Maximal Ancestral Graph Structure Learning via Exact Search

Rantanen, Kari

AUAI Press
2021

Rantanen , K , Hyttinen , A & Järvisalo , M 2021 , Maximal Ancestral Graph Structure Learning via Exact Search . in C P de Campos , M H Maathuis & E Quaeghebeur (eds) , Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021) . Proceedings of Machine Learning Research , vol. 161 , AUAI Press , pp. 1237-1247 , 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021) , 27/07/2021 . < https://proceedings.mlr.press/v161/rantanen21a.html >

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

Maximal Ancestral Graph Structure Learning via Exact Search

Kari Rantanen[1] **Antti Hyttinen**[1] **Matti Järvisalo**[1]

[1]HIIT, Department of Computer Science, University of Helsinki, Finland

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.[1] 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 [Triantafillou and Tsamardinos, 2016, Tsamardinos et al., 2006]. Further studies in this direction include Chobtham and Constantinou

[1]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].

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1237–1247.

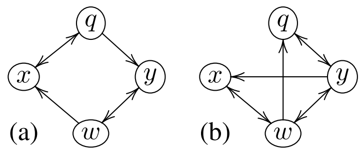

Figure 1: (a) A (MAG). (b) A mixed graph that is not a MAG: there is an inducing path between x and q.

[2020], who combine structure and causal effect estimation, and GFCI which combines score-based and constraint-based learning [Ogarrio et al., 2016]. However, all of these approaches employ forms of greedy search without guarantees on the quality of the obtained solutions.

In this work, we develop score-based learning of MAGs via exact search, motivated by the success of exact approaches to BN structure learning. In particular, we formulate the MAGSL problem of learning score-optimal MAGs employing the Gaussian BIC score. We outline sparsity conditions under which exact search is possible, generalizing the parent set bounds standardly employed in exact approaches to BN structure learning. Lifting optimality-preserving score pruning techniques essential for scaling up exact BN structure learning algorithms, we develop methodology of local score computation and score pruning for MAGs. From score computation to search for optimal MAGs, combining dynamic programming and branch-and-bound we present a first exact search algorithm for learning MAGs for given local scores. We evaluate this exact score-based MAG learning approach both in terms of runtime efficiency, comparing to a baseline approach based on employing declarative methods (namely, answer set programming), and also empirically show that our approach finds better scoring MAGs than its direct competitors. All in all, our contributions open up new avenues for developing increasingly efficient and accurate exact approaches to learning MAGs.

2 MAG STRUCTURE LEARNING

Following Zhang [2008a], we consider MAGs without undirected edges [Richardson and Spirtes, 2002].

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 $v_i \rightarrow \cdots \rightarrow v_i$. An *almost 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 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., $\leftrightarrow v_i \leftrightarrow$, $\rightarrow v_i \leftrightarrow$, $\leftrightarrow v_i \leftarrow$ or $\rightarrow v_i \leftarrow$. 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.

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.*

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

We define MAGSL as follows.

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

$$G^* \in \mathrm{argmax}_{G \in \mathcal{G}}\ s(G). \tag{1}$$

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]

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

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, Corrollary 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 $\text{pa}_G(v)$. Furthermore, we denote the parents of a c-component C over nodes $v_1, \ldots, v_l$ with a list $\text{pa}_G(C) = (\text{pa}_G(v_1), \ldots, \text{pa}_G(v_l))$.

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

$$s(G) = \sum_{i=1}^{l} s(C_i, \text{pa}_G(C_i)), \qquad (3)$$

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 $\mathcal{G}$, 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., $\sum_{x \in C_i} |\text{pa}(x)| \leq p$. 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 $s(C_i, \text{pa}_G(C_i)) = s(G) - s(G')$, 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 [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 RCIF 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, RCIF usually converges from random starting points to only a few different likelihood values; see Section 5. For any c-component of size 1, RCIF 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 $\sum_{x \in C} |\text{pa}(x)| \leq p$, 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_{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.

