(S)$. Conversely, if we pick new parent sets for the vertices in S such that all the conditions C1–C5 are satisfied, then by construction of $\Phi_{D,f}(S)$, the corresponding truth assignment τ satisfies $\Phi_{D,f}(S)$, and its weight is $\sum_{v \in S} f'_P(v) = f(D_S^{\mathrm{new}}) - \alpha(S)$. In particular, let K_0 be the weight of the truth assignment which corresponds to the parent sets of S as defined by the input DAG D. We summarize these observations in the following theorem.

Theorem 2. *$\Phi_{D,f}(S)$ has a solution of weight K if and only if there are new parent sets for the vertices in S giving rise to a DAG D^{new} with $f(D^{\mathrm{new}}) - f(D) = K - K_0$.*

Experimental Evaluation

The current state-of-the-art heuristic algorithms for solving the treewidth-bounded BN structure learning problem are the k-MAX algorithm by Scanagatta et al. (2018) and the ETL algorithms by Benjumeda, Bielza, and Larrañaga (2019) (available as two variants–the default variant ETL$_\mathrm{d}$ and the poly-time variant ETL$_\mathrm{p}$), therefore, we analyze the benefit of applying BN-SLIM on top of these algorithms. It is worth noting that both k-MAX and BN-SLIM are anytime algorithms, i.e., they run indefinitely long and can be halted at any instant to output the best solution found so far; ETL, on the other hand, as per the available implementation, is deterministic and terminates when it fails to find any new improvements. This distinction affects the nature of the experiments conducted to draw a comparison between the different algorithms. However, for the most part, we closely follow the experimental setup (including data sets, timeouts, comparison metrics) used by Scanagatta et al. (2018) to compare k-MAX with previous approaches.

Name	n	Name	n	Name	n	Name	n	Name	n	Name	n
NLTCS	16	Accidents	111	Pumsb-star	163	Book	500	Movie reviews	1001	r2*	2000
MSNBC	17	Mushrooms	112	DNA	180	EachMovie	500	munin*	1041	r3*	2000
KDDCup2k	65	Adult	123	Kosarek	190	NIPS	500	BBC	1058	r4*	2000
Plants	69	Connect4	126	andes*	223	link*	724	Voting	1359	r5*	4000
Audio	100	OCR Letters	128	MSWeb	294	WebKB	839	Ad	1556	r6*	4000
Jester	100	Retail	135	diabetes*	413	Reuters-52	889	r0*	2000	r7*	4000
Netflix	100	RCV-1	150	pigs*	441	20 NewsGroup	910	r1*	2000		

Table 1: Data sets used for experimentation. Data sets marked with a * are synthetic. n is the number of random variables. The number of samples ranges from 100 to 291326 for the real data sets, while, 5000 samples were drawn for the synthetic data sets.

Since BN-SLIM needs an initial heuristic solution, we enlist either k-MAX, ETL$_\text{d}$, or ETL$_\text{p}$ for this purpose. We denote by BN-SLIM(X), the algorithm which applies BN-SLIM on an initial solution provided by X where X $\in$ {k-MAX, ETL$_\text{d}$, ETL$_\text{p}$}. We run all our experiments with treewidth bounds 2, 5, 8 for each data set following Scanagatta et al. (2018). All reported BN-SLIM results are averages over three random seeds (see supp. material for details).

Setup. We run all our experiments on a 4-core Intel Xeon E5540 2.53 GHz CPU, with each process having access to 8GB RAM. We use UWrMaxSat as the MaxSAT-solver primarily due to its anytime nature (available at the 2019 MaxSAT Evaluation webpage[1]). We tried other solvers but found that UWrMaxSat works best for our use case. We use the BNGenerator package (Ide 2015) in conjunction with the BBNConvertor tool (Guo 2002) to generate and reformat random Bayesian Networks. We also use the implementation of the k-MAX algorithm available as a part of the BLIP package (Scanagatta 2015). For the ETL algorithms we use the software made available[2] by Benjumeda, Bielza, and Larrañaga (2019). We implement the local improvement algorithm in Python 3.6.9, using the NetworkX 2.4 graph library (Hagberg, Schult, and Swart 2008). The source code is attached as supplementary material, and we intend to make it publicly available.

