Machine Learning manuscript No.
(will be inserted by the editor)

Approximate Structure Learning for Large Bayesian Networks

Mauro Scanagatta · Giorgio Corani · Cassio P. de Campos · Marco Zaffalon

Abstract We present approximate structure learning algorithms for Bayesian networks. We discuss on the two main phases of the task: the preparation of the cache of the scores and structure optimization, both with bounded and unbounded treewidth. We improve on state-of-the-art methods that rely on an ordering-based search by sampling more effectively the space of the orders. This allows for a remarkable improvement in learning Bayesian networks from thousands of variables. We also present a thorough study of the accuracy and the running time of inference, comparing bounded-treewidth and unbounded-treewidth models.

1 Introduction

Score-based structure learning of Bayesian networks is the task of finding the highest-scoring directed acyclic graph (DAG), where the score function measures the appropriateness of the DAG for the data. This task is NP-hard (Chickering et al., 2014), and is the subject of intense, cutting-edge research. Even using the most recent theoretical advances (Cussens et al., 2017) exact learning can be impractical, even if one restricts themselves to cases where the best DAG has at most two parents per node. Hence, approximate methods are necessary to tackle structure learning, especially in order to scale to domains with a large number of variables.

The task is usually accomplished in two phases: identification of a list of candidate parent sets for each node (which we call *parent set identification*) and optimal assignment of the parent set of each node (which we call *structure optimization*). Most research so far has focused on structure optimization. For instance, there are *exact* approaches based on dynamic programming (Koivisto & Sood, 2004; Silander & Myllymaki, 2006), branch and bound (de Campos & Ji, 2011; de Campos et al., 2009), linear and integer programming (Jaakkola et al., 2010), shortest-path heuristics (Yuan & Malone, 2013, 2012), to name a few. A state-of-the-art approach is implemented by the software Gobnilp (Bartlett & Cussens, 2017; Cussens, 2011), which adopts a branch-and-cut idea using linear integer programming.

M. Scanagatta, G. Corani, and M. Zaffalon are with
Istituto Dalle Molle di studi sull'Intelligenza Artificiale (IDSIA), Manno, Switzerland
C. P. de Campos is with
Queen's University Belfast, Northern Ireland, UK and Utrecht University, Netherlands
E-mail:
{mauro,giorgio,zaffalon}@idsia.ch
c.decampos@uu.nl

This is an anytime algorithm; thus it provides an approximate solution at any step of the computation (as long as a first valid graph has been found) until it eventually reaches the exact solution. Yet, exact solvers do not scale to domains with a large number of variables. Experiments so far arrive up to few hundreds of variables at their best, and only in particular instances. One of the most scalable approaches with good empirical results to score-based structure learning is the Acyclic Selection Ordering-Based Search (Scanagatta et al., 2015); this is an approximate algorithm that scales up to thousands of variables.

Once the Bayesian network has been learned, one can gain insights into how variables relate to each other; for instance, this is an important goal when studying gene regulatory networks. Yet, in other applications, efficient inference is required in order to make predictions using the model. The time complexity of (exact) inference grows exponentially with a property of the DAG called *treewidth* (Bodlaender et al., 2001). The solvers discussed so far learn expressive models that are suitable to understanding the domain and analyzing the relationship among variables, but they perform structure learning without bounding the treewidth of the learned model. Learning Bayesian networks with bounded treewidth is indeed very challenging, since it generalizes the unbounded problem (and even determining the treewidth of a DAG is NP-hard). Because of that, the machine learning community has put effort in developing alternative modeling techniques in order to obtain tractable inference, such as arithmetic circuits (Darwiche, 2009, Chap.12) and their learning from data (Lowd & Domingos, 2008), sum-product networks (Poon & Domingos, 2011; Rooshenas & Lowd, 2014), among others.

Recent advances have made possible to achieve some results for structure learning of Bayesian networks with bounded treewidth. Some exact methods (Berg et al., 2014; Korhonen & Parviainen, 2013; Parviainen et al., 2014) have been proposed, but they do not scale to more than hundreds of variables. Later, some approximate approaches have been devised (Nie et al., 2015), which scale to a few hundreds of variables. A recent breakthrough in the number of variables is the k-G algorithm (Scanagatta et al., 2016), which is able to learn bounded-treewidth models of reasonable accuracy from thousands of variables.

In this paper we present a series of *approximate* techniques for score-based structure learning of Bayesian networks; they include methods for parent set identification and structure optimization, both with bounded and unbounded treewidth. The algorithms scale up to several *thousands* of variables, even in the more challenging case of bounded treewidth. We build a unified presentation based on findings from Scanagatta et al. (2015, 2016). The algorithms for structure optimization that we present are ordering-based search methods; as a novel contribution we propose an approach for effectively sampling the orders, which remarkably improves the performance of both algorithms. Such an approach is general and might be helpful for any ordering-based algorithm.

2 Bayesian Networks

Consider the task of learning the structure of a Bayesian Network from a data set of N instances $\mathcal{D} = \{D_1, ..., D_N\}$. The set of n random variables is $\mathcal{X} = \{X_1, ..., X_n\}$. We assume the variables to be *categorical* (with a finite number of states) and the data set to be *complete*. The goal is to find the highest-scoring DAG $\mathcal{G}$ over nodes $\mathcal{X}$ by defining the set of parents $\Pi_1, ..., \Pi_n$ of each variable. Such a graph induces a joint probability distribution, because of the assumed Markov condition: every variable is conditionally independent of its non-descendant variables given its parent variables.

Different score functions can be used to assess the quality of the DAG; see for example (Liu et al., 2012) for a thorough discussion. In this work we adopt the *Bayesian Information Criterion* (BIC), which is asymptotically proportional to the posterior probability of the DAG. The BIC score is *decomposable*, being constituted by the sum of the scores of each variable and its parent set:

$$\mathrm{BIC}(\mathcal{G}) = \sum_{i=1}^{n} \mathrm{BIC}(X_i, \Pi_i) = \sum_{i=1}^{n} \left(\mathrm{LL}(X_i|\Pi_i) + \mathrm{Pen}(X_i, \Pi_i)\right) ,$$

where $\mathrm{LL}(X_i|\Pi_i)$ denotes the log-likelihood of X_i and its parent set:

$$\mathrm{LL}(X_i|\Pi_i) = \sum\nolimits_{\pi \in \Pi_i,\ x \in X_i} N_{x,\pi} \log \hat{\theta}_{x|\pi} ,$$

while $\mathrm{Pen}(X_i, \Pi_i)$ is the complexity penalization for X_i and its parent set:

$$\mathrm{Pen}(X_i, \Pi_i) = -\frac{\log N}{2}(|X_i| - 1)(|\Pi_i|) .$$

$\hat{\theta}_{x|\pi}$ is the maximum likelihood estimate of the conditional probability $P(X_i = x|\Pi_i = \pi)$; $N_{x,\pi}$ represents the number of times $(X = x \wedge \Pi_i = \pi)$ appears in the data set; $|\cdot|$ indicates the size of the Cartesian product space of the variables given as argument. Thus $|X_i|$ is the number of states of X_i and $|\Pi_i|$ is the product of the number of states of the parents of X_i.

