Integer Programming Based Methods and Heuristics for Causal Graph Learning

Sanjeeb Dash **João Gonçalves** **Tian Gao**
IBM Research, Yorktown Heights, NY 10598, USA

Abstract

Acyclic directed mixed graphs (ADMG) – graphs that contain both directed and bidirected edges but no directed cycles – are used to model causal and conditional independence relationships between a set of random variables in the presence of latent or unmeasured variables. Bow-free ADMGs, Arid ADMGs, and Ancestral ADMGs (AADMG) are three widely studied classes of ADMGs where each class is contained in the previously mentioned class. There are a number of published methods – primarily heuristic ones – to find score-maximizing AADMGs from data. Bow-free and Arid ADMGs can model certain equality restrictions – such as Verma constraints – between observed variables that maximal AADMGs cannot. In this work, we develop the first exact methods – based on integer programming – to find score-maximizing Bow-free and Arid ADMGs. Our methods work for data that follows a continuous Gaussian distribution and for scores that linearly decompose into the sum of scores of c-components of an ADMG. To improve scaling, we develop an effective linear-programming based heuristic that yields solutions with high parent set sizes and/or large districts. We show that our proposed algorithms obtain better scores than other state-of-the-art methods and return graphs that have excellent fits to data.

1 INTRODUCTION

Bayesian networks (BNs), or directed acyclic graphs (DAGs), are often used to model causal relationships: a directed edge from node i to another node j in the graph indicates that a change in variable i will cause a change in variable j. In score-based BN learning methods a score such as the Bayesian information criterion (BIC) is used to measure how well a graph explains the data at hand, and a discrete search procedure is used to learn a graph with a high score, or even a provably maximum score (Yuan and Malone, 2013; Cussens et al., 2016; Gao and Wei, 2018). Constraint-based structure learning algorithms use conditional independence tests to decide on the existence of edges between all pairs of variables; see Spirtes et al. (2000) and Pearl (2000). In situations that involve latent or unmeasured variables, DAG models cannot capture all causal relationships, and acyclic directed mixed graphs (ADMG) are used instead. These contain directed and bidirected edges (which are used to indicate correlations between a pair of variables caused by latent confounding variables).

Ancestral ADMGs or AADMGs were proposed by Richardson and Spirtes (2002) as a generalization of DAG models that are closed under marginalization and can capture conditional independence relations among observed variables. The FCI algorithm (Spirtes et al., 2000; Zhang, 2008) and the conservative FCI (cFCI) (Ramsey et al., 2012) are constraint-based algorithms that can learn AADMGs. For data that follows a continuous Gaussian distribution, some heuristic score-based approaches are given in (Triantafillou and Tsamardinos, 2016; Tsirlis et al., 2018; Bernstein et al., 2020; Chobtham and Constantinou, 2020). An exact score-based method based on solving an integer programming (IP) formulation is proposed in Chen et al. (2021).

Maximal ancestral ADMGs can model all ordinary conditional independence constraints, but cannot capture more general non-parametric equality constraints such as Verma constraints (Verma and Pearl, 1990). Evans (2018) showed that general ADMGs can capture such equality constraints for discrete observed random variables; however it can be difficult to learn such models. For Gaussian random variables, a widely studied class of models are *linear structural equation models with correlated errors* (SEM) associated with an ADMG. These

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

are quite popular in the social and behavioral sciences (e.g., psychology) (Tarka, 2018) and high-quality software such as LISREL (Jöreskog, 1973; Jöreskog and Sörbom, 1978) and AMOS (Arbuckle, 1989) are available to learn such models. Only certain subclasses of ADMGs (and associated SEMs) are known to be identifiable: bow-free ADMGs are almost-everywhere identifiable (Brito and Pearl, 2002), and arid ADMGs are globally identifiable and form a smooth curved exponential family (Drton et al., 2010; Shpitser et al., 2018). We focus on causal discovery problems for bow-free and arid ADMGs. Existing greedy-search methods such as GBAP(Nowzohour et al., 2017) or the continuous optimization approach in (Bhattacharya et al., 2020) – which we call ABIC – lack consistency guarantees.

Main Contributions We propose an integer programming (IP) formulation and solution algorithm to find provably score-maximizing bow-free ADMGs and arid ADMGs that can be composed of *c-components* in an input candidate list when the input data is assumed to follow a Gaussian distribution and the graph score decomposes into the sum of c-component scores. If the input candidate list contains all possible c-components over a collection of nodes, then our approach yields an exact algorithm to find bow-free (arid) ADMGs with optimal scores (and thus guaranteed to be at least as good as scores returned by ABIC or GBAP). We build upon the IP formulations for DAG learning in GOBNILP (Bartlett and Cussens, 2017) and AADMG learning in Chen et al. (2021), using similar sets of variables as in the latter paper (Chen et al., 2021), but add new constraints to enforce bow-freeness or aridity and novel cutting planes to speed up the solution of the IP formulations. Based on a polyhedral analysis of the convex hull of solutions of these IP formulations, we assert that the cutting planes we derive are not dominated by other cutting planes when the number of nodes is small.

We show experimentally that when *district* (a component of bidirected edges) and parent set sizes are bounded and the enumeration and scoring of all possible c-components is tractable, our IP-based methods obtain graphical structures that have as good scores as the gound-truth graph and excellent precision and recall. To deal with ground-truth graphs that have large parent set sizes or large districts, we develop a linear programming (LP) based heuristic that takes an input list $\mathcal{C}$ of c-components and adds relevant 'large' c-components to get an output list $\mathcal{C}'$. When we use our previous IP formulation to find optimal graphs that can be composed of c-components in $\mathcal{C}'$, the resulting solutions often have better scores than solutions returned by existing state-of-the-art methods such as GBAP or ABIC, and also fit the data very well as measured by other metrics. Further, our combined heuristic has better run-time scaling with problem size.

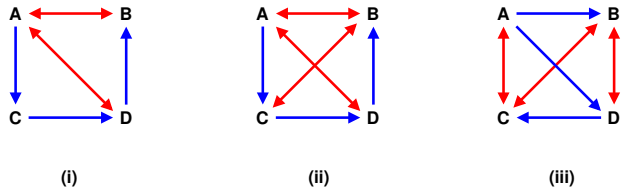

Figure 1: Three Bow-Free Graphs With Four Nodes That Have Almost Directed Cyles: (i) Has a Verma Constraint; (ii) Is Bow-Free but Non-Arid; (iii) Is Arid.

2 LEARNING PROBLEM

2.1 Preliminaries

We depict a directed edge by an arrow $\rightarrow$ and a bidirected edge by $\leftrightarrow$. A directed graph contains only directed edges, while a directed mixed graph (DMG) has both directed and bidirected edges. Consider a DMG $\mathcal{G} = (V, E_d, E_b)$ with node set V, directed edge set $E_d \subseteq \{(i,j) : i,j \in V, i \neq j\}$ and bidirected edge set $E_b \subseteq \{\{i,j\} : i,j \in V, i \neq j\}$. We call node a an ancestor of node b in $\mathcal{G}$ if there is a directed path from a to b in $\mathcal{G}$ or $a = b$. We call node a a spouse (resp., parent) of node b in $\mathcal{G}$ if there is a bidirected edge between a and b (resp., a directed edge from a to b) in $\mathcal{G}$. We denote the set of ancestors, the set of parents and the set of spouses of node a in $\mathcal{G}$ by $\text{an}_{\mathcal{G}}(a)$, $\text{pa}_{\mathcal{G}}(a)$ and $\text{sp}_{\mathcal{G}}(a)$, respectively. Similarly, given a set $D \subset V$ of nodes in $\mathcal{G}$, we denote the union of parents of nodes in D by $\text{pa}_{\mathcal{G}}(D)$. An acyclic DMG (or ADMG) contains no directed cycle $(a \rightarrow c \rightarrow \ldots \rightarrow b \rightarrow a)$. An ADMG is called an *AADMG* if it does not contain an almost directed cycle $a \rightarrow c \rightarrow \ldots \rightarrow b \leftrightarrow a$ (i.e., $\{a,b\} \in E_b$ is a bidirected edge and $a \in \text{an}_{\mathcal{G}}(b)$).

A *bow-free* graph is an ADMG that does not have a pair of nodes a, b such that $a \rightarrow b$ and $a \leftrightarrow b$ (such a structure is called a bow). The definition of an *arid* graph is more involved. Given a set of nodes $S \subseteq V$, we say a rooted arborescence on S is a subgraph T of $\mathcal{G}$ consisting of directed edges that define a rooted, directed tree on S: there is one node r in T with outdegree 0 but positive indegree, all other nodes in T have outdegree 1, and T is acyclic and connected. We say that an ADMG is arid if there is no subset of nodes S in $\mathcal{G}$ such that the induced bidirected subgraph of $\mathcal{G}$ on S forms a connected component, and the induced directed subgraph on S contains a rooted arborescence spanning S. Note that an arid graph does not contain a bow, and therefore all arid graphs are bow-free. Further, any AADMG is arid. See Figure 1.

The district for node a is defined as the connected component containing a in the subgraph of $\mathcal{G}$ induced by all bidirected edges i.e.,

$$\{b \in V : b \leftrightarrow \ldots \leftrightarrow a \text{ in } \mathcal{G} \text{ or } a = b\}.$$

The districts define a set of equivalence classes of nodes in $\mathcal{G}$. In a DAG, each district consists of a single node, and each node forms a single-node district. Given a district D of $\mathcal{G}$, we define a DMG $\mathcal{G}_D$ with node set $D \cup \text{pa}_{\mathcal{G}}(D)$. The bidirected edges in $\mathcal{G}_D$ are the bidirected edges in $\mathcal{G}$ that connect pairs of nodes in D. The directed edges in $\mathcal{G}_D$ are the directed edges in $\mathcal{G}$ that point to nodes in D. We call $\mathcal{G}_D$ the subgraph of $\mathcal{G}$ implied by the district D and call it a c-component. For a c-component C, we let D_C stand for the district of C, and let $P_C = \text{pa}_C(D_C)$, and $P_{C,i} = \text{pa}_C(i)$ where i is a node in D_C. See Figure 2

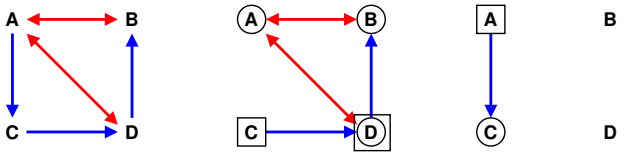

Figure 2: The bow-free graph from Figure 1 followed by the two c-components that form the graph. District nodes are circled, while parent nodes are indicated by squares. Node D in the first c-component is in the district but also a parent of a different district node.

