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 Chibcham and Constantinou [2020].

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, 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].
directed path \( v_i \to \cdots \to v_j \). An almost directed cycle is a directed path \( v_i \to \cdots \to v_j \) with a bi-directed edge \( v_i \leftrightarrow v_j \). For example, the mixed graph shown in 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., \( \leftrightarrow v_i \leftrightarrow v_i \). 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 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 \( \leftarrow l \rightarrow \) explicitly marking the unobserved \( l \). This gives intuition to inducing paths, nodes (e.g. \( x, q \) in 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.

\textbf{Lemma 1.} Every inducing path of length \( n \geq 3 \) nodes in a MAG between \( x \) and \( y \) is of the form \( x \leftrightarrow z_1 \leftrightarrow \ldots \leftrightarrow z_{n-2} \leftrightarrow y \), 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 \( x \).

\textbf{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 \( x \leftrightarrow y \). Node \( z_1 \) cannot be an ancestor of \( x \) since that would introduce a directed cycle \( x \to z_1 \to \cdots \to x \) or an almost directed cycle \( x \leftrightarrow z_1 \to \cdots \to x \). 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 \( x \leftrightarrow z_1 \leftrightarrow \ldots \leftrightarrow z_{n-2} \leftrightarrow y \) since otherwise \( z_1 \) would not be a collider. The inducing path can also not be of the form \( x \leftrightarrow z_1 \leftrightarrow \ldots \leftrightarrow z_{n-2} \leftrightarrow y \) since \( z_1 \) is an ancestor of \( y \) and thus otherwise there would be an almost directed cycle \( z_1 \to \cdots \to y \leftrightarrow z_{n-2} \to \cdots \to x \to z_1 \). This same logic can be applied symmetrically to \( y \). Therefore the inducing path must be of the form \( x \leftrightarrow z_1 \leftrightarrow \ldots \leftrightarrow z_{n-2} \leftrightarrow y \). \( \square \)

We define MAGSL as follows.

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

\[ G^* \in \arg \max_{G \in \mathcal{G}} s(G). \]

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 [Richardson and Spirtes, 2002; Tsirlis et al., 2018]

\[ s(G) = \ln L_G(\hat{\theta}) - (|E| + 2|V|)/2 \cdot \ln N, \] (2)

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 [Richardson and Spirtes, 2002; Tsirlis et al., 2018]. Furthermore, since Markov-equivalent MAGs can represent the same multivariate Gaussians distributions [Richardson and Spirtes, 2002; Corollary 8.19] and Markov-equivalent MAGs share adjacencies [Spirtes and Richardson, 1996], the BIC score is score-equivalent.

Fortunately, the score function \( s \) factorizes according to so-called c-components [Tian and Pearl, 2002; Richardson, 2009], which we define for the purpose of this paper as follows. We denote the set of parents of variable \( v \) in a MAG \( G \) by \( pa_G(v) \). Furthermore, we denote the parents of a c-component \( C \) over nodes \( v_1, \ldots, v_l \) with a list \( pa_G(C) = (pa_G(v_1), \ldots, pa_G(v_l)) \).

Definition 2 (local c-component). A local c-component is a pair \((C_i, pa(C_i))\), where \( C_i \) is a graph the nodes of which are strongly connected via bidirected edges and \( pa(C_i) \) is a list of the parent sets of the nodes in \( C_i \).

In Figure 1(a), \((x \leftrightarrow q, \{w\}, \emptyset)\) and \((w \leftrightarrow y, \{q\})\) are local c-components. The score of a MAG \( G \) containing \( l \) maximal local c-components indexed by \( i \) is

\[ s(G) = \sum_{i=1}^{l} s(C_i, pa_G(C_i)), \] (3)

where \( s(\cdot, \cdot) \) denote the local scores for the c-components.