We first conducted a preliminary analysis on 20 data sets to find out the best values for the *budget* (maximum number of random variables in a subinstance) and the *timeout* (per MaxSAT call) of BN-SLIM. We tested out budget values 7, 10, and 17, and timeout values 1s, 2s, and 5s, and finally settled on a budget of 10 and a timeout of 2 seconds for our experiments.

Data sets. We consider 99 data sets for our experiments. 84 of these come from *real-world* benchmarks. These are based on the benchmarks introduced by Lowd and Davis (2010); Van Haaren and Davis (2012); Bekker et al. (2015); Larochelle, Bengio, and Turian (2010), a subset of which has been used by Scanagatta et al. (2018). These benchmarks are publicly available [3] in the form of pre-partitioned data sets. There are three data sets corresponding to each of the 28 benchmarks (see Table 1).

The remaining 15 data sets are classified as *synthetic* as they are obtained by drawing 5000 samples from known BNs (see Table 1). Five of these BNs are commonly used in the literature as benchmarks[4], and we generated the remaining 10 BNs randomly using the BNGenerator tool with more random variables than the previously mentioned data sets. Overall, the collection of data sets provides a wide variety of the data's nature and the different parameters.

Both k-MAX and BN-SLIM take a score function cache as input, while ETL requires the samples themselves and computes the required scores on-the-fly. We thus compute the score function cache using the scoring module provided as a part of ETL's source code. More specifically, we first obtain the parent set tuples using independence selection (available in the BLIP package), and then we recompute the scores for these tuples using ETL's scoring module. This cache is used as input to both BN-SLIM and k-MAX. This provides a level playing field and improves comparability between the different algorithms.

While computing these score function caches, the scoring function module was unable to process two data sets and hence we discarded these two data sets. The final list of data sets is shown in Table 1. Further, k-MAX crashes for 3 data sets and hence we disregard these for any experiments involving k-MAX or BN-SLIM(k-MAX).

Evaluation metric. For evaluating our algorithm's performance, we use the same metric as Scanagatta et al., i.e., ΔBIC, which is the difference between the BIC scores of two solutions. Given a DAG D, the BIC score approximates the logarithm of the marginal likelihood of D. Thus, given two DAGs D_1 and D_2, the difference in their BIC scores approximates the ratio of their respective marginal likelihoods which is the Bayes Factor (Raftery 1995). A positive ΔBIC score signifies positive evidence towards D_1 and a negative ΔBIC score signifies positive evidence towards D_2. The ΔBIC values can be mapped to a scale of qualitative categories (Raftery 1995) as follows:

	Category	ΔBIC		Category	ΔBIC
🟥	extremely neg.	$(-\infty, -10)$	🟦	extremely pos.	$(10, \infty)$
🟥	strongly neg.	$(-10, -6)$	🟦	strongly pos.	$(6, 10)$
🟥	negative	$(-6, -2)$	🟦	positive	$(2, 6)$

Experimental results. The primary focus of our experimentation is to analyze the benefit gained by applying

[1]https://maxsat-evaluations.github.io/2019/descriptions.html
[2]https://github.com/marcobb8/et-learn
[3]https://github.com/arranger1044/DEBD
[4]https://www.bnlearn.com/bnrepository/

