Maximal Ancestral Graph Structure Learning via Exact Search

Rantanen, Kari

AUAI Press
2021


http://hdl.handle.net/10138/338682

unspecified
publishedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.
Abstract

Generalizing Bayesian networks, maximal ancestral graphs (MAGs) are a theoretically appealing model class for dealing with unobserved variables. Despite significant advances in developing practical exact algorithms for learning score-optimal Bayesian networks, practical exact algorithms for learning score-optimal MAGs have not been developed to-date. We develop here methodology for score-based structure learning of directed maximal ancestral graphs. In particular, we develop local score computation employing a linear Gaussian BIC score, as well as score pruning techniques, which are essential for exact structure learning approaches. Furthermore, employing dynamic programming and branch and bound, we present a first exact search algorithm that is guaranteed to find a globally optimal MAG for given local scores. The experiments show that our approach is able to find considerably higher scoring MAGs than previously proposed in-exact approaches.

1 INTRODUCTION

Exact approaches to Bayesian network structure learning [Bartlett and Cussens, 2017, van Beek and Hoffmann, 2015, Yuan and Malone, 2013] have flourished in recent years. Motivated by guarantees on learning globally optimal structures over a justified Bayesian score and the resulting gains in accuracy from limited number of observational data points, advances in scalability of exact structure learning have been achieved through harnessing different search techniques and optimality-preserving score pruning.

However, despite their popularity, Bayesian networks (BNs) offer only a rather limited independence and causal model. In particular, any causal inference based on BNs must assume causal sufficiency, which is rarely satisfied in real-world settings [Spirtes et al., 1993]. Thus, the demand for accurate structure learning in the presence of latent variables is not answered by the various exact approaches developed for BN structure learning (BNSL). Maximal ancestral graphs (MAGs), on the other hand, are a theoretically appealing generalization of BNs to account for unobserved variables and latent confounding [Richardson and Spirtes, 2002, Spirtes et al., 1993]. Since MAGs can represent marginalizations of BNs and retain many of their important properties, they have been extensively used in causal inference [Richardson and Spirtes, 2003, Zhang, 2008a, b, Jaber et al., 2019, Perkovic et al., 2017, Spirtes et al., 1993].

In contrast to BNs, for which significant advances in exact structure learning algorithms have been made, exact approaches to learning MAGs have not been developed, despite the potential gains in applicability. Several constraint-based learning algorithms for MAGs or their equivalence classes [Ramsey et al., 2012, Colombo and Maathuis, 2014, Zhang, 2008b, Claassen and Heskes, 2012, Bernstein et al., 2020, Claassen et al., 2013] are in-exact and use independence tests, which are not very accurate for large conditioning sets and relatively small sample sizes. As for score-based learning, [Tsirlis et al., 2018] recently presented a greedy score-based approach under the BIC score (a consistent scoring criterion for linear Gaussian parameterization of MAGs [Richardson and Spirtes, 2002]) in which the needed maximum likelihood estimates can be computed with the so-called residual iterative conditional fitting algorithm (RICF) [Drton et al., 2009]. This greedy search approach builds on earlier developments for GSMAG and MMHC [Tsamardinos and Tsamardinos, 2016, Tsamardinos et al., 2006]. Further studies in this direction include Chib and Constantinou [2017].

Note that while there is recent work on even more general model classes and settings; they are either inexact [Nowzohour et al., 2017, Améndola et al., 2020] or operate on arguably less justified objective functions [Rantanen et al., 2020, Triantafillou and Tsamardinos, 2015, Hyttinen et al., 2014] as BIC or Bayesian scores for discrete or linear Gaussian data [Geiger and Heckerman, 1994, Consonni and Rocca, 2012].
Definition 1 (MAG). A mixed graph \((V, E)\) over a set of nodes \(V\), and edge relation \(E = E_{\rightarrow} \cup E_{\leftrightarrow}\) consisting of both directed and bidirected edges is a (directed) maximal ancestral graph (MAG) if

i. the graph does not contain any directed or almost directed cycles (ancestrality); and

ii. there is no inducing path between any two non-adjacent vertices (maximality).

Ancestors of \(v_i\) are the nodes \(v_j\) from which there is a directed path \(v_j \rightarrow \cdots \rightarrow v_i\) to the node \(v_i\). A directed cycle is a directed path \(v_i \rightarrow \cdots \rightarrow v_j\) with a bi-directed edge \(v_i \leftrightarrow v_j\). For example, the mixed graph shown in \(\text{Figure 1}(a)\) does not include an almost directed cycle. A collider on a path has the adjacent edges on the path pointing towards the node, i.e., \(\leftrightarrow v_i \leftrightarrow v_i \leftrightarrow v_i \leftrightarrow v_i \leftrightarrow v_i\). An inducing path is a path \(v_i \cdots v_j\) on which every vertex (except for the endpoints \(v_i, v_j\)) is a collider on the path and every collider is an ancestor of an endpoint \(v_i, v_j\) of the path. The path \(x \leftrightarrow w \leftrightarrow y \leftrightarrow q\) in \(\text{Figure 1}(b)\) is an inducing path. The graphical separation criterion for MAGs is \(m\)-separation, which is essentially \(d\)-separation after bi-directed edges \(\leftrightarrow\) are replaced with structures \(l \rightarrow \) explicitly marking the unobserved \(l\). This gives intuition to inducing paths, nodes (e.g., \(x, q\) in \(\text{Figure 1}(b)\)) connected by an inducing path cannot be \(m\)-separated by any conditioning set.