The conditions of being bow-free or arid are a property of the districts of a graph. The following result is an important part of our algorithmic approach.

Proposition 2.1. *A DMG $\mathcal{G}$ is bow-free (or arid) if and only if the subgraph of $\mathcal{G}$ induced by each district is bow-free (or arid, respectively).*

All proofs are included in the appendix.

2.2 Graphical Models

We assume the ground-truth data comes from a multivariate Gaussian distribution [1] and is well-represented by the following structural equation model (SEM) (Pearl, 2000):

$$X = BX + \epsilon. \tag{1}$$

Here B is a $d \times d$ matrix, $X = (X_1, \ldots, X_d)^T$ are variables of the model, and $\epsilon = (\epsilon_1, \ldots, \epsilon_d)^T$ is a noise vector following a multivariate Gaussian distribution $N(0, \Omega)$. We assume that the nonzero entry pattern in the matrices B and Ω is modeled by a bow-free or arid ADMG $\mathcal{G}$ that satisfies

$$B_{ij} = 0 \text{ if } i = j \text{ or } j \to i \text{ is not in } \mathcal{G}, \tag{2}$$
$$\Omega_{ij} = 0 \text{ if } i \leftrightarrow j \text{ is not in } \mathcal{G}. \tag{3}$$

The absence of a directed cycle in $\mathcal{G}$ means that there is a topological ordering of nodes in $\mathcal{G}$/variables in X such that there is an equation in (1) for every variable X_i of the form $X_i = \sum_{j<i} B_{ij} X_j + \epsilon_i$. In other words, the value of ith variable X_i in a data sample can be computed uniquely from the value of its parent variables in $\mathcal{G}$ and the value of ϵ_i. It is well-known that X follows a multivariate Gaussian distribution $N(0, \Sigma)$ where

$$\Sigma = (I - B)^{-T} \Omega (I - B)^{-1}. \tag{4}$$

Here we say that the distribution $N(0, \Sigma)$ (or the covariance matrix Σ) is consistent with $\mathcal{G}$. Further, for a specific arid graph, a consistent covariance matrix Σ always has a unique decomposition in terms of B and Ω matrices in equation (4), whereas this is almost always true for bow-free graphs. However, a Gaussian distribution can be consistent with multiple bow-free graphs and there is no complete theory (Nowzohour et al., 2017) identifying this class of graphs. Given a finite set of input data samples from $N(0, \Sigma)$ (and neither B nor Ω are given), our goal is to find a bow-free (arid) ADMG $\mathcal{G}$ such that there exists an associated parametrization (B, Ω) satisfying equations (2) and (3) that fits the data at least as well as other bow-free (arid) graphs. As the sample covariance matrix $\tilde{\Sigma}$ may differ from Σ, it may not be possible to recover the ground-truth Σ (in the sense of equation (4)).

2.3 Graph Scores

We assign each graph $\mathcal{G}$ a score that indicates how well the graph structure represents the observed data (in the sense described above) and find the graph with the maximum score. We primarily use the Bayesian Information Criterion (BIC) (Schwarz, 1978) score, which is given, for a graph $\mathcal{G}$, by

$$\textbf{BIC}_{\mathcal{G}} = 2 \ln(l_{\mathcal{G}}(\hat{\Sigma})) - \ln(n)(2|V| + |E|). \tag{5}$$

Here $\hat{\Sigma}$ is the maximum likelihood estimate of Σ given the graphical model $\mathcal{G}$ and $\ln(l_{\mathcal{G}}(\hat{\Sigma}))$ is the associated log-likelihood while n denotes the number of samples, $|V|$ and $|E|$ denote the number of nodes and edges in $\mathcal{G}$, respectively. According to Nowzohour et al. (2017), $\ln(l_{\mathcal{G}}(\hat{\Sigma}))$ can be decomposed by c-components in $\mathcal{G}$. Let $\mathcal{C}$ denote all c-components of $\mathcal{G}$. For c-component C, let $\hat{\Sigma}_C$ be the maximum log-likelihood and $\hat{\sigma}_j$ be the

[1] We focus on multivariate Gaussian cases in this work as the scoring function can be factorized, although our work may also apply to discrete cases per different factorization rules (Richardson, 2009).

diagonal entry of $\hat{\Sigma}_C$ corresponding to node j. Then

$$\ln(l_{\mathcal{G}}(\hat{\Sigma})) = -\frac{n}{2}\sum_{C\in\mathcal{C}}\Big[|D_C|\ln(2\pi)+ \log(\frac{|\hat{\Sigma}_C|}{\prod_{j\in P_C\setminus D_C}\hat{\sigma}_j^2})+ \frac{n-1}{n}tr(\hat{\Sigma}_C^{-1}S_C - |P_C\setminus D_C|)\Big]. \quad (6)$$

We focus on BIC scores for the following reasons: (1) Asymptotically, the true model is assigned the highest BIC score among all models (Hannan and Quinn, 1979); (2) It is common to use BIC scores to compare models (Scanagatta et al., 2018; Jaakkola et al., 2010) (3) The BIC score decomposes into the sum of scores of c-components and this is essential for our proposed IP method. On the other hand, the process of estimating parameters in (6) via maximum likelihood is notably intricate. Given a predefined graph structure and the empirical covariance matrix, a standard approach is to use Residual Iterative Conditional Fitting (RICF) (Drton et al., 2009). However, RICF may only converge to a local solution. Moreover, the above BIC score may not approximate the Bayes marginal likelihood well when the regularity property (which requires that the mapping from parameters to a probability distribution is unique and the Fisher information matrix is always positive definite (Watanabe, 2013; Drton and Plummer, 2017; Liu and Suzuki, 2024)) is violated, as we are not aware of any BIC scores for non-regular AADMGs. This limitation may impact practical applications of the learning algorithms, and further research is needed to address this limitation. Our approach should be applicable to any new scores, as the novelty of our proposed method lies in the optimization procedure and is score-agonistic (as long as the score is decomposable and consistent).

Another way of evaluating solution quality in DAG learning when the ground truth graph is known is to calculate precision and recall for the directed edges in a graph. These metrics do not capture accuracy perfectly for bow-free graphs. For DAG/AADMG learning, there is a complete theory of equivalence classes of graphs that can produce the same set of joint distributions (or matrices Σ in (4)). However, Nowzohour et al. (2017) say that "a graphical characterization of the (distributional) equivalence classes for BAP [bow-free graphs] models is not yet known" (other than that the graph skeleton is invariant) and this "makes it difficult to evaluate simulation results, since the graphs corresponding to the ground truth and the optimal solution may be distinct and yet still represent the same model." Further, precision and recall do not measure how well the model fits the data. RMSEA (root mean square error of approximation) (Steiger, 1990) is a popular (Savalei, 2021) measure of quality of model fit to data and we also report these values. The AMOS software (IBM, 2022) reports over a dozen model fit metrics.

If a graph score (e.g., BIC) can be decomposed into the sum of scores of its c-components, one can solve the learning problem for bow-free (or arid) graphs by enumerating all combinations of c-components that correspond to different bow-free (or arid) graphs. By proposition 2.1, one need only enumerate all bow-free (or arid) c-components for bow-free (or arid) graph learning. Since the number of possible c-components grows exponentially with the number of nodes n, the above approach is impractical for large problem dimensions. A common practice is to only consider graphical models where each node has a bounded number of parents. In Chen et al. (2021), the authors bound the district size and the number of parents of each node in a district to limit the number of relevant c-components to a polynomial number (as a function of n, the number of nodes in the graph). This idea can be used to make a discrete search approach more tractable. The model we propose can handle general c-components, and in our experiments we show some results with all candidate c-components with 'bounded size' so that the model can be solved within a reasonable amount of time.

A standard approach in graphical model learning when scores are decomposable is to reduce the size of the problem by pruning the list of candidate c-components taking into account score values. As removing directed and bidirected edges cannot create new bows or new rooted arborescences, it is clear that removing edges from c-components in an optimal graph does not affect optimality as long as the score of the c-component does not decrease. This is our first type of pruning. For example, if the c-component $\{A\}\to B\leftrightarrow C\leftarrow\{D\}$ (here $\{A\}\to B$ means that the parent set of node B consists of node A; $B\leftrightarrow C$ means that there is a bidirected edge between B and C and both belong to the district of the c-component) has a score lower than that of $\emptyset\to B\leftrightarrow C\leftarrow\{D\}$, we can prune the first c-component. See the Appendix for the precise pruning rules we use and a proof of their validity.

3 IP METHODS FOR BOW-FREE AND ARID GRAPHS

Bartlett and Cussens (2017) developed the state-of-the-art IP-based DAG learning solver GOBNILP. Chen et al. (2021) built upon this work and developed an IP formulation for AADMG learning by (1) adding variables for c-components with multi-node districts; (2) redefining *cluster inequalities* for DAG learning in terms of new variables; (3) developing inequalities to

disallow almost directed cycles (forbidden structures for AADMGs). We keep (1) and (2), update pruning rules, and do preprocessing on input c-components to disallow those with bows (for bow-free learning) and c-trees (for arid learning) due to Proposition 2.1 and obtain a valid IP formulation. To strengthen this formulation we derive new classes of *cutting planes* discussed below.

Assume we have n random variabes, represented by n nodes, and we want to learn a maximum-score bow-free graph on these n nodes that has all its c-components contained in an input list of c-components $\mathcal{C}$. Let z_C be a binary variable for each $C \in \mathcal{C}$, where $z_C = 1$ if and only if C is used in the learned graph. Let s_C be the local score of c-component C. Let $\mathcal{G}(z)$ stand for the learned DMG indicated by the vector of variables $z = (z_C)_{C\in\mathcal{C}}$. We formulate the bow-free learning problem as the optimization problem:

$$\max \sum_{C\in\mathcal{C}} s_C z_C \tag{7}$$

$$s.t. \sum_{C:i\in D_C} z_C = 1,\ i \in \{1,\dots,d\} \tag{8}$$

$$\mathcal{G}(z) \text{ contains no directed cycles} \tag{9}$$

$$\mathcal{G}(z) \text{ contains no bows} \tag{10}$$

$$z_C \in \{0,1\} \tag{11}$$

Constraints (8) enforce the condition that each node belongs to the district of exactly one c-component. We next discuss how to enforce (9) by an exponential size family of linear inequalities and how to enforce (10).