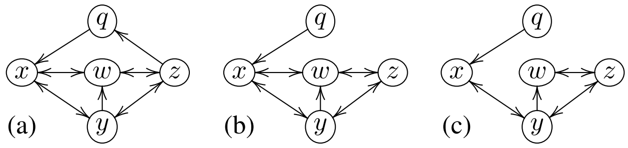

Figure 2: Pruning a local score for (b) due to a higher score for (c) may prevent finding an optimal MAG (a), as replacing (b) in (a) with (c) results in an inducing path btw. x and w.

The benefit of the above formulation is that we only need to check acyclicity and maximality when constructing S_{internal} which can be multiple times smaller in size compared to S_{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 $pa_G(y) \subseteq pa_G(x)$.

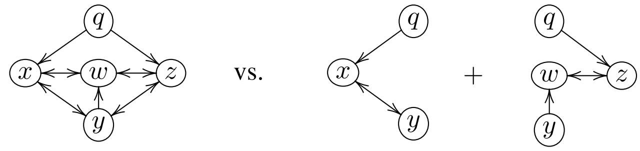

Figure 3: The local score for the left can be pruned if the sum of local scores for the center and right is higher.

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) $\sum_i s(C_i, P_i) \geq s(C', P')$;
(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

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 $pa_G(x) \subseteq pa_G(y)$, then there can be no inducing path between x and y.*

Proof. Assume that $\mathrm{pa}_G(x) \subseteq \mathrm{pa}_G(y)$. 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. $\square$

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. $\square$

Lemma 4. *If $pa_G(x) \subseteq pa_G(y)$ 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. $\square$

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 x (Lemma 1). Hoever, 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. $\square$

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.

function UB(nodes U, reach R_U, almost reach A_U)
  if $U = \emptyset$ **then return** 0
  Let $u_0 \in U$: lexicographically least node in U.
  Let $\mathcal{C}$: the possible c-component and parent set combinations (C, pa(C)) such that $u_0 \in C \subseteq U$.
  Remove each $(C, \mathrm{pa}(C)) \in \mathcal{C}$ that would introduce an (almost) directed cycle given R_U, A_U.
  Let $m \leftarrow -\infty$
  for each $(C, \mathrm{pa}(C)) \in \mathcal{C}$ **do**
    Let $R_{U \setminus C}$ and $A_{U \setminus C}$ be updated versions of R_U and A_U given $(C, \mathrm{pa}(C))$.
    $\hat{m} \leftarrow \mathrm{UB}(U \setminus C, R_{U \setminus C}, A_{U \setminus C})$
    $m \leftarrow \max(m, s(C, \mathrm{pa}(C)) + \hat{m})$
  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 $x \in C_i$, $pa(x) \in P_i$ is a subset of the corresponding $pa'(x) \in P'$, and
(3) if $x \leftrightarrow y \notin C'$ then $x \leftrightarrow y \notin C_i$ for all $x, y \in C_i$, and
(4) if $x \leftrightarrow y \in C'$ and $x \leftrightarrow y \notin C_i$ for some $x, y \in C_i$, then $\{x,y\}$ must be maximality-preserving 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 $C_1, \ldots, C_n$ 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 5). (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 (a) $\mathrm{pa}_G(x) \subseteq \mathrm{pa}_G(y)$ or (b) all paths of bidirectional edges between x and y have less than 4 nodes. By Lemmas 4 and 3, 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. $\square$

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).

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

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

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

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 $v_i \rightarrow v_j$ is in G. R_U (A_U) implies a(n 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 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 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$(V, S_\emptyset)$ where $S_\emptyset$ 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, Forré 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

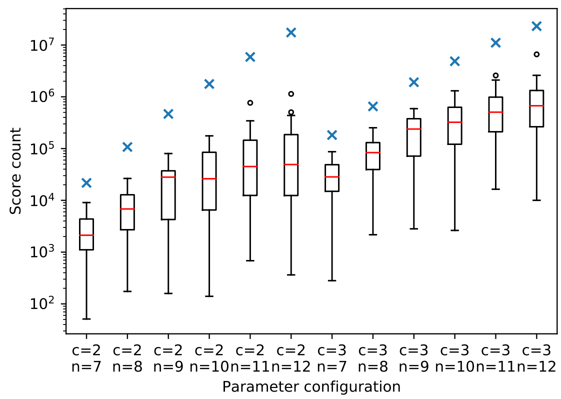

Figure 4: Score counts for the synthetic datasets.