The following lemma further characterizes inducing paths in the context of MAGs.

Lemma 1. Every inducing path of length \(n \geq 3\) nodes in a MAG between \(x\) and \(y\) is of the form \(x \leftrightarrow z_1 \leftrightarrow \cdots \leftrightarrow z_{n-2} \leftrightarrow y\), where each \(z_i\) is an ancestor of either \(x\) or \(y\). In particular, \(z_1\) is an ancestor of \(y\) and \(z_{n-2}\) is an ancestor of \(x\).

Proof. Let \(G\) be a MAG with an inducing path between \(x\) and \(y\) going through the nodes \(z_1, \ldots, z_{n-2}\). The edges between each \(z_i, z_{i+1}\) must be bi-directed \(z_i \leftrightarrow z_{i+1}\) (otherwise \(z_i\) or \(z_{i+1}\) would not be a collider). Due to maximality, \(G\) has to contain either \(x \leftrightarrow y, x \leftrightarrow y\) or \(y \leftrightarrow x\). Node \(z_1\) cannot be an ancestor of \(x\) since that would introduce a directed cycle \(x \rightarrow z_1 \rightarrow \cdots \rightarrow x\) or an almost directed cycle \(x \leftrightarrow z_1 \rightarrow \cdots \rightarrow x\). Thus \(z_1\) is an ancestor of \(y\) and, by symmetry, \(z_{n-2}\) is an ancestor of \(x\). There cannot be an inducing path of the form \(x \leftrightarrow z_1 \leftrightarrow \cdots \leftrightarrow z_{n-2} \leftrightarrow y\) since otherwise \(z_1\) would not be a collider. The inducing path can also not be of the form \(x \rightarrow z_1 \leftrightarrow \cdots \leftrightarrow z_{n-2} \leftrightarrow y\) since \(z_1\) is an ancestor of \(y\) and thus otherwise there would be an almost directed cycle \(z_1 \rightarrow \cdots \rightarrow y \leftrightarrow z_{n-2} \rightarrow \cdots \rightarrow x \rightarrow z_1\). This same logic can be applied symmetrically to \(y\). Therefore the inducing path must be of the form \(x \leftrightarrow z_1 \leftrightarrow \cdots \leftrightarrow z_{n-2} \leftrightarrow y\). \(\square\)

We define MAGSL as follows.

Problem 1 (MAGSL). Find a MAG \(G^* = (V, E^*)\) such that

\[
G^* \in \text{argmax}_{G \in \mathcal{G}} s(G),
\]

where \(\mathcal{G}\) denotes the class of MAGs and \(s\) gives a score for each MAG \(G\).

Note that the highest-scoring MAG is a representative of its equivalence class; the common structural features of the members of the equivalence class can then be further inspected by other methods such as FCI [Spirtes et al. 1993].
We use the BIC scoring function \cite{Richardson2002,Tsirlis2018}
\[ s(G) = \ln L_G(\hat{\theta}) - (|E| + 2|V|)/2 \cdot \ln N, \]
where \( N \) is sample size, \( L_G \) is the multivariate Gaussian likelihood function and \( \hat{\theta} \) are maximum likelihood parameters for the linear Gaussian model over \( G \). The number of parameters accounts here for the mean and variance for each node, and one coefficient per directed or bi-directed edge.

BIC is asymptotically consistent scoring criterion for MAGs \cite{Richardson2002,Tsirlis2018}. Furthermore, since Markov-equivalent MAGs can represent the same multivariate Gaussians distributions \cite{Richardson2002,Corollary8.19}, Markov-equivalent MAGs share adjacencies \cite{SpirtesRichardson1996}. The score of a MAG in BNSL as we need to consider not only c-components but also bi-directed edges. Further, the intricate properties of the model class imply that an iterative method (with strong convergence guarantees) is needed for local score computation and that special care should be taken when pruning local scores in order not to jeopardize exactness.

### 3.1 Computing a Local Score

We compute local scores as \( s(C_i, \text{pa}_G(C_i)) = s(G) - s(G') \), where \( G \) is a MAG with the local c-component \( C_i \) and its parents; \( G' \) is the empty MAG over all parents of \( C_i \) not in \( C_i \). The scores \( s(G') \) are cached to speed up scoring, similarly as in BNSL and in M3HC \cite{TianPearl2002,Tsirlis2018}. The maximum likelihood estimates for the BIC score are computed using residual iterative conditional fitting (RICF) of \cite{Drton2009}. The strong convergence properties of RICF state that from a given starting point RICF converges to an accumulation point, all of which give the same value for the likelihood function \cite{Drton2009,Theorem13}. There can be several local optima especially for limited sample sizes \cite{Drton2004}. For any c-component of size 1, RICF converges in one step and the produced local scores exactly equal the corresponding BNSL local scores computed with Gobnilp \cite{Bartlett2017}.