To make local score computation and exact learning possible for non-trivial instances, we use two restrictions for \( G \), similarly as what is used for BNSL. We use \( c \) to denote the local c-component size limit (in terms of nodes in \( C_i \)). We use \( p \) to denote the limit on parent relations for a c-component, i.e., \( \sum_{x \in C_i} |pa(x)| \leq p \). This means that configuration \( c = 1, p \) is BNSL with maximum number of parents \( p \).

3 LOCAL SCORE COMPUTATION

The first challenge in learning MAGs is computing local scores. There are vastly more local scores to compute than in BNSL as we need to consider not only c-components consisting of subsets of variables but also all different ways the variables in a c-component can be connected with the 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 to not jeopardize exactness.

3.1 COMPUTING A LOCAL SCORE

We compute local scores as \( s(C_i, pa_G(C_i)) = s(G) - s(G') \), 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 [Geiger and Heckerman, 1994; Tsirlis et al., 2018]. The maximum likelihood estimates for the BIC score are computed using residual iterative conditional fitting (RICF) of [Drton et al., 2009]. 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 [Drton et al., 2009; Theorem 13]. There can be several local optima especially for limited sample sizes [Drton and Richardson, 2004; Améndola et al., 2020]. In practice, for local c-components of limited size, RICF usually converges from random starting points to only a few different likelihood values; see Section 5 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 [Bartlett and Cussens, 2017].

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 \( pa(x_i) \in P \) describes the parents of \( x_i \). We say that \( z \in pa(x_i) \) 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 \( \sum_{x \in C} |pa(x)| \leq p \), 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 \( S_{internal} \) 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 \( S_{internal} \) and go over all the ways of adding external parents to them (as long as the parent limit \( p \) is not exceeded), forming \( S_{external} \).
3. Return the local scores in \( S_{internal} \) and \( S_{external} \) which cannot be pruned.
We will now propose two rules which provide a way of (such as the MAG in (a)).

The benefit of the above formulation is that we only need to check acyclicity and maximality when constructing $S_{\text{internal}}$ which can be multiple times smaller in size compared to $S_{\text{external}}$ (depending on the choice of $p$).

### 3.3 Local Score Pruning for MAGS

For BNSL, scaling up exact structure learning is made possible by score pruning techniques [de Campos and Ji [2011]]. 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 $x \leftrightarrow y \leftrightarrow z \leftrightarrow w$ 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:

(a) there is no path of bidirected edges between $x$ and $y$ containing more than 3 nodes; or

(b) $pa_G(x) \subseteq pa_G(y)$ or $pa_G(y) \subseteq pa_G(x)$.

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 $pa(x) \in P$ is a subset of the corresponding $pa(x) \in P'$;
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. $\sum s(C_i, P_i) \geq s(C', P')$;
2. each $pa(x) \in P$ is a subset of the corresponding $pa(x) \in P_i$ (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...
We are finally ready to establish the correctness of Pruning 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 pa_C(x) ⊆ pa_G(y), then there can be no inducing path between x and y.

Proof. Assume that pa_G(x) ⊆ 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} ↔ 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 ↔ 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 ↔ z ↔ 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 pa_G(x) ⊆ pa_G(y) for nodes x and y in a MAG G, then G will remain a MAG after removing edge x ↔ y.

Proof. A direct consequence of Lemma 2 □

Lemma 5. A MAG remains a MAG after removing an arbitrary edge x → y.

Proof. Let G be a MAG and let G' be G after removing an edge x → 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} ↔ y where z_{n-2} is an ancestor x (Lemma 1). However, since G has the edge x → y, this would mean that G contains an almost directed cycle x → y ↔ z_{n-2} → ⋯ → 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 R_U, almost reach A_U )
2: if U = ∅ then return 0
3: Let u0 ∈ U: lexicographically least node in U.
4: Let C: the possible c-component and parent set combinations (C, pa(C)) such that u0 ∈ C ⊆ U.
5: Remove each (C, pa(C)) ∈ C that would introduce an (almost) directed cycle given R_U, A_U.
6: Let m ← −∞
7: for each (C, pa(C)) ∈ C do
8: Let R_{U\setminus C} and A_{U\setminus C} be updated versions of R_U and A_U given (C, pa(C)).
9: m ← max(m, s(C, pa(C)) + m)
10: return m