By exploiting decomposability, structure learning can be accomplished in two phases. The first phase is to identify a list of candidate parent sets (called *parent set identification*), which can be done independently for each variable. The second phase is to decide the parent set of each node in order to maximize the score of the resulting DAG (called *structure optimization*). The ultimate goal is to find

$$\mathcal{G}^* \in \operatorname*{argmax}_{\mathcal{G}} \mathrm{BIC}(\mathcal{G}),$$

where we avoided using the symbol for equality because there might be multiple optima. The usual first step to achieve such a goal is the task of finding the *candidate parent sets* for a given variable X_i (a candidate parent set cannot contain itself). It regards the construction of L_i, the cache of possible parent sets Π_i for X_i alongside their scores $\mathrm{BIC}(X_i, \Pi_i)$, which without any restriction has 2^{n-1} possible parent sets, since every subset of $\mathcal{X} \setminus \{X_i\}$ is a candidate. This becomes quickly prohibitive with the increase of n. If we apply a bound d on the number of parents that a variable can have (that is, a limit on the *in-degree* of a node), then the size of

$$L_i = \{\langle \Pi_i, \mathrm{BIC}(X_i, \Pi_i)\rangle \mid s(\Pi_i) \leq d\}$$

reduces from 2^{n-1} to $\Theta(n^d)$, which might still be too large and we might be losing optimality (this is the case if any optimal DAG would have more than d parents for X_i). $s(\cdot)$ represents the actual cardinality of a set (for instance, $s(\{X_i\}) = 1$ and $s(\Pi_i)$ is the number of elements in Π_i).

There is no known manner of avoiding some loss, as this problem is hard itself. The goal is to find the best approximate idea that still keeps in the cache the most promising candidate sets. Pruning based on limiting the number of parents is not enough (nor the most appropriate) if n is large, as we discuss in the next section.

3 Parent set identification

The first objective is to produce a sensible approximation for the cache L_i of each node. The most common approach in the literature is to explore parent sets in sequential order until a certain time-limit is reached: first the empty parent set is included in L_i, then all the parent sets of size one, then all the parent sets of size two, and so on and so forth, up to size d, the *maximum in-degree*. We refer to this approach as *sequential exploration*. Since the number of candidate parent sets increases exponentially with d, sequential exploration implies the adoption of a low d when n increases, if one wants this method to finish in a reasonable amount of time. For instance, $d = 2$ has been used when dealing with more than hundred variables (Bartlett & Cussens, 2017; Cussens et al., 2013). This approach prevents detecting higher-scoring parent sets with larger in-degree, while needing the computation of scores for many low-scoring parent sets of low in-degree.

Pruning rules (de Campos et al., 2009) detect sub-optimal parent sets and allow to discard parts of the search space, avoiding to spend time in computing their scores. Although they do reduce the time required to explore the space of parent sets, they are not effective enough and hence do not allow us to deal with much larger in-degrees, in particular when the number of variables is large. Scanagatta et al. (2015) propose an approximate search of candidate parent sets that explores them without limiting a priori the in-degree d. We describe such an approach in the next section. It is worth mentioning that the methods for structure optimization that we discuss in this paper work with *any decomposable score function*. However, the procedure for efficient exploration of the space of the parent sets of this section exists only for BIC, and it is an open question to devise similar ideas for other score functions.

3.1 Approximate exploration of parent sets

The main idea of Scanagatta et al. (2015) is to quickly identify the most promising parent sets through an approximate scoring function that does not require scanning the data set. Later on, only the scores of the most promising parent sets are computed. The approximate scoring function is called BIC^*. The BIC^* of a parent set $\Pi = \Pi_1 \cup \Pi_2$ constituted by the union of two non-empty and disjoint parent sets Π_1 and Π_2 is:

$$\mathrm{BIC}^*(X, \Pi_1, \Pi_2) = \mathrm{BIC}(X, \Pi_1) + \mathrm{BIC}(X, \Pi_2) + \mathrm{inter}(X, \Pi_1, \Pi_2)\,, \tag{1}$$

that is, the sum of the BIC scores of the two parent sets and of an interaction term, which ensures that the penalty term of $\mathrm{BIC}^*(X, \Pi_1, \Pi_2)$ matches the penalty term of $\mathrm{BIC}(X, \Pi_1 \cup \Pi_2)$. In particular, $\mathrm{inter}(X, \Pi_1, \Pi_2) = \frac{\log N}{2}(|X| - 1)(|\Pi_1| + |\Pi_2| - |\Pi_1||\Pi_2| - 1) - \mathrm{BIC}(X, \varnothing)$. The $\mathrm{BIC}^*(X, \Pi)$ score is equal to the $\mathrm{BIC}(X, \Pi)$ score if the interaction information $ii(X; \Pi_1; \Pi_2)$ is zero (Scanagatta et al., 2015). Yet, this condition is generally false; for this reason, $\mathrm{BIC}^*(X, \Pi)$ is an *approximate* score, but it is efficiently computable. If $\mathrm{BIC}(X, \Pi_1)$ and $\mathrm{BIC}(X, \Pi_2)$ are known, then BIC^* is computed in constant time (with respect to data accesses).

The *independence selection* algorithm (Scanagatta et al., 2015) exploits BIC^* to quickly estimate the score of a large number of parent sets. It is described in Algorithm 1. It adopts two lists: (1) *open*: a list for the parent sets to be explored, ordered by their BIC^* score; (2) *closed*: a list of already explored parent sets, along with their actual BIC score. The algorithm returns the content of the *closed* list, which becomes L_i for each variable X_i. The procedure is repeated for every variable and can be easily parallelized. Independence selection prioritizes the computation of the BIC score of the most promising parent sets (those with highest BIC^*

score) without constraining the in-degree. It consistently increases the scores achieved by different structure optimization algorithms when compared to sequential ordering (Scanagatta et al., 2015), and thus we adopt it in all our experiments.

Algorithm 1 IndependenceSelection(X)

for all $Y \in \mathcal{X} \setminus \{X\}$ **do** ▷ Compute the BIC scores of all single parents
  $\Pi_Y \leftarrow \{Y\}$
  $closed.put(\Pi_Y, BIC(\Pi_Y))$
end for

for all $Y_1 \in \mathcal{X} \setminus \{X\}$ **do** ▷ Compute the BIC* of all bivariate parent sets
  for all $Y_2 \in \mathcal{X} \setminus \{Y_1, X\}$ **do**
    $\Pi \leftarrow \{Y_1, Y_2\}$
    $open.put(\Pi, \text{BIC}^*(\Pi))$
  end for
end for

▷ Explore the space of parent sets, computing at each iteration the BIC score of the parent set with highest BIC*
while $open \neq \varnothing \ \ \& \ \ thereIsTime()$ **do**
  $\Pi \leftarrow popBest(open)$
  $closed.put(\Pi, BIC(\Pi))$
  for all $Y \in \mathcal{X} \setminus \{X \cup \Pi\}$ **do**
    $\Pi_n \leftarrow \Pi \cup Y$
    if $\Pi_n \notin closed \ \ \& \ \ \Pi_n \notin open$ **then**
      $open.put(\Pi_n, BIC^*(\Pi_n))$
    end if
  end for
end while

4 Structure optimization

The objective of structure optimization is to select the parent set of each node from its cache in order to maximize the score of the resulting DAG. At this stage, all caches of scores are assumed to be available. The number of possible Bayesian networks structures increases super-exponentially with the number of variables, so the task is very hard. When restricted to the available caches, there are still $\prod_{i=1}^{n} s(L_i)$ graphs to evaluate if it were to use a brute-force approach.