### 3.2 Computing All Local Scores

Suppose \( s(C, P) \) is a local score where \( C \) is a c-component over nodes \( x_1, \ldots, x_k \) and each \( \text{pa}(x_i) \in P \) describes the parents of \( x_i \). We say that \( z \in \text{pa}(x_i) \) is an internal parent if \( z \in C \) and otherwise an external parent. If the c-component includes inducing paths or (almost) directed cycles, i.e., violations of the MAG properties, it cannot be a part of an optimal MAG. Thus, we call \( s(C, P) \) a valid local score if and only if the local c-component \( C \) with the parents in \( P \) is a MAG itself. A valid score will always remain valid if we only add external parents to it.

Every valid local score \( s(C, P) \) over observed variables \( V \), where the number of nodes in \( C \) is at most \( c \) and number of parent relations is limited by \( \sum_{x \in C} |\text{pa}(x)| \leq p \), can be computed as follows.

1. For all \( Y \subseteq V \) with \( 1 \leq |Y| \leq c \), iterate over every possible MAG over nodes \( Y \) in which every node belongs to one and the same c-component. This corresponds to iterating all valid local scores \( S_{\text{internal}} \) that only contain internal parents. Ignore scores in which the total number of parent relations is more than \( p \).
2. Iterate over each local score in \( S_{\text{internal}} \) and go over all the ways of adding external parents to them (as long as the parent limit \( p \) is not exceeded), forming \( S_{\text{external}} \).
3. Return the local scores in \( S_{\text{internal}} \) and \( S_{\text{external}} \) which cannot be pruned.
We will now propose two rules which provide a way of pruning MAGs: the extension of standard score pruning from BNSL to MAGSL would prune score \( s(C', P') \) if there is a score \( s(C, P) \) such that (1) \( s(C, P) > s(C', P') \) and (2) the mixed graph induced by \((C, P)\) is a subgraph of the mixed graph induced by \((C', P')\). This is based on the following intuition: if the optimal graph would include \((C', P')\), then replacing it with \((C, P)\) would give a higher scoring solution. Since, the graph with \((C', P')\) is acyclic and only edges are taken out, no cycles can be induced.

However, such a pruning rule is correct only if maximum \( c \)-component size employed is \( c \leq 3 \), as removing edges from a local \( c \)-component may end up violating the maximality property (recall that all nodes connected by an inducing path must be adjacent). Consider Figure 2. The local \( c \)-component in \((b)\) included in the MAG in \((a)\) cannot be replaced by the local \( c \)-component in \((c)\), as the endpoints \( x, w \) of the inducing path \( x \leftrightarrow w \) would no longer be adjacent. Thus pruning the local score of the local \( c \)-component in \((b)\) based on a higher scoring local \( c \)-component \((c)\) may jeopardize finding optimal solutions (such as the MAG in \((a)\)).

We will now propose two rules which provide a way of score pruning in MAGSL for any choice of \( c \) in a way that is guaranteed to maintain an optimal solution. Note that for MAGs, local scores can be pruned based on a set of other local scores, albeit with certain technical conditions.

For presentation of the rules, we use the concept of maximality-preserving pair of nodes in a MAG which intuitively can only have a trivial inducing path between them (i.e., an edge). If a MAG is non-maximal, it is not due to a missing edge between any maximality-preserving pair.

**Definition 3.** Let \( x \) and \( y \) be nodes in graph \( G \). We call \( \{x, y\} \) a maximality-preserving pair in \( G \) if:

1. There is no path of bidirected edges between \( x \) and \( y \) containing more than 3 nodes; or
2. Each parent set \( \text{pa}_G(x) \subseteq \text{pa}_G(y) \) or \( \text{pa}_G(y) \subseteq \text{pa}_G(x) \).

The correctness of the following pruning rules will be proven later in this section. The first rule is a non-trivial generalization of the standard pruning in BNSL.

**Pruning Rule 1.** Let \( C \) and \( C' \) be \( c \)-components over a same set of nodes and let their parent set collections be \( P \) and \( P' \), respectively. The score \( s(C', P') \) can be pruned if:

1. \( s(C, P) \geq s(C', P') \);
2. Each parent set \( \text{pa}(x) \subseteq \text{pa}(y) \) or \( \text{pa}(y) \subseteq \text{pa}(x) \);
3. If \( x \leftrightarrow y \) is missing from \( C' \), then it is missing from \( C \) as well; and
4. If \( x \leftrightarrow y \) exists in \( C' \) but not in \( C \), then \( \{x, y\} \) must be a maximality-preserving pair in \( C \).

The second rule generalizes the first by also considering partitionings of a local \( c \)-component \( C \) into a set of smaller local \( c \)-components \( C_1, \ldots, C_n \).

**Pruning Rule 2.** Let \( C_1, \ldots, C_n \) and \( C' \) be \( c \)-components with parent set collections \( P_1, \ldots, P_n \) and \( P' \), respectively. Suppose the nodes in \( C_1, \ldots, C_n \) partition the nodes in \( C' \) to distinct subsets. The score \( s(C', P') \) can be pruned if:

1. \( \sum_{i=1}^{n} s(C_i, P_i) \geq s(C', P') \);
2. Each parent set \( \text{pa}(x) \subseteq \text{pa}(y) \) or \( \text{pa}(y) \subseteq \text{pa}(x) \) (if \( x \) was partitioned into \( C_i \));
3. If \( x \leftrightarrow y \) is missing from \( C' \), then it is not featured in any \( C_i \), either; and
4. If \( x \leftrightarrow y \) exists in \( C' \) but not in some \( C_i \), then either \( x \) or \( y \) was not partitioned into \( C_i \) or \( \{x, y\} \) is a maximality-preserving pair in \( C_i \).