3.1 IP Formulation

A class of anti-cycle constraints named *cluster inequalities* was introduced in Jaakkola et al. (2010). Bartlett and Cussens (2017) showed that cluster inequalities often yield tight integer programming formulations for DAG learning problems. Given that there are exponentially many cluster inequalities, efficiently finding relevant cluster inequalities is essential in this context. A variant, called *strengthened cluster inequalities*, were introduced in the context of ADMG learning by Chen et al. (2021) and have the following form:

$$\sum_{C\in\mathcal{C}:\exists i\in S\cap D_C, P_{C,i}\cap S=\emptyset} z_C \geq 1,\quad \forall S \subseteq \{1,2,\dots,d\}. \tag{12}$$

Inequalities (12) encode the constraint that for any set S of nodes in an acyclic graph, there must exist a node in S whose parent set has no intersection with S.

It was proved in Chen et al. (2021) that if $z \in \{0,1\}^{\mathcal{C}}$ satisfies (8) and (12), then z corresponds to an ADMG, i.e., $\mathcal{G}(z)$ contains no directed cycles. Let BIP($\mathcal{C}$) be the integer programming problem associated with a list of candidate c-components $\mathcal{C}$ (which we abbreviate to BIP if $\mathcal{C}$ is clear from the context):

$$\max \sum_{C\in\mathcal{C}} s_C z_C \tag{13}$$

$$s.t. \sum_{C:i\in D_C} z_C = 1,\ \forall i \in \{1,\dots,d\} \tag{14}$$

$$\sum_{C\in\mathcal{C}:\exists i\in S\cap D_C, P_{C,i}\cap S=\emptyset} z_C \geq 1, \quad \forall S \subseteq \{1,2,\dots,d\} \tag{15}$$

$$z_C = 0 \ \text{ if } C \text{ contains a bow} \tag{16}$$

$$z_C \in \{0,1\} \tag{17}$$

We let ARIP be the model obtained from BIP when we replace the condition "C contains a bow" in (16) with "C is non-arid". Proposition 2.1 implies that $\mathcal{G}(z)$ must be bow-free (arid) if each c-component in $\mathcal{G}(z)$ is bow-free (arid). This latter condition is implied by constraint (16). We have the following theorem.

Theorem 3.1. *BIP (ARIP) is a valid IP formulation of the optimal bow-free (arid) graph learning problem: solving BIP (ARIP) leads to a score-maximizing bow-free (arid) graph.*

Let BLP be the LP relaxation of BIP obtained by replacing (17) with $0 \leq z_C \leq 1$ for all $C \in \mathcal{C}$. An important question is whether BLP is a good approximation to the convex hull of its integral solutions (or solutions of BIP); call this convex hull cBIP. We derive new classes of *cutting planes* for BIP i.e., linear inequalities satisfied by all solutions of BIP but not by all solutions of BLP. Adding cutting planes to BLP yields a tighter approximation of cBIP than BLP itself. We demonstrate the computational value of some of these cutting planes; the resulting IP formulation can be solved 30% faster (for one test set) by standard IP solvers than BIP. The first class we give generalizes the *set packing* constraints in (Bartlett and Cussens, 2017, Eqn 4) that rule out directed cycles in DAGs.

Theorem 3.2. *Let $\emptyset \neq S \subseteq V$ and let T be the set of c-components $C \in \mathcal{C}$ such that $D_C \cap S \neq \emptyset$, and any node in $S \setminus D_C$ is a parent of a node in $D_C \cap S$. Then $\sum_{C\in T} z_C \leq 1$ is a cutting plane for BIP.*

Theorem 3.3. *Let $\emptyset \neq S \subseteq V$ with $|S|$ odd and T be the set of c-components $C \in \mathcal{C}$ such that $|D_C \cap S| \geq 2$. Then $\sum_{C\in T} \lfloor |D_C\cap S|/2 \rfloor z_C \leq \lfloor |S|/2 \rfloor$ is a cutting plane for BIP.*

A natural question in polyhedral combinatorics is whether a given class of cutting planes is strongest possible or can be dominated by other cutting planes. If the cutting planes in Theorems 3.2 and 3.3 define facets of cBIP, then no other cutting planes can dominate them. Following (Cussens et al., 2017), we define

a closely related polyhedron (to cBIP) as follows: drop the c-component variables with single node districts and empty parent sets from BIP – thereby converting equations (8) to ≤ 1 inequalities – and let fBIP(d) be the convex hull of solutions when we have d nodes and all other possible c-component variables. We call fBIP(d) the family variable polytope on d nodes.

Theorem 3.4. *The inequality in Theorem 3.2, when $|S| = 2$ or $|S| = 3$, and the inequality in Theorem 3.3, when $|S| = 3$, define facets of fBIP(3).*

3.2 Solving the IP formulation

Though the inequalities defined in Theorems 3.2 and 3.3 are valid for different sizes of S, we only consider S with $|S| = 2$ or 3 in the case of Theorem 3.2 and $|S| = 3$ for Theorem 3.3 in our computations. Let BLP/BIP augmented with these inequalities be denoted by BLP$_+$/BIP$_+$. Note that BIP$_+$ and BIP have the same set of solutions (and optimal solutions). A similar property does not hold for BLP$_+$ and BLP.

Let BLP- be the LP obtained from BLP by removing the inequalities (15). To solve BIP$_+$, our algorithm first solves BLP-; assume the optimal solution is z^*. We then use a heuristic (described in Chen et al. (2021)) to find any constraints of the form (15) that are violated by z^*. If any violated inequalities of this type are found, they are added to the previous LP, which is then resolved to obtain a new optimal solution. This process is repeated till no violated lifted cluster inequalities can be found (and we have approximately optimized over BLP). We then look for all violated inequalities of the types given in Theorems 3.2 and 3.3 for all $|S|$ with $2 \leq |S| \leq 3$ and repeatedly add such violated inequalities till we do not find any. At this point we employ a branch-and-bound strategy to solve BIP$_+$. Whenever an integral solution is obtained during the branch-and-bound process, the code tests if the integer solution violates any lifted cluster inequalities (this test can be performed in polynomial time). On termination of the branch-and-bound process one has an optimal bow-free graph that can be composed from the input list of bow-free c-components in $\mathcal{C}$.

The upper bounds on the optimal value of BIP obtained from BLP and BLP$_+$ differ. In the Appendix, we show that the integrality gap of BLP (= |BLP value - BIP value| / |BIP value|) is roughly 4.6% for a specific dataset; the integrality gap obtained with BLP$_+$ is 3.7%. Further, we take less time when we use BLP$_+$.

3.3 Exact Optimality for Bow-Free and Arid Graphs

It is easy to find optimal bow-free graphs when the number of nodes is 3 or 4. If $\mathcal{C}$ is the list of all possible 1,226 c-components on 4 node DMGs, then solving BIP($\mathcal{C}$) returns the optimal bow-free graph. The number of all c-components is much larger for graphs with 5 or more nodes (there are 83,897 c-components on 5 nodes where each district node has at most one parent) and BIP is difficult to solve when $|\mathcal{C}|$ is very large.

To formulate ARIP, we need to test for the aridity of each input c-component, or equivalently, the graph induced by the district of a c-component. As bow-free graphs with up to 3 nodes are arid, a non-arid c-component must have at least 4 nodes in its district. For c-components with districts having 4 or more nodes, we first check for a bow or a directed cycle on the district nodes as existence of a bow or a directed cycle implies non-aridity. Assuming neither structure is present, we check if the c-component contains a c-tree, which is a subgraph of the c-component containing a spanning tree T_1 of bidirected edges and a spanning rooted arborescence T_2 formed by the directed edges on the nodes in T_1. Each of these structures can be checked by BFS/DFS style operations. If no c-tree exists, then the c-component is arid.

In Figure 1 we give three bow-free graphs that are not AADMGs. In Figure 1(i) we give an example from Bhattacharya et al. (2020) that encodes a Verma constraint linking nodes B and C. We can recover this graph from a specific data sample. For a different data sample generated from Figure 1(i), if we change the scoring function by not penalizing number of edges, then the optimal bow-free graph is (ii), while the optimal arid graph is (iii) with a slightly lower BIC score.

4 LP HEURISTIC

BIP is hard to solve when $\mathcal{C}$ is too large, e.g., when all possible bow-free c-components are in $\mathcal{C}$. To get a tractable IP, we restrict district sizes of c-components in $\mathcal{C}$ as well as parent set sizes. If the ground-truth graph has larger c-components than those in $\mathcal{C}$, then BIP($\mathcal{C}$) will not recover the ground-truth. We develop a heuristic LPH that dynamically generates new c-components with larger parent set and district sizes and augments $\mathcal{C}$. The idea is to treat BLP$_+$ as a good approximation of BIP, the active c-components in a BLP$_+$ solution as important c-components, and then create a list of neighboring c-components (by adding edges) and augmenting $\mathcal{C}$ with this list. We start off with an input list $\mathcal{C}_0 = \mathcal{C}$ of bounded size c-components and find an optimal solution $\bar{z}$ to BLP$_+(\mathcal{C}_i)$ for $i = 0$.

For each c-component C with $\bar{z}_C > 0$: (1) we add a node to the parent set of a district node; (2) we add a spouse to a district node; (3) we convert a parent node to a spouse. We apply these operations to each district node and to each parent node to create a new list of c-components from C. We add all newly generated c-components to $\mathcal{C}$ to create $\mathcal{C}_1$. We then solve $\text{BLP}_+(\mathcal{C}_i)$ for $i = 1, 2, \ldots, k$ for k iterations. Finally, we solve $\text{BIP}_+(\mathcal{C}_k)$ to optimality. As our heuristic augments $\mathcal{C}$, the objective value of the final solution can be no worse than the initial objective function value. Our heuristic is different from the typical hill climbing (HC) procedure (Tsirlis et al., 2018) which starts with a candidate graph, and then iteratively adds, removes, or changes the orientations of some edges.

5 EMPIRICAL STUDY

We focus on bow-free graphs. We start with a (randomly generated or real-life) bow-free graph, then generate structural equations (1) that are consistent with the graph (defined by matrices B in (2) and Ω in (3)), and then generate samples drawn from the distribution (4) implied by the structural equations. We create a list of c-components and calculate their scores from the samples. These scores are inputs to our exact IP models and heuristic approach that search for graphs maximizing the BIC score. In additional to the final solution BIC score, we also report RMSEA values (calculated using the IBM Amos software IBM (2022)), and the F-score of the precision and recall values (see the appendix for a definition) when compared to the skeleton of the ground-truth graph (ignoring edge directions).