**Theorem 1.** Let C' and C_1, . . . , C_n be c-components with parent sets P' and P_1, . . . , P_n respectively. Suppose the nodes in C_1, . . . , 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, pa(x) in P_i is a subset of the corresponding pa'(x) in P', and

3) if x ↔ y ∉ C' then x ↔ y ∉ C_i for all x, y in C_i, and

4) if x ↔ y in C' and x ↔ y ∉ C_i for some x, y in C_i, then \{x,y\} must be maximality-preserving pair 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, . . . , C_n and P_1, . . . , 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 → y in G' but not in G sequentially (Lemma 2). (3) Suppose that G is missing an edge x ↔ y ∈ G'. If x ∉ C_i or y ∉ C_i for all C_i, there are no inducing path between x and y in G by Lemma 1 and so x ↔ y is allowed to be absent. Otherwise, if x, y ∈ C_i for some C_i, we require \{x,y\} to be a maximality-preserving pair in C_i. Therefore we have either (a) pa_G(x) ⊆ pa_G(y) or (b) all paths of bidirectional edges between x and y have less than 4 nodes. By Lemma 2 and 3 resp., the edge x ↔ 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. □

**4 EXACT STRUCTURE SEARCH**

We turn to developing a first exact search procedure for MAGSL. Several approaches for BNSL [Koivisto and Sood 2004; Silander and Myllymäki 2006; Yuan and Malone 2013] can use the fact that if a sink node and edges to it are removed from an optimal BN structure, then the remaining BN structure is optimal with respect to the remaining vari-
Algorithm 2 Exact branch-and-bound algorithm for finding score-optimal maximal and ancestral solutions $S^*$ (MAGs).

\begin{algorithm}
\begin{algorithmic}[1]
\Function{SEARCH}{nodes $U$, partial solution $S$}
\If{$S$ is non-maximal} \Return \EndIf
\If{$U = \emptyset$} \Return \EndIf
\If{$s(S') < s(S)$} $S' \leftarrow S$ \EndIf
\State $R_U, A_U \leftarrow$ reachability in $S$
\If{$s(S) + UB(U, R_U, A_U) \leq S^*$} \Return \EndIf
\State Let $\hat{S}$: $S$ extended with the UB solution.
\If{$\hat{S}$ is maximal} \Return \EndIf
\If{$s(S') < s(\hat{S})$} $\hat{S} \leftarrow S'$ \EndIf
\State Let $u_0 \in U$: lexicographically least node in $U$.
\State Let $C$: all possible c-component and parent set combinations $(C, pa(C))$ such that $u_0 \in C \subseteq U$.
\State Remove each $(C, pa(C)) \in C$ that would introduce an (almost) directed cycle given $R_U, A_U$.
\For{each $(C, pa(C)) \in C$}
\State Let $S'$ be $S$ with $(C, pa(C))$ added to it.
\State \Call{SEARCH}{$U \setminus C, S'$}
\EndFor
\EndFunction
\end{algorithmic}
\end{algorithm}

Although all MAGs do have sink nodes, unfortunately removing a sink node does not always preserve optimality: to see this, consider the covariance matrix

$$C_x = \begin{bmatrix}
1.00 & 0.00 & 0.20 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.20 \\
0.20 & 0.00 & 1.04 & 0.50 \\
0.00 & 0.20 & 0.50 & 1.04
\end{bmatrix}$$

with $N = 100$ samples. The optimal solution over all four nodes is $v_1 \rightarrow v_3 \leftrightarrow v_4 \leftarrow v_2$, while without $v_4$ the optimal solution is the empty graph. Intuitively, the correlation between $v_1$ and $v_3$ is only worth the additional parameter under the BIC score if the correlation of $v_4$ on $v_3$ is also taken into account. Symmetrically, without $v_3$ the optimal solution is also the empty graph. The decomposability of the score in Eq. 3, though directly implies that removing a sink c-component (from which there are no edges out) from an optimal MAG, retains optimality. However, not all MAGs have so-called sink c-components. Figure 1(a) shows a MAG for which there is no order for the c-components and thus no sink c-components [Richardson and Spirtes, 2002].