We take the software Gobnilp (Bartlett & Cussens, 2017; Cussens, 2011) as a benchmark for our evaluations, since it is the state of the art for *exact* structure learning. It is available from `https://www.cs.york.ac.uk/aig/sw/gobnilp/` and is an anytime algorithm. When provided with enough time, it certainly finds the highest-scoring graph, while it provides an approximate solution (provided that a certain amount of time has been given for it to find the first valid graph) whenever it has not yet reached an optimum or has not yet been able to prove it is an optimum. In terms of approximate methods, an effective approach in large domains (thousands of variables) is *Acyclic Selection Ordering-Based Search* (ASOBS) (Scanagatta et al., 2015). It consistently outperforms Gobnilp on data sets containing more than 500 variables, while Gobnilp generally wins on smaller data sets. In this section we describe ASOBS and then we propose a novel variant named ASOBS$_{\text{ENT}}$, which improves on ASOBS by applying a better scheme to sample orders.

4.1 Acyclic Selection Ordering-Based Search

Sampling from the space of orders rather than from the space of structures is certainly appealing since the space of orders is significantly smaller than the space of structures. For any fixed ordering of the n variables, the decomposability of the score enables efficient optimization over all DAGs compatible with the ordering (Cooper & Herskovits, 1992). Moreover, identifying the highest-scoring network *consistent* with a variable order is time-efficient. A network is consistent with the order $\prec$ if $\forall X_i : \forall X \in \Pi_i : X \prec X_i$; we call this condition the *consistency rule*. A network consistent with an order is necessarily acyclic. In order to identify the highest-scoring network given an order, we have to choose independently the best parent set for each node X_i among those containing only variables that are antecedent of X_i in that order. Finding the highest-scoring network given the ordering is thus simple and efficient (can be done in linear time in the size of the caches).

The Ordering-Based Search (OBS) algorithm by Teyssier & Koller (2005) learns the highest-scoring network given an order, according to the consistency rule. Then it greedily explores several neighboring orders by considering swaps between variables that are adjacent in the order. Updating the score of the network after swapping two adjacent variables is efficient thanks to score decomposability. ASOBS performs a similar computation but relaxes the consistency rule in order to retrieve higher-scoring DAGs that are *inconsistent* with the provided order. In particular, it allows arcs from a variable to its successors (*back-arcs*) if they do *not* introduce a directed cycle. Scanagatta et al. (2015) prove that, for a given order, ASOBS achieves equal or higher score than OBS. Moreover, ASOBS empirically outperforms OBS in every analyzed data set (Scanagatta et al., 2015).

We recall that X_j is an *ancestor* of X_i if there is a directed path from X_j to X_i. In this case, we equivalently say that X_i is a *descendant* of X_j. The ASOBS algorithm is as follows:

1. Build a Boolean square matrix `isDescendantOf` that tracks the descendants of each node; thus its entry in position (X_i, X_j) is true if and only if X_i is a descendant of X_j. All entries of `isDescendantOf` are initialized as *false* as we start from an empty structure.
2. For each X_j in the order, with $j = n, \ldots, 1$:
   (a) Select the highest-scoring parent set Π_j that contains no descendants of X_j, as checked through matrix `isDescendantOf`.
   (b) Update matrix `isDescendantOf` to reflect that:
      – Each variable belonging to Π_j is ancestor of X_j;
      – Each ancestor of X_j has as descendants X_j, the descendants of X_j and any other node in the path between the ancestor and X_j.

While the algorithm is quite simple and details about the implementation could be omitted, we would like to point out that a smart representation for the data structure of ancestors/descendants allows us to achieve an overall computational complexity that is asymptotically equal to OBS.

In (Scanagatta et al., 2015), ASOBS is implemented by uniformly sampling the space of orders and repeating the above procedure for each sampled order. In the following we introduce a novel approach, which more effectively samples from the space of orders and might be applied to any structure learning algorithm based on ordering search.

4.2 Entropy-based sampling

We investigate which parent sets are especially important for the score of a DAG, with the aim of improving the sampling strategy and thus to cover better regions of the space of orders. To analyze this problem, we introduce the following *range* statistic:

$$\text{range}\left[\text{BIC}(X_i, \Pi_i)\right] = \text{BIC}(X_i, \Pi_i) - \min_{\Pi_i' \in \mathcal{P}(X_i)} \text{BIC}(X_i, \Pi_i'), \tag{2}$$

where $\mathcal{P}(X_i)$ denotes the set of the feasible parent sets for X_i. Thus the range measures how much the score improves when assigning to X_i the parent set Π_i instead of the lowest-scoring parent set (usually the empty parent set). Intuitively, high-entropy variables should achieve higher differences in score when compared to using no parents, since their likelihood (and thus their score) is low without parents. Thus their score can be largely improved through careful selection of their parent set. To verify this intuition, we compute the range score of every parent set of the most entropic and the least entropic variables of different data sets, listed in Tab. 1.

Results referring to four data sets are given in Fig. 1; note that to understand which variables can most affect the overall score by a more careful selection of their parent sets, we are interested in the highest values of range. High-entropy variables (shown in black, last five boxplots of each graph) yield much higher values of range than low-entropy variables (shown in red, first five boxplots of each graph). For instance, in the data set Reuters-52, high-entropy variables yield ranges that are up to an order of magnitude larger than those of low-entropy variables. Moreover, ranges have a much larger span for high-entropy variables than for low-entropy variables. Again with reference to Reuters-52, the difference between the maximum and the median range is about 500 for high-entropy variables and lower than 25 for low-entropy variables.

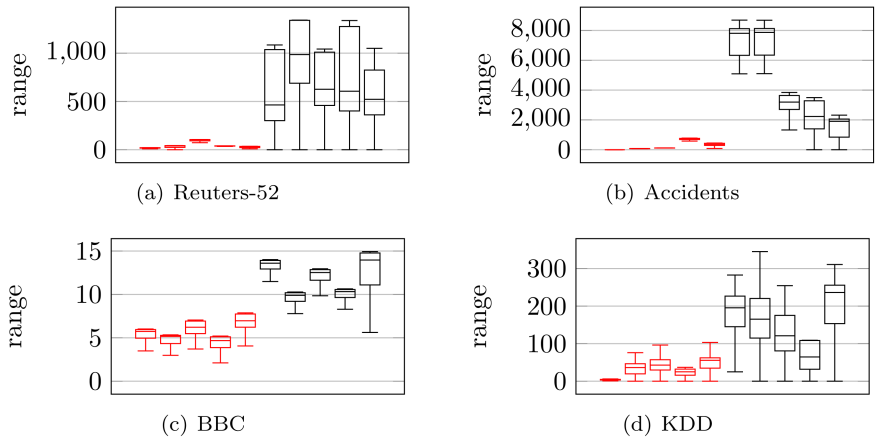


Fig. 1 The red boxplots on the left (first five boxplots of each figure) refer to the five *least* entropic variables of each data set. The black boxplots on the right refer to the five *most* entropic variables of each data set. Each boxplot represents the distribution of the range statistic, across variables belonging to the same data set.

