On Pruning with the MDL Score

Eunice Yuh-Jie Chen, Adnan Darwiche, Arthur Choi
Computer Science Department, University of California, Los Angeles

Abstract

The space of Bayesian network structures is prohibitively large and hence numerous techniques have been developed to prune this search space, but without eliminating the optimal structure. Such techniques are critical for structure learning to scale to larger datasets with more variables. Prior works exploited properties of the MDL score to prune away large regions of the search space that can be safely ignored by optimal structure learning algorithms. In this paper, we propose new techniques for pruning regions of the search space that can be safely ignored by algorithms that enumerate the k-best Bayesian network structures. Empirically, these techniques allow a state-of-the-art structure enumeration algorithm to scale to datasets with significantly more variables.

Keywords: Bayesian networks, structure learning

1. Introduction

Learning the structure of a Bayesian network is a fundamental problem in artificial intelligence and machine learning. In particular, we seek a structure, a directed acyclic graph (DAG), that best explains a given dataset (Darwiche, 2009; Koller & Friedman, 2009; Murphy, 2012). In practice, learning a single optimal DAG may not be sufficient, especially when the dataset has few examples and is otherwise noisy. Thus, we are interested in discovering other likely DAGs, and not just the best one.

Recently, a number of algorithms have been proposed to *enumerate* the k-most likely DAGs from a given dataset (Tian et al., 2010; Cussens et al., 2013;

Chen & Tian, 2014; Chen et al., 2015, 2016). For example, using dynamic programming, we can enumerate the 100-best networks for real-world datasets with 17 variables (Tian et al., 2010). Using heuristic search methods, we can enumerate the $1,000$-best networks for real-world datasets with 23 variables, which is the current state-of-the-art (Chen et al., 2015). In this paper, we show how to extend the reach of such systems further, allowing us to enumerate structures for datasets with 29 variables. Each of these advances is quite significant, when we consider how quickly the search space grows, as we increase the number of variables.[1]

More specifically, we propose techniques that can greatly reduce the search space of Bayesian network structures, by safely eliminating regions of the search space that do not contain any of the k-most likely DAGs. By exploiting properties of the popular MDL score for Bayesian networks, we identify an upper bound on the number of parents that a node can have, in any of the k-best structures. Any structure enumeration algorithm that can incorporate such a bound (including all of the aforementioned approaches) can benefit from the techniques that we propose. In fact, our bounds generalize those proposed for the problem of learning a single optimal structure (Suzuki, 1996; Tian, 2000; De Campos & Ji, 2011). Such bounds are broadly used in the literature, and the scalability of modern structure learning algorithms depend critically on such bounds.

This paper is organized as follows. In Section 2, we review score-based structure learning and the MDL score. In Section 3, we propose our techniques for pruning the search space for the purposes of enumerating the k-best Bayesian network structures. We evaluate our approach empirically in Section 4, and conclude in Section 5. Proofs are provided in the Appendix.

[1] For n variables, there are $O(n! \cdot 2^{\binom{n}{2}})$ BN structures. More precisely, for $n = 17, 23$ and 29, there are $6.27 \cdot 10^{52}$, $6.97 \cdot 10^{94}$ and $2.51 \cdot 10^{148}$ structures, respectively; for more on counting Bayesian network structures (and labeled DAGs), see `https://oeis.org/A003024`.

2. Technical Preliminaries and Related Work

In this section, we first review the score-based structure learning of Bayesian networks, and the problem of enumerating the k-best structures. We then review the MDL score, and prior works that have exploited the MDL score to prune the search space of Bayesian network structures.

We use upper case letters (X) to denote variables and lower case letters (x) to denote their values. Variable sets are denoted by bold-face upper case letters ($\mathbf{X}$) and their instantiations by bold-face lower case letters ($\mathbf{x}$). We use $|X|$ to denote the number of values of a discrete variable X, and $|\mathbf{X}|$ to denote the number of variables in a set $\mathbf{X}$. Generally, we will use X to denote a variable in a Bayesian network and $\mathbf{U}$ to denote its parents. We refer to a variable X and its parents $\mathbf{U}$ as a family, which we denote by $X\mathbf{U}$.

2.1. Score-Based Structure Learning

Score-based approaches for learning the structure of a Bayesian network are based on searching for a DAG that minimizes a given scoring metric, which generally rates the quality of a DAG based (in part) on how well a given structure fits a given dataset $\mathcal{D}$ (which is typically complete). Structure scores often decompose into a sum of local scores, over the families $X\mathbf{U}$ of the DAG:

$$\mathsf{score}(G \mid \mathcal{D}) = \sum_{X\mathbf{U}} \mathsf{score}(X\mathbf{U} \mid \mathcal{D}). \tag{1}$$

For example, MDL and BDeu scores are decomposable (note that we negate such scores as needed to obtain minimization problems). For more on score-based structure learning, see, e.g., (Darwiche, 2009; Koller & Friedman, 2009; Murphy, 2012).

2.2. Learning the k-Best Structures

In the problem of enumerating the k-best DAGs, we simply want to find k different DAGs whose scores are the smallest. Enumerating such DAGs can provide a number of insights about a dataset, beyond learning just a single

best DAG. First, the single best DAG may be (Markov) equivalent to other equally good DAGs, or otherwise, there may be other DAGs with comparable scores. In either case, we would like to be aware of all such DAGs. Next, if the score that we use corresponds to the probability of a DAG given the data, then enumerating the k-best DAGs gives us a (truncated) view of the posterior over DAGs. If the aggregate probability of the k-best DAGs consumes most of the available probability, then we can further say that the remaining DAGs are unlikely to be relevant. By enumerating the k-best DAGs, we can also look for the structural features that are prominent in the most likely DAGs, as a more tractable approach to Bayesian model averaging; see, e.g., (Tian et al., 2010; Chen & Tian, 2014).

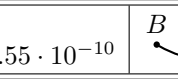
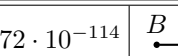
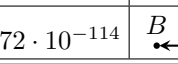
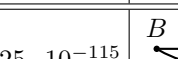
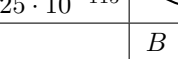
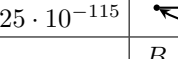
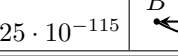
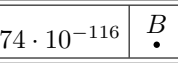
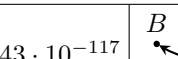
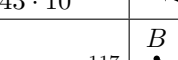
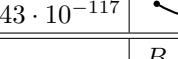
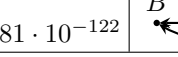
For a concrete example, consider Figure 1 where we have enumerated all 25 DAGs learned from a dataset $\mathcal{D}$ over 3 variables: B, D and L. The dataset $\mathcal{D}$ was simulated from a known Bayesian network, and hence we know the ground-truth structure: $B \rightarrow D \rightarrow L$. First, each DAG has been ranked by its probability (using its normalized MDL score, which corresponds to the BIC score). DAGs with the same rank and score are grouped together (in this case, each set corresponds to a Markov equivalence class). We note that the top set of two DAGs together have 66.28% probability, although there is a second set of three DAGs which are also relatively likely at 33.60% probability. The remaining 20 DAGs (including the ground truth DAG $B \rightarrow D \rightarrow L$) are relatively unlikely, given the dataset $\mathcal{D}$. Note that our dataset $\mathcal{D}$ is relatively small in this case, with only 100 examples. In Figure 2, we enumerated all DAGs from a dataset $\mathcal{D}$ over 1,000 examples. Here, the ground-truth DAG is now among the 3 most likely DAGs, which together have 98.01% probability. In Figure 1, there is a 99.9990% probability that there is an edge connecting variables D and L; in Figure 2, the probability that there is no such edge is essentially negligible.