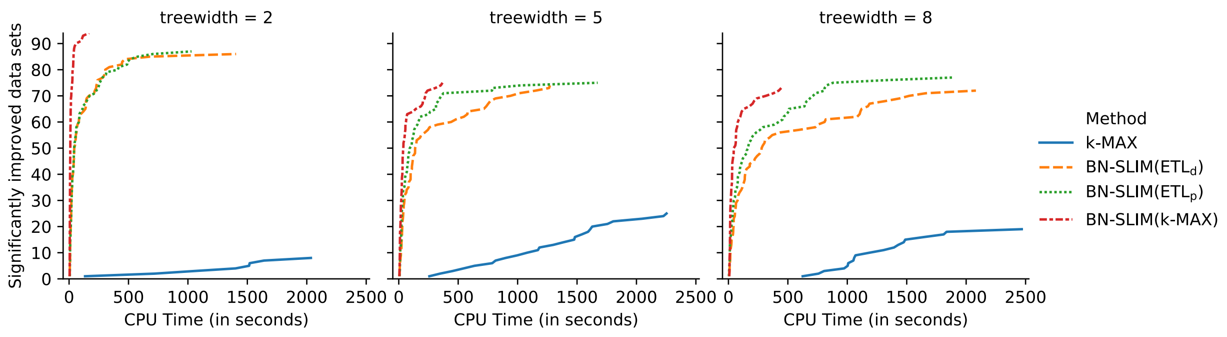

Figure 2: CDF plots showing the number of significantly improved data sets (ΔBIC ≥ 10) across 94 data sets

BN-SLIM on top of other heuristics and not to compare between the different heuristics. To this end, we run BN-SLIM for 60 minutes on top of the initial solution provided by k-MAX, ETL$_d$, and ETL$_p$ and measure the time required for BN-SLIM to obtain a solution that counts as extremely positive evidence with respect to the initial solution. The initial solution by k-MAX is the solution captured at the 30-minute mark, whereas the initial solution by ETL is the final solution obtained upon termination. The maximum time required for computing the initial solution on any individual instance, by both ETL$_d$ and ETL$_p$, is around 3.5 hours. For comparison, we let k-MAX continue running for 60 more minutes after it has produced the initial solution.

Fig. 2 shows the results of this analysis. We consider a data set to be *significantly improved* if BN-SLIM is able to improve by at least 10 BIC points over the initial heuristic solution. We observe that BN-SLIM improves over k-MAX much more efficiently as over ETL. Giving k-MAX more time for computing the initial solution increases this discrepancy even further, as the improvement rate of k-MAX rapidly slows down after 30 minutes. Averaging over all the heuristics, BN-SLIM can produce a solution with extremely positive evidence for 95%, 79%, and 78% of instances for treewidth bounds 2, 5, and 8, respectively.

Fig. 3 (top) shows the ΔBIC values from comparing the BN-SLIM(ETL) solution after 30 minutes to the corresponding initial solution by ETL. We can see that BN-SLIM(ETL) can secure extremely positive evidence for a significant number of data sets across all tested treewidth bounds, with a smaller treewidth being more favorable.

Due to the anytime nature of k-MAX, we can compare it against BN-SLIM(k-MAX) in a "race." We run both simultaneously for one hour, where out of the time allotted to BN-SLIM(k-MAX), 30 minutes are used to generate the initial solution, and the remaining 30 minutes are used to improve this initial solution. Fig. 3 (bottom) shows the ΔBIC values of comparing k-MAX and BN-SLIM(k-MAX) at the one hour mark. Similar to BN-SLIM(ETL) we observe that BN-SLIM(k-MAX) outperforms k-MAX on a significant number of instances, and on all instances for treewidth 2.

The experimental evaluation demonstrates BN-SLIM approach's effectiveness and the combined power as a heuristic method of BN-SLIM(k-MAX) and BN-SLIM(ETL).