4.1 A BRANCH-AND-BOUND APPROACH

Nevertheless, the conditions of a mixed graph being a MAG (Definition 1) are rather restrictive in terms of search. Placing a single edge forbids the presence of a number of other edges through the MAG conditions. Our approach takes advantage of this: after placing a local c-component (with its parents) we disregard a large number of other c-components whose edges would now result in non-MAG structures.

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.

Algorithm 1 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 $C$, containing a node $u_0$ (Line 9–10). 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 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 2 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 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 1. If $\hat{S}$ is maximal, we backtrack and also update $S^*$ if $\hat{S}$ has better score.

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 [Hyttinen et al., 2014; Sonntag et al., 2015; Magliacane et al., 2016; Forér and Mooij, 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
chosen c-components, and each node is required to be contained 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 computation 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 probable. The coefficients and covariances for the directed and bidirected edges were drawn uniformly from \([0.2, 0.8]\). Variances of disturbances were drawn as \(0.5 \cdot s + 1\), with \( s \sim N(0, 1)\). 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 variables, 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 performance of the local score computation part on the synthetic data. Figure 4 shows the number of local scores produced 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 results in substantial increase in the number of local scores. The increase in the number of local scores for each additional 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 computation 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 5 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 solution 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.].
Table 1: Results on the benchmark datasets.

<table>
<thead>
<tr>
<th>Benchmark dataset</th>
<th>Time</th>
<th>BB, c=2</th>
<th>BB, c=3</th>
<th>BN</th>
<th>FCI + bootstrap</th>
<th>GFCI + bootstrap</th>
<th>MHCIR + bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>magic-irri 100</td>
<td>8</td>
<td>45</td>
<td>1091.50</td>
<td>-1091.50</td>
<td>-1093.47</td>
<td>-1092.25</td>
<td>-1096.67</td>
</tr>
<tr>
<td>magic-niab 200</td>
<td>8</td>
<td>169</td>
<td>-2169.11</td>
<td>-2167.97</td>
<td>-2174.51</td>
<td>-2175.21</td>
<td>-2171.45</td>
</tr>
<tr>
<td>magic-irri 200</td>
<td>8</td>
<td>269</td>
<td>-4228.14</td>
<td>-4229.13</td>
<td>-4245.32</td>
<td>-4233.32</td>
<td>-4224.88</td>
</tr>
<tr>
<td>magic-niab 200</td>
<td>10</td>
<td>1283</td>
<td>-8358.48</td>
<td>-8367.11</td>
<td>-8363.27</td>
<td>-8360.05</td>
<td>-8358.48</td>
</tr>
<tr>
<td>magic-irri 100</td>
<td>10</td>
<td>421</td>
<td>-2396.24</td>
<td>-2398.19</td>
<td>-2428.62</td>
<td>-2403.56</td>
<td>-2405.28</td>
</tr>
<tr>
<td>magic-irri 100</td>
<td>10</td>
<td>1108</td>
<td>-4677.92</td>
<td>-4735.4</td>
<td>-4684.57</td>
<td>-4693.9</td>
<td>-4681.44</td>
</tr>
<tr>
<td>magic-irri 100</td>
<td>10</td>
<td>2421</td>
<td>-9440.41</td>
<td>-9449.40</td>
<td>-9444.91</td>
<td>-9441.98</td>
<td>-9440.41</td>
</tr>
<tr>
<td>magic-irri 100</td>
<td>10</td>
<td>1605</td>
<td>-1256.84</td>
<td>-1263.84</td>
<td>-1243.38</td>
<td>-1238.89</td>
<td>-1242.15</td>
</tr>
<tr>
<td>magic-irri 200</td>
<td>8</td>
<td>169</td>
<td>-2169.11</td>
<td>-2170.97</td>
<td>-2174.51</td>
<td>-2175.21</td>
<td>-2171.45</td>
</tr>
<tr>
<td>magic-niab 400</td>
<td>8</td>
<td>-8396.98</td>
<td>-8358.48</td>
<td>-8367.11</td>
<td>-8363.27</td>
<td>-8360.05</td>
<td>-8358.48</td>
</tr>
<tr>
<td>magic-niab 800</td>
<td>3</td>
<td>-2185.37</td>
<td>-2185.90</td>
<td>-1855.29</td>
<td>-1853.83</td>
<td>-1851.44</td>
<td>-1856.23</td>
</tr>
<tr>
<td>magic-niab 200</td>
<td>8</td>
<td>4110.33</td>
<td>-4110.33</td>
<td>-4113.24</td>
<td>-4112.03</td>
<td>-4118.7</td>
<td>-4110.93</td>
</tr>
<tr>
<td>magic-niab 400</td>
<td>8</td>
<td>42</td>
<td>-4633.63</td>
<td>-4633.64</td>
<td>-4641.52</td>
<td>-4639.44</td>
<td>-4646.11</td>
</tr>
<tr>
<td>magic-niab 800</td>
<td>8</td>
<td>46</td>
<td>-9787.39</td>
<td>-9780.91</td>
<td>-9789.05</td>
<td>-9788.03</td>
<td>-9785.92</td>
</tr>
<tr>
<td>magic-niab 1600</td>
<td>8</td>
<td>2154</td>
<td>-19712.27</td>
<td>-19720.19</td>
<td>-19729.26</td>
<td>-19716.07</td>
<td>-19719.3</td>
</tr>
</tbody>
</table>