One of the first approaches for enumerating the k-best DAGs was based on dynamic programming (DP) (Tian et al., 2010), and was an extension of a DP-based approach for learning a single optimal DAG (Koivisto & Sood, 2004; Singh & Moore, 2005; Silander & Myllymäki, 2006). Another approach encodes the

rank	score	DAG
1	0.3314	
–	0.3314	
3	0.1120	
–	0.1120	
–	0.1120	
6	0.0012	
7	$3.35 \cdot 10^{-5}$	
–	$3.35 \cdot 10^{-5}$	
–	$3.35 \cdot 10^{-5}$	
10	$6.02 \cdot 10^{-6}$	
11	$2.03 \cdot 10^{-6}$	
–	$2.03 \cdot 10^{-6}$	
13	$3.58 \cdot 10^{-7}$	

rank	score	DAG
14	$1.21 \cdot 10^{-7}$	
–	$1.21 \cdot 10^{-7}$	
–	$1.21 \cdot 10^{-7}$	
–	$1.21 \cdot 10^{-7}$	
–	$1.21 \cdot 10^{-7}$	
–	$1.21 \cdot 10^{-7}$	
20	$6.09 \cdot 10^{-10}$	
–	$6.09 \cdot 10^{-10}$	
22	$2.06 \cdot 10^{-10}$	
–	$2.06 \cdot 10^{-10}$	
–	$2.06 \cdot 10^{-10}$	
25	$2.20 \cdot 10^{-12}$	

Figure 1: All 25 Bayesian network structures over 3 variables (B, D and L), learned from a small dataset over 100 examples, and sorted by probability. The ground-truth structure is $B \to D \to L$, taken from the CHILD Bayesian network (Spiegelhalter & Cowell, 1992). A dash (–) indicates that a DAG has the same rank and score as the DAG above it.

rank	score	DAG
1	0.3267	
–	0.3267	
–	0.3267	
4	0.0090	
–	0.0090	
6	0.0007	
–	0.0007	
–	0.0007	
9	$1.78 \cdot 10^{-7}$	
10	$1.29 \cdot 10^{-8}$	
–	$1.29 \cdot 10^{-8}$	
–	$1.29 \cdot 10^{-8}$	
–	$1.29 \cdot 10^{-8}$	
–	$1.29 \cdot 10^{-8}$	
–	$1.29 \cdot 10^{-8}$	

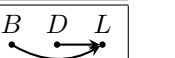
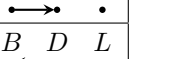
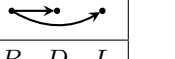
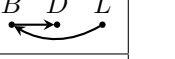
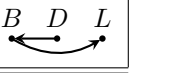
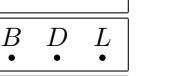
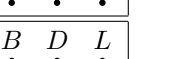
rank	score	DAG
16	$3.55 \cdot 10^{-10}$	
17	$1.72 \cdot 10^{-114}$	
–	$1.72 \cdot 10^{-114}$	
19	$1.25 \cdot 10^{-115}$	
–	$1.25 \cdot 10^{-115}$	
–	$1.25 \cdot 10^{-115}$	
22	$4.74 \cdot 10^{-116}$	
23	$3.43 \cdot 10^{-117}$	
–	$3.43 \cdot 10^{-117}$	
25	$6.81 \cdot 10^{-122}$	

Figure 2: All 25 Bayesian network structures over 3 variables (B, D and L), learned from a dataset over 1,000 examples, and sorted by probability. The ground-truth structure is $B \to D \to L$, taken from the CHILD Bayesian network (Spiegelhalter & Cowell, 1992). A dash (–) indicates that a DAG has the same rank and score as the DAG above it.

enumeration problem as a series of integer linear programming (ILP) problems (Cussens et al., 2013). The first ILP problem encodes the problem of finding a single optimal DAG (Jaakkola et al., 2010; Cussens, 2011; Cussens et al., 2013). This solution is then eliminated by adding a constraint to the ILP problem. The solution of the new ILP problem gives us the second best DAG. We repeat this process until each of the k-best DAGs is enumerated.

The current state-of-the-art for enumerating the k-best DAGs is based on heuristic search methods such as A*, which was previously observed to be orders-of-magnitude more efficient than the above approaches based on DP and ILP (Chen et al., 2015). It is based on navigating a seemingly intractable search space over all DAGs. The complexity of this search can be mitigated, however, by exploiting an oracle that can find a single optimal DAG. This search space, called the BN graph, can also be used to learn Bayesian network structures with non-decomposable priors and constraints (Chen et al., 2015).

2.3. Evaluating Bayesian Network Structures Using the MDL Score

Suppose we are given a (complete) dataset $\mathcal{D}$ containing N examples over a set of variables $\mathbf{X}$. The MDL score of a DAG G given dataset $\mathcal{D}$ is denoted by $\mathsf{MDL}(G|\mathcal{D})$ and defined as:

$$\mathsf{MDL}(G|\mathcal{D}) = \sum_{X\mathbf{U}} \mathsf{MDL}(X|\mathbf{U}),$$

which decomposes into local scores $\mathsf{MDL}(X|\mathbf{U})$ over the families $X\mathbf{U}$ of G:

$$\mathsf{MDL}(X|\mathbf{U}) = H(X|\mathbf{U}) + c \cdot K(X|\mathbf{U}),$$

where $c = \frac{1}{2}\log_2 N$ is a constant. The MDL score balances between two objectives: we want to minimize the terms $H(X|\mathbf{U})$ (to improve the fit of the data) and minimize the terms $K(X|\mathbf{U})$ (to reduce the model complexity).

First, to improve the fit of the data, we seek to minimize the conditional entropy:

$$H(X|\mathbf{U}) = -N \cdot \sum_{x\mathbf{u}} Pr_{\mathcal{D}}(x\mathbf{u}) \log_2 Pr_{\mathcal{D}}(x|\mathbf{u})$$

where $H(X|\mathbf{U})$ is the conditional entropy of a variable X given its parents $\mathbf{U}$ (and scaled by N here). This conditional entropy is computed with respect to the empirical distribution $Pr_{\mathcal{D}}$ induced by the data, i.e., $Pr_{\mathcal{D}}(\mathbf{x}) = \frac{1}{N}\mathcal{D}\#(\mathbf{x})$, where $\mathcal{D}\#(\mathbf{x})$ is the number of times instance $\mathbf{x}$ appears in the dataset $\mathcal{D}$. Roughly, the conditional entropy $H(X|\mathbf{U})$ is the expected uncertainty in the value of variable X, when we observe the parents $\mathbf{U}$. Hence, the lower the conditional entropy, the better the parents $\mathbf{U}$ are at predicting the value of X (and hence, providing a better fit of the data). Correspondingly, the conditional entropy $H(X|\mathbf{U})$ can only decrease (or stay the same value) as we add more parents $\mathbf{U}$ to variable X. Finally, the conditional entropy is non-negative, and upper-bounded by $H_{\max}(X) = N \cdot \log_2 |X|$. This upper bound corresponds to the entropy of the uniform distribution over X, where $Pr(x) = \frac{1}{|X|}$.

The second objective of the MDL score is to minimize the model complexity:

$$K(X|\mathbf{U}) = (|X| - 1) \prod_{U \in \mathbf{U}} |U|$$

where $K(X|\mathbf{U})$ is the number of free parameters in the conditional distribution of X given $\mathbf{U}$. For a given DAG G, the sum of all $K(X|\mathbf{U})$ is the total number of free parameters in the corresponding Bayesian network.[2]

Consider the following dataset of $N = 100$ examples:

A	B	$\mathcal{D}\#(a, b)$
a	b	22
a	$\bar{b}$	18
$\bar{a}$	b	38
$\bar{a}$	$\bar{b}$	22

In the following table, we enumerate the three possible DAGs G, with their maximum likelihood parameters, aggregate entropy $H(G) = \sum_{X\mathbf{U}} H(X|\mathbf{U})$, aggregate complexity $K(G) = \sum_{X\mathbf{U}} K(X|\mathbf{U})$, and MDL scores:

[2] We note that the MDL and BIC scores are numerically equivalent, and that both scores are asymptotically equivalent to the BDeu score; see, e.g., (Koller & Friedman, 2009).