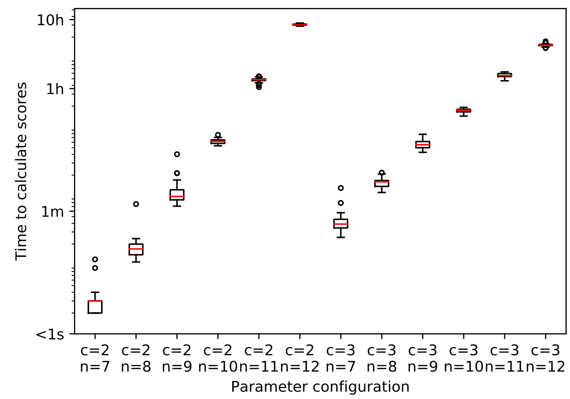

Figure 5: Scoring and pruning time on synthetic data.

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 $\pm[0.2, 0.8]$. 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 6 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 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.,

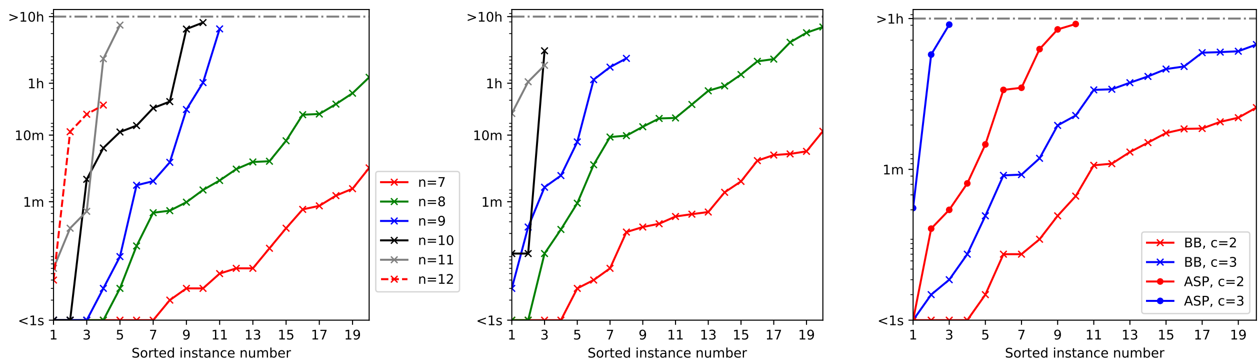

Figure 6: Runtimes: BB for $c = 2$ (left) and $c = 3$ (middle) on synthetic data; BB vs ASP on benchmark data (right).

Table 1: Results on the benchmark datasets.