The results about the range statistic suggest that the choice of the parent sets of high-entropy variables has the largest impact on the eventual score of the DAG. We thus modify

ASOBS in order to better assign the parent sets of the high-entropy variables. We do so by putting them at the end of the order provided to ASOBS, so that they are offered plenty of different parent sets to choose from. In particular, we sample the variables of the order in a fashion that is *proportional* to their entropy. Suppose we have already chosen the first p variables of the ordering (p= 0,1, . . . , n-1). Without loss of generality, denote a still unselected variable as X_j ($j = p + 1, \ldots, n$). The $(p + 1)$-th position of the ordering is chosen by assigning to each of the $n - p$ variables not already selected a weight proportional to their entropy:

$$w_j = \frac{H(X_j)}{\sum_{i=p+1}^{n} H(X_i)},$$

where $H(\cdot)$ denotes the empirical entropy. The next variable of the order is sampled from a discrete distribution, whose probabilities are constituted by the above weights w_j. We call $\text{ASOBS}_{\text{ENT}}$ this variant of ASOBS, since it is equipped with the entropy-based sampling.

4.3 Experiments

We compare $\text{ASOBS}_{\text{ENT}}$, ASOBS and Gobnilp using the twenty data sets of Tab. 1: such a collection of data sets has been previously used for instance by Rooshenas & Lowd (2014) and by others referenced therein. The data sets are available for instance from `https://github.com/arranger1044/awesome-spn#dataset`. They contain between 16 and 1556 binary-valued variables. Each data set is split in three subsets, producing a total of 60 structure learning experiments. Moreover, in order to test the algorithms in domains with a large number of variables, we generate further 15 synthetic data sets as follows. Using the BNgenerator package[1], we generate five networks containing 2,000 variables, five networks containing 4,000 variables and five networks containing 10,000 variables. From each generated network we then sample a data set of N=5,000 instances.

In each experiment we run Gobnilp, ASOBS and $\text{ASOBS}_{\text{ENT}}$ for one hour, on the same computer and providing them with the same caches of candidate parent sets, which were pre-computed using the BIC^* approach of Sect. 3.1. When computing the caches of parent sets, we allowed one minute per variable with no maximum in-degree. The detailed results of all the experiments (scores obtained by each method in each data set) of this paper are available at: `http://ipg.idsia.ch/papers/scanagatta2017b/supplementary.pdf`.

We also considered further competitors for structure learning of Bayesian networks. For instance the package *urlearning*[2] implements search methods based on (Yuan & Malone, 2013). Yet, it could not learn from data set containing more than 100 variables; similar limits are indeed acknowledged also by the authors. We also tried *WinMine*[3], but it failed to provide results within an hour when dealing with more than 500 variables. On smaller data sets, it was anyway outperformed by both Gobnilp and ASOBS-ENT (notice that we ran these experiments using the BDeu score, as WinMine does not support the BIC score). We report the results in the supplementary material.

We analyze the results in Tab. 2 by separating small data sets ($n \leq 200$), large data sets ($200 < n < 2000$) and very large data sets ($n \geq 2000$). In each experiment we measure the difference in terms of BIC scores between $\text{ASOBS}_{\text{ENT}}$ and ASOBS, and between $\text{ASOBS}_{\text{ENT}}$ and Gobnilp. We denote this difference by ΔBIC.

[1] `http://sites.poli.usp.br/pmr/ltd/Software/BNGenerator/`

[2] `www.urlearning.org/`

[3] `www.microsoft.com/en-us/research/project/winmine-toolkit/`

Real data sets							
	n		n		n		n
Nltcs	16	Jester	100	DNA	180	WebKB	839
Msnbc	17	Netflix	100	Kosarek	190	Reuters-52	889
Kdd	64	Accidents	111	MSWeb	294	C20NG	910
Plants	69	Retail	135	Book	500	BBC	1058
Audio	100	Pumsb-star	163	EachMovie	500	Ad	1556

Synthetic data sets					
	n		n		n
rand2000-0	2000	rand4000-0	4000	rand10000-0	10000
rand2000-1	2000	rand4000-1	4000	rand10000-1	10000
rand2000-2	2000	rand4000-2	4000	rand10000-2	10000
rand2000-3	2000	rand4000-3	4000	rand10000-3	10000
rand2000-4	2000	rand4000-4	4000	rand10000-4	10000

Table 1 Data sets sorted according to the number n of variables.

A positive ΔBIC provides evidence in favor of the higher-scoring model. The ΔBIC values can be interpreted as follows (Raftery, 1995):

- ΔBIC $>$10: extremely positive evidence;
- 6 $<\Delta$BIC $<$10: strongly positive evidence;
- 2 $<\Delta$BIC $<$6: positive evidence;
- ΔBIC $<$2: neutral evidence.

We perform the sign test considering one method as winning over the other when there is a ΔBIC of at least 2 in its favor, and treating as ties the cases in which $|\Delta\text{BIC}| <2$. The statistically significant differences (p-value $<$0.01) are boldfaced in Tab. 2. The supplementary material shows in detail the scores obtained by each solver on each data set.

	$n \leq 200$		$200 < n < 2000$		$n \geq 2000$	
$\text{ASOBS}_{\text{ENT}}$ vs	ASOBS	Gobnilp	ASOBS	Gobnilp	ASOBS	Gobnilp
$\{\Delta\text{BIC}\}$						
extremely positive	21	4	**21**	**23**	**15**	**-**
strongly positive	0	0	1	1	0	-
positive	0	0	1	0	0	-
neutral	0	0	0	0	0	-
negative	0	0	0	0	0	-
strongly negative	0	0	0	0	0	-
very negative	15	**32**	1	0	0	-
p-value	.58	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	

Table 2 Comparison of $\text{ASOBS}_{\text{ENT}}$ against ASOBS and Gobnilp. The table shows the number of occurrences of each scenario and p-values of a sign test.

As for the comparison of $\text{ASOBS}_{\text{ENT}}$ and ASOBS, $\text{ASOBS}_{\text{ENT}}$ performs better in all three categories. Its advantage is more prominent in large and very large data sets: in almost every data set it improves the BIC score by more than ten points from ASOBS's values. The number of victories in favor of $\text{ASOBS}_{\text{ENT}}$ is significant both in large and very large data sets.

Regarding the comparison of $\text{ASOBS}_{\text{ENT}}$ and Gobnilp, Gobnilp is significantly better than $\text{ASOBS}_{\text{ENT}}$ in small data sets ($n \leq 200$), while in large data sets the situation is reversed: $\text{ASOBS}_{\text{ENT}}$ outperforms Gobnilp in *all* large data sets. When dealing with very large data sets ($n \geq 2000$), Gobnilp failed to provide a solution after one hour of computation.