G	parameters	$H(G)$	$K(G)$	$\mathsf{MDL}(G\|D)$
$A \quad B$	$\theta_a = \frac{40}{100}$, $\theta_b = \frac{60}{100}$	194.19	2	200.83
$A \rightarrow B$	$\theta_a = \frac{40}{100}$, $\theta_{b\|a} = \frac{22}{40}$, $\theta_{b\|\bar{a}} = \frac{38}{60}$	193.69	3	203.66
$A \leftarrow B$	$\theta_{a\|b} = \frac{22}{60}$, $\theta_{a\|\bar{b}} = \frac{18}{40}$, $\theta_b = \frac{60}{100}$	193.69	3	203.66

where $c = \frac{1}{2}\log_2 100 = 3.32$. First, note that DAGs $A \rightarrow B$ and $A \leftarrow B$ are Markov equivalent, and they have the same entropy and complexity. Next, these two DAGs have a lower (better) entropy $H(G)$ than the empty DAG with no edge. Finally, the lowest (best) MDL score is obtained by the empty DAG, since the difference in complexity was greater than the difference in entropy. It is this balance, between entropy and complexity, that we analyze in this paper.

2.4. Pruning Parent Sets with the MDL Score

The MDL score balances the fit of the data with the complexity of the model. Prior works have studied this balance, finding ways to prune the search space of DAGs, thus simplifying the learning problem (Tian, 2000; Teyssier & Koller, 2005; De Campos & Ji, 2011), at least for the case of learning a single optimal DAG.

First, we consider one of the most elementary pruning rules. Consider a variable X and a candidate parent set $\mathbf{U}$. The following theorem identifies a simple and general condition that guarantees that $\mathbf{U}$ will never appear in a DAG minimizing any decomposable score (MDL or otherwise) (Teyssier & Koller, 2005).

Theorem 1 (Teyssier & Koller, 2005)**.** *Let $\mathcal{D}$ denote a dataset, with variable X and two candidate sets of parents $\mathbf{U}$ and $\mathbf{U}'$. If $\mathbf{U}' \subset \mathbf{U}$ and $\mathsf{score}(X\mathbf{U}'|\mathcal{D}) < \mathsf{score}(X\mathbf{U}|\mathcal{D})$, then no DAG G that minimizes $\mathsf{score}(G|\mathcal{D})$ contains the family $X\mathbf{U}$.*[3]

Consider a DAG G that contains $X\mathbf{U}$ as a family. We can replace the family $X\mathbf{U}$ with a smaller family $X\mathbf{U}'$ and obtain a new DAG G', since replacing $X\mathbf{U}$

[3] We note that non-strict inequalities lead to more pruning, although we use strict inequalities here to simplify the discussion.

with a smaller family will not introduce any directed cycles. Moreover, if the smaller family $X\mathbf{U}'$ has a better score as well, the new DAG G' will also have a better score (since the scores of other families do not change). Hence, an optimal DAG would not contain such a family $X\mathbf{U}$, as we can obtain a better DAG with a strictly better score. We can thus ignore the family $X\mathbf{U}$ when searching for an optimal Bayesian network in this case.

The ability to prune many families in this manner is critical to the efficiency and scalability of structure learning (De Campos & Ji, 2011; Cussens, 2012). Unfortunately, Theorem 1 is not practical enough by itself, as there are exponentially many pairs of parent sets $\mathbf{U}$ and $\mathbf{U}'$ to test. However, the MDL score lends itself to a simple test on the *size* of a parent set, that allows us to eliminate a large number of structures at once. Roughly, there is a trade-off in the score of a family $\mathsf{MDL}(X\mathbf{U}|\mathcal{D})$ when we try to add a new parent U to X. Adding a parent increases the fit of the data, but it also increases the complexity of the model. For the MDL score, there is a point after which adding new parents cannot provide a better fit of the data, compared to the additional complexity it would introduce. This is summarized by the following theorem (Suzuki, 1996; Tian, 2000; De Campos & Ji, 2011).

Theorem 2. *Given a dataset with N examples, there exists an optimal DAG under the MDL score whose families $X\mathbf{U}$ have parents where $|\mathbf{U}| \leq \lfloor \log_2 \frac{N}{c} \rfloor$, and where $c = \frac{1}{2} \log_2 N$.*

Theorem 2 provides an upper bound d on the number of parents $\mathbf{U}$ that a variable X needs to have in an optimal Bayesian network. Note that this bound depends only on the size of the dataset N. For example, the following table enumerates the bounds d given dataset sizes N from $2^6 = 128$ to $2^{16} = 65,536$:

N	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
d	4	5	6	6	7	8	9	10	11	12	13

For example, when $N = 2^{10} = 1,024$, it suffices to consider only those families $X\mathbf{U}$ where there are at most $d = 7$ parents $\mathbf{U}$. Note that the bounds are more useful when the dataset size is not too large.

If a structure learning algorithm can accommodate such a bound, then it can potentially eliminate an exponential number of candidate structures. For example, in frameworks based on heuristic search methods such as A*, one can prune away nodes (representing DAGs) in the search space when they violate the bounds (Chen et al., 2015). Similarly, in frameworks based on dynamic programming, one can simplify the underlying recurrence relations so that they do not consider sub-problems that violate the bounds. As another example, approaches based on ILP can significantly benefit from pruning away irrelevant families (those with too many parents). Such families do not need to be encoded into the ILP, and can lead to exponentially fewer ILP variables in the encoding of a structure learning problem (Jaakkola et al., 2010; Cussens, 2011). For structure learning approaches based on decomposable scores (as in Equation 1), one typically has to solve local optimization sub-problems of the form $\min_{\mathbf{U}} \mathsf{score}(X\mathbf{U}|\mathcal{D})$. Some approaches perform some pre-computations on these sub-problems, allowing for more efficient lookups to be performed during structure learning itself (De Campos & Ji, 2011; Yuan & Malone, 2013). However, for datasets over larger sets of variables, such an approach is only feasible when (1) there are not too many scores to process, and (2) the associated data structures can fit in memory (both are indeed enabled by the above bounds).

3. Pruning Parent Sets While Enumerating Structures

In this section, we generalize the pruning techniques from the previous section for the task of enumerating the k-best Bayesian network structures. For example, Theorem 2 can only guarantee that X does not have too many parents in an *optimal* DAG—but it says nothing about how many parents a variable can have in the 2^{nd} best DAG.

Our goal in this paper is to prune the space of structures that we need to consider when enumerating the k-best DAGs. Ultimately, we will propose a single algorithm for computing an upper bound on the size of the largest family that we will need to consider. Along the way, we will analyze simpler bounds

that will conceptually motivate the final algorithm that we propose.

First, we present a simple generalization of Theorem 1.

Theorem 3. *Let $\mathcal{D}$ denote a dataset, with variable X and a candidate set of parents $\mathbf{U}$. If there are k parents sets $\mathbf{U}' \subset \mathbf{U}$ where $\mathsf{score}(X\mathbf{U}'|\mathcal{D}) < \mathsf{score}(X\mathbf{U}|\mathcal{D})$, then none of the k-best DAGs contains the family $X\mathbf{U}$.*

Consider a DAG G that contains $X\mathbf{U}$ as a family. We can replace the family $X\mathbf{U}$ with any of the k-better families $X\mathbf{U}'$ (none of which will introduce a cycle), and obtain k new DAGs G with a strictly lower score. Hence, none of the k-best DAGs will contain such a family $X\mathbf{U}$, since we can always find k better DAGs with lower scores. Hence, we can eliminate the family $X\mathbf{U}$ from consideration when enumerating the k-best DAGs.[4]

The following observation provides another simple condition where we can safely prune a family from an enumeration problem.