Benchmark dataset			Time	BIC score							
Name	N	n	BB, c=2	BB, c=2	BN	FCI	+bootstrap	GFCI	+bootstrap	M3HC	+bootstrap
arth150	100	8	7963s	-233.27	-234.83	-283.86	**-228.19**	-257.75	-235.54	-443.16	-342.16
arth150	200	8	16550s	-419.11	-421.39	-465.41	-409.08	-457.46	**-402.72**	-586.98	-457.29
arth150	400	8	30276s	-777.26	-778.73	-819.2	-769.37	-905.43	**-744.04**	-849.61	-801.53
arth150	800	8	$\geq$10h	timeout	-1423.85	-1492.67	**-1367.92**	-1567.07	-1450.01	-1547.12	-1440.8
arth150	1600	8	$\geq$10h	timeout	-2780.39	-2928.61	**-2584.09**	-3057.74	-2643.8	-2880.17	-2786.27
ecoli70	100	8	5439s	-701.23	-701.23	-749.24	**-684.43**	-745.36	-701.23	-791.43	-707.0
ecoli70	200	8	8285s	-1378.43	-1378.43	-1386.35	-1378.43	-1386.35	**-1252.9**	-1386.35	-1378.43
ecoli70	400	8	12256s	-2684.62	-2685.26	-2696.84	**-2413.51**	-2696.84	-2685.26	-2696.84	-2684.62
ecoli70	800	8	18096s	**-5330.34**	-5332.49	-5332.61	-5332.61	-5332.61	-5332.49	-5332.61	-5332.61
ecoli70	1600	8	19717s	**-10728.61**	**-10728.61**	**-10728.61**	**-10728.61**	**-10728.61**	**-10728.61**	**-10728.61**	**-10728.61**
magic-irri	100	8	45s	-1091.59	-1091.59	-1097.93	**-1085.44**	-1093.47	-1092.7	-1096.7	-1094.05
magic-irri	200	8	169s	**-2169.11**	-2170.97	-2174.51	-2174.51	-2175.51	-2171.45	-2174.81	-2169.22
magic-irri	400	8	269s	**-4228.14**	-4229.13	-4245.32	-4230.37	-4233.32	-4229.09	-4244.8	-4230.23
magic-irri	800	8	1283s	**-8358.48**	**-8358.48**	-8367.11	-8363.27	-8360.05	-8360.05	**-8358.48**	**-8358.48**
magic-irri	1600	8	2085s	**-16718.88**	**-16718.88**	-17169.02	-16726.55	-16720.1	-16718.89	-16730.5	-16719.3
magic-irri	100	9	160s	-1236.84	-1236.84	-1243.38	**-1230.9**	-1238.92	-1238.15	-1242.15	-1239.7
magic-irri	200	9	4215s	**-2396.24**	-2398.1	-2428.62	-2403.56	-2402.64	-2398.84	-2405.28	-2401.46
magic-irri	400	9	11081s	**-4677.92**	-4678.9	-4735.4	-4684.57	-4693.9	-4681.44	-4701.12	-4683.74
magic-irri	800	9	2421s	**-9440.41**	**-9440.41**	-9449.04	-9445.2	-9441.98	-9441.98	**-9440.41**	**-9440.41**
magic-irri	1600	9	$\geq$10h	timeout	**-18492.31**	-19076.71	-18500.66	-18498.54	-18496.35	-18503.37	**-18492.31**
magic-irri	100	10	1595s	-1352.6	-1352.6	-1359.14	**-1346.66**	-1354.68	-1353.91	-1357.91	-1355.27
magic-irri	200	10	12354s	**-2665.85**	-2667.71	-2698.23	-2673.17	-2672.25	-2668.44	-2674.73	-2668.81