Figure 3 shows an example of applying Pruning Rule 2. The \( c \)-component over \( x, w, y, z \) with external parent \( q \) for \( x \) and \( z \) (left) can be pruned if the \( c \)-components over \( x, y \) and \( w, z \) (center, right) have a higher sum of scores. Note that removing edges \( x \leftrightarrow w \) and \( y \leftrightarrow z \) is permitted since the absence of inducing paths between \( x, w \) and \( y, z \), respectively, is guaranteed in any resulting graph.

To prove the correctness of the score pruning rules, we need additional theory, which builds on important earlier work on...
MAgs [Richardson and Spirtes 2002, Zhang and Spirtes 2005]. The first lemma characterizes the types of inducing paths possible in MAGs.

Lemma 2. Let x and y be non-adjacent nodes in a MAG G. If \( \text{pa}_G(x) \subseteq \text{pa}_G(y) \), then there can be no inducing path between x and y.

Proof. Assume that \( \text{pa}_G(x) \subseteq \text{pa}_G(y) \). This means that any ancestor of x in G is also an ancestor of y. By Lemma 1 an inducing path between x and y would imply the existence of the edge \( z_{n-2} \leftrightarrow y \) where \( z_{n-2} \) is an ancestor of x. Clearly \( z_{n-2} \) cannot be also an ancestor of y since this would introduce an almost directed cycle. Thus there is no inducing path between x and y in G.

The next three lemmas identify which edges can be removed without making nodes connected by inducing paths non-adjacent, thus preserving maximality (Definition 1).

Lemma 3. A MAG remains a MAG after removing an edge \( x \leftrightarrow y \) if the longest inducing path between x and y has a length less than 4 nodes.

Proof. The case where the only inducing path between x and y has length 2 nodes is trivial. Moreover, there cannot be an inducing path of length 3 nodes in a MAG, as it would be of the form \( x \leftrightarrow z \leftrightarrow y \) where z is an ancestor of x or y (Lemma 1), which would form an almost directed cycle through x or through y.

Lemma 4. If \( \text{pa}_G(x) \subseteq \text{pa}_G(y) \) for nodes x and y in a MAG G, then G will remain a MAG after removing edge \( x \leftrightarrow y \).

Proof. A direct consequence of Lemma 2.

Lemma 5. A MAG remains a MAG after removing an arbitrary edge \( x \rightarrow y \).

Proof. Let G be a MAG and let \( G' \) be G after removing an edge \( x \rightarrow y \). Clearly, \( G' \) does not introduce new cycles, almost directed cycles or inducing paths. \( G' \) could only be non-maximal if there were an inducing path between x and y in \( G' \). This would mean that in the original G we had an edge \( z_{n-2} \leftrightarrow y \) where \( z_{n-2} \) is an ancestor of x (Lemma 1). However, since G has the edge \( x \rightarrow y \), this would mean that G contains an almost directed cycle \( x \rightarrow y \leftrightarrow z_{n-2} \rightarrow \cdots \rightarrow x \), which is not possible. Hence there is no inducing path between x and y, and \( G' \) is maximal.

We are finally ready to establish the correctness of Pruning Rule 2 (and thus also of Pruning Rule 1).

Algorithm 1 A dynamic programming algorithm for computing upper bounds, i.e., score-optimal ancestral solutions.

```
1: function UB( nodes \( U \), reach \( \text{R}_U \), almost reach \( \text{A}_U \) )
2: if \( U = \emptyset \) then return 0
3: Let \( u_0 \in U \): lexicographically least node in U.
4: Let \( C \): the possible c-component and parent set combinations \( (C, \text{pa}(C)) \) such that \( u_0 \in C \subseteq U \).
5: Remove each \((C, \text{pa}(C)) \in C \) that would introduce an (almost) directed cycle given \( \text{R}_U, \text{A}_U \).
6: Let \( m \leftarrow -\infty \).
7: for each \((C, \text{pa}(C)) \in C \) do
8: Let \( \text{R}_{U \setminus C} \) and \( \text{A}_{U \setminus C} \) be updated versions of \( \text{R}_U \) and \( \text{A}_U \) given \((C, \text{pa}(C)) \).
9: \( \hat{m} \leftarrow \text{UB}(U \setminus C, \text{R}_{U \setminus C}, \text{A}_{U \setminus C}) \).
10: \( m \leftarrow \max(m, s(C, \text{pa}(C)) + \hat{m}) \).
return m
```

Theorem 1. Let \( C' \) and \( C_1, \ldots, C_n \) be c-components with parent sets \( P' \) and \( P_1, \ldots, P_n \) respectively. Suppose the nodes in \( C_1, \ldots, C_n \) partition the nodes in \( C' \) to distinct subsets. Score \( s(C', P') \) can be pruned if:

1. \( \sum_i s(C_i, P_i) \geq s(C', P') \), and
2. for all \( x \in C_i \), \( \text{pa}(x) \in P_i \) is a subset of the corresponding \( \text{pa}(x) \in P' \), and
3. if \( x \leftrightarrow y \notin C' \) then \( x \leftrightarrow y \notin C_i \) for all \( x, y \in C_i \), and
4. for \( x \leftrightarrow y \in C' \) and \( x \leftrightarrow y \notin C_i \) for some \( x, y \in C_i \), then \( \{x, y\} \) must be maximality-preserving in \( C_i \).