Theorem 4. *Let $\mathcal{D}$ denote a dataset, with variable X and a candidate set of parents $\mathbf{U}$. Let G denote the DAG with the best $\mathsf{score}(G|\mathcal{D})$, among all DAGs containing the family $X\mathbf{U}$. If there are k better DAGs G' where $\mathsf{score}(G'|\mathcal{D}) < \mathsf{score}(G|\mathcal{D})$, then none of the k-best DAGs of dataset $\mathcal{D}$ contains the family $X\mathbf{U}$.*[5]

Theorems 3 and 4 are simple and intuitive, but they are of limited practical utility by themselves. In particular, in order to eliminate a family $X\mathbf{U}$ from consideration, we would need a way to determine whether there are k smaller families (or DAGs) with better scores. Under the MDL score, we can identify some stronger and more practical conditions, which we shall propose next. Before we proceed, we remark that Theorem 3 is based on reasoning locally about

[4] We remark that Theorem 3 was also implicitly used in the k-best enumeration algorithms of (Tian et al., 2010) and (Chen & Tian, 2014), although we make the underlying concept explicit here.

[5] If G is the best DAG containing the family $X\mathbf{U}$, and if G' is a DAG with a strictly better score, then G' cannot contain the family $X\mathbf{U}$ (by the optimality of G).

families $X\mathbf{U}$, and Theorem 4 is based on reasoning globally about DAGs G. We consider instances of such local and global tests, next.

3.1. Pruning with Local Tests

Theorem 2 provided a practical realization of Theorem 1, for the MDL score. It provided an upper bound on the number of parents that a variable needs to have in an optimal DAG, which allows us to eliminate a large proportion of DAG structures at once. Theorem 2 was based on the trade-off between the conditional entropy (fit to the data) and the model complexity (number of parameters). The following result establishes an analogous bound for the problem of enumerating the k-best structures.

Theorem 5. *Let $\mathcal{D}$ denote a dataset, with variable X and a candidate set of parents $\mathbf{U}$. If $H_{\max}(X) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$, then every proper subset $\mathbf{U}'$ of $\mathbf{U}$ has a local score $\mathsf{MDL}(X\mathbf{U}'|\mathcal{D}) \leq \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$, where $c = \frac{1}{2}\log_2 N$.*

Intuitively, this theorem identifies a limit on the number of parents $\mathbf{U}$ that a variable needs to have, where the added complexity of a large enough parent set $\mathbf{U}$ makes *every* smaller parent set $\mathbf{U}' \subset \mathbf{U}$ preferable to $\mathbf{U}$ itself. In particular, if a given parent set $\mathbf{U}$ satisfies the conditions of Theorem 5, then there are $2^{|\mathbf{U}|} - 1$ smaller parent sets $\mathbf{U}' \subset \mathbf{U}$ that are at least as good as $\mathbf{U}$. Thus, if we are interested in enumerating the k-best DAGs, and if $k \leq 2^{|\mathbf{U}|} - 1$, then we know that $X\mathbf{U}$ is not needed to find the k-best DAGs, via Theorem 3. As a result, we obtain the following result that is analogous to Theorem 2.

Corollary 1. *Given a dataset with N examples, there exists a set of k-best DAGs under the MDL score where families $X\mathbf{U}$ have parents where $|\mathbf{U}| \leq d = \lfloor \log_2 \frac{2N}{c} \rfloor$ when $k \leq 2^{d+1} - 1$ and where $c = \frac{1}{2}\log_2 N$.*

Note that first, this theorem identifies a bound d on the number of parents in $\mathbf{U}$ that we need to consider; for any parent set $\mathbf{U}$ of size $d+1$, every smaller parent set $\mathbf{U}' \subset \mathbf{U}$ will have a better score. Next, we must verify that this number of smaller parent sets (i.e., $2^{d+1} - 1$) eliminates at least as many parent sets as DAGs that we want to enumerate.

Note that the bound d itself depends only on the size of the dataset N. For example, the following table enumerates the bounds given dataset sizes N from $2^6 = 128$ to $2^{16} = 65,536$:

N	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
d	5	6	7	7	8	9	10	11	12	13	14

For example, when $N = 2^{10} = 1,024$, then $d = 8$ and there are at least $2^{8+1}-1 = 511$ better families for any family $X\mathbf{U}$ with 9 parents or more. Hence, if we want to enumerate the 511-best DAGs (or fewer), it suffices to consider families with no more than 8 parents. For any parent set $\mathbf{U}$ with size greater than d, it still holds that every smaller parent set $\mathbf{U}' \subset \mathbf{U}$ will lead to a better score. Hence, if we wanted to enumerate instead the $1,000$-best DAGs (for the same dataset size), then we find the smallest parent set size $d' > d$ where $k \leq 2^{d'+1} - 1$, which is $d' = 9$ in this case (and we consider only parent sets of size d' or less).

Like Theorem 2, Corollary 1 provides an upper bound on the number of parents $\mathbf{U}$ that a variable X needs to have, but now for the k-best Bayesian networks. However, when k is large, we may have to inflate the bound to accommodate the additional networks, which may make the bound too loose to be practical. Next, we will consider more sophisticated tests that can allow us to provide tighter bounds when enumerating the k-best Bayesian networks, even for larger values of k.

3.2. Pruning with Global Tests

We now provide a practical realization of Theorem 4, which states that if there are k DAGs better than the best DAG containing a family $X\mathbf{U}$, then we can prune any DAG containing the family $X\mathbf{U}$ for the purposes of enumerating the k-best DAGs. While Theorem 3 reasons locally about families $X\mathbf{U}$, Theorem 4 reasons globally about DAGs. The importance of this distinction is that it will allow us to more aggressively prune the space of DAGs.

We shall now propose a practical method for pruning a family $X\mathbf{U}$, by reasoning globally about DAGs. We shall proceed in multiple steps. First, we will

discuss a technique that allows us to determine when there are k better DAGs than a given DAG G. Next, we will apply this technique so that we can bound the size of a family $X\mathbf{U}$ under the assumption that a DAG's topological ordering is fixed (which will be given in Algorithm 1). Finally, we will generalize across all possible topological orderings, allowing us to bound the size of a family $X\mathbf{U}$ for the original learning problem over all DAGs (which will be given in Algorithm 2). This final algorithm is the one we shall evaluate in our experiments, in Section 4.

Consider again Theorem 5, which tells us if a family $X\mathbf{U}$ has too many parents $\mathbf{U}$, then under the MDL score, the complexity of the corresponding model is high enough so that *every* smaller family $X\mathbf{U}'$ will be preferred, where $\mathbf{U}' \subset \mathbf{U}$. This reasoning can be extended to *multiple* families $Y\mathbf{V}$, for a given set of variables $\mathbf{Y}$. That is, the size of the family $X\mathbf{U}$ can introduce enough model complexity so that we can reduce the size of *other* families $Y\mathbf{V}$ as well, and still obtain a better scoring DAG. This is formalized in the following theorem.

Theorem 6. *Suppose we are given a dataset with N examples over variables $\mathbf{X}$, and a DAG G with (1) a family $X\mathbf{U}$ and (2) a set of variables $\mathbf{Y} \subseteq \mathbf{X} \setminus X$ with families $Y\mathbf{V}$ for each $Y \in \mathbf{Y}$. Suppose further that*

$$H_{\max}(X) + \sum_{Y \in \mathbf{Y}} H_{\max}(Y) \leq \frac{1}{2} c \cdot K(X|\mathbf{U})$$

where $c = \frac{1}{2} \log_2 N$. If we construct a new DAG G' from G where we:

1. *replace the parents $\mathbf{U}$ of variable X with some proper subset $\mathbf{U}' \subset \mathbf{U}$,*
2. *for each $Y \in \mathbf{Y}$, replace the parents $\mathbf{V}$ of variable Y with some subset $\mathbf{V}' \subseteq \mathbf{V}$.*

then any such DAG G' satisfies $\mathsf{MDL}(G'|\mathcal{D}) \leq \mathsf{MDL}(G|\mathcal{D})$.