magic-irri	400	10	33771s	**-5209.08**	-5210.07	-5266.57	-5215.74	-5225.06	-5212.61	-5232.29	-5214.12
magic-irri	800	10	-	memout	**-9788.63**	-9797.26	-9793.43	-9790.2	-9790.2	**-9788.63**	**-9788.63**
magic-irri	1600	10	$\geq$10h	timeout	**-20669.1**	-21253.5	-20677.45	-20675.33	-20673.15	-20680.17	-20672.53
magic-niab	100	8	<1s	**-974.23**	**-974.23**	-976.16	-976.04	-976.55	**-974.23**	-988.99	-985.05
magic-niab	200	8	3s	**-1850.27**	-1850.91	-1855.29	-1853.83	-1853.77	-1851.43	-1861.7	-1856.23
magic-niab	400	8	2s	**-4110.33**	**-4110.33**	-4113.24	-4113.02	-4118.7	-4110.93	-4127.8	-4113.53
magic-niab	800	8	5s	**-8776.99**	-8777.64	-8780.52	-8780.14	-8777.64	-8777.64	-8784.93	-8779.73
magic-niab	1600	8	195s	**-15306.12**	**-15306.12**	-15364.39	-15324.25	-15310.85	-15308.73	-15322.14	-15311.86
magic-niab	100	9	1s	**-1103.85**	**-1103.85**	-1108.14	-1106.59	-1109.02	-1106.0	-1121.46	-1116.43
magic-niab	200	9	11s	**-2127.89**	-2128.7	-2133.84	-2132.4	-2132.33	-2129.99	-2140.25	-2134.78
magic-niab	400	9	42s	**-4633.3**	-4633.64	-4641.52	-4639.44	-4644.11	-4636.36	-4653.24	-4638.94
magic-niab	800	9	60s	**-9787.39**	-9788.03	-9790.91	-9790.53	-9788.03	-9788.03	-9795.32	-9790.93
magic-niab	1600	9	2154s	**-19712.27**	**-19712.27**	-19770.19	-19729.26	-19716.07	-19713.93	-19729.03	-19718.87
magic-niab	100	10	29s	**-1185.68**	-1185.87	-1191.14	-1189.02	-1191.04	-1187.08	-1204.67	-1199.64
magic-niab	200	10	137s	**-2382.11**	-2383.33	-2388.17	-2388.82	-2387.41	-2386.28	-2395.33	-2392.04
magic-niab	400	10	1360s	**-4639.42**	-4639.61	-4657.5	-4652.51	-4650.7	-4642.03	-4664.04	-4650.03
magic-niab	800	10	550s	**-10819.85**	-10820.72	-10834.1	-10826.23	-10820.51	-10820.51	-10841.72	-10828.19
magic-niab	1600	10	15520s	**-21758.51**	**-21758.51**	-21818.07	-21777.13	-21762.3	-21760.18	-21775.26	-21767.31
magic-niab	100	11	159s	**-1189.28**	-1189.47	-1195.64	-1195.29	-1195.46	-1191.42	-1209.09	-1204.06
magic-niab	200	11	2597s	**-2390.1**	-2390.61	-2397.73	-2399.39	-2395.77	-2394.64	-2405.08	-2399.61
magic-niab	400	11	-	memout	**-5080.62**	-5099.19	-5095.97	-5085.56	-5083.75	-5108.45	-5095.06
magic-niab	800	11	-	memout	**-11870.66**	-11889.45	-11886.63	-11874.24	-11872.92	-11890.16	-11880.38
magic-niab	1600	11	-	memout	**-23697.44**	-23757.0	-23723.23	-23701.24	-23699.11	-23714.19	-23704.74