Proof. Suppose a MAG \( G' \) includes the local c-component \( C' \) with parent sets \( P' \). We can replace this component with \( C_1, \ldots, C_n \) and \( P_1, \ldots, P_n \), to form a valid MAG G: (1) G does not contain any edges which are not in \( G' \). (2) A MAG remains a MAG after removing edges \( x \rightarrow y \) in \( G' \) but not in G sequentially (Lemma 2). (3) Suppose that G is missing an edge \( x \leftrightarrow y \in G' \). If \( x \notin C_i \) or \( y \notin C_i \) for all \( C_i \), there are no inducing path between x and y in G by Lemma 1 and so \( x \leftrightarrow y \) is allowed to be absent. Otherwise, if \( x, y \in C_i \) for some \( C_i \), we require \( \{x, y\} \) to be a maximality-preserving pair in \( C_i \). Therefore we have either either (a) \( \text{pa}_G(x) \subseteq \text{pa}_G(y) \) or (b) all paths of bidirectional edges between x and y have less than 4 nodes. By Lemmas 4 and 5 resp., the edge \( x \leftrightarrow y \) may be absent in either case. As \( \sum_i s(C_i, P_i) \geq s(C', P') \), we do not need \( P' \) and \( C' \) to form an optimal MAG.
Algorithm 2 Exact branch-and-bound algorithm for finding score-optimal maximal and ancestral solutions $S^*$ (MAGs).

1. \textbf{function} \textsc{search}(nodes $U$, partial solution $S$)
2. \hspace{1em} if $S$ is non-maximal then \textbf{return}
3. \hspace{1em} if $U = \emptyset$ then
4. \hspace{2em} if $s(S') < s(S)$ then $S' \leftarrow S$
5. \hspace{1em} \textbf{return}
6. $R_U, A_U \leftarrow$ reachability in $S$
7. if $s(S) + \text{ub}(U, R_U, A_U) \leq S^*$ then \textbf{return}
8. Let $\hat{S}$: $S$ extended with the UB solution.
9. if $\hat{S}$ is maximal then
10. \hspace{1em} if $s(S') < s(\hat{S})$ then $S' \leftarrow \hat{S}$
11. \textbf{return}
12. Let $u_0 \in U$: lexicographically least node in $U$.
13. Let $C$: all possible c-component and parent set combinations $(C, \text{pa}(C))$ such that $u_0 \in C \subseteq U$.
14. Remove each $(C, \text{pa}(C)) \in C$ that would introduce an (almost) directed cycle given $R_U, A_U$.
15. for each $(C, \text{pa}(C)) \in C$ do
16. \hspace{1em} Let $S'$ be $S$ with $(C, \text{pa}(C))$ added to it.
17. \textsc{search}(U \setminus C, S')

To use this, we keep track of the reach $R_U$ and almost reach $A_U$ for any subset of nodes $U \subseteq V$. A node $x$ reaches a node $y$ in a graph $G$ if there exists a directed path from $x$ to $y$ in $G$. Moreover, $x$ almost reaches $y$ in $G$ if there exists nodes $v_1, \ldots, v_n$ with $x = v_1, y = v_n$ such that the edge $v_j \leftrightarrow v_{j+1}$ exists in $G$ for some $j \in \{1, \ldots, n-1\}$ and for all other $i \in \{1, \ldots, n-1\}$ the edge $v_i \rightarrow v_{i+1}$ is in $G$. $R_U (A_U)$ implies an (almost) directed cycle if a node can (almost) reach itself.

4.2 AN ALTERNATIVE ASP-BASED APPROACH

Motivated by the successes of declarative programming techniques, in particular answer set programming (ASP) [Gebser et al. 2012], for learning globally optimal structures from various graphical model classes and settings [Hyytininen et al. 2014; Sonntag et al. 2015; Magliacane et al. 2016; Forré and Moor 2018; Zhalama et al. 2019], we developed a first ASP-based approach to MAGSL as a comparative baseline exact approach. As the approach turns out to show weaker performance than the branch-and-bound approach, we provide the detailed encoding as part of our code package. In short, the ASP encoding is based on representing in a natural way the search space of MAGs. This requires encoding the ruling out of directed and almost directed cycles as well as confirming that inducing paths only appear between adjacent nodes. Existence of specific edges is implied by the mean.