By reasoning about the additional families of variables $\mathbf{Y}$, we can identify *many* more DAGs that are better than a given DAG G. In particular, when the conditions of Theorem 6 are met, then any sub-DAG G' of G has an MDL score that is at least as good, after we reduce the sizes of the families $X\mathbf{U}$ and

$Y\mathbf{V}$. More specifically, for the family $X\mathbf{U}$, there are $2^{|\mathbf{U}|} - 1$ proper subsets of $\mathbf{U}$. For each family $Y\mathbf{V}$, there are $2^{|\mathbf{V}|}$ subsets of $\mathbf{V}$. We can pick any sub-DAG by taking any combination of these families, hence there are

$$(2^{|\mathbf{U}|} - 1) \cdot \prod_{\substack{Y\mathbf{V} \\ Y \in \mathbf{Y}}} 2^{|\mathbf{V}|}$$

possible sub-DAGs G' whose scores are at least as good as G. This is in contrast to Theorem 5 which considers a *single* family $X\mathbf{U}$, and consequently, it identifies only $2^{|\mathbf{U}|}-1$ better DAGs. Based on this analysis, we obtain the following upper bound on the number of parents that a variable needs to have.

Corollary 2. *Suppose we are given a dataset with N examples over variables $\mathbf{X}$, and a DAG G with (1) a family $X\mathbf{U}$ and (2) a set of variables $\mathbf{Y} \subseteq \mathbf{X} \setminus X$ with families $Y\mathbf{V}$ for each $Y \in \mathbf{Y}$. Suppose that $\log_2 |Y| \leq \alpha \cdot (|X| - 1)$ for all $Y \in \mathbf{Y}$, for some constant α. Let*

$$d(\mathbf{Y}) = \left\lfloor \log_2 \left(\frac{2N}{c} \cdot \alpha \cdot (|\mathbf{Y}| + 1) \right) \right\rfloor.$$

where $c = \frac{1}{2} \log_2 N$. There exists a set of k-best DAGs under the MDL score where families $X\mathbf{U}$ have parents where $|\mathbf{U}| \leq d(\mathbf{Y})$ when

$$k \leq (2^{d(\mathbf{Y})+1} - 1) \cdot \prod_{\substack{Y\mathbf{V} \\ Y \in \mathbf{Y}}} 2^{|\mathbf{V}|}.$$

If the conditions of Corollary 2 are met, then the conditions of Theorem 6 are met as well (as shown in the Appendix). Hence, in this case, we can find a set of k-best DAGs where X has at most $d(\mathbf{Y})$ parents.

Note that the bound $d(\mathbf{Y})$ depends on the size of the dataset N, the number of variables $\mathbf{Y}$, and the cardinalities $|Y|$ of variables $Y \in \mathbf{Y}$. Suppose all variables are binary, and hence we can assume $\alpha = 1$; in general, we can set $\alpha = \max_{Y \in \mathbf{Y}} \frac{\log_2 |Y|}{|X|-1}$. If $\mathbf{Y}$ is empty, then $d(\mathbf{Y}) = \lfloor \log_2 \frac{2N}{c} \rfloor$, which corresponds to the bound of Corollary 1. Remember that Corollary 1 provides a bound d on the number of parents when the number k of DAGs that we want to enumerate satisfies $k \leq 2^{d+1} - 1$. If k is larger, then Corollary 2 can provide a tighter

bound by reasoning about other families $Y\mathbf{V}$. In particular, if we include larger families $Y\mathbf{V}$, then we can accommodate larger values of k. We also want to minimize the number of variables Y to include in $\mathbf{Y}$, as this also loosens the bound $d(\mathbf{Y})$. We discuss how to optimize this set $\mathbf{Y}$, next.

3.2.1. A Special Case: A Fixed Topological Ordering

Let π denote a topological ordering $\langle X_1, \ldots, X_n \rangle$ of the n variables $\mathbf{X}$, and let $\pi_{1:i}$ (and $\pi_{i:n}$) denote the set of the first i (and last $n-i+1$) variables in the order. Consider the optimal DAG $G^\star$ that contains the family $X\mathbf{U}$, but also respects the given ordering π:

$$\begin{aligned} G^\star &= \operatorname*{argmin}_{\substack{G \sim \pi \\ X\mathbf{U} \in G}} \mathsf{MDL}(G|\mathcal{D}) \\ &= \mathsf{MDL}(X\mathbf{U}|\mathcal{D}) + \sum_{\substack{i \in \{1,\ldots,n\} \\ X_i \neq X}} \min_{\mathbf{U}_i \subseteq \pi_{1:i-1}} \mathsf{MDL}(X_i\mathbf{U}_i|\mathcal{D}). \end{aligned}$$

Since the DAG must respect the given ordering (which we denote by $G \sim \pi$), the optimal DAG G can find the optimal families independently (apart from $X\mathbf{U}$, which we keep fixed). This corresponds to the K2 structure learning algorithm (Cooper & Herskovits, 1992). For networks of the scale that we are interested in, these local sub-problems $\min_{\mathbf{U}} \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$ are tractable in practice. Hence, we can also compute such a DAG $G^\star$.

Next, we want to identify an upper bound on the number of parents $\mathbf{U}$ that a variable X needs to have, without eliminating the k-best DAGs for a given topological ordering π. For a given family $X\mathbf{U}$, we can find the best DAG $G^\star$ with family $X\mathbf{U}$ respecting π, as described above. Using Theorem 5, we can determine if there are k better families smaller than $X\mathbf{U}$. Using Theorem 6, we can determine if there are k better DAGs than $G^\star$, which can lead to a tighter bound on the number of parents $\mathbf{U}$ that we need to consider, by reasoning about the other families $Y\mathbf{V}$ in $G^\star$. In particular, we want to find a large enough set $\mathbf{Y} \subseteq \mathbf{X} \setminus X$, such that the DAG $G^\star$ meets the conditions of Theorem 6 (and Corollary 2). That is, we need to find a large enough set of variables $\mathbf{Y}$ so that we can guarantee that there are k better DAGs than $G^\star$.

Although the bound is easy to evaluate given a particular set $\mathbf{Y}$, there are exponentially many subsets $\mathbf{Y}$. Thus, we first propose a simple greedy method for the case where the topological ordering π is fixed (we shall relax this assumption later). In particular, after first initializing $\mathbf{Y}$ to the empty set, we propose to add variables one-by-one to $\mathbf{Y}$ in the reverse order of π, until the conditions of Corollary 2 are met (i.e., we can enumerate at least k better DAGs). Since finding larger families will help us enumerate more sub-DAGs, we propose to add variables to $\mathbf{Y}$ by reverse topological order, since variables at the end of the order can have larger families. In contrast, variables at the start of the order must have smaller families (given a topological order, a variable may only have parents that appear earlier in the order). This procedure is summarized in Algorithm 1, which returns an upper bound on number of parents that we need to consider when enumerating the k-best DAGs, given a topological ordering π.

