On Pruning with the MDL Score

Eunice Yuh-Jie Chen EYJCHEN@CS.UCLA.EDU
Arthur Choi AYCHOI@CS.UCLA.EDU
Adnan Darwiche DARWICHE@CS.UCLA.EDU
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

The space of Bayesian network structures is forbiddingly 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, we show that these techniques allow a state-of-the-art structure enumeration algorithm to scale to datasets with significantly more variables.

Keywords: Bayesian networks, structure learning, model selection, minimum description length.

1. Introduction

Learning the structure of a Bayesian network is a fundamental problem in artificial intelligence and machine learning, where one seeks a structure (i.e., DAG) that best explains a given dataset. In practice, learning a single optimal DAG may not be sufficient, especially when the dataset size is small and 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 and Tian, 2014; Chen et al., 2015, 2016). For example, using dynamic programming, Tian et al. (2010) can enumerate the 100-best networks for real-world datasets with 17 variables. Using heuristic search methods, Chen et al. (2015) can enumerate the $1,000$-best networks for real-world datasets with 23 variables, which is the current state-of-the-art. In this paper, we show how to extend the reach of systems like Chen et al. (2015) further, and allow it 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 that do not contain any of the k-most likely DAGs that we are interested in. 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

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; see `https://oeis.org/A003024`.

bounds generalize those proposed for the problem of learning a single optimal structure (Suzuki, 1996; Tian, 2000; De Campos and 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.

2. Technical Preliminaries and Related Work

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 to 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:

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

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 and Friedman (2009); Murphy (2012).

In the problem of enumerating the k-best DAGs, we simply want k DAGs with the smallest scores. One of the first approaches for solving this problem optimally was due to Tian et al. (2010), which extended a dynamic programming (DP) approach originally intended for learning a single optimal DAG (Koivisto and Sood, 2004; Singh and Moore, 2005; Silander and Myllymäki, 2006).

Another approach for enumerating the k-best DAGs is to encode the learning 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). The corresponding solution is then added as a constraint to the current ILP problem, whose new solution gives us the second best DAG. This process continues until we can enumerate each of the k-best DAGs.

The current state-of-the-art for enumerating the k-best DAGs is based on heuristic search methods such as A* (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).

2.2 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:

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

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. First, we seek to minimize the conditional entropy:

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

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). Finally, we remark that 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:

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

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]

2.3 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 and Koller, 2005; De Campos and Ji, 2011), at least for the case of learning a single optimal DAG.

Consider now a variable X and two candidate parent sets $\mathbf{U}$ and $\mathbf{U}'$, where $\mathbf{U} \subset \mathbf{U}'$. The following theorem identifies a general situation where we can guarantee that $\mathbf{U}'$ will never appear in a DAG minimizing a decomposable score (MDL or otherwise).

Theorem 1 (Teyssier and Koller (2005)) *Let $\mathcal{D}$ denote a dataset with a 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]

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 and Friedman (2009).
3. Although non-strict inequalities lead to more pruning, we use strict inequalities to simplify the discussion.

Consider a DAG G that contains $X\mathbf{U}$ as a family. We can replace the family $X\mathbf{U}$ with the smaller family $X\mathbf{U}'$ and obtain a new DAG G', since replacing $X\mathbf{U}$ with a smaller family will not introduce any directed cycles. Moreover, since the smaller family $X\mathbf{U}'$ has a better score, the new DAG G' also has 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 always obtain a better DAG with a strictly better score. We can thus eliminate the family $X\mathbf{U}$ from consideration when searching for an optimal Bayesian network.

The ability to prune many families in this manner is critical to the efficiency and scalability of structure learning (De Campos and 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.

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

Theorem 2 provides an upper bound on the number of parents $\mathbf{U}$ that a variable X needs to have in an optimal Bayesian network. 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 $\max_{\mathbf{U}} \mathsf{score}(X\mathbf{U}|\mathcal{D})$. Some approaches perform some pre-computations on these sub-problems, allowing for more efficient lookups to be performed during structure learning itself (De Campos and Ji, 2011; Yuan and 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 2nd 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. 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. That is, no DAG that contains $X\mathbf{U}$ as a family will be found in the first k best scoring DAGs. We remark that Theorem 3 was also implicitly used in the k-best enumeration algorithms of Tian et al. (2010) and Chen and Tian (2014), although we make the underlying concept explicit here.