Algorithm \textsc{search} uses dynamic programming [Koivisto and Sood 2004; Silander and Myllymäki 2006; Tian and He 2009] to compute an upper bound for the score of an optimal solution for nodes $U$ such that, together with $R_U, A_U$, (almost) directed cycles are not formed (Line 5). The algorithm iterates over the local c-components $C_i$ containing a node $u_0$ (Line 8-10). This solves a relaxation of the MAGSL optimization problem as solutions need only be ancestral: inducing paths between non-adjacent nodes are allowed. We cache the solutions of each $\text{ub}(U, R_U, A_U)$ to avoid solving subproblems repeatedly. For efficiency, we do not store the almost reach from node $x$ to node $y$ if $x$ can reach $y$.

Algorithm \textsc{search} uses branch and bound [Suzuki 1996; Tian 2000; Rantanen et al. 2017] to find an optimal, maximal and ancestral solution $S^*$, by extending the partial solution $S$ over $V \setminus U$ to also cover the remaining nodes $U$. The algorithm is invoked by \textsc{search}(V, S_0) where $S_0$ contains no nodes and $S^*$ is initialized to a valid MAG (e.g. the empty MAG). At each step, we ensure that $S$ is maximal by checking that there are no inducing paths between any unadjacent $x, y \in V \setminus U$ (Line 2). If there are no remaining nodes and $S$ has a better score than $S^*$, we update $S^*$ (Line 4). Otherwise, if improving $S^*$ by assigning the remaining nodes $U$ to $S$ is impossible, we backtrack in the search (Line 7). On Line 8 we create $\hat{S}$ by extending $S$ with the upper bound solution obtained from Algorithm \textsc{search} if $\hat{S}$ is maximal, we backtrack and also update $S^*$ if $\hat{S}$ has better score.
chosen c-components, and each node is required to be con-
tained in exactly one c-component. The objective of finding
a best-scoring MAG structure is represented as weighted
soft constraints on the input c-component candidates.

5 EMPIRICAL EVALUATION

We empirically evaluate the efficiency of the score compu-
tation approach, and our branch-and-bound algorithm (“BB”)
against competing learning algorithms in terms of runtime
efficiency and quality of the MAGs found. We implemented
the local score computation and Algorithms 1 and 2 in C++.
The score computation software includes an efficient C++
implementation of the RICF algorithm as well as the score
pruning. The experiments were run on on 2.83-GHz Intel
Xeon E5440 nodes under 100-GB memory limit.

Data We generated synthetic data by sampling MAGs over
n = 7..12 nodes with expected degree is 2 or 3 and the
presence of directed and bi-directed edges equally proba-
ble. The coefficients and covariances for the directed and
bidirected edges were drawn uniformly from ±[0.2, 0.8].
Variances of disturbances were drawn as |0.5 · s| + 1, with
s ∼ N(0, 1). Rejection sampling was used to ensure the
models were valid linear Gaussian MAGs. For each model
we generated N = 200 samples. In addition to synthetic
MAGs, we use four Gaussian Bayesian network benchmarks
from the bnlearn network repository: ecoli70, magic-niab,
magic-irri and arth150 [Scutari, 2010]. We sampled from
N = 100 to N = 1600 samples from each benchmark.
To inspect structure learning in the presence of latent vari-
able, we marginalized the data to datasets of n = 7..11
highly correlated variables, reducing the set of variables by
repeatedly removing the variable for which the sum of the
absolute values of correlations to the remaining variables
was minimized.

Score Computation and Pruning We first look at the per-
formance of the local score computation part on the syn-
thetic data. Figure 4 shows the number of local scores pro-
duced for different numbers of nodes and c-component size
limits without (blue x) and with pruning (box plot). Note
that for c = 2 we use parent limit p = 8 and for c = 3 we
use p = 4, as these are still feasible choices. We observe
that increasing the c-component size limit from 2 to 3 re-
results in substantial increase in the number of local scores.
The increase in the number of local scores for each addi-
tional node is perhaps more modest. There is considerable
instance-specific variation on the number local scores after
pruning. Figure 5 shows the runtime for local score com-
putation for different configurations. The runtime grows
exponentially with the number of nodes but we can compute
local scores on a single thread for up to 12 nodes in 10
hours. This is inline with BNSL, also there the computation
of local scores may take considerable amount of time.

Runtime Performance Figure 4 left and middle show runtimes (sorted by increasing runtime for each line) of
our branch and bound under the c-component size limits
c = 2 and c = 3 on the synthetic data. Within approximately
1 hour, the approach scales up to 8-variable instances for
c = 2, 3. For c = 2 we can solve many instances with 9-12
nodes within the time limit of 10 hours. Runtime comparison
of BB with the alternative ASP-based approach (Section 4.2)
using the state-of-the-art Clingo ASP solver [Gebser et al.,
2012] is shown in Figure 6 right for the benchmark datasets
marginalized to 7-node instances. BB is considerably faster
than the ASP approach. Table 1(c = 2) and Table 2(c =
3, 4) show further per-instance runtimes of our BB approach
on the benchmark instances. There is some variation to the
runtimes depending on the dataset. BB scales best on magic-
irri and magic-niab, solving many of the 10-node instances
for c = 2 within 10 hours.