4.4 Discussion

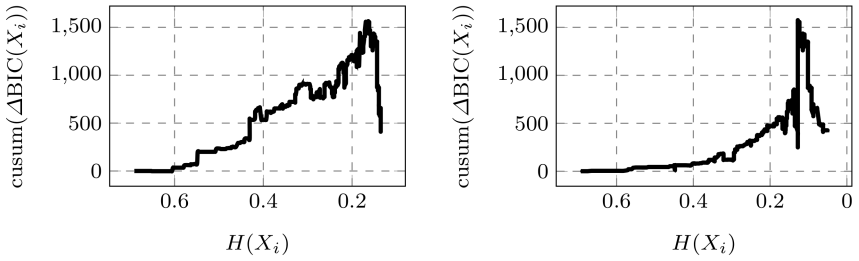
We further investigate the difference between $\text{ASOBS}_{\text{ENT}}$ and ASOBS. We do this by tracking the difference between BIC scores achieved by $\text{ASOBS}_{\text{ENT}}$ and ASOBS on each individual variable X_i:

$$\Delta\text{BIC}(X_i, \Pi_i) = \text{BIC}_{\text{ASOBS}_{\text{ENT}}}(X_i, \Pi_i) - \text{BIC}_{\text{ASOBS}}(X_i, \Pi_i)\,. \tag{3}$$

We expect $\Delta\text{BIC}(X_i, \Pi_i)$ to be positive for high-entropy variables and negative for low-entropy variables. For each variable X_i, we then compute the following statistic:

$$\text{cusum}(\Delta\text{BIC}(X_i)) = \sum_{X_j : H(X_j) \geq H(X_i)} \Delta\text{BIC}(X_j, \Pi_j)\,.$$

This statistic measures the advantage of $\text{ASOBS}_{\text{ENT}}$ over ASOBS on the variables that are more than or equally entropic to X_i itself. In Fig. 2, we plot this statistic as a function of $H(X_i)$; it shows that $\text{ASOBS}_{\text{ENT}}$ builds a large advantage (largely positive cusum) on high-entropy and medium-entropy variables. This advantage is eventually lost as we reach the least entropic variables, but it remains largely positive in the end. Because of that, $\text{ASOBS}_{\text{ENT}}$ yields higher-scoring networks than ASOBS.



(a) BBC (b) WebKB

Fig. 2 Cusum statistic as a function of $H(X_i)$ for different data sets. The value of the cusum at the end of the curve (i.e., in relation to least entropic variables) equals the difference between the DAG identified by $\text{ASOBS}_{\text{ENT}}$ and ASOBS.

5 Treewidth-bounded structure optimization

The structure learning approaches discussed so far (including $\text{ASOBS}_{\text{ENT}}$) do not bound the treewidth of the DAG. They are therefore a good choice when one wants to learn an expressive model to understand how variables relate to each other. However, it may be important to learn Bayesian networks with bounded treewidth when one needs to do efficient inferences with the model. We discuss this problem in this section. Before we present some ideas and algorithms, we need some background material.

5.1 Treewidth and k-trees

We denote an undirected graph as $U = (V, E)$, where V is the vertex set and E is the edge set. A *tree decomposition* of U is a pair $(\mathcal{C}, \mathcal{T})$ where $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ is a collection of subsets of V and $\mathcal{T}$ is a tree over $\mathcal{C}$, so that:

- $V = \cup_{i=1}^{m} C_i$;
- for every edge that connects the vertices v_1 and v_2, there is a subset C_i that contains both v_1 and v_2;
- for all i, j, k in $\{1, 2, ..m\}$, if C_j is in the path between C_i and C_k in $\mathcal{T}$, then $C_i \cap C_k \subseteq C_j$.

The width of a tree decomposition is $\max_i s(C_i) - 1$, where $s(C_i)$ is the number of vertices in C_i. The treewidth of U is the minimum width among all possible tree decompositions of U. Treewidth can be equivalently defined in terms of triangulations of U. A triangulated graph is an undirected graph in which every cycle of length greater than three contains a chord. The treewidth of a triangulated graph is the size of its maximal clique minus one. The treewidth of U is the minimum treewidth over all the possible triangulations of U.

The treewidth of a DAG is characterized with respect to all possible triangulations of its *moral graph*. The moral graph of a DAG is an undirected graph that includes an edge $(i - j)$ for every arc $(i \to j)$ in the DAG and an edge $(p - q)$ for every pair of edges $(p \to i)$, $(q \to i)$ in the DAG. The treewidth of a DAG is the minimum treewidth over all the possible triangulations of its moral graph. Thus the maximal clique of any moralized triangulation of G is an upper bound on the treewidth of the model.

A complete graph is a *clique*. A clique containing $k + 1$ nodes is a $(k + 1)$-clique; it has treewidth k. A clique is *maximal* if it is not a subset of a larger clique. A $(k + 1)$-clique thus contains multiple non-maximal k-cliques. An undirected graph is a k-tree if it has treewidth k and the addition of any edge increases its treewidth. A k-tree can be inductively built as follows (Patil, 1986). We start with a $(k + 1)$-clique. Then we connect a new node to a k-clique of the original graph, obtaining an updated graph. Other nodes can be added one at a time following the same procedure. A *partial k-tree* is a subgraph of a k-tree; as such, it has treewidth bounded by k. An example of the iterative construction of a k-tree ($k = 2$) is given in Fig. 3. We start with the clique over the variables A, B, C. Then we link D to the 2-clique $\{A, B\}$. Then we link E to the 2-clique $\{C, A\}$, and F to the 2-clique $\{C, E\}$. Fig. 3 also shows in blue the tree decomposition at each iteration. The nodes of the tree have size three; thus the treewidth is two.

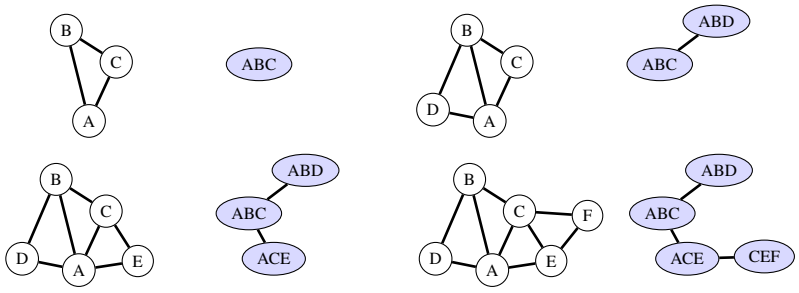


Fig. 3 Iterative construction of a k-tree (white nodes) with treewidth k=2. The corresponding tree decomposition is shown alongside (blue nodes).

5.2 Learning Bayesian networks with bounded treewidth

Learning Bayesian networks with bounded treewidth is very challenging, in particular because treewidth is a *global* property of the DAG and determining it is already NP-hard. A pioneering approach, polynomial in both the number of variables and the treewidth bound, has been proposed by Elidan & Gould (2008). It incrementally builds the network; at each arc addition it provides an upper-bound on the treewidth of the learned structure. The limit of this approach is that, as the number of variables increases, the gap between the bound and the actual treewidth becomes large, leading to sparse networks.