We compare our code[2] against the bow-free graph learning codes from Bhattacharya et al. (2020) and Nowzohour et al. (2017). We refer to these codes as ABIC and GBAP. ABIC uses an approximate BIC score (hence the name) during optimization as a proxy for the BIC score. We take the final solution graphs returned by these codes and use our BIC score calculation routines before reporting results. Let δBIC(model1, model2) = BIC(model 1) - BIC(model 2). Raftery Raftery (1995) proposes that δBIC provides evidence in favor of model 1 as follows: very strong if $\delta\text{BIC} > 10$, strong if $6 < \delta\text{BIC} < 10$, positive if $2 < \delta\text{BIC} < 6$, weak if $\delta\text{BIC} < 2$. We run our algorithms on a 2.8 GHz Linux machine with 60 cores (referred to as Machine 1 in the tables) but use at most 32 cores (mainly while solving BIP). We use CPLEX IBM (2019) to solve LPs and IPs. We let ABIC use all 60 cores. We run GBAP on a 2.4 GHz machine (referred to as Machine 2) with 40 cores and, for the purpose of comparing running times, we also run our algorithms on that machine.

[2] `https://github.com/IBM/causal-graph-learning`

We first experiment with randomly generated bow-free graphs with $d = 10$ nodes. After generating a graph, we generate B in equation (2) and Ω in equation (3) as described in the Appendix. We mostly follow the settings used in Bhattacharya et al. (2020) other than increasing the influence of bidirected edges. We then draw n samples of the error vector ϵ from the multivariate Gaussian distribution $N(0, \Omega)$ and apply equation 1 to obtain n samples of X. We generate four different categories of graphs, each with 10 randomly generated graphs. First, we generate ground-truth graphs that can be decomposed into 'small' c-components (with small districts and small parent sets; see Section 5.1). We then generate graphs where all districts have a single node but parent sets can be large. Existing IP-based DAG learning algorithms work with bounded size parent sets; we will show that our proposed LP-based heuristic can obtain better scores. We then consider bow-free graphs with large districts and small parent sets. Finally, we considered bow-free graphs with both large districts and large parent sets. See the Appendix.

In order to recover a graph with either large parent sets or large districts using our exact algorithm, one would have to enumerate all 'large' c-components, and solve BIP for this very large list of c-components. This is impractical. We instead run our exact algorithm on an input list of 'small' c-components. We also run LPH and let it dynamically generate large c-components.

5.1 Sparse Bow-Free Graphs with Small C-Components

For the first category of randomly generated graphs, we restrict parent set sizes for single node districts to at most 3, restrict districts to have at most 2 nodes and nodes in 2 node districts to have at most 2 parents. Let $\mathcal{C}$ be the list of all c-components satisfying the size bounds. We optimize over the set of AADMGs and bow-free graphs (this set includes the ground-truth) that can be composed using $\mathcal{C}$.

In Table 1, we give averages across 10 instances of several metrics to compare the optimal AADMG (column AA) and bow-free graphs (column BF) that can be constructed from $\mathcal{C}$ with those of the ground-truth (column Orig.) and also with the graphs returned by ABIC and GBAP. We first give averages for the BIC score (the individual scores for each instance are given in Table 4 in the Appendix), then for Δ(BIC) (e.g., for BF, Δ(BIC) stands for BIC(ORIG) - BIC(BF)). We then give standard deviations of Δ(BIC), average RMSEA scores and their standard deviation, average F-scores, and running time in minutes.

The BIC scores for the optimal bow-free graphs are *at least as good as* the ground-truth scores (and sometimes

Table 1: Scores and Running Time (Minutes) for 10 Sparse Randomly Generated 10-Node Graphs With Small C-Components and Parent Sets

Metric	Orig.	ABIC	GBAP	AA	BF
Avg. BIC	-18059.7	-18228.8	-18200.6	-18071.3	-18058.8
Δ(BIC)		169.1	140.8	11.6	-1.0
Std. Δ(BIC)		435.5	224.2	17.7	1.9
Avg. RMSEA	.010	.027	.014	.021	.008
Std. RMSEA	.009	.019	.008	.023	.009
Avg. F-score		.891	.918	.927	.972
Avg. Time (Machine 1)		123		19	14
Avg. Time (Machine 2)			82	16	14

strictly larger than AADMG scores indicating a difference in these graph classes) and *always as good* as the heuristically obtained ABIC and GBAP scores – see Table 4. The quality of fit to the samples is essentially as good as the original model (as measured by average RMSEA values). A model fit is considered to be excellent if RMSEA < 0.05. For some graphs in Table 1, the skeleton of the solution graph matches the ground-truth skeleton exactly when we obtain the same score as the ground truth. However the directed and bidirected edge sets *do not match those of the ground-truth*. These experiments show that if our list of candidate c-components contains those of the ground-truth solution, we can always recover a graph with at least as good a BIC score. Table 5 in the appendix shows that we get better precision, recall, and F-score values with BF than with ABIC, GBAP, and AA.

In Table 2, we give our results on the 3rd and 4th categories of graphs. ABIC takes about 2 hours/4 hours for the 3rd/4th categories, respectively. GBAP takes about 1.5 hours. We run LPH with a time limit of an hour to solve BIP. LPH produces the best average scores and exhibits less variability. Using Raftery's criteria on the the 3rd category of graphs, there is strong evidence in favor of LPH models versus GBAP models for 2 instances, and the reverse is true for 2 instances; these numbers are 7 and 2, respectively for the 4th category. Comparing LPH to ABIC, these numbers are 7 (LPH >> ABIC) and 0 (ABIC >> LPH) for both the 3rd and 4th categories. Also see the appendix for results on 20 node graphs; our code and ABIC still get reasonable results (ours are better than ABIC), while GBAP does not terminate within 2 days.

5.2 Real-World Graphs

We also experimented with variants of two real-world DAGS, `Barley` and `Insurance`, downloaded from Scutari (2024). The nodes correspond to discrete random variables in the original datasets. We treat two variables/nodes as latent and drop them and connect the child nodes with a tree of bidirected edges. We then generate a single data set with 1000 samples using our default parameters creating a continuous model. For `Insurance`, we drop `Age` and `Mileage` to get a graph with 25 nodes. For `Barley`, we drop `jordtype` and `saatid` to get a graph with 46 nodes. We initialize the c-component list $\mathcal{C}$ for BF/LPH with parent set size 2 for 1-node districts, and parent set size 0 for 2-node districts. We run LPH for 4 hours. ABIC and GBAP struggle to obtain solutions in a reasonable amount of time and do not terminate in a day for `Barley`. For an instance of `Insurance` with 1000 samples, ABIC takes 25 hours to terminate. `Insurance` has low average parent set size and all codes perform reasonably well. LPH obtains an F-score of 88% and RMSEA of 0.015 (compared to the original RMSEA of 0.009). ABIC obtains 78% and 0.030 respecively. LPH obtains a worse solution quality for `Barley` than for `Insurance`.

5.3 Scalability

For DAGs with n nodes and indegree at most k, it is well known that the number of possible c-components (and therefore the number of binary variables) in integer programming formulations such as those of GOBNILP (and in ours) is roughly $O(n^{k+1})$. On the other hand, if one assumes the ground truth is not a DAG, but a bow-free graph with districts of size at most p and the size of each parent set of a node in a district is at most k, then the number of possible c-components is roughly $O(n^{p(k+1)})$. For $p \geq 2$, the number of c-components in the latter case is at least as large as the square of the number of c-components in the DAG case. The number of variables in BIP is the same as the number of c-components in the input candidate list which currently consists of all c-components with at

Table 2: Summary of Results - Scores and Running Times for Graphs With Either Large Parent Sets or Large C-Components

	Table 8				Table 10			
Metric	Orig.	ABIC	GBAP	LPH	Orig.	ABIC	GBAP	LPH
Avg. BIC	-20880	-20903	-20910	-20889	-20005.0	-20081.6	-20200.3	-20014.1
Avg. Δ(BIC)		32.8	30.2	8.8		76.6	195.3	9.1
Std. Δ(BIC)		43.6	67.5	6.3		127.0	254.8	7.3
Avg. RMSEA	.010	.085	.014	.011	.007	.032	.020	.017
Std. RMSEA	.009	.185	.011	.007	.008	.018	.014	.016
Avg. F-score		.854	.920	.875		.866	.845	.841
Avg. Time (Machine 1)		139		45		229		68
Avg. Time (Machine 2)			91	40			81	65

most p nodes in a district and at most k parents per node in a district for some p (usually 2) and k (usually 3) which may differ from the largest district size and parent set size in the ground truth graph. The number of constraints in BLP or BIP is potentially exponential, but in our implementation we dynamically generate only the relevant constraints; in practice we generate a few hundred constraints before we have solved BLP or BLP$_+$ to optimality. We then solve BIP/BIP$_+$ via a branch-and-bound process that enumerates a few thousand branch-and-bound nodes per hour.

6 CONCLUSION

We presented an IP based algorithm that takes an input candidate list of c-components $\mathcal{C}$ and returns a score maximizing bow-free/arid graph that can be composed from $\mathcal{C}$. Our method can handle any graph score that decomposes into the sum of c-component scores, though we only tested BIC scores. If the ground-truth graph has larger c-components than those in $\mathcal{C}$, our LP heuristic LPH is able to augment $\mathcal{C}$ with relevant c-components. Our algorithm yields better BIC scores, RMSEA values, and run-time scaling compared to state-of-the-art score-based methods for bow-free ADMG learning. Our method can handle structure learning of linear Gaussian SEMs given a decomposable scoring function. Scoring general SEMs is a nontrivial problem, but if a tractable scoring function is developed, our method can be applied to general SEMs.

We are able to run LPH on 20 node graphs and even larger graphs; see (see Tables 12 and 13 in the Appendix, and the previous section on real-world graphs). Further, we already obtain a speedup by limiting the size of the initial c-components to be less than the maximum c-component size that is consistent with the ground truth. The main challenge in applying the method to larger graphs is enumerating all c-components with size below a fixed bound. An approach that avoids complete enumeration of small c-components could use a greedy hill-climbing style algorithm (such as M3HC) that starts out with an initial graph and then performs a number of random edge insertions, deletions, direction flips, and swaps between directed and bidirected edges and retains the new graph if the score improves. One could take the c-components generated in the greedy algorithm solution process (or in the final solution) and then give these to LPH. LPH is guaranteed to find a solution with the same score or better. Such a combined heuristic would help LPH start with more "relevant" and "informed" c-components than the current implementation.