For example, suppose all variables are binary, and hence we can assume $\alpha = 1$. Further, suppose that $N = 2^8$ and that we want to enumerate $k = 1,000$ DAGs. First, if $\mathbf{Y} = \emptyset$, then Corollary 2 (and Corollary 1) tells us that $d(\emptyset) = 7$ but we can only guarantee that $2^{7+1} - 1 = 255$ of the best DAGs are included. Suppose we add the last variable Y_n of the topological ordering π to our set $\mathbf{Y}$. Further, say that Y_n has 2 parents in $G^\star$. In this case, $d(\mathbf{Y}) = 8$ and $(2^{8+1} - 1) \cdot 2^2 = 2,044$ of the best DAGs are included. Hence, this bound is sufficient for $k = 1,000$, and we know that we only need to consider families $X\mathbf{U}$ where parent sets $\mathbf{U}$ have size at most 8, given the topological ordering π.[6]

3.2.2. The General Case

Our final goal is to determine an upper bound on the number of parents that a given variable can have in the k-best DAGs. We have just identified an upper bound for the special case where all k-best DAGs respect a given topological ordering. If we find such a bound for all $n!$ orderings, then the loosest of these

[6] Using Corollary 1, we could also inflate the bound until it includes enough DAGs. For $k = 1,000$ DAGs, this leads to a bound of $d = 9$, which is a looser bound in this case.

Algorithm 1: `BoundParents`($\mathcal{D}$,π,X,k)

Data: dataset $\mathcal{D}$ over variables $\mathbf{X}$; topological ordering π; variable X; count k to enumerate.

Result: upper bound on $|\mathbf{U}|$ for X.

$(\mathbf{Y}, s) \leftarrow (\{\}, 1)$
for $i \leftarrow |\mathbf{X}|$ **to** 1 **do**
  $X_i \leftarrow \pi_{i:i}$
  if $X_i \neq X$ **then**
    $\mathbf{U}_i^* \leftarrow \text{argmin}_{\mathbf{U}_i \subseteq \pi_{1:i-1}} \mathsf{MDL}(X_i \mathbf{U}_i | \mathcal{D})$
    $(\mathbf{Y}, s) \leftarrow (\mathbf{Y} \cup X_i, s \cdot 2^{|\mathbf{U}_i^*|})$
  if $k \leq s \cdot (2^{d(\mathbf{Y})+1} - 1)$ **then**
    return $d(\mathbf{Y})$

bounds will give us an upper bound for the general case that we seek.

Here, we propose to navigate the permutation tree of all $n!$ orderings, where each (partial) path on the permutation tree corresponds to a (partial) ordering of the variables; see Figure 3. The key observation here is that walking down a path on the permutation tree can be viewed as an instance of Algorithm 1 using the corresponding order. That is, Algorithm 1 incrementally adds variables one-by-one from the end of the ordering π. However, once Algorithm 1 finds a large enough set $\mathbf{Y}$ satisfying Corollary 2, then it returns an upper bound of $d(\mathbf{Y})$. Note that if it constructs a set $\mathbf{Y} = \pi_{i:n}$, then the bound obtained from $\mathbf{Y}$ would be the same as the one that would be obtained for any other topological ordering with the same suffix (i.e., each run of the different orderings would perform precisely the same steps in Algorithm 1). Hence, a run of Algorithm 1 that returns early can be viewed as obtaining bounds for (exponentially) many permutations at once. We can backtrack to the next unexplored branch of the permutation tree, and continue running Algorithm 1 at that point. By repeating

Algorithm 2: `BoundParents`($\mathcal{D}$,X,k)

Data: dataset $\mathcal{D}$ over variables $\mathbf{X}$; variable X; count k to enumerate.

Result: upper bound on $|\mathbf{U}|$ for X.

$Q \leftarrow$ queue initialized with tuple $(\mathbf{X}, \{\}, 1)$
$\mathbf{Y}_{\max} \leftarrow \{\}$
while Q is not empty **do**
  extract the first item $(\mathbf{Z}, \mathbf{Y}, s)$ from Q
  if $|\mathbf{Y}_{\max}| < |\mathbf{Y}|$ **then** $\mathbf{Y}_{\max} \leftarrow \mathbf{Y}$
  foreach $Z \in \mathbf{Z}$ **do**
    $\mathbf{Z}' \leftarrow \mathbf{Z} \setminus Z$
    if $Z \neq X$ **then**
      $\mathbf{U}^* \leftarrow \text{argmin}_{\mathbf{U} \subseteq \mathbf{Z}'} \mathsf{MDL}(Z\mathbf{U}|\mathcal{D})$
      $(\mathbf{Y}', s') \leftarrow (\mathbf{Y} \cup Z, s \cdot 2^{|\mathbf{U}^*|})$
    else $(\mathbf{Y}', s') \leftarrow (\mathbf{Y}, s)$
    if $k > s' \cdot (2^{d(\mathbf{Y}')+1} - 1)$ **then**
      insert $(\mathbf{Z}', Y', s')$ into Q
return $d(\mathbf{Y}_{\max})$

this process, we can perform an exhaustive search of the permutation tree over $n!$ orderings, where we prune a downward path when Algorithm 1 terminates early. The loosest bound that we observe during this exhaustive search gives us an upper bound on the number of parents that a given variable can have in the k-best DAGs. This procedure is described in Algorithm 2, where the permutation tree is implicitly navigated using a queue, which is used to enumerate all of the needed suffixes.

We finally remark that while the search space over permutations is large, depending on the number k of DAGs that we want to enumerate, we may only need to traverse the permutation tree up to a shallow depth. We shall illustrate

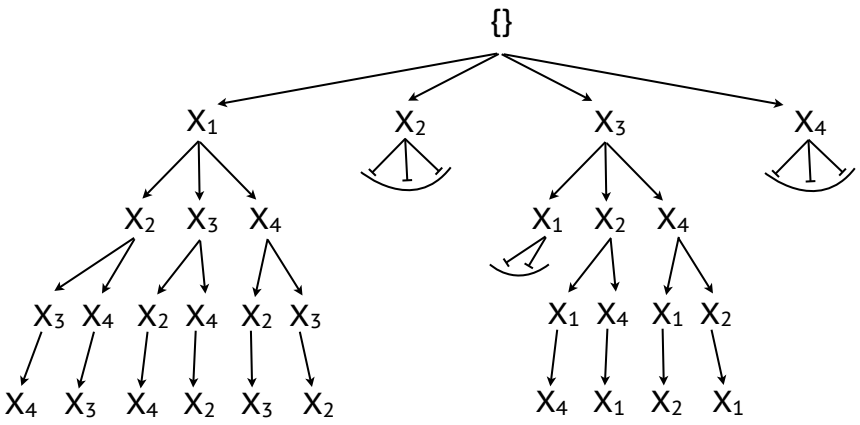


Figure 3: A (pruned) permutation tree over 4 variables: each path from the root to a node corresponds to a (partial) ordering π. The permutation spaces below the (partial) orderings $\langle X_2 \rangle$, $\langle X_3, X_1 \rangle$, and $\langle X_4 \rangle$ have been pruned here.

this in our experiments next.

4. Experiments

In the section, we evaluate our proposed method for pruning the search space of Bayesian network structures, for the problem of enumerating the k-best DAGs (more specifically, we evaluate Algorithm 2). We shall first evaluate the effectiveness of our approach in reducing the space of DAGs. We next evaluate the impact that this has on the state-of-the-art system for enumerating the k-best network structures, which is due to (Chen et al., 2015). Our experiments were performed on a 2.67GHz Intel Xeon X5650 CPU with a memory limit of 64 GB. We use real-world datasets from the UCI ML Repository (Bache & Lichman, 2013).[7] All timing results are averages over 10 runs.