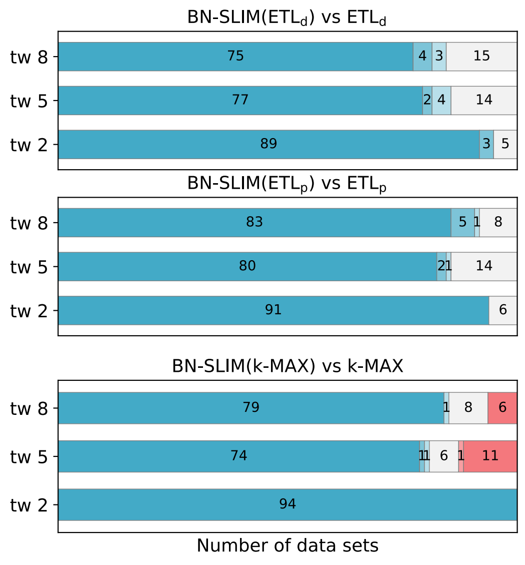

Figure 3: Comparison between BN-SLIM(X) and BN-SLIM

Conclusion

With BN-SLIM, we have presented a novel method for improving the outcome of treewidth-bounded BN structure learning heuristics. We have demonstrated its robustness and performance by applying BN-SLIM to the solution provided by the state-of-the-art heuristics k-MAX, ETL$_d$, and ETL$_p$. The approach of BN-SLIM is based on exact reasoning via MaxSAT, which is fundamentally different from the mentioned heuristics. Consequently, both approaches complement each other, and their combination provides significantly better solutions than any of the heuristics alone. Simultaneoulsy, the combination still scales to large instances with thousands of random variables, which are far out of reach for exact methods alone. Thus, BN-SLIM combines the best of both worlds.

The highly encouraging experimental outcome suggests several avenues for future work, which include the development of more sophisticated subinstance selection schemes, the inclusion of variable fidelity sampling (crude for the global solver, fine-grained for the local solver), as well as more complex collaboration protocols between local and global solver in a distributed setting.

Acknowledgements

The authors acknowledge the support by the FWF (projects P32441 and W1255) and by the WWTF (project ICT19-065).

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE transactions on automatic control* 19(6): 716–723.

Bekker, J.; Davis, J.; Choi, A.; Darwiche, A.; and Van den Broeck, G. 2015. Tractable learning for complex probability queries. In *Advances in Neural Information Processing Systems*, 2242–2250.

Benjumeda, M.; Bielza, C.; and Larrañaga, P. 2019. Learning tractable Bayesian networks in the space of elimination orders. *Artificial Intelligence* 274: 66–90.

Berg, J.; Järvisalo, M.; and Malone, B. M. 2014. Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, 86–95. JMLR.org.

Bodlaender, H. L.; and Möhring, R. H. 1993. The Pathwidth and Treewidth of Cographs. *SIAM J. Discrete Math.* 6(2): 181–188. doi:10.1137/0406014.

Chickering, D. M. 2002. Learning Equivalence classes of Bayesian-Network Structures. *J. Mach. Learn. Res.* 2: 445–498.

Elidan, G.; and Gould, S. 2009. Learning Bounded Treewidth Bayesian Networks. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, 417–424. Curran Associates, Inc.

Fichte, J. K.; Hecher, M.; Lodha, N.; and Szeider, S. 2018. An SMT Approach to Fractional Hypertree Width. In Hooker, J. N., ed., *Proceedings of CP 2018, the 24rd International Conference on Principles and Practice of Constraint Programming*, volume 11008 of *Lecture Notes in Computer Science*, 109–127. Springer Verlag. doi:10.1007/978-3-319-98334-9_8.

Fichte, J. K.; Lodha, N.; and Szeider, S. 2017. SAT-Based Local Improvement for Finding Tree Decompositions of Small Width. In Gaspers, S.; and Walsh, T., eds., *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, 401–411. Springer Verlag. doi:10.1007/978-3-319-66263-3_25.

Ganian, R.; Lodha, N.; Ordyniak, S.; and Szeider, S. 2019. SAT-Encodings for Treecut Width and Treedepth. In Kobourov, S. G.; and Meyerhenke, H., eds., *Proceedings of ALENEX 2019, the 21st Workshop on Algorithm Engineering and Experiments*, 117–129. SIAM. doi:10.1137/1.9781611975499.10.

Guo, H. 2002. BBNConvertor – Bayesian Networks Formats Convertor. URL http://kdd.cs.ksu.edu/KDD/Groups/Probabilistic-Reasoning/convertor.html. [Last Accessed: March 15, 2021].

Hagberg, A. A.; Schult, D. A.; and Swart, P. J. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15. Pasadena, CA USA.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20(3): 197–243.

Ide, J. S. 2015. BNGenerator – A generator for random Bayesian network. URL http://sites.poli.usp.br/pmr/ltd/Software/BNGenerator. [Last Accessed: March 15, 2021].

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Korhonen, J. H.; and Parviainen, P. 2013. Exact Learning of Bounded Tree-width Bayesian Networks. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Workshop and Conference Proceedings*, 370–378. JMLR.org.