References

Arbuckle, J. (1989). Amos: Analysis of moment structures. *The American Statistician*, pages 66–67.

Bartlett, M. and Cussens, J. (2017). Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271.

Bernstein, D., Saeed, B., Squires, C., and Uhler, C. (2020). Ordering-based causal structure learning in the presence of latent variables. In *International Conference on Artificial Intelligence and Statistics*, pages 4098–4108.

Bhattacharya, R., Nagarajan, T., Malinsky, D., and Shpitser, I. (2020). Differentiable causal discovery under unmeasured confounding. *arXiv preprint arXiv:2010.06978*.

Brito, C. and Pearl, J. (2002). A new identification condition for recursive models with correlated errors. *Structural Equation Modeling*, 9(4):459–474.

Chen, R., Dash, S., and Gao, T. (2021). Integer programming for causal structure learning in the presence of latent variables. In *International Conference on Machine Learning*.

Chobtham, K. and Constantinou, A. C. (2020). Bayesian network structure learning with causal effects in the presence of latent variables. *arXiv preprint arXiv:2005.14381*.

Christof, T. and Loebel, A. (1997). PORTA - POlyhedron Representation Transformation Algorithm. `https://porta.zib.de`.

Cussens, J., Haws, D., and Studený, M. (2016). Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*, pages 1–40.

Cussens, J., Järvisalo, M., Korhonen, J. H., and Bartlett, M. (2017). Bayesian network structure learning with integer programming: polytopes, facets and complexity. *JAIR*, pages 185–229.

Drton, M., Eichler, M., and Richardson, T. (2009). Computing maximum likelihood estimates in recursive linear models with correlated errors. *Journal of Machine Learning Research*, 10(10):2329–2348.

Drton, M., Foygel, R., and Sullivant, S. (2010). Global identifiability of linear structural equation models. *Annals of Statistics - ANN STATIST*, 39.

Drton, M. and Plummer, M. (2017). A bayesian information criterion for singular models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(2):323–380.

Evans, R. J. (2018). Margins of discrete bayesian networks. *Annals of Statistics*, 46(6A):2623–2656.

Gao, T. and Wei, D. (2018). Parallel bayesian network structure learning. In *International Conference on Machine Learning*, pages 1671–1680.

Hannan, E. J. and Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41:190–195.

IBM (2019). IBM CPLEX Optimizer, version 12.10.

IBM (2022). IBM SPSS Amos, version 29.

Jaakkola, T., Sontag, D., Globerson, A., and Meila, M. (2010). Learning bayesian network structure using lp relaxations. In *International Conference on Artificial Intelligence and Statistics*, pages 358–365.

Jöreskog, K. (1973). A general method for estimating a linear structural equation system. In *Structural Equation Models in the Social Sciences, A.S. Goldberger and O. D. Duncan (eds.)*, pages 83–112. Academic Press, New York.

Jöreskog, K. and Sörbom, D. (1978). *LISREL: A General Computer Program for Estimation of a Linear Structural Equation System by Maximum Likelihood Methods*. National Educational Resources, Chicago.

Liu, L. and Suzuki, J. (2024). Learning under singularity: an information criterion improving wbic and sbic. *Japanese Journal of Statistics and Data Science*, pages 1–16.

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

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological Methodology*, 25:111–163.

Ramsey, J., Zhang, J., and Spirtes, P. L. (2012). Adjacency-faithfulness and conservative causal inference. *arXiv preprint arXiv:1206.6843*.

Richardson, T. (2009). A factorization criterion for acyclic directed mixed graphs. In *Conference on Uncertainty in Artificial Intelligence*, page 462–470.

Richardson, T. and Spirtes, P. (2002). Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030.

Savalei, V. (2021). Improving fit indices in structural equation modeling with categorical data. *Multivariate Behavioral Research*, 56:390–407.

Scanagatta, M., Corani, G., de Campos, C., and Zaffalon, M. (2018). Approximate structure learning for large bayesian networks. *Machine Learning*, 107:1209–1227.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Scutari, M. (2024). bnlearn - an R package for bayesian network learning and inference. `https://www.bnlearn.com/`.

Shpitser, I., Evans, R. J., and Richardson, T. S. (2018). Acyclic linear sems obey the nested markov property. In *Uncertainty in artificial intelligence: proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 2018. NIH Public Access.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, prediction, and search*. MIT press.

Steiger, J. (1990). Structural model evaluation and modi-fication: An interval estimation approach. *Multivariate Behavioral Research*, 25:173–180.

Tarka, P. (2018). An overview of structural equation modeling: its beginnings, historical development, usefulness and controversies in the social sciences. *Qual Quant.*, 52:313–354.

Tillman, R. and Spirtes, P. (2011). Learning equivalence classes of acyclic models with latent and selection variables from multiple datasets with over-

lapping variables. In *International Conference on Artificial Intelligence and Statistics*, pages 3–15.

Triantafillou, S. and Tsamardinos, I. (2016). Score-based vs constraint-based causal learning in the presence of confounders. In *UAI Workshop on Causation: Foundation to Application*, pages 59–67.

Tsirlis, K., Lagani, V., Triantafillou, S., and Tsamardinos, I. (2018). On scoring maximal ancestral graphs with the max–min hill climbing algorithm. *International Journal of Approximate Reasoning*, 102:74–85.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *In Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*.

Watanabe, S. (2013). A widely applicable bayesian information criterion. *The Journal of Machine Learning Research*, 14(1):867–897.

Yuan, C. and Malone, B. (2013). Learning optimal bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65.

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

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A PROOFS

In this section, we give proofs for the propositions and theorems in the paper, other than Theorem 3.4, which we prove later on. For convenience, we first restate the propositions and theorems to be proven. We note that when we say that a district is bow-free or arid in the main paper, we mean that the subgraph induced by the district is bow-free or arid. Given a DMG $\mathcal{G}$ and subset of nodes S, we refer to the subgraph of $\mathcal{G}$ induced by S to be $\mathcal{G}[S]$; it contains all directed and bidirected edges that contain pairs of nodes in S.

Proposition 2.1. *A DMG $\mathcal{G}$ is bow-free (or arid) if and only if the subgraph of $\mathcal{G}$ induced by each district is bow-free (or arid, respectively).*

Proof. If $\mathcal{G}$ is bow-free, then every induced subgraph is also bow-free. Next assume a graph $\mathcal{G}$ is not bow-free. Then there exists a pair of nodes a, b in $\mathcal{G}$ such that $a \to b$ and $a \leftrightarrow b$, that is, there is a directed edge from a to b in $\mathcal{G}$ and also a bidirected edge in $\mathcal{G}$ between the two nodes. The fact that a and b are connected by a bidirected edge implies that they are in the same component in the subgraph defined by bidirected edges, and therefore in the same district, say D, of $\mathcal{G}$. Now a is a parent of b and the edge from a to b is in the induced subgraph $\mathcal{G}[D]$ implying that $\mathcal{G}[D]$ is not bow-free. Therefore, we have a proof by contradiction of the fact that if all subgraphs of $\mathcal{G}$ induced by districts are bow-free then so is $\mathcal{G}$.

Similarly, if $\mathcal{G}$ is arid, then every induced subgraph must be arid. On the other hand, assume $\mathcal{G}$ is not arid; then it has a c-tree T, i.e., a collection of nodes U contained in $\mathcal{G}$ that has a spanning tree T_1 of bidirected edges from $\mathcal{G}$, and a rooted arborescence T_2 within the directed edges connecting pairs of nodes in U. The first fact implies that U is contained within a single component in the subgraph of $\mathcal{G}$ defined by bidirected edges, and therefore U is contained in a district D of $\mathcal{G}$. As the directed edges in T_2 connect pairs of nodes in U, it means that the c-tree T is contained in the subgraph of $\mathcal{G}$ induced by D and $\mathcal{G}[D]$ is not arid. The result follows. □

As the subgraph induced by a district D is contained in the c-component $\mathcal{G}_D$, we have the following corollary of the above result.

Corollary A.1. *A DMG $\mathcal{G}$ is bow-free (or arid) if and only if each c-component in G is bow-free (or arid, respectively).*

Before giving the proof of Proposition A.2, we first formally define the pruning rules that we use to prune an input list of c-components. Let $\sigma(G)$ be the score of a DMG G. We refer to a "bidirected subgraph" of a DMG as the subgraph formed by taking only the bidirected edges; in such a subgraph, an edge $u \leftrightarrow v$ is a *cut-edge* if removing the edge means that there is no path from u to v via bidirected edges in the resulting subgraph.

P1 Given a c-component C with district D, let $v \in D$ and u be a parent of v. Let C' be the c-component obtained from C after removing the edge $u \to v$. If the score of C' is greater than or equal to the score of C, then delete C from $\mathcal{C}$.

P2 Let C and v be defined as in the previous rule. Let u be a spouse of v. Let C' and C'' be the two c-components that are obtained from C after removing the edge $u \leftrightarrow v$ if $u \leftrightarrow v$ is a cutedge in the bidirected subgraph of C; if it is not a cutedge, assume we get C' (and let $C'' = \emptyset$ with $\sigma(C'') = 0$). If $\sigma(C') + \sigma(C'') \geq \sigma(C)$, then delete C from $\mathcal{C}$.

P3 Let C be a district $\{a, b\}$ with two nodes having the bidirected edge $a \leftrightarrow b$. Let W be the set of parents of b in C. Consider the c-component C' with single node district $\{a\}$ and empty parent set, and let C'' be the c-component with single node district $\{b\}$ with parent set $W' \subseteq W \cup \{a\}$. If $\sigma(C') + \sigma(C'') \geq \sigma(C)$, then delete C from $\mathcal{C}$.

In all our experiments, we allow a larger parent set size for nodes in single node districts than for nodes in multi-node districts. This is because the number of candidate c-components with multi-node districts grows rapidly with the district size. We formalize this detail in the next proposition.