An *exact* method has been proposed by Korhonen & Parviainen (2013), which finds the highest-scoring network with the desired treewidth. However, its complexity increases exponentially with the number of variables n and it has been applied in experiments with fewer than 16 variables. Parviainen et al. (2014) adopted an anytime integer linear programming (ILP) approach, called TWILP. If the algorithm is given enough time, it finds the highest-scoring network with bounded treewidth. Otherwise, it returns a sub-optimal DAG with bounded treewidth. Such an ILP problem has an exponential number of constraints in the number of variables; this limits its scalability, even if the constraints can be generated online. Typically it cannot handle data sets containing more than 100 variables. Berg et al. (2014) cast the problem of structure learning with bounded treewidth as a problem of weighted partial maximum satisfiability. They solved the problem exactly through a MaxSAT solver and performed experiments with at most 30 variables. Nie et al. (2014) proposed a more efficient anytime ILP approach with a polynomial number of constraints in the number of variables. Yet, they reported that the quality of the solutions quickly degrades as the number of variables exceeds a few dozens, and that no satisfactory solutions are found with data sets containing more than 50 variables.

To scale to larger domains, one has to resort to approximate approaches. The S2 algorithm (Nie et al., 2015) samples uniformly the space of k-trees; the sampled k-trees are assessed via a heuristic scoring function (called *informative score*). The DAG is then recovered as a sub-graph of the k-tree with highest informative score. Nie et al. (2016) further refined this idea, obtaining via A* the k-tree that is guaranteed to maximize the informative score. In (Scanagatta et al., 2016) the authors presented the k-G algorithm. It consistently yields higher-scoring networks than S2 for different tested treewidths. The advantage becomes especially important in the largest data sets that contain thousands of variables. An algorithm named k-MAX that follows a similar idea was recently presented in (Scanagatta et al., 2018). We describe k-G in the next section, before we present an improvement of it.

5.3 The k-G algorithm

Like ASOBS, k-G is based on sampling orders. In particular, it samples an order and then it *greedily* (whence the G letter in the name k-G) searches for the highest-scoring DAG with bounded treewidth consistent with the order. The DAG is built iteratively; one variable is added at each iteration while keeping the moral graph of the DAG as a subgraph of a k-tree, which guarantees that the final DAG will have treewidth bounded by k. The algorithm is discussed in the following.

The algorithm starts by choosing an initial k-tree. Such an initial k-tree $\mathcal{K}_{k+1}$ consists of the complete clique over the first $k+1$ variables in the order. Then the initial DAG $\mathcal{G}_{k+1}$ is learned over the same $k+1$ variables. Since $k+1$ often regards a tractable number of variables, we can exactly learn $\mathcal{G}_{k+1}$ adopting a solver such as Gobnilp. Because the moral

graph of $\mathcal{G}_{k+1}$ is a subgraph of $\mathcal{K}_{k+1}$, we have that $\mathcal{G}_{k+1}$ has bounded treewidth. We then iteratively add each remaining variable, according to the order. Consider the next variable in the order, $X_{\prec i}$, where $i \in \{k+2, ..., n\}$. Let us denote by $\mathcal{G}_{i-1}$ and $\mathcal{K}_{i-1}$ the DAG and the k-tree that must be updated after adding $X_{\prec i}$. We add $X_{\prec i}$ to $\mathcal{G}_{i-1}$, constraining its parent set $\Pi_{\prec i}$ to be a (subset of a) k-clique in $\mathcal{K}_{i-1}$. This yields the updated DAG $\mathcal{G}_i$. We then update the k-tree, connecting $X_{\prec i}$ to such a k-clique. This yields the k-tree $\mathcal{K}_i$; it contains an additional $k+1$-clique compared to $\mathcal{K}_{i-1}$. By construction, $\mathcal{K}_i$ is also a k-tree. Because the moral graph of $\mathcal{G}_i$ cannot have arcs outside this $(k+1)$-clique, it is a subgraph of $\mathcal{K}_i$.

In order to choose the parent set of the variable being added to the graph, k-G chooses the highest-scoring parent set among the feasible ones. We denote the set of existing k-cliques in $\mathcal{K}$ as $\mathcal{K}_C$. Thus k-G chooses as parent set for $X_{\prec i}$ the highest-scoring parent set that is a subset of an existing k-clique in $\mathcal{K}_C$.

$$\Pi_{X_{\prec i}} \in \underset{\pi \subset C, C \in \mathcal{K}_C}{\operatorname{argmax}} \operatorname{BIC}(X_{\prec i}, \pi_{\prec i}) \,.$$

Thus k-G finds a *locally optimal* DAG consistent with a given order and whose treewidth is bounded by k.

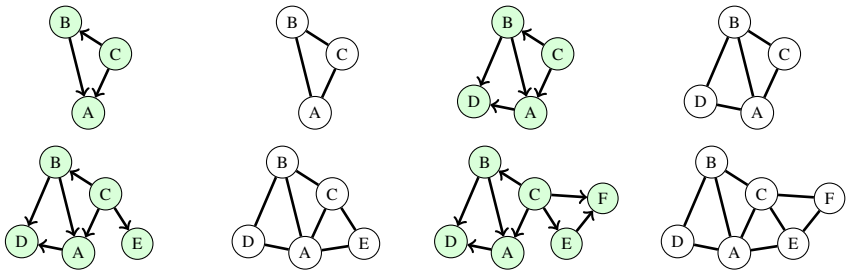


Fig. 4 Example of a treewidth-bounded DAG (k=2) being built iteratively. On the left, the DAG at each step (green nodes) is presented, and on the right the resulting k-tree (white nodes). We start with the variables $\{A, B, C\}$ and we add the remaining ones one at a time (D, E and F).

5.4 Sampling orders for k-G

The insights of Sect. 4.2 about high-entropy and low-entropy variables can be applied also to k-G. We thus modify k-G by applying the entropy-based approach for sampling the orders, as discussed in Sect. 4.2. We call this new approach k-G$_{\mathrm{ENT}}$ as an extension of the original algorithm k-G.

We compare k-G and k-G$_{\mathrm{ENT}}$ on small, large and very large data sets already introduced in Sect. 4. We provide each solver with the same cache of candidate parent sets, pre-computed using BIC*, allowing one minute per variable and no maximum in-degree. Each solver is executed for one hour on the same computer and we track the BIC score obtained by the two algorithms for treewidths $k \in \{2, 4, 5, 6, 8\}$. We summarize the results by again separating small data sets ($n \leq 200$, Tab. 3), large data sets ($200 < n < 2000$, Tab. 4) and very large data sets ($n \geq 2000$, Tab. 4).

k-G$_{\mathrm{ENT}}$ performs better than k-G even in small data sets ($n \leq 200$). In spite of that, the sign test does not claim significance when analyzing the number of wins and losses of

the two methods on such data sets. On the other hand, k-G_{ENT} outperforms k-G in almost every large and very large data set; the analysis of the number of wins and losses shows significance for each tested treewidth. In most cases, the obtained $\Delta\mathrm{BIC}$ is larger than 10, providing very strong evidence in favor of the model learned by k-G_{ENT}. The difference is significant (sign-test, p<0.01) for every tested treewidth.