Figure 6: Runtimes: BB for c = 2 (left) and c = 3 (middle) on synthetic data; BB vs ASP on benchmark data (right).
Bayes factor vs. optimal BN

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>Name</td>
<td>N</td>
<td>n=7</td>
</tr>
<tr>
<td>magic-niab 100 7</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>magic-niab 200 7</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>magic-niab 400 7</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>magic-niab 100 8</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>magic-niab 200 8</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>magic-niab 400 8</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>magic-irri 100 7</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>magic-irri 200 7</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>magic-irri 400 7</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>magic-irri 100 8</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>magic-irri 200 8</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>magic-irri 400 8</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>ecoli70 100 7</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>ecoli70 200 7</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>ecoli70 400 7</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>ecoli70 100 8</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>ecoli70 200 8</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>ecoli70 400 8</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>arth150 100 7</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>arth150 200 7</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>arth150 400 7</td>
<td>456.5</td>
<td>516.2</td>
</tr>
<tr>
<td>arth150 100 8</td>
<td>227.64</td>
<td>287.48</td>
</tr>
<tr>
<td>arth150 200 8</td>
<td>337.2</td>
<td>396.9</td>
</tr>
<tr>
<td>arth150 400 8</td>
<td>456.5</td>
<td>516.2</td>
</tr>
</tbody>
</table>

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

6 CONCLUSION

There is noticeable variation (see Figure 7) with respect to the dataset: for many the MAGs found are 10 times more likely than BNs, for some BNs are equally likely. FCI and GFCI perform somewhat poorly, suggesting that they cannot find very high scoring MAGs over the 200 data samples given. The MAGs found by bootstrapped M3HC are better, often more likely than BNs. The MAGs found by our approach are still better, 2-10 times more likely than the optimal BNs. Note that our method is feasible with different parent relation limits p, i.e., p = 2 for c = 2 and p = 4 for c = 3, 4. There is an interesting trade-off as to whether to allow for larger c-components or more parent relations.

Table 1 and 2 give the BIC scores obtained on the benchmark data, with highest score for each instance in bold. Optimal BNs achieve the highest score for a minority of the instances. Relative performance of the in-exact algorithms depends noticeably on the dataset and the number of samples. FCI, GFCI and M3HC in some cases (especially for arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions. For arth150 and ecoli70) find high-scoring solutions.
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