Kwisthout, J.; Bodlaender, H. L.; and van der Gaag, L. C. 2010. The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 237–242. IOS Press.

Larochelle, H.; Bengio, Y.; and Turian, J. 2010. Tractable multivariate binary density estimation and the restricted Boltzmann forest. *Neural Computation* 22(9): 2285–2307.

Lodha, N.; Ordyniak, S.; and Szeider, S. 2016. A SAT Approach to Branchwidth. In Creignou, N.; and Berre, D. L., eds., *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, 179–195. Springer Verlag. doi:10.1007/978-3-319-40970-2_12.

Lodha, N.; Ordyniak, S.; and Szeider, S. 2017. SAT-Encodings for Special Treewidth and Pathwidth. In Gaspers, S.; and Walsh, T., eds., *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, 429–445. Springer Verlag. doi:10.1007/978-3-319-66263-3_27.

Lodha, N.; Ordyniak, S.; and Szeider, S. 2019. A SAT Approach to Branchwidth. *ACM Trans. Comput. Log.* 20(3): 15:1–15:24. doi:10.1145/3326159. URL http://www.ac.tuwien.ac.at/files/tr/ac-tr-19-010.pdf.

Lowd, D.; and Davis, J. 2010. Learning Markov network structure with decision trees. In *2010 IEEE International Conference on Data Mining*, 334–343. IEEE.

Nie, S.; de Campos, C. P.; and Ji, Q. 2015. Learning Bounded Tree-Width Bayesian Networks via Sampling. In Destercke, S.; and Denoeux, T., eds., *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 of *Lecture Notes in Computer Science*, 387–396. Springer Verlag.

Nie, S.; de Campos, C. P.; and Ji, Q. 2016. Learning Bayesian Networks with Bounded Tree-width via Guided Search. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 3294–3300. AAAI Press.

Parviainen, P.; Farahani, H. S.; and Lagergren, J. 2014. Learning Bounded Tree-width Bayesian Networks using Integer Linear Programming. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, 751–759. JMLR.org.

Peruvemba Ramaswamy, V.; and Szeider, S. 2020. MaxSAT-Based Postprocessing for Treedepth. In Simonis, H., ed., *Principles and Practice of Constraint Programming*, 478–495. Cham: Springer International Publishing. ISBN 978-3-030-58475-7.

Pisinger, D.; and Ropke, S. 2010. Large neighborhood search. In *Handbook of metaheuristics*, 399–419. Springer.

Raftery, A. E. 1995. Bayesian Model Selection in Social Research. *Sociological Methodology* 25: 111–163. ISSN 00811750, 14679531. URL http://www.jstor.org/stable/271063.

Samer, M.; and Veith, H. 2009. Encoding Treewidth into SAT. In *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, 45–50. Springer Verlag.

Scanagatta, M. 2015. BLIP – Bayesian Network learning and inference package . URL https://ipg.idsia.ch/software/blip. [Last Accessed: March 15, 2021].

Scanagatta, M.; Corani, G.; de Campos, C. P.; and Zaffalon, M. 2016. Learning Treewidth-Bounded Bayesian Networks with Thousands of Variables. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 1462–1470.

Scanagatta, M.; Corani, G.; Zaffalon, M.; Yoo, J.; and Kang, U. 2018. Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets. *Int. J. Approx. Reason* 95: 152–166.

Schidler, A.; and Szeider, S. 2020. Computing Optimal Hypertree Decompositions. In Blelloch, G.; and Finocchi, I., eds., *Proceedings of ALENEX 2020, the 22nd Workshop on Algorithm Engineering and Experiments*, 1–11. SIAM.

Schidler, A.; and Szeider, S. 2021. SAT-based Decision Tree Learning for Large Data Sets. In *Proceedings of AAAI'21, the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press.

Schwarz, G. 1978. Estimating the dimension of a model. *The Annals of Statistics* 6(2): 461–464.

Van Haaren, J.; and Davis, J. 2012. Markov network structure learning: A randomized feature generation approach. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.