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 $X\mathbf{U}$.*[4]

Theorems 3 and 4 are simple and intuitive, but they are of limited practical utility by themselves. However, we explicate them as they conceptually underlie more practical and sophisticated tests that we shall soon propose. Before we proceed, we first remark that Theorem 3 is based on reasoning locally about families $X\mathbf{U}$. In contrast, 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 (but without eliminating the optimal DAG). 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 we have $H_{\max}(X) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$, 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,

4. Note that 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).

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.

Theorem 6 *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 $k \leq 2^{d+1} - 1$.*

Like Theorem 2, Theorem 6 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. The above bound may not be tight, in that there may be a tighter bound on the number of parents, perhaps using a more complex test. Next, we will indeed consider some more sophisticated tests that can allow us to prune more candidate structures for the k-best enumeration problem.

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 the family $X\mathbf{U}$, Theorem 4 reasons globally about DAGs. The importance of this distinction is that it will allow us to be more aggressive in pruning 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 prune a family $X\mathbf{U}$ under the assumption that a DAG's topological ordering is fixed. Finally, we will generalize across all possible topological orderings, allowing us to prune $X\mathbf{U}$ as a candidate family for the original learning problem over all DAGs.

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 7 *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 7 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 8 *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*

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

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

If the conditions of Corollary 8 are met, then we can find a set of k-best DAGs where X has at most $d(\mathbf{Y}) - 1$ parents. A question remains: how do we find the smallest set $\mathbf{Y}$ that provides the tightest bound $d(\mathbf{Y}) - 1$? We discuss this 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:

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

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 and Herskovits, 1992). For networks of the scale that we are interested in, these local sub-problems $\max_{\mathbf{U}} \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$ are tractable in practice. Hence, we can also compute such a DAG $G^\star$.

Now, if we can find a set $\mathbf{Y} \subseteq \mathbf{X} \setminus X$, such that the DAG $G^\star$ meets the conditions of Corollary 8, then we need $d(\mathbf{Y}) - 1$ or fewer parents for variable X, to obtain the k-best DAGs that respect an ordering π. Essentially, we will have found k DAGs at least as good as $G^\star$ (the best DAG with family $X\mathbf{U}$ and respecting π), as in Theorem 7. Ideally, we want a smaller set $\mathbf{Y}$, leading to a tighter bound on the number of parents. However, there are exponentially many subsets $\mathbf{Y}$. Thus,

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

```
Data: dataset $\mathcal{D}$ over variables $\mathbf{X}$;
topological ordering $\pi$;
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
        $\mathbf{U}_i^* \leftarrow \text{argmin}_{\mathbf{U}_i \subseteq \pi_{1:i-1}} \mathsf{MDL}(X_i\mathbf{U}_i|\mathcal{D})$
        $(\mathbf{Y}, s) \leftarrow (\mathbf{Y} \cup X_i, s2^{|\mathbf{U}_i^*|})$
    if $k \leq s(2^{d(\mathbf{Y})} - 1)$ then
        return $d(\mathbf{Y}) - 1$
```