Proposition A.2. *Let $\mathcal{C}$ be a list of all c-components that have districts with size at most k and parent set size for nodes in a district of size t to be at most l_t where $l_t > l_s$ if $t < s$. The optimal bow-free or arid graph score that is obtainable from $\mathcal{C}$ is the same as that obtainable after pruning $\mathcal{C}$ according to the pruning rules enumerated above.*

Proof. Consider an optimal bow-free (or arid graph) $\mathcal{G}$ that can be composed from the c-components in an input candidate list $\mathcal{C}$ satisfying the conditions of the proposition. Assume it contains a c-component C that would be affected by pruning rule 1 and assume the remaining c-components form a set S. Note that removing the edge $u \to v$ from $\mathcal{G}$ only changes C (to C'), does not affect c-components in S, and does not cause the new graph $\mathcal{G}'$ to have a directed cycle or have a bow (or become non-arid). Further by the definition of $\mathcal{C}$, C' is also contained in $\mathcal{C}$. Therefore $\mathcal{G}'$ is a valid solution. Furthermore, the $\sigma(\mathcal{G}) = \sigma(C) + \sum_{s \in S} \sigma(s)$. $\mathcal{G}'$ has exactly the same c-components other than C, which becomes C' in $\mathcal{G}'$ and

$$\sigma(\mathcal{G}') = \sigma(C') + \sum_{s \in S} \sigma(s) \geq \sigma(\mathcal{G}) \geq \sigma(\mathcal{G}').$$

The first inequality above follows from the fact that $\sigma(C') \geq \sigma(C)$. The second follows from the fact that $\mathcal{G}'$ is composed of c-components in $\mathcal{C} \setminus \{C\}$ whereas $\mathcal{G}$ is composed of components from $\mathcal{C}$.

The proof for the second pruning rule is similar; it is enough to note that C' and C'' (if C'' is nonempty) are both contained in $\mathcal{C}$ and removing $u \leftrightarrow v$ does not create any new directed cycles, bows, or c-trees (see the proof of Proposition 2.1 for a definition of a c-tree).

For the third pruning rule, let $\mathcal{G}$ be an optimal bow-free (or arid) graph that can be built from $\mathcal{C}$ and let C be a c-component of $\mathcal{G}$. Assume we perform the edge deletion and replacement operations indicated in pruning rule 3 to get graph $\mathcal{G}'$. Replacing $a \leftrightarrow b$ with $a \to b$ certainly does not create a new bow. As we also delete the parents of a from the graph $\mathcal{G}$ at the same time we perform the replacement, no new directed cycles containing the edge $a \to b$ are present in the new graph $\mathcal{G}'$. Finally, we do not create a new c-tree as a and b are not part of the same district anymore (they were originally in a two-node district).

The c-component C' is created by deleting all parents of a in C and C'' is created by replacing the edge $a \leftrightarrow b$ by $a \to b$. The parent set size in C' is zero, and therefore $C' \in \mathcal{C}$. Similarly, $C'' \in \mathcal{C}$ as C'' has a single node district $\{b\}$ and b is allowed to have at least one more parent in C'' than in C. We also have $\sigma(\mathcal{G}') \geq \sigma(\mathcal{G})$ given the score inequality on C, C', C''. The result follows. □

Theorem 3.1. *BIP (ARIP) is a valid IP formulation of the optimal bow-free (arid) graph learning problem: solving BIP (ARIP) leads to a score-maximizing bow-free (arid) graph.*

Proof. Assume we have a candidate list of c-components $\mathcal{C}$. We wish to establish that BIP($\mathcal{C}$) is a valid integer programming formulation for the bow-free graph learning problem in the sense that an optimal solution $\bar{z}$ of BIP($\mathcal{C}$) yields a score-maximizing bow-free graph $\mathcal{G}(\bar{z})$ that can be composed of c-components in $\mathcal{C}$.

We abbreviate BIP($\mathcal{C}$) by BIP. Let $\bar{z} = (\bar{z}_C)_{C \in \mathcal{C}}$ be a vector that satisfies all the constraints of BIP, namely constraints (13) - (16). Consider the set Q of c-components for which $\bar{z}_C = 1$; the rest of the variables values in $\bar{z}$ are 0 by definition. The c-components in Q have disjoint districts and these districts cover all nodes, by constraint (13) which forces each node to be contained in exactly one district. Now consider the DMG $\mathcal{G}(\bar{z})$ formed by taking the union of the c-components in Q. Constraint (14) guarantees that $\mathcal{G}(\bar{z})$ has no directed cycles. By constraint (15), each c-component in Q is bow-free, and therefore by Corollary A.1, $\mathcal{G}(\bar{z})$ is bow-free. This shows that any solution of (13) - (16) is a bow-free graph. Further, consider any bow-free graph that is composed of components from $\mathcal{C}$; if we set the corresponding z variables to 1, and the rest to 0, then we get a feasible solution of (13) - (16). The result for BIP follows and the result for ARIP is very similar. □

Theorem 3.2. *Let $\emptyset \neq S \subseteq V$ and let T be the set of c-components $C \in \mathcal{C}$ such that $D_C \cap S \neq \emptyset$, and each node in $S \setminus D_C$ is a parent of a node in $D_C \cap S$. Then $\sum_{C \in T} z_C \leq 1$ is a cutting plane for BIP.*

Proof. To prove that the above inequality is valid for all solutions of BIP, we will prove, by contradiction, that no two of the variables in the left-hand-side of the inequality can simultaneously be one in a solution.

Assume this latter statement is false, and assume that $\bar{z}_{C_1} = 1$ and $\bar{z}_{C_2} = 1$ in a solution $\bar{z}$ of BIP, where C_1, C_2 are two c-components involved in the l.h.s. of the inequality. Let $U_1 = D_{C_1} \cap S \neq \emptyset$ and let $V_1 = S \setminus U_1$. Define U_2, V_2 in a similar manner for C_2. If $U_1 \cap U_2 \neq \emptyset$, then constraint (14) is violated for any node in $U_1 \cap U_2$, a contradiction to $\bar{z}$ being a solution of BIP.

So assume that $U_1 \cap U_2 = \emptyset$, which means that $U_2 \subseteq V_1$. and $U_1 \subseteq V_2$. By definition, for $i = 1, 2$, C_i has a directed edge from each node in V_i to a node in U_i. Therefore, C_1 has a directed edge from each node in U_2 to

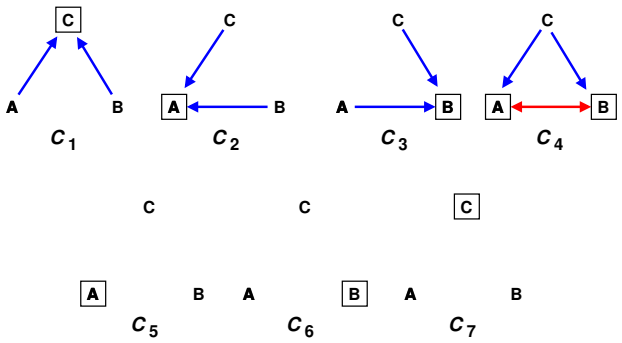

Figure 3: Seven C-Components $C_1, \ldots, C_7$ on Three Nodes Labeled A, B, C. The District nodes in Each C-Component Are Indicated by a Square Around the Node.

some node in U_1 (let T_1 be the set of these edges), and C_2 has a directed edge from each node in U_1 to some node in U_2 (let T_2 be the set of these edges). We will show that there is a directed cycle in $T_1 \cup T_2$. Assume, without loss of generality, that $|U_1| \leq |U_2|$. Let u_1 in U_1 be an arbitrary node and start a directed walk as follows: starting from u_i with $i = 1, 2, \ldots$ follow a directed edge in T_2 from u_i to some node in U_2 that is distinct from $v_1, \ldots, v_{i-1}$ and call this node v_i. Then follow a directed edge in T_1 from v_i to a node in U_1 distinct from $u_1, \ldots, u_i$ and call this u_{i+1} and repeat. If we cannot find a distinct new node, then we have found a directed cycle. As $|U_1| \leq |U_2|$, this process must stop after $v_{|U_1|}$ is marked and a node in U_1 repeated (which means we have found a directed cycle). This is a contradiction to $\bar{z}$ satisfying the strengthened cluster inequalities which rule out existence of directed cycles.

Assume $\mathcal{G}$ has 3 nodes (i.e., assume $V = \{\text{A,B,C}\}$) and $\mathcal{C}$ consists of all valid c-components on V and $S = V$. We define a point $\bar{z} \in \{0,1\}^{\mathcal{C}}$ via Figure 3: we give the c-components C for which $\bar{z}_C > 0$ and assume $\bar{z}_C = 0$ for all other $C \in \mathcal{C}$. Let $\bar{z}_{C_7} = 2/3$ and $\bar{z}_{C_1} = \cdots \bar{z}_{C_6} = 1/3$. It is easy to verify that $\bar{z}_C$ is a solution of BLP but does not satisfy the inequality in the theorem. The latter condition follows from the fact that $\bar{z}_{C_1} + \bar{z}_{C_2} + \bar{z}_{C_3} + \bar{z}_{C_4} = 4/3 > 1$ but this sum should at most 1 by the first part of the proof. □

Theorem 3.3. *Let $\emptyset \neq S \subseteq V$ with $|S|$ odd and T be the set of c-components $C \in \mathcal{C}$ such that $|D_C \cap S| \geq 2$. Then $\sum_{C \in T} \lfloor |D_C \cap S|/2 \rfloor z_C \leq \lfloor |S|/2 \rfloor$ is a cutting plane.*

Proof. We will show that the inequality in the theorem is a Gomory-Chvátal cutting plane. The sum of the constraints in (14) for all nodes in S becomes

$$\sum_{C : D_C \cap S \neq \emptyset} |D_C \cap S| z_C = |S|.$$

Recall that the variables z_C are nonnegative in BLP. Therefore, if we divide all coefficients in the above equation by 2 and then round down the coefficients of all variables, we get the following valid inequality for BLP:

$$\sum_{C : |D_C \cap S| \geq 2} \lfloor |D_C \cap S|/2 \rfloor z_C \leq |S|/2.$$

For any integral solution, the left-hand-side of the above inequality is integral but the right-hand is fractional as $|S|$ is odd. Rounding down the right-hand-side, we get the inequality in the theorem as a valid inequality for integral solutions of BLP. To see that it is a cutting plane, consider the point $\bar{z} \in \{0,1\}^{\mathcal{C}}$ defined via Figure 4. Assume $\bar{z}_{C_1} = \bar{z}_{C_2} = \bar{z}_{C_3} = 1/2$ and $\bar{z}_C = 0$ for all other $C \in \mathcal{C}$. Clearly, $\bar{z}_{C_1} + \bar{z}_{C_2} + \bar{z}_{C_3} = 3/2$ which violates the inequality defined in the theorem which requires that $z_{C_1} + z_{C_2} + z_{C_3} \leq 1$. □

B RANDOM SAMPLE AND GRAPH GENERATION PARAMETERS

To generate bow-free graphs we iterate over the nodes $1, \ldots, d$ in increasing order, and for each ij with $1 \leq i < j \leq d$, we create a directed edge $i \to j$ with probability p_{dir}. If we do not create $i \to j$, we then create $i \leftrightarrow j$ with probability p_{bidir}. The resulting graph is acyclic and bow-free. We delete some of the edges of the graph (and stay bow-free) to satisfy different bounds as explained below.