$n < 200$	Treewidth			
k-G_{ENT} vs k-G	2	4	6	8
ΔBIC				
Very strong (>10)	24	19	23	22
Very negative (<-10)	12	17	13	14
p-value	0.06	0.87	0.13	0.24

Table 3 k-G_{ENT} often beats k-G in small data sets, but the number of victories is not statistically significant.

$200 < n < 2000$	Treewidth			
k-G_{ENT} vs k-G	2	4	6	8
ΔBIC				
Very strong (>10)	**24**	**22**	**22**	**24**
Strongly positive (between 6 and 10)	0	1	1	0
Strongly negative (between -10 and -6)	0	0	0	0
Very negative (<-10)	0	1	1	0
p-value	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$

$n > 2000$	Treewidth			
k-G_{ENT} vs k-G	2	4	6	8
ΔBIC				
Very strong (>10)	**12**	**14**	**14**	**14**
Very negative (<-10)	3	1	1	1
p-value	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$

Table 4 k-G_{ENT} consistently achieves a larger BIC score than k-G in large and very large data sets. In both settings, the amount of victories is statistically significant.

Scanagatta et al. (2016) shows that k-G outperforms S2; for the sake of completeness, we have compared k-G_{ENT} to S2. It further increases the advantage achieved by k-G over S2. Out of the 75 data sets used in this experiments, of which 36 have are small, 24 are large, and 15 are very large, k-G_{ENT} *always* yields a higher score than S2. In the great majority of cases the improvement is larger than 10; smaller improvements are found only in some data sets with less than 70 variables.

5.5 Inference on real data sets

One of the main reasons to learn Bayesian networks of bounded treewidth is to ensure that their use for inferences later on can be performed exactly and efficiently. In this section we compare the performance (in terms of inferential results between models that were learned with bounded and unbounded treewidths). We consider the 20 real data sets of Tab. 1. The inference task that we take on is the computation of the probability of evidence $P(e)$ of *five* randomly selected variables, which we set to random states.

When dealing with large real data sets, the actual value of $P(e)$ may be unknown (since it is a computationally challenging problem). We thus (approximately) assume that the ground-truth is the highest-scoring network of unbounded treewidth that is available to us; that is, we assume that the true networks is either the one obtained by ASOBS$_{\mathrm{ENT}}$ or by Gobnilp (whichever achieves the best score). We then compute $P(e)$ by performing exact inference on such a network, using the algorithm Iterative Join Graph Propagation (Mateescu et al., 2010) and running it until convergence. This software is available from `http://www.hlt.utdallas.edu/~vgogate/ijgp.html`.

We assess the difference between $P(e)$ computed using the assumed ground-truth and using the network with bounded treewidth (k=2,4 6,8) learned by k-G$_{\mathrm{ENT}}$. For each data set, we run 100 queries and we measure the mean absolute error (mae) of the resulting inference for each bounded-treewidth model:

$$\mathrm{mae} = \frac{1}{q} \sum_i |P_i(e) - \hat{P}_i(e)| \,,$$

where q denotes the total number of queries, $P_i(e)$ and $\hat{P}_i(e)$ are the probability of evidence on the i-th query computed by respectively the model assumed as ground-truth and the bounded-treewidth model. We show in Fig. 5 how mae varies with the treewidth. Overall, the difference in mae goes down as the treewidth increases, but it almost vanishes at $k = 6$ to $k = 8$, suggesting that a treewidth larger than 8 should be rarely necessary.

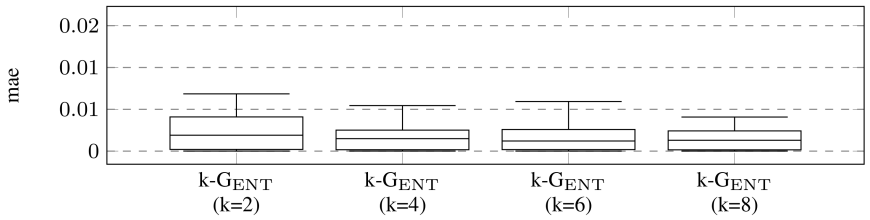


Fig. 5 Distribution of mean absolute errors, for bounded-treewidth models with different treewidths. Each boxplot represents mae measures taken on 20 data sets.

Besides mae, we analyze the time required by the inferences. We report summary results (obtained by averaging over all data sets) in Fig. 6, including also the unbounded model in the comparison. The most striking result is that the bounded-treewidth models are at least *one order of magnitude* faster than the unbounded-treewidth ones, even with treewidth as large as 8. Such large differences are partially due to the fact that the query involves multiple variables. Smaller differences are observed when computing marginals for these real data sets. Tab. 5 reports mean inference time and mae for models with different treewidths. Yet,

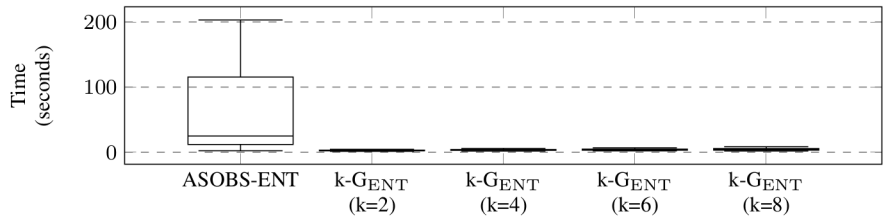


Fig. 6 Distribution of computational times on the 20 data sets in Tab. 1 for each model. On each of the 20 data sets we record the *mean* running time of 100 inferences; thus eventually we represent such 20 means for each model with a boxplot.

we show that orders of magnitude of difference in the running time are observed also when computing marginals, when when deal with domains containing thousands of variables, as we study in the next section.

asobs		**k-2**		**k-4**		**k-6**		**k-8**
time	mae	time	mae	time	mae	time	mae	time
64.2	0.0030	2.66	0.0018	3.6	0.0018	4.1	0.0016	4.52

Table 5 Mean inference results on 20 data sets, running 100 queries for each data set. Times are expressed in *seconds*.

5.6 Inference on synthetic data sets

Now we take on inferential tasks over domains where the true networks is known, so we have access to the true probability of evidence. In this way, we compare inferential results obtained from the networks learned by ASOBS$_{\mathrm{ENT}}$ and by k-G$_{ENT}$, using different treewidths ($k = 2, 4, 6, 8$). We consider the 15 very large synthetic data sets ($n \geq 2000$) of Tab. 1.

Unbounded-treewidth models containing this amount of variables pose serious challenges for inference. Even *marginals* cannot be computed exactly. In several cases we have had no convergence of the inference after 30 minutes of computation. We thus resorted to approximate inference. Considering that inference with bounded-treewidth models take consistently less than 0.1 seconds (recall that we are computing marginals), we allow one minute (that is, a time superior by *two orders of magnitude*) of approximate inference for the queries applied to the true unbounded-treewidth models. In these experiments, we do not compute the probability of joint observations, which would even been even more demanding. For each network, we perform inference regarding the *marginal* probability of 100 different variables. We select the *leaves* of the network as variables to be queried, as this requires marginalizing out the largest number of variables. This setting should be especially challenging for the accuracy of bounded-treewidth models, which have limited expressiveness. If some intermediate potentials are poorly estimated, errors would propagate. In case the network contains less than 100 leaves, we run the remaining queries on randomly chosen parents of the leaves. We again perform the inferences using Iterative Join Graph Propagation (Mateescu et al., 2010).