First, we obtain an upper bound p on the maximum number of parents that any variable needs to have. In particular, we apply Algorithm 2 on every variable. Table 1 highlights the results. First, we note that the bound p ranges

[7] The datasets used were discretized, and are available at `http://urlearning.org/`

benchmark				10-best		100-best		1,000-best	
name	n	N	S	p	s	p	s	p	s
hepatitis	20	126	0.16	6	0.01	6	0.01	7	0.03
imports	22	205	0.69	6	0.03	6	0.03	7	0.07
parkinsons	23	195	1.44	6	0.04	6	0.04	8	0.10
sensors	25	5456	6.25	10	1.69	10	1.69	10	1.69
autos	26	159	13.00	6	0.10	6	0.10	8	1.46
horse	28	300	56.00	7	0.53	7	0.53	8	0.70
flag	29	194	116.00	6	0.22	6	0.22	7	0.73

Table 1: Full vs. pruned score lists for enumerating the k-best DAGs. For each benchmark we report: the number of variables n in the dataset, the size of the dataset N, the sizes of the full (S) and pruned (s) score lists in GBs, and the bound p on the number of parents.

from 24% (with flag) and 40% (with sensors) of the total number of variables. Next, we compare the memory required to represent the full list of scores of all families $X\mathbf{U}$ versus the pruned list. We assume a neutral representation (data structure) of a score list, where we use 64-bits to represent the parents (as a bit set), and another 64-bits to represent the score itself (using floating-point).[8][9] Table 1 shows that the pruned score lists use a much smaller amount of memory, compared to full score lists. While the pruned score lists always use less than 2GB of memory, it would not be possible to store the full score list in memory for a dataset like flag (given our 64GB limit). For dataset flag, the pruned list is a 158× savings in space. Finally, we note that for the datasets considered in Table 1, the upper bounds p are computed in less than 5 minutes.

We now use our pruned score list to learn the k-best DAGs using the state-

[8] We assume that for each variable X, the scores $\mathsf{score}(X\mathbf{U}|\mathcal{D})$ are stored in the same data structure, and then indexed by the parents $\mathbf{U}$ (which is represented as a bit set).

[9] For a dataset over n variables, there are $n \cdot 2^{n-1}$ total families $X\mathbf{U}$. For a bound p on the number of parents, there are $n \cdot \sum_{i=0}^{p} \binom{n-1}{i}$ unpruned families. Hence, a full score list uses $128 \cdot n \cdot 2^{n-1}$ bits, and a pruned score list uses $128 \cdot n \cdot \sum_{i=0}^{p} \binom{n-1}{i}$ bits, which is reported in Table 1.

benchmark		10-best			100-best			1,000-best		
name	n	E_h	T_h	T_{A*}	E_h	T_h	T_{A*}	E_h	T_h	T_{A*}
hepatitis	20	155	1.71	0.17	2188	3.32	0.80	6427	5.13	14.23
imports	22	111	63.26	0.16	232	73.83	0.20	1041	134.97	0.72
parkinsons	23	110	666.23	1.23	741	973.44	1.71	4313	3143.19	10.61
sensors	25	354	10219.25	3.65	482	13991.11	4.76	1342	23237.06	10.49
autos	26	1199	2098.97	6.46	2909	3242.36	8.96	9185	4062.17	13.78
horse	28	1095	2045.58	8.96	11653	2449.30	21.92	48069	5908.90	55.98
flag	29	1248	4454.21	19.79	26766	11093.91	45.22	110272	21959.47	257.27

Table 2: Enumerating the k-best DAGs using a pruned score list. For each benchmark we report: the number of variables n in the dataset, the number E_h of black-box invocations, and the times to compute the heuristic function (T_h) and to navigate the BN graph with A* search (T_{A*}), in seconds. See (Chen et al., 2015) for details.

of-the-art system of (Chen et al., 2015), which previously scaled to datasets over 23 variables, using full score lists. Table 2 highlights the results, showing that the $1,000$-best structures can be enumerated for datasets with as many as 29 variables. Note that the improvement from 23 variables to 29 variables is quite significant, considering the relative sizes of these search spaces (from Footnote 1).

Table 2 reports some of the relevant statistics of the system of (Chen et al., 2015). Basically, this system is performing A* search in a search space called the BN graph, which is a search space over all possible DAGs. A* search is typically used to find a single best solution, but continuing A* search can yield the k-best solutions as well (Dechter et al., 2012). While the BN graph is normally too large of a search space for methods such as A* search, (Chen et al., 2015) proposed to use an oracle that learns a single optimal DAG, as a heuristic for enumerating the k-best DAGs using A* search.

In Table 2, E_h is the number of times this black-box oracle was invoked, T_h is the total time to evaluate the heuristic function (i.e., the oracle), and T_{A*} is the total time spent in A* search. As expected, the majority of time (T_h) is

spent evaluating the structure learning oracle (Chen et al., 2015). We remark that the oracle only finds a single best solution, and hence uses the classical MDL bounds (i.e., when $k = 1$). However, A* search in the BN graph benefited significantly from the more general bounds proposed in this paper (primarily in terms of pruning the BN graph).

5. Conclusion

In this paper, we proposed new techniques for pruning the search space of DAGs, for the purposes of enumerating the k-best Bayesian networks. These techniques identify an upper bound on the number of parents that a node can have, among the k-best DAGs. These bounds exploit properties of the MDL score, and generalize widely-used bounds for the case of finding a single optimal DAG. In our experiments, our techniques allowed a state-of-the-art system for enumerating the k-best DAGs to scale from datasets over 23 variables to larger datasets over 29 variables.

Acknowledgments

This work has been partially supported by NSF grant #IIS-1514253, ONR grant #N00014-15-1-2339 and DARPA XAI grant #N66001-17-2-4032.

References

Bache, K., & Lichman, M. (2013). UCI Machine Learning Repository. URL: `http://archive.ics.uci.edu/ml`.

Chen, E. Y.-J., Choi, A., & Darwiche, A. (2015). Learning Bayesian networks with non-decomposable scores. In *GKR Workshop* LNAI (pp. 50–71).

Chen, E. Y.-J., Choi, A., & Darwiche, A. (2016). Enumerating equivalence classes of Bayesian networks using EC graphs. In *AISTATS*.

Chen, Y., & Tian, J. (2014). Finding the k-best equivalence classes of Bayesian network structures for model averaging. In *UAI*.

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

Cussens, J. (2011). Bayesian network learning with cutting planes. In *UAI* (pp. 153–160).

Cussens, J. (2012). An upper bound for BDeu local scores. In *ECAI Workshop on AIGM*.

Cussens, J., Bartlett, M., Jones, E. M., & Sheehan, N. A. (2013). Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology*, *37*, 69–83.

Darwiche, A. (2009). *Modeling and reasoning with Bayesian networks*. Cambridge University Press.

De Campos, C. P., & Ji, Q. (2011). Efficient structure learning of Bayesian networks using constraints. *JMLR*, *12*, 663–689.

Dechter, R., Flerova, N., & Marinescu, R. (2012). Search algorithms for m best solutions for graphical models. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence*.

Jaakkola, T., Sontag, D., Globerson, A., & Meila, M. (2010). Learning Bayesian network structure using LP relaxations. In *AISTATS* (pp. 358–365).

Koivisto, M., & Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *JMLR*, *5*, 549–573.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. The MIT Press.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Silander, T., & Myllymäki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In *UAI* (pp. 445–452).

Singh, A. P., & Moore, A. W. (2005). *Finding optimal Bayesian networks by dynamic programming*. Technical Report CMU-CALD-050106.

Spiegelhalter, D., & Cowell, R. G. (1992). Learning in probabilistic expert systems. *Bayesian statistics*, *4*, 447–465.

Suzuki, J. (1996). Learning Bayesian belief networks based on the minimum description length principle. In *ICML*.

Teyssier, M., & Koller, D. (2005). Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *UAI* (pp. 584–590).

Tian, J. (2000). A branch-and-bound algorithm for MDL learning Bayesian networks. In *UAI* (pp. 580–588).

Tian, J., He, R., & Ram, L. (2010). Bayesian model averaging using the k-best Bayesian network structures. In *UAI* (pp. 589–597).

Yuan, C., & Malone, B. (2013). Learning optimal Bayesian networks: A shortest path perspective. *JAIR*, *48*, 23–65.