Quality of Solutions We evaluate the performance of our
exact branch-and-bound search for MAGSL in terms of solu-
tion quality against competing in-exact approaches: the hill-
climbing-based in-exact MAGSL algorithm M3HC [Tsirlis
et al., 2018], the constraint-based FCI [Zhang, 2008b
Spirtes et al., 1993] and its variant GFCI [Ogarrio et al.,
2018].
Figure 6: Runtimes: BB for $c = 2$ (left) and $c = 3$ (middle) on synthetic data; BB vs ASP on benchmark data (right).

Table 1: Results on the benchmark datasets.

<table>
<thead>
<tr>
<th>Benchmark dataset</th>
<th>Time N BC $c = 2$</th>
<th>BB N BC $c = 2$</th>
<th>BN</th>
<th>FCI</th>
<th>$\pm$ bootstrap</th>
<th>GFCI $\pm$ bootstrap</th>
<th>MHC</th>
<th>$\pm$ bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>magic-irri 100 8 10h</td>
<td>119717</td>
<td>110728.61</td>
<td>110728.61</td>
<td>110728.61</td>
<td>110728.61</td>
<td>110728.61</td>
<td>110728.61</td>
<td>110728.61</td>
</tr>
<tr>
<td>magic-irri 200 8</td>
<td>4215</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
</tr>
<tr>
<td>magic-irri 400 8</td>
<td>11801</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
</tr>
<tr>
<td>magic-irri 800 8</td>
<td>2421</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
</tr>
<tr>
<td>magic-irri 100 9</td>
<td>160</td>
<td>-1256.84</td>
<td>-1256.84</td>
<td>-1256.84</td>
<td>-1256.84</td>
<td>-1256.84</td>
<td>-1256.84</td>
<td>-1256.84</td>
</tr>
<tr>
<td>magic-irri 200 9</td>
<td>4215</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
<td>2396.24</td>
</tr>
<tr>
<td>magic-irri 400 9</td>
<td>11801</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
<td>-4677.92</td>
</tr>
<tr>
<td>magic-irri 800 9</td>
<td>2421</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
<td>-9440.41</td>
</tr>
<tr>
<td>magic_irri 100 10</td>
<td>1395</td>
<td>-1352.6</td>
<td>-1352.6</td>
<td>-1352.6</td>
<td>-1352.6</td>
<td>-1352.6</td>
<td>-1352.6</td>
<td>-1352.6</td>
</tr>
<tr>
<td>magic_irri 200 10</td>
<td>12354</td>
<td>2665.85</td>
<td>2665.85</td>
<td>2665.85</td>
<td>2665.85</td>
<td>2665.85</td>
<td>2665.85</td>
<td>2665.85</td>
</tr>
<tr>
<td>magic_irri 400 10</td>
<td>33771</td>
<td>-5209.08</td>
<td>-5209.08</td>
<td>-5209.08</td>
<td>-5209.08</td>
<td>-5209.08</td>
<td>-5209.08</td>
<td>-5209.08</td>
</tr>
<tr>
<td>magic_irri 800 10</td>
<td>63710</td>
<td>-9788.63</td>
<td>-9788.63</td>
<td>-9788.63</td>
<td>-9788.63</td>
<td>-9788.63</td>
<td>-9788.63</td>
<td>-9788.63</td>
</tr>
<tr>
<td>magic_irri 1600 10</td>
<td>20855</td>
<td>-16798.88</td>
<td>-16789.5</td>
<td>-16789.5</td>
<td>-16789.5</td>
<td>-16789.5</td>
<td>-16789.5</td>
<td>-16789.5</td>
</tr>
<tr>
<td>magic-nab 100 8</td>
<td>10h</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
<td>&lt;1s</td>
</tr>
<tr>
<td>magic-nab 200 8</td>
<td>3s</td>
<td>-4850.27</td>
<td>-4850.27</td>
<td>-4850.27</td>
<td>-4850.27</td>
<td>-4850.27</td>
<td>-4850.27</td>
<td>-4850.27</td>
</tr>
<tr>
<td>magic-nab 400 8</td>
<td>2s</td>
<td>-4110.33</td>
<td>-4110.33</td>
<td>-4110.33</td>
<td>-4110.33</td>
<td>-4110.33</td>
<td>-4110.33</td>
<td>-4110.33</td>
</tr>
<tr>
<td>magic-nab 1600 9</td>
<td>11h</td>
<td>-10338.57</td>
<td>-10338.57</td>
<td>-10338.57</td>
<td>-10338.57</td>
<td>-10338.57</td>
<td>-10338.57</td>
<td>-10338.57</td>
</tr>
<tr>
<td>magic-nab 1600 10</td>
<td>21h</td>
<td>-19712.27</td>
<td>-19712.27</td>
<td>-19712.27</td>
<td>-19712.27</td>
<td>-19712.27</td>
<td>-19712.27</td>
<td>-19712.27</td>
</tr>
<tr>
<td>magic-nab 1600 11</td>
<td>11h</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
</tr>
<tr>
<td>magic-nab 1600 12</td>
<td>2h</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
<td>-9942.23</td>
</tr>
</tbody>
</table>

[2016] that combines score-based and constraint-based learning. We employ the FCI and GFCI implementation from the causal-cmd v1.2.1 package with Fisher’s test $\alpha = 0.05$ and semi-bic-score. We directly obtained a single representative MAG of the equivalence class returned by these algorithms. As the other methods are considerably faster per a single run (finishing often within minutes), we also bootstrapped the data 50 times and then report the best result based on score w.r.t. the original data. We also report (as “BN”) the scores of the optimal Bayesian networks found by using Gobnilp [Bartlett and Cussens, 2017]. Figure 7 shows the Bayes factors of the found MAGs with re-
### Table 2: Results on the benchmark datasets.