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-z-test $\alpha = 0.05$ and sem-bic-score. We directly obtained a single representative MAG of the equivalence class returned by these algorithms. As the other methods are considerable 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.

Benchmark dataset			Time		BIC score								
Name	N	n	BB, c=3	BB, c=4	BB, c=3	BB, c=4	BN	FCI	+bootstrap	GFCI	+bootstrap	M3HC	+bootstrap
arth150	100	7	528s	6323s	-224.74	-224.74	-226.3	-233.97	**-219.27**	-244.24	-227.01	-395.44	-305.74
arth150	200	7	1482s	9848s	-377.96	-377.96	-379.95	-411.06	**-358.42**	-430.95	-379.52	-538.38	-418.23
arth150	400	7	1426s	7629s	-722.47	-722.47	-723.94	-749.46	**-661.35**	-775.41	-727.62	-746.74	-731.87
arth150	800	7	1447s	7599s	-1380.36	-1380.36	-1380.36	-1449.18	**-1235.14**	-1523.58	-1389.68	-1476.19	-1389.73
arth150	1600	7	1778s	8901s	-2696.91	-2696.91	-2696.91	-2527.26	**-2500.61**	-3153.36	-2724.3	-2815.5	-2703.04
ecoli70	100	7	519s	3523s	-602.8	-602.8	-602.8	-606.68	-571.23	-602.8	**-540.04**	-691.1	-602.8
ecoli70	200	7	631s	4317s	-1180.74	-1180.74	-1180.74	-1188.65	-1181.78	-1188.65	**-1055.2**	-1188.65	-1180.74
ecoli70	400	7	748s	4886s	**-2321.64**	**-2321.64**	**-2321.64**	-2333.22	**-2321.64**	-2333.22	**-2321.64**	-2333.22	**-2321.64**
ecoli70	800	7	914s	8984s	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**	**-4614.11**
ecoli70	1600	7	980s	9640s	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**	**-9326.18**
magic-irri	100	7	17s	426s	**-977.01**	**-977.01**	**-977.01**	-983.5	-978.75	-978.89	**-977.01**	-982.12	-978.75
magic-irri	200	7	51s	858s	**-1951.54**	**-1951.54**	-1952.47	-1956.0	-1955.01	-1957.01	-1952.94	-1955.77	-1953.68
magic-irri	400	7	52s	886s	**-3802.84**	**-3802.84**	-3803.83	-3818.42	-3804.46	-3808.49	-3803.83	-3817.89	-3804.46
magic-irri	800	7	198s	1739s	**-7520.48**	**-7520.48**	**-7520.48**	-7529.57	-7525.01	-7528.95	**-7520.48**	**-7520.48**	**-7520.48**
magic-irri	1600	7	259s	3353s	**-15051.99**	**-15051.99**	**-15051.99**	-15498.72	-15054.51	-15062.79	-15052.84	-15064.45	**-15051.99**
magic-irri	100	8	1144s	$\geq$10h	-1091.59	timeout	-1091.59	-1097.93	**-1085.44**	-1093.47	-1092.7	-1096.7	-1094.05
magic-irri	200	8	3121s	$\geq$10h	**-2169.11**	timeout	-2170.97	-2174.51	-2174.51	-2175.51	-2171.45	-2174.81	-2169.22
magic-irri	400	8	4930s	$\geq$10h	**-4228.14**	timeout	-4229.13	-4245.32	-4230.37	-4233.32	-4229.09	-4244.8	-4230.23
magic-irri	800	8	20289s	$\geq$10h	**-8358.48**	timeout	**-8358.48**	-8367.11	-8363.27	-8360.05	-8360.05	**-8358.48**	**-8358.48**
magic-irri	1600	8	$\geq$10h	$\geq$10h	timeout	timeout	**-16718.88**	-17169.02	-16726.55	-16720.1	-16718.89	-16730.5	-16719.3
magic-niab	100	7	$<$1s	15s	**-845.77**	**-845.77**	**-845.77**	-848.0	-846.58	**-845.77**	**-845.77**	-857.54	-856.6
magic-niab	200	7	3s	86s	-1616.51	**-1616.27**	-1616.99	-1620.13	-1617.77	-1618.42	-1617.4	-1623.32	-1618.87
magic-niab	400	7	2s	125s	**-3593.07**	**-3593.07**	-3593.15	-3595.95	-3593.21	-3594.29	-3593.17	-3603.88	-3594.87
magic-niab	800	7	6s	288s	**-6618.23**	**-6618.23**	-6618.88	-6621.76	-6621.76	-6618.88	-6618.42	-6626.17	-6620.72
magic-niab	1600	7	81s	1249s	**-13206.57**	**-13206.57**	**-13206.57**	-13249.49	-13209.71	-13206.65	-13206.65	-13222.01	-13214.8
magic-niab	100	8	3s	8757s	**-974.23**	**-974.23**	**-974.23**	-976.16	-976.04	-976.55	**-974.23**	-988.99	-985.05
magic-niab	200	8	74s	26899s	**-1850.11**	**-1850.11**	-1850.91	-1855.29	-1853.83	-1853.77	-1851.43	-1861.7	-1856.23
magic-niab	400	8	61s	$\geq$10h	**-4109.82**	timeout	-4110.33	-4113.24	-4113.02	-4118.7	-4110.93	-4127.8	-4113.53
magic-niab	800	8	131s	$\geq$10h	**-8776.99**	timeout	-8777.64	-8780.52	-8780.14	-8777.64	-8777.64	-8784.93	-8779.73
magic-niab	1600	8	4700s	$\geq$10h	**-15306.12**	timeout	**-15306.12**	-15364.39	-15324.25	-15310.85	-15308.73	-15322.14	-15311.86

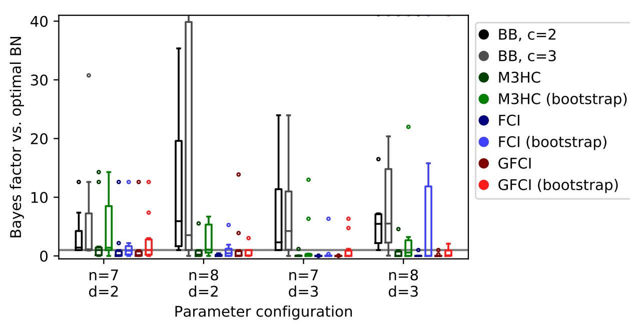

Figure 7: Bayes factors against the optimal BN on synthetic data from MAGs with average node degree d.