Appendix A. Proofs

The proof of Theorem 3 is given in the text. The proof of Theorem 4 is immediate.

Lemma 1. *For parent sets* $\mathbf{U}$ *and* $\mathbf{U}'$ *where* $\mathbf{U}' \subset \mathbf{U}$*:*

$$\frac{1}{2}c \cdot K(X|\mathbf{U}) \leq c \cdot K(X|\mathbf{U}) - c \cdot K(X|\mathbf{U}').$$

Proof From the definition of $K(X|\mathbf{U})$,

$$K(X|\mathbf{U}) = K(X|\mathbf{U}') \cdot \prod_{U \in \mathbf{U} \setminus \mathbf{U}'} |U| \geq K(X|\mathbf{U}') \cdot 2.$$

We obtain the Lemma after rearranging. □

Theorem 7 (Tian, 2000)**.** *For parent sets* $\mathbf{U}$ *and* $\mathbf{U}'$ *where* $\mathbf{U}' \subset \mathbf{U}$*, if*

$$H_{\max}(X) \leq c \cdot K(X|\mathbf{U}) - c \cdot K(X|\mathbf{U}'),$$

then $\mathbf{U}$ *and any of its supersets* $\mathbf{U}''$ *satisfies*

$$\mathsf{MDL}(X\mathbf{U}' \mid \mathcal{D}) \leq \mathsf{MDL}(X\mathbf{U}'' \mid \mathcal{D}).$$

Proof of Theorem 5 If $H_{\max}(X) \leq \frac{1}{2}c \cdot K(X \mid \mathbf{U})$ and $\mathbf{U}' \subset \mathbf{U}$ then

$$H_{\max}(X) \leq c \cdot K(X|\mathbf{U}) - c \cdot K(X|\mathbf{U}')$$

by Lemma 1. By Theorem 7, we have $\mathsf{MDL}(X\mathbf{U}'|\mathcal{D}) \leq \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$. □

Proof of Corollary 1 Using Theorem 5, it suffices to show that

$$H_{\max}(X) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$$

for any $|\mathbf{U}| \geq d+1$. In this case, any $\mathbf{U}' \subset \mathbf{U}$ will have at least as good of a score, i.e., when $|\mathbf{U}'| \leq d$. We have

$$\begin{aligned}
&H_{\max}(X) = N \log_2 |X| \\
&\leq N \cdot (|X| - 1) && \text{by } \log_2 x \leq x - 1 \text{ for } x \geq 2 \\
&\leq \frac{1}{2}c \cdot (|X| - 1) \cdot 2^{d+1} && \text{by } d \leq \log_2 \tfrac{2N}{c} \leq d+1 \\
&\leq \frac{1}{2}c \cdot (|X| - 1) \cdot \prod_{U \in \mathbf{U}} |U| && \text{by } |U| \geq 2 \text{ and } |\mathbf{U}| \geq d+1 \\
&= \frac{1}{2}c \cdot K(X \mid \mathbf{U}) && \text{by definition of } K(X|\mathbf{U})
\end{aligned}$$

as desired. □

Proof of Theorem 6 Consider a DAG G with variables X and $\mathbf{Y}$. We shall denote these variables by X_i for $1 \leq i \leq m$, where $m = |\mathbf{Y}| + 1$ and where $X = X_1$ and $\mathbf{Y} = \{X_2, \ldots, X_m\}$. Let $\mathbf{U}_i$ denote the parents of variable X_i. Let G' denote a sub-DAG of G where X_1 has parents $\mathbf{U}_1' \subset \mathbf{U}_1$ and where X_i has

parents $\mathbf{U}'_i \subseteq \mathbf{U}_i$ for $2 \leq i \leq m$. We then have

$$
\begin{aligned}
&\mathsf{MDL}(G') \\
&= \mathsf{MDL}(G) + \textstyle\sum_{1 \leq i \leq m} \left[H(X_i|\mathbf{U}'_i) + c \cdot K(X_i|\mathbf{U}'_i) - H(X_i|\mathbf{U}_i) - c \cdot K(X_i|\mathbf{U}_i) \right] \\
&\leq \mathsf{MDL}(G) + \textstyle\sum_{1 \leq i \leq m} H_{\max}(X_i) + \sum_{1 \leq i \leq m} \left[c \cdot K(X_i|\mathbf{U}'_i) - c \cdot K(X_i|\mathbf{U}_i) \right] \\
&\qquad\qquad \text{by } H(X_i|\mathbf{U}'_i) \leq H_{\max}(X_i) \text{ and } H(X_i|\mathbf{U}_i) \geq 0 \\
&\leq \mathsf{MDL}(G) + \tfrac{1}{2} c \cdot K(X_1 \mid \mathbf{U}_1) + \textstyle\sum_{1 \leq i \leq m} \left[c \cdot K(X_i|\mathbf{U}'_i) - c \cdot K(X_i|\mathbf{U}_i) \right] \\
&\qquad\qquad \text{by assumption} \\
&\leq \mathsf{MDL}(G) + \textstyle\sum_{2 \leq i \leq m} \left[c \cdot K(X_i|\mathbf{U}'_i) - c \cdot K(X_i|\mathbf{U}_i) \right] \\
&\qquad\qquad \text{by Lemma 1} \\
&\leq \mathsf{MDL}(G)
\end{aligned}
$$

since $K(X_i|\mathbf{U}_i) \geq K(X_i|\mathbf{U}'_i)$. □

Proof of Corollary 2 Using Theorem 6, it suffices to show that

$$H_{\max}(X) + \sum_{Y \in \mathbf{Y}} H_{\max}(Y) \leq \frac{1}{2} c \cdot K(X|\mathbf{U})$$

for any $|\mathbf{U}| \geq d(\mathbf{Y}) + 1$. In this case, any $\mathbf{U}' \subset \mathbf{U}$ and any $\mathbf{V}' \subseteq \mathbf{V}$ will have at least as good of a score, i.e., when $|\mathbf{U}'| \leq d(\mathbf{Y})$. First, by assumption:

$$d(\mathbf{Y}) \leq \log_2 \left(\frac{2N}{c} \cdot \alpha \cdot (|\mathbf{Y}| + 1) \right) \leq d(\mathbf{Y}) + 1$$

and thus

$$\alpha N \cdot (|\mathbf{Y}| + 1) \leq \frac{1}{2} c \cdot 2^{d(\mathbf{Y})+1}. \tag{A.1}$$

Next,

$$
\begin{aligned}
& H_{\max}(X) + \sum_{Y \in \mathbf{Y}} H_{\max}(Y) \\
&= N \log_2 |X| + \sum_{Y \in \mathbf{Y}} N \log_2 |Y| \\
&\leq N \cdot (|X| - 1) + \sum_{Y \in \mathbf{Y}} N \log_2 |Y| && \text{by } \log_2 x \leq x - 1 \text{ for } x \geq 2 \\
&\leq N \cdot (|X| - 1) + \sum_{Y \in \mathbf{Y}} \alpha N \cdot (|X| - 1) && \text{by assumption} \\
&\leq \alpha N \cdot (|X| - 1) + \sum_{Y \in \mathbf{Y}} \alpha N \cdot (|X| - 1) && \text{by } \alpha \geq 1 \\
&= \alpha N \cdot (|\mathbf{Y}| + 1) \cdot (|X| - 1) \\
&\leq \frac{1}{2} c \cdot (|X| - 1) \cdot 2^{d(\mathbf{Y})+1} && \text{by Equation A.1} \\
&\leq \frac{1}{2} c \cdot (|X| - 1) \cdot \prod_{U \in \mathbf{U}} |U| && \text{by } |U| \geq 2 \text{ and } |\mathbf{U}| \geq d(\mathbf{Y}) + 1 \\
&= \frac{1}{2} c \cdot K(X|\mathbf{U})
\end{aligned}
$$

as desired. □