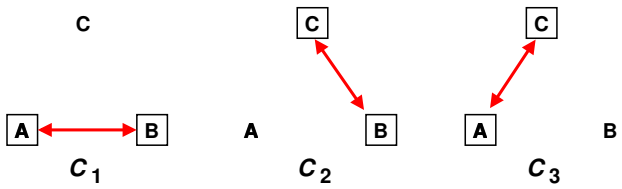

Figure 4: Three C-Components $C_1, \ldots, C_3$ on Three Nodes Labeled A, B, C. The District Nodes in Each C-Component Are Indicated by a Square Around the Node.

For the sparse graphs used in Table 1, we allow each node to have at most 3 incoming arrows and each district to have size at most 2. We use $p_{dir} = 0.4$ and $p_{bidir} = 0.3$ in the graph generation process. Subsequently, for each node we randomly delete incident bidirected edges until it has at most one incident bidirected edge. This operation ensures all districts have size at most 2. Then for each node i, we remove randomly selected edges $j \to i$ or $j \leftrightarrow i$ till node i has at most three such edges. The output graphs have the desired c-component sizes.

After generating a graph, we generate B in equation (2) and Ω in equation (3) as follows. We mostly follow the settings used in Bhattacharya et al. (2020); the main difference is that we increase the magnitude of nonzero off-diagonal entries in Ω to increase correlations between variables connected by bidirected edges. The goal is to increase the influence of bidrected edges. We first set the appropriate entries of B and Ω to zero. The nonzero entries of B are chosen by uniformly sampling from $[-2.0, -0.5] \cup [0.5, 2.0]$, the non-diagonal nonzero entries of Ω are chosen by uniformly sampling from $[-2.0, -1.0] \cup [1.0, 2.0]$, and the diagonal nonzero entries of Ω are chosen by uniformly sampling from $[0.7, 1.2]$ and then adding $\sum_{j \neq i} |\Omega_{ij}|$ to the sample value for row i. This last step ensures Ω is strictly diagonally dominant and therefore positive definite. We then draw n samples of the error vector ϵ from the multivariate Gaussian distribution $N(0, \Omega)$ and apply equation 1 to obtain n samples of X.

To decide on the appropriate value of n, we create 10 graphs (in the fourth category, with large districts and large parent sets) and 5 instances per graph (with n samples in each instance) for each each sample size n where $n \in \{500, 1000, 2000, 5000\}$. For each of the 50 instances, we run LPH with the settings used for large districts and large parent set sizes. We take the solution graph, compute its score and divide by n. In Figure 5, for each sample size n, we plot the average of score/n and the standard error across the 50 instances. This plot shows that our method is quite stable (the mean values do not change much and standard error is small) starting with about 1000 samples, and in all subsequent experiments with 10 node graphs we use 1000 samples. Further, given the high running time of ABIC and GBAP, we only consider 10 graphs and a single instance per graph for the different graph categories discussed earlier.

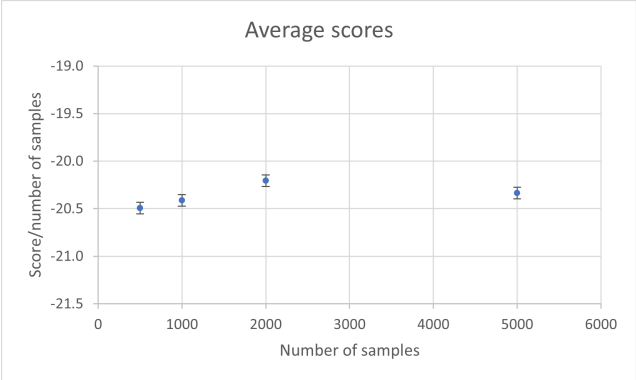

Figure 5: Average Scores as a Function of Sample Size.

C CUTTING PLANES

We used the PORTA software (Christof and Loebel, 1997) to explicitly find the convex hull of fBIP(3). Below, we describe the inequalities obtained. Based on the output from PORTA we can confirm the validity of Theorem 3.4.

Table 3: Gap Between BIC Score of LP Solution at the End of the Root Node for Bow-Free Graphs and BIC Score of Ground-Truth graph. Running Time is given in minutes. Averages are taken over 10 Sparse Randomly Generated 10-Node Graphs With Small C-Components and Parent Sets

Metric	BF-cuts	BF-3cuts	BF
Avg. Gap	4.6%	3.9%	3.7%
Avg. Time	14	11	10

There are 62 bow-free graphs for a set of 3 nodes. Each of these graphs can be assembled from a subset of a set $\mathcal{C}$ of 31 bow-free c-components. The set of c-components consists of 9 c-components with 1-node districts and 1 or 2 parents, 12 c-components with 2-node districts with or without the other node as parent (of one of the nodes in the district or of both nodes in the district), and 10 c-components with 3-node districts. Note that the c-components with 1-node district and no parents are not included in the set of c-components considered, which means that representing graphs that have nodes with in-degree zero is done by selecting a subset of the c-components that don't include those nodes in their districts.

The facet-defining inequalities found can be grouped in the following three groups:

1. The first group consists of 31 non-negativity inequalities, one for each variable associated with a c-component in the set $\mathcal{C}$.

2. The second group consists of 26 inequalities and includes the inequality that says that the sum of all the variables associated with c-components with at least one bidirected edge is less than or equal to 1. This inequality is the same as the inequality that results from applying theorem 3.3 to a 3-node graph with the set S being the 3 nodes in the graph. All other inequalities in this group are variations of this inequality and are formed by dropping one or more variables associated with c-components with 2-node districts and adding, from the list of variables associated with 1-node district c-components, the same number of variables that were dropped earlier. One of those inequalities is the same as the inequality that results from applying theorem 3.2 to a 3-node graph with the set S being the 3 nodes in the graph. In addition, three of those inequalities are the same as the inequalities that result from applying theorem 3.2 to a 3-node graph with the set S being each pair of nodes in the graph.

3. The last group consists of 27 inequalities and includes the inequality that says that the sum of all variables associated with c-components having 1 or 2-node districts (excluding the 3 c-components that have two nodes connected by a bidirected edge and the third node is a parent of the other two) plus two times the sum of all remaining variables (including the 3 c-components excluded in the first sum) is less than or equal to 2. This inequality is the inner version of the strengthened cluster inequality applied to a 3-node graph with the set S being the 3 nodes in the graph. All other inequalities in this group are variations of this inequality and are formed by dropping one or more variables associated with c-components with 1-node districts and adding the coefficient 2 to the same number of variables dropped from the list of variables associated with 2-node district c-components.

We now present computational results showing the efficacy of the inequalities in Theorems 3.2 and 3.3 for the case where the ground-truth graphs can be decomposed into 'small' c-components (with small districts and small parent sets). In Table 3, we give the average gap between the value of BLP_+ at the end of the root node of the branch-and-bound tree and the optimal value of BIP (gap = |BLP_+ value - BIP value| / |BIP value|) as well as the average time. Column 'BF' corresponds to the case where we used our bow-free graph code with the inequalities from Theorem 3.2 (with $|S| = 2$ and 3) and from Theorem 3.2 (with $|S| = 3$), column 'BF-3cuts' corresponds to the case where we didn't include the inequalities for $|S| = 3$, and column 'BF-cuts' corresponds to the case where we don't use any of the new inequalities. The results show that the addition of the new inequalities reduces the average gap as well as the average solution time.

Table 4: Scores for 10 Sparse Randomly Generated 10-Node Graphs With Small C-Components and Parent Sets

Orig.	ABIC	GBAP	AA	BF
-17741.6	-17765.1	-17742.1	**-17741.6**	**-17741.6**
-17508.5	-17511.9	**-17508.5**	**-17508.5**	**-17508.5**
-17872.5	**-17872.5**	**-17872.5**	*-17871.2*	*-17871.2*
-19055.6	-19123.7	-19149.9	-19093.6	**-19055.6**
-17888.1	-17908.4	-17909.6	*-17884.1*	*-17881.6*
-18584.9	-18625.4	-18587.9	-18595.5	**-18584.9**
-17791.2	-17795.6	-17917.1	*-17790.1*	*-17789.5*
-18964.8	-20438.8	-19545.4	-19010.8	**-18964.8**
-17562.1	-17565.6	-17562.9	**-17562.1**	**-17562.1**
-17627.9	-17681.6	-18209.7	-17655.9	**-17627.9**
-18059.7	-18228.8	-18200.6	-18071.3	-18058.8

D ADDITIONAL EXPERIMENTS AND RESULTS

Precision (see Tillman and Spirtes (2011)) is defined as the percentage of edges in the solution graph which are also in the ground-truth graph. Recall represents the percentage of edges in the ground-truth graph which are also in the solution graph. The F-score combines precision and recall into a single number and is the harmonic mean of precision and recall:

$$\text{F-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

D.1 Additional Results for Graphs with Small C-Components

In Table 4, for each ground-truth instance, we compare the BIC scores of the optimal AADMG (column AA) and bow-free graphs (column BF) that can be constructed from $\mathcal{C}$ with that of the ground-truth (column Orig.) and also with the scores returned by ABIC and GBAP. We show in bold the scores that match the ground-truth scores and in italics the scores that are slightly better. In the last row, we give the averages across the 10 instances.

We also compare the solution graphs from Table 4 against the ground-truth graphs and report average precision, recall, and F-score values in Table 5 for the skeleton, for the set of directed edges, and for the set of bidirected edges (with the caveat that while skeleton values have some relevance, it is hard to say what the values for directed and bidirected edges mean). All values for the optimal bow-free graphs are better than the corresponding heuristically obtained ABIC and GBAP values.