spect to the baseline optimal Bayesian network. The Bayes factors are based on the Gaussian BIC scoring function, which was computed for output MAGs from all competing methods. To score the found MAGs, every c-component was scored by running ICF 100 times from different starting points; BIC was calculated according to the highest likelihood value found. We also used this strategy for computing the local scores used by our method for Figure 7. After pruning, on average 0.28% of the remaining local scores had two ICF accumulation points for $c = 2$, and 0.45% had two or three ICF accumulation points for $c = 3$. Specifically, 6 out of the 40 instances for $c = 2$ had local score(s) remaining with two ICF accumulation points. For $c = 3$, 9 out of the 40 instances had local score(s) remaining with two ICF accumulation points, and 6 instances had local score(s) remaining with three ICF accumulation points.

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., $p = 8$ 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.

Tables 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 can in some cases (especially for arth150 and ecoli70) find high-scoring solutions. For magic-niab and magic-irri our BB obtains clearly better solutions (using $c = 2, 3, 4$) on this metric than its in-exact competitors. Moreover, the solutions of BB are guaranteed globally optimal over the input scores, i.e., higher-scoring solutions are known not to exist within the limits c and p.

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

C. Améndola, P. Dettling, M. Drton, F. Onori, and J. Wu. Structure learning for cyclic linear causal models. In *UAI*, volume 124 of *PMLR*, pages 999–1008. PMLR, 2020.

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

D. Bernstein, B. Saeed, C. Squires, and C. Uhler. Ordering-based causal structure learning in the presence of latent variables. In *AISTATS*, pages 4098–4108. PMLR, 2020.

D. Chickering. Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, 3:507–554, 2002.

K. Chobtham and A. C. Constantinou. Bayesian network structure learning with causal effects in the presence of latent variables. In *PGM*, volume 138 of *PMLR*, pages 101–112, 2020.

T. Claassen and T. Heskes. A Bayesian approach to constraint based causal inference. In *UAI*, pages 207–216. AUAI Press, 2012.

T. Claassen, J. M. Mooij, and T. Heskes. Learning sparse causal models is not NP-hard. In *UAI*, pages 172–181. AUAI Press, 2013.

D. Colombo and M. H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, 2014.

G. Consonni and L. L. Rocca. Objective Bayes factor for Gaussian directed acyclic graphical models. *Scand. Stat. Theory Appl.*, 39(4):743–756, 2012.

C. P. de Campos and Q. Ji. Efficient structure learning of bayesian networks using constraints. *J. Mach. Learn. Res.*, 12:663–689, 2011.

M. Drton and T. S. Richardson. Multimodality of the likelihood in the bivariate seemingly unrelated regressions model. *Biometrika*, 91(2):383–392, 06 2004.

M. Drton, M. Eichler, and T. S. Richardson. Computing maximum likelihood estimates in recursive linear models with correlated errors. *J. Mach. Learn. Res.*, 10:2329–2348, 2009.

P. Forré and J. M. Mooij. Constraint-based causal discovery for non-linear structural causal models with cycles and latent confounders. In *UAI*, pages 269–278. AUAI Press, 2018.

M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.*, 187:52–89, 2012.

D. Geiger and D. Heckerman. Learning Gaussian networks. In *UAI*, pages 235–243. Morgan Kaufmann, 1994.

A. Hyttinen, F. Eberhardt, and M. Järvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *UAI*, pages 340–349. AUAI Press, 2014.

A. Jaber, J. Zhang, and E. Bareinboim. On causal identification under Markov equivalence. In *IJCAI*, pages 6181–6185. ijcai.org, 2019.

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

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

J. Kuipers and G. Moffa. Partition MCMC for inference on acyclic digraphs. *J. Am. Stat. Assoc.*, 112(517):282–299, 2017.

C. Lee and P. van Beek. Metaheuristics for score-and-search bayesian network structure learning. In *Canadian Conference on AI*, volume 10233 of *LNCS*, pages 129–141. Springer, 2017.

Z. A. Liao, C. Sharma, J. Cussens, and P. van Beek. Finding all Bayesian network structures within a factor of optimal. In *AAAI*, pages 7892–7899. AAAI Press, 2019.

S. Magliacane, T. Claassen, and J. M. Mooij. Ancestral causal inference. In *NIPS*, pages 4466–4474. Curran Associates, 2016.