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
            $\mathbf{U}^* \leftarrow \text{argmin}_{\mathbf{U} \subseteq \mathbf{Z}'} \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$
            $(\mathbf{Y}', s') \leftarrow (\mathbf{Y} \cup Z, s2^{|\mathbf{U}^*|})$
        else $(\mathbf{Y}', s') \leftarrow (\mathbf{Y}, s)$
        if $k > s'(2^{d(\mathbf{Y}')} - 1)$ then
            insert $(\mathbf{Z}', Y', s')$ into $Q$
return $d(\mathbf{Y}_{\max}) - 1$
```

we first propose a simpler greedy method (which we shall again extend shortly). Since we are given a topological ordering of the variables π, we simply add variables one-by-one to $\mathbf{Y}$ in the reverse order of π, until the conditions of Corollary 8 are met.[5] 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.

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 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 1. 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 8, then it returns an upper bound of $d(\mathbf{Y}) - 1$. 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 then backtrack to the next unexplored branch of the permutation tree, and continue running

5. We use the reverse order since variables at the end of the order generally have larger families, which helps us enumerate more sub-DAGs, which in turn lets us obtain a tighter bound on the number of parents.

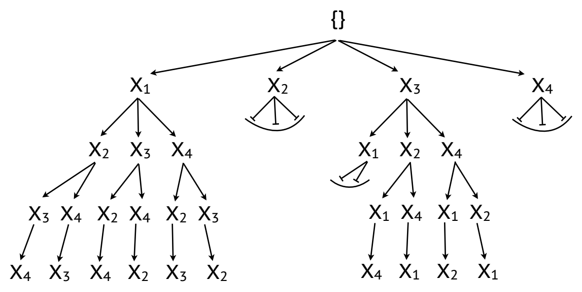

Figure 1: A (pruned) permutation tree over 4 variables: each path from the root to a node corresponds to a (partial) ordering π. The permutation spaces below the (partial) orderings $\langle X_2 \rangle$, $\langle X_3, X_1 \rangle$ and $\langle X_4 \rangle$ have been pruned here.

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.

Algorithm 1 at that point. By repeating 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 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

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.

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 and Lichman, 2013).[6] 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 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).[7][8] 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 prune list is a $158\times$ 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-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). We also report the timings of the enumeration algorithm, broken down into two parts (the total time is thus $T_h + T_{A*}$); see Chen et al. (2015) for details, although we remark that the majority of time (T_h) is spent evaluating the structure learning oracle, i.e., the heuristic function (Chen et al., 2015).

6. Discretized, and available at `http://urlearning.org/`

7. 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).

8. 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.

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 and ONR grant #N00014-15-1-2339.

References

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

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

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

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

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

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

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

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

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

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

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

M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *JMLR*, 5:549–573, 2004.

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

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

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

A. P. Singh and A. W. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, CMU-CALD-050106, 2005.

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

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

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

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

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

Appendix A. Proofs

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

Lemma 9 *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 $K(X|\mathbf{U})$, and since $|X| \geq 2$ for every variable X. ■

Theorem 10 (Tian, 2000) *Consider* $\mathbf{U}' \subset \mathbf{U}$. *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 Since $H_{\max}(X) \leq \frac{1}{2}c \cdot K(X \mid \mathbf{U})$, if $\mathbf{U}' \subset \mathbf{U}$ then $H_{\max}(X) \leq c \cdot K(X|\mathbf{U}) - c \cdot K(X|\mathbf{U}')$, by Lemma 9. By Theorem 10, $\mathsf{MDL}(X\mathbf{U}'|\mathcal{D}) \leq \mathsf{MDL}(X\mathbf{U}|\mathcal{D})$. ■

Proof of Theorem 6 By Theorem 5, it suffices to show $H_{\max}(X) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$ for $|\mathbf{U}| \geq d+1$. That is: $H_{\max}(X) = N \log_2 |X| \leq N(|X|-1) \leq \frac{1}{2}c \cdot (|X|-1) \cdot 2^{d+1} \leq \frac{1}{2}c \cdot K(X \mid \mathbf{U})$ ■

Proof of Theorem 7 Consider a DAG G, where X_i has parents $\mathbf{U}_i$ for $1 \leq i \leq m$ (i.e., let $\mathbf{Y} = \{X_2, \ldots, X_m\}$ where $m = |\mathbf{Y}| + 1$). Let G' denote the same DAG but where X_1 has parents $\mathbf{U}_1' \subset \mathbf{U}_1$ and X_i has parents $\mathbf{U}_i' \subseteq \mathbf{U}_i$ for $2 \leq i \leq m$. Then

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

Proof of Corollary 8 First we show that if $\alpha N(|\mathbf{Y}|+1)(|X|-1) \leq \frac{1}{2}c \cdot K(X|\mathbf{U}')$, then we have that $H_{\max}(X) + \sum_{Y \in \mathbf{Y}} H_{\max}(Y) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$. Note that

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

By Theorem 7, there exist $(2^{|\mathbf{U}|}-1) \cdot \prod_{Y\mathbf{V}, \text{ where } Y \in \mathbf{Y}} 2^{|\mathbf{V}|}$ many G' such that $\mathsf{MDL}(G') \leq \mathsf{MDL}(G)$. Finally, note that when $|\mathbf{U}| \geq d(\mathbf{Y})$, then $\alpha N(|\mathbf{Y}|+1)(|X|-1) \leq \frac{1}{2}c \cdot K(X|\mathbf{U})$. ■