Let us denote the true marginal of X by $P(X)$ and the marginal estimated by a model by $\hat{P}(X)$. Let us denote by $\mathcal{Q}$ the set of 100 variables selected for as queries. The mean absolute error (mae) of the marginal query is then:

$$\mathrm{mae} = \frac{1}{s(\mathcal{Q})} \sum_{X \in \mathcal{Q}} \frac{1}{|X|} \sum_{x \in X} |P(x) - \hat{P}(x)| \ . \tag{4}$$

In order to compare the bounded-treewidth models against the unbounded models learned with $\mathrm{ASOBS}_{\mathrm{ENT}}$, we divide mae of each bounded-treewidth model by the mae obtained on the same data set by the model learned with $\mathrm{ASOBS}_{\mathrm{ENT}}$. We call this measure *relative* mae. We show in Fig. 7 boxplots of relative mae for each learned model (with $k = 2, 4, 6, 8$). Remarkably, the median of the relative mae is close to one already for a low treewidth bound of 2, and becomes slightly lower than one for larger treewidths, indicating a slightly *better* accuracy for the treewidth-bounded models. One might wonder why the treewidth-unbounded models do not yield more accurate inferences than the treewidth-bounded ones, which have lower score. A first conjecture is that we are considering marginal queries, while treewidth-unbounded models might have an advantage in joint queries involving many variables. Another conjecture is that their accuracy might be deteriorated by approximate inference. We leave this type of investigation for future experiments.

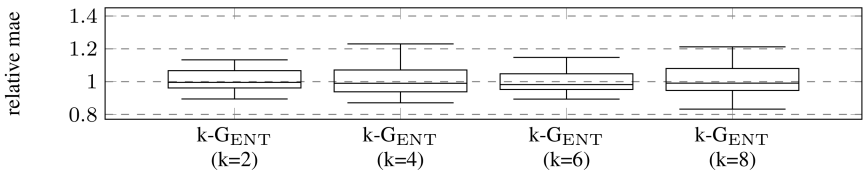


Fig. 7 Relative mae between the bounded-treewidth models and the unbounded-treewidth models learned by $\mathrm{ASOBS}_{\mathrm{ENT}}$. Each model performs 100 marginal inferences in each of the 15 very large data sets. We average the results referring to the same data set, obtaining 15 observations for each model. The boxplots visualize such observations.

Hence, besides delivering huge computational savings, bounded-treewidth models are competitive with unbounded-treewidth models as for the accuracy of the inference. This corroborates with findings of Elidan & Gould (2008), who pointed out that bounding the treewidth prevents selecting overly complicated structures of dependences, thereby reducing the chance of overfitting.

6 Conclusions

We have presented a set of approximate algorithms for structure learning of Bayesian networks. They include parent set identification, structure optimization and structure optimization under bounded treewidth. Taken together they allow for a remarkable improvement regarding the Bayesian network structure learning task for domains with thousands of variables, both for bounded and unbounded treewidth learning.

We foresee two main directions for future work. From the methodological viewpoint, it would be interesting to extend the procedure of BIC^* for the preparation of the cache of the parents also to other scoring function for Bayesian networks, such as the *Bayesian Dirichlet*

equivalent uniform (BDeu) score. From the empirical viewpoint, it would be interesting to compare bounded-treewidth Bayesian networks against other probabilistic models that allow tractable inference and scale to thousands of variables, such as sum-product networks, in terms of their computational performance and accuracy of inferences.

Acknowledgments

Work partially supported by the Swiss NSF grants nos. 200021_146606 / 1 and IZKSZ2_162188.

References

Bartlett, M. and Cussens, J. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

Berg, J., Järvisalo, M., and Malone, B. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pp. 86–95, 2014.

Bodlaender, H. L., Koster, A. M. C. A., van den Eijkhof, F., and van der Gaag, L. C. Pre-processing for triangulation of probabilistic networks. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pp. 32–39, 2001.

Chickering, D. M., Heckerman, D., and Meek, C. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2014.

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

Cussens, J. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 153–160, 2011.

Cussens, J., Malone, B., and Yuan, C. IJCAI 2013 tutorial on optimal algorithms for learning Bayesian networks (https://sites.google.com/site/ijcai2013bns/slides), 2013.

Cussens, J., Järvisalo, M., Korhonen, J. H., and Bartlett, M. Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *Journal of Artificial Intelligence Research*, 58:185–229, 2017.

Darwiche, A. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

de Campos, C. P. and Ji, Q. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

de Campos, C. P., Zeng, Z., and Ji, Q. Structure learning of Bayesian networks using constraints. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 113–120, 2009.

Elidan, G. and Gould, S. Learning bounded treewidth Bayesian networks. *Journal of Machine Learning Research*, 9:2699–2731, 2008.

Jaakkola, T., Sontag, D., Globerson, A., and Meila, M. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 358–365, 2010.

Koivisto, M. and Sood, K. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

Korhonen, J. and Parviainen, P. Exact learning of bounded treewidth Bayesian networks. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pp. 370–378, 2013.

Liu, Z., Malone, B., and Yuan, C. Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics*, 13(15):1–16, 2012.

Lowd, D. and Domingos, P. Learning arithmetic circuits. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pp. 383–392, 2008.

Mateescu, R., Kask, K., Gogate, V., and Dechter, R. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research*, 37:279–328, 2010.

Nie, S., Mauá, D. D., de Campos, C. P., and Ji, Q. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems 27*, pp. 2285–2293, 2014.

Nie, S., de Campos, C. P., and Ji, Q. Learning bounded treewidth Bayesian networks via sampling. In *Proceedings of the 13th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 387–396, 2015.

Nie, S., de Campos, C. P., and Ji, Q. Learning Bayesian networks with bounded treewidth via guided search. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 3294–3300, 2016.

Parviainen, P., Farahani, H. S., and Lagergren, J. Learning bounded treewidth Bayesian networks using integer linear programming. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pp. 751–759, 2014.

Patil, H. P. On the structure of k-trees. *Journal of Combinatorics, Information and System Sciences*, pp. 57–64, 1986.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 689–690, 2011.

Raftery, A. E. Bayesian model selection in social research. *Sociological Methodology*, 25: 111–164, 1995.

Rooshenas, A. and Lowd, D. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 710–718, 2014.

Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 28*, pp. 1855–1863, 2015.

Scanagatta, M., Corani, G., de Campos, C. P., and Zaffalon, M. Learning treewidth-bounded Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 29*, pp. 1462–1470, 2016.

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

Silander, T. and Myllymaki, P. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pp. 445–452, 2006.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pp. 584–590, 2005.

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

Yuan, C. and Malone, B. An improved admissible heuristic for learning optimal Bayesian networks. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pp. 924–933, 2012.