C. Nowzohour, M. H. Maathuis, R. J. Evans, and P. Bühlmann. Distributional equivalence and structure learning for bow-free acyclic path diagrams. *Electron. J. Stat.*, 11(2):5342–5374, 2017.

J. M. Ogarrio, P. Spirtes, and J. Ramsey. A hybrid causal search algorithm for latent variable models. In *PGM*, volume 52 of *JMLR W&CP*, pages 368–379. JMLR.org, 2016.

E. Perkovic, J. Textor, M. Kalisch, and M. H. Maathuis. Complete graphical characterization and construction of adjustment sets in Markov equivalence classes of ancestral graphs. *J. Mach. Learn. Res.*, 18:220:1–220:62, 2017.

J. Ramsey, J. Zhang, and P. Spirtes. Adjacency-faithfulness and conservative causal inference. *CoRR*, abs/1206.6843, 2012.

K. Rantanen, A. Hyttinen, and M. Järvisalo. Learning chordal Markov networks via branch and bound. In *NIPS*, pages 1845–1855, 2017.

K. Rantanen, A. Hyttinen, and M. Järvisalo. Discovering causal graphs with cycles and latent confounders: An exact branch-and-bound approach. *Int. J. Approx. Reason.*, 117:29–49, 2020.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Ann. Stat.*, 30(4):962–1030, 2002.

T. S. Richardson. A factorization criterion for acyclic directed mixed graphs. In *UAI*, pages 462–470. AUAI Press, 2009.

T. S. Richardson and P. Spirtes. Causal inference via ancestral graph models. *Oxford Statistical Science Series*, 1 (27):83–105, 2003.

M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon. Learning Bayesian networks with thousands of variables. In *NIPS*, pages 1864–1872, 2015.

M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

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

R. Silva and Z. Ghahramani. Bayesian inference for Gaussian mixed graph models. In *UAI*, pages 453–460. AUAI Press, 2006.

D. Sonntag, M. Järvisalo, J. M. Peña, and A. Hyttinen. Learning optimal chain graphs with answer set programming. In *UAI*, pages 822–831. AUAI Press, 2015.

P. Spirtes and T. Richardson. A polynomial time algorithm for determining DAG equivalence in the presence of latent variables and selection bias. In *AISTATS*, pages 489–500, 1996.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*. Springer, 1993.

J. Suzuki. Learning Bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B & B technique. In *ICML*, pages 462–470. Morgan Kaufmann, 1996.

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

J. Tian and R. He. Computing posterior probabilities of structural features in Bayesian networks. In *UAI*, pages 538–547. AUAI Press, 2009.

J. Tian and J. Pearl. A general identification condition for causal effects. In *AAAI*, pages 567–573. AAAI Press, 2002.

S. Triantafillou and I. Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *J. Mach. Learn. Res.*, 16:2147–2205, 2015.

S. Triantafillou and I. Tsamardinos. Score-based vs constraint-based causal learning in the presence of confounders. In *CFA@UAI*, volume 1792 of *CEUR Workshop Proceedings*, pages 59–67. CEUR-WS.org, 2016.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65(1):31–78, 2006.

K. Tsirlis, V. Lagani, S. Triantafillou, and I. Tsamardinos. On scoring maximal ancestral graphs with the max-min hill climbing algorithm. *Int. J. Approx. Reason.*, 102: 74–85, 2018.

P. van Beek and H. Hoffmann. Machine learning of Bayesian networks using constraint programming. In *CP*, volume 9255 of *LNCS*, pages 429–445. Springer, 2015.

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

Zhalama, J. Zhang, F. Eberhardt, W. Mayer, and M. J. Li. ASP-based discovery of semi-Markovian causal models under weaker assumptions. In *IJCAI*, pages 1488–1494. ijcai.org, 2019.

J. Zhang. Causal reasoning with ancestral graphs. *J. Mach. Learn. Res.*, 9:1437–1474, 2008a.

J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artif. Intell.*, 172(16-17):1873–1896, 2008b.

J. Zhang and P. Spirtes. A transformational characterization of Markov equivalence for directed acyclic graphs with latent variables. In *UAI*, pages 667–674. AUAI Press, 2005.