<table>
<thead>
<tr>
<th>Benchmark dataset</th>
<th>Time</th>
<th>BIC score</th>
</tr>
</thead>
<tbody>
<tr>
<td>arth150 10 7</td>
<td>528s</td>
<td>-224.74</td>
</tr>
<tr>
<td>arth150 20 7</td>
<td>1428s</td>
<td>-377.96</td>
</tr>
<tr>
<td>arth150 40 7</td>
<td>1426s</td>
<td>-722.47</td>
</tr>
<tr>
<td>arth150 80 7</td>
<td>1447s</td>
<td>-1380.36</td>
</tr>
<tr>
<td>arth150 160 7</td>
<td>1771s</td>
<td>-2069.91</td>
</tr>
<tr>
<td>ecoli70 10 7</td>
<td>519s</td>
<td>-602.8</td>
</tr>
<tr>
<td>ecoli70 20 7</td>
<td>631s</td>
<td>-1100.74</td>
</tr>
<tr>
<td>ecoli70 40 7</td>
<td>748s</td>
<td>-2321.64</td>
</tr>
<tr>
<td>ecoli70 80 7</td>
<td>914s</td>
<td>-4611.14</td>
</tr>
<tr>
<td>ecoli70 160 7</td>
<td>980s</td>
<td>-9326.18</td>
</tr>
<tr>
<td>magic-irri 10 7</td>
<td>17s</td>
<td>-977.01</td>
</tr>
<tr>
<td>magic-irri 20 7</td>
<td>51s</td>
<td>-1951.54</td>
</tr>
<tr>
<td>magic-irri 40 7</td>
<td>52s</td>
<td>-3802.84</td>
</tr>
<tr>
<td>magic-irri 80 7</td>
<td>198s</td>
<td>-7520.48</td>
</tr>
<tr>
<td>magic-irri 160 7</td>
<td>259s</td>
<td>-15051.99</td>
</tr>
<tr>
<td>magic-irri 10 8</td>
<td>1144s</td>
<td>-1099.59</td>
</tr>
<tr>
<td>magic-irri 20 8</td>
<td>312s</td>
<td>-2169.11</td>
</tr>
<tr>
<td>magic-irri 40 8</td>
<td>493s</td>
<td>-4228.14</td>
</tr>
<tr>
<td>magic-irri 80 8</td>
<td>2028s</td>
<td>-8358.48</td>
</tr>
<tr>
<td>magic-irri 160 8</td>
<td>10ch</td>
<td>-16718.88</td>
</tr>
<tr>
<td>magic-niab 10 7</td>
<td>3s</td>
<td>-1616.51</td>
</tr>
<tr>
<td>magic-niab 20 7</td>
<td>3s</td>
<td>-1616.51</td>
</tr>
<tr>
<td>magic-niab 40 7</td>
<td>2s</td>
<td>-3593.07</td>
</tr>
<tr>
<td>magic-niab 80 7</td>
<td>6s</td>
<td>-6618.23</td>
</tr>
<tr>
<td>magic-niab 160 7</td>
<td>8s</td>
<td>-13206.57</td>
</tr>
<tr>
<td>magic-niab 10 8</td>
<td>3s</td>
<td>-974.23</td>
</tr>
<tr>
<td>magic-niab 20 8</td>
<td>74s</td>
<td>-1860.11</td>
</tr>
<tr>
<td>magic-niab 40 8</td>
<td>61s</td>
<td>-4109.82</td>
</tr>
<tr>
<td>magic-niab 80 8</td>
<td>131s</td>
<td>-8776.99</td>
</tr>
<tr>
<td>magic-niab 160 8</td>
<td>470s</td>
<td>-15306.12</td>
</tr>
</tbody>
</table>

### Figure 7: Bayes factors against the optimal BN on synthetic data from MAGs with average node degree $d$.

The MAGs are a generalization of BNs, allowing for unobserved variables. We developed algorithmic solutions—including

### 6 CONCLUSION

MAGs are a generalization of BNs, allowing for unobserved variables. We developed algorithmic solutions—including
score computation, score pruning, and a practical exact search algorithm—to learning score-optimal MAG, motivated by the impact of such solutions in the realm of BNs. Potential avenues for developing further approaches to MAG structure learning include approaches with improved scalability [Lee and van Beek, 2017, Tsamardinos et al., 2006, Chickering 2002], liftings of techniques currently mostly limited to DAG structures, e.g., Bayesian model averaging through exact computation [Liao et al., 2019, Koivisto and Sood, 2004, Tian and He 2009] or MCMC sampling [Kuipers and Moffa, 2017, Koller and Friedman 2009, Silva and Ghahramani, 2006], and techniques for focusing search on promising local c-components towards scaling to large-scale structures [Scanagatta et al., 2015].

Acknowledgements

This work was funded by Academy of Finland under grants 316771, 322869 and 328718.

References