Table 5: Comparison Against Original Graph

	Precision			Recall			F-score		
	skel.	dir.	bidir.	skel.	dir.	bidir.	skel.	dir.	bidir.
ABIC	.839	.749	.179	.949	.774	.383	.891	.761	.244
GBAP	.891	.706	.352	.947	.693	.458	.918	.699	.398
AA	.906	.711	.450	.950	.818	.283	.927	.761	.347
BF	**.969**	**.812**	**.633**	**.975**	**.873**	**.517**	**.972**	**.841**	**.569**

D.2 Heuristic Results on DAG Learning

In this set of experiments, all districts have a single node but parent sets can be large (thus we are learning DAGs). We set $p_{dir} = 0.4$ and $p_{bidir} = 0.0$ while generating graphs, and then restrict all nodes to have indegree at most 6.

For each instance, we compute the optimal DAG that can be constructed using the c-components in $\mathcal{C}$ that do not have bidirected edges. To do so, we use the IP model implemented in the code of (Chen et al., 2021): cluster inequalities are used to eliminate directed cycles and no bidirected edges are allowed. Two experiments were conducted with this code. In one experiment, $\mathcal{C}$ consists of c-components with one node districts and up to three parents (two node districts are not relevant here). We refer to this experiment as DAG-3. In the other experiment, which we refer to as DAG-6, $\mathcal{C}$ consists of c-components with one node districts and up to six parents. We also conducted an experiment with our LP heuristic, where the operation of adding a node to the parent set is the only one allowed. In this experiment, $\mathcal{C}$ consists of c-components with one node districts and up to three parents. Finally, we also compared with the state-of-the-art IP-based DAG learning solver GOBNILP developed by (Bartlett and Cussens, 2017). For this solver, we conducted two experiments, where we set the parameter controlling the maximum parent set size to three and six, and we refer to these experiments as GOBNILP-3 and GOBNILP-6, respectively.

In Table 6 we give results for 10 instances as well as the average in the last row. The DAG-3 scores are at least as good as the ground-truth scores in only 2 instances and worse in the remaining 8 instances. By comparison, the scores from the LP-based heuristic are at least as good as the ground-truth scores in 7 instances (we indicate scores that match the ground truth scores in **bold**, and those that are better in *italics*). These results demonstrate that for the 10 node graphs considered here, our LP heuristic does an excellent job in finding relevant large parent sets. The DAG-6 scores are always as good as the ground-truth scores and slightly better in half of the instances. Finally, as expected, the DAG-3 scores are very similar to the GOBNILP-3 scores and the DAG-6 scores are also very similar to the GOBNILP-6 scores as both codes optimize over the same family of bow-free graphs. The slight differences are likely due to solver tolerances. Most IP solvers such as CPLEX or Gurobi have a default optimality tolerance of 10^{-4}. This means that they terminate when the relative gap between the current best integral solution and upper bound on the solution is 10^{-4} or less. The maximum difference between the GOBNILP solution and the DAG solution is 1.6 which yields a relative gap of less than 10^{-4}.

In Table 7, we provide averages of the precision, recall, and F-score metrics as well as average time for all the experiments. The best values of the precision, recall, and F-score metrics are given in **bold**. We note that the precision, recall, and F-score values of the LP heuristic, for both the skeleton and directed edges, fall between the values of DAG-3 and the values of DAG-6. This is also true for the time values.

Table 6: Scores for Randomly Generated Datasets with One Node Districts and Large Parent Sets

Orig.	GOBNILP-3	GOBNILP-6	DAG-3	DAG-6	LPH
-16930.1	-17006.3	**-16930.1**	-17006.3	**-16930.1**	**-16930.1**
-16952.2	-17330.8	**-16952.2**	-17330.8	**-16952.2**	-16967.5
-16960.0	-17303.3	**-16960.0**	-17302.9	*-16958.9*	*-16958.9*
-16933.9	-17283.8	**-16933.9**	-17283.8	**-16933.9**	**-16933.9**
-16960.9	-17009.4	**-16960.9**	-17009.4	*-16960.3*	*-16960.3*
-16933.3	-17376.3	**-16933.3**	-17376.3	**-16933.3**	**-16933.3**
-17029.4	-17080.2	**-17029.4**	-17079.9	*-17029.1*	-17038.8
-16894.8	-16894.8	**-16894.8**	*-16893.2*	*-16893.2*	*-16893.2*
-16997.1	-17204.9	**-16997.1**	-17204.9	**-16997.1**	-17017.9
-17030.8	*-17029.8*	*-17029.8*	*-17029.6*	*-17029.6*	*-17029.6*
-16962.2	-17152.0	-16962.2	-17151.7	-16961.8	-16966.4

Table 7: Average Precision and Recall for Randomly Generated Datasets with One Node Districts and Large Parent Sets and average time

	Precision		Recall		F-score		Time (s)
	skel.	dir.	skel.	dir.	skel.	dir.	
GOBNILP-3	.811	.574	.842	.590	.826	.582	24
GOBNILP-6	**.994**	.921	**.989**	.917	**.991**	.919	137
DAG-3	.804	.569	.847	.597	.825	.583	5
DAG-6	.971	**.926**	**.989**	**.943**	.980	**.934**	80
LPH	.894	.785	.973	.841	.932	.812	27

Table 8: Scores for Randomly Generated Graphs with Large Districts and Small Parent Sets

Orig.	ABIC	GBAP	LPH
-21274.3	-21286.5	-21272.5	-21274.4
-21449.9	-21469.7	-21449.6	-21466.6
-21178.5	-21193.2	-21183.0	-21183.7
-21021.8	-21047.9	-21033.3	-21032.4
-21373.2	-21402.6	-21385.0	-21391.9
-20894.5	-20910.0	-20907.5	-20908.9
-20807.7	-20832.9	-20838.7	-20817.3
-19409.7	-19571.4	-19640.3	-19417.9
-20359.3	-20362.3	-20357.0	-20358.4
-21033.0	-21053.2	-21037.0	-21038.3
-20880.2	-20903.0	-20910.4	-20889.0

D.3 Heuristic Results for Graphs with Large Districts and Small Parent Sets

We generate bow-free graphs with large districts and small parent sets. We set $p_{dir} = 0.3$ and $p_{bidir} = 0.8$. For each node i in a generated graph, we randomly remove edges $j \to i$ or $j \leftrightarrow i$ till node i has at most three edges with inward pointing arrows (at most three edges of the form $j \to i$ or $j \leftrightarrow i$). We let $\mathcal{C}$ be the same as in Section 5.1.

In Table 8, we compare the ground-truth scores with ABIC, GBAP and our LP heuristic (LPH). For all instances, LPH yields better scores than ABIC. However, in half the instances GBAP obtains a better score than LPH and worse in the other half. Further, both LPH and GBAP obtain scores very close to the ground truth in two instances. However, the average scores for GBAP are noticeably worse than the average LPH score.

We give precision, recall, and F-score values in Table 9.

Table 9: Average Precision and Recall for Graphs with Large Districts and Small Parent Sets

	Precision			Recall			F-score		
	skel.	dir.	bidir.	skel.	dir.	bidir.	skel.	dir.	bidir.
ABIC	.761	.432	.539	.974	.957	.348	.854	.595	.423
GBAP	.905	.699	.673	.935	.784	.629	.920	.739	.650
LPH	.806	.353	.608	.958	.783	.196	.875	.487	.296

Table 10: Scores for Randomly Generated Graphs with Large Districts and Large Parent Sets

Orig.	ABIC	GBAP	AA	BF	LPH
-19057.4	-19071.4	-19484.6	-19169.2	-19117.4	-19060.0
-19802.3	-19830.9	-20625.3	-20082.1	-19916.3	-19822.6
-20606.4	-20613.9		-21074.8	-20857.5	-20615.0
-21178.7	-21207.7	-21908.4	-21332.9	-21267.9	-21194.1
-20865.8	-20876.5	-20881.0	-20993.5	-20962.1	-20870.1
-18846.5	-18848.3	-18895.1	-19031.6	-18936.4	-18852.3
-21268.7	-21716.6	-21282.0	-21405.1	-21347.0	-21288.0
-18906.2	-18927.6	-18829.1	-18924.9	-18921.7	-18908.4
-22152.7	-22226.1	-22394.6	-22517.5	-22320.3	-22172.0
-21059.0	-21110.3	-21309.0	-21118.6	-21100.4	-21059.2
-20374.4	-20442.9	-20623.2	-20565.0	-20474.7	-20384.2

D.4 Heuristic Results for Graphs with Large Districts and Large Parent Sets

In these experiments we study general DMG learning where both the districts and the parent sets can be large. We set $p_{dir} = 0.4$ and $p_{bidir} = 0.3$. We let $\mathcal{C}$ be the same as in Section 5.1.

In Table 10, we compare the scores of LPH with those of the ground-truth (column 'Orig.'), optimal AADMG (column 'AA'), optimal bow-free graph (column 'BF'), and the ABIC heuristic (Bhattacharya et al., 2020) (column 'ABIC'). The results show clearly that LPH has the best performance of all methods. In 8 out of 10 instances, LPH scores are better than ABIC scores. In addition, they are significantly better than the optimal bow-free graph (that can be formed from $\mathcal{C}$) score for all instances, which in turn are better than the optimal AADMG in all instances as well.

In Table 11, we report precision, recall, and F-score for the four methods.

D.5 Heuristic Results for Sparse 20-Node Graphs

We also generated 10 sparse bow-free graphs with 20 nodes as follows. We first generate a DAG with $p_{dir} = 0.2$ containing 22 nodes. We then delete randomly selected edges until every node has at most in-degree 4 and out-degree 4. Then, we randomly select two nodes, which have no parents and out-degree at least 2. We remove those two nodes from the graph and connect the children from each of those nodes with bidirected edges forming a path with the children ordered randomly. Finally, we delete any direct edge that is part of a bow making the graph bow-free. The list of c-components that we generate to start LPH contains c-components with a single node district and parent set sizes at most 3, and two node districts where the nodes have no parents. We use a

Table 11: Average Precision and Recall for Graphs with Large Districts and Large Parent Sets

	Precision			Recall			F-score		
	skel.	dir.	bidir.	skel.	dir.	bidir.	skel.	dir.	bidir.
ABIC	.799	.641	.388	.946	.783	.398	.866	.705	.393
GBAP	.845	.528	.331	.845	.525	.299	.845	.526	.314
AA	.840	.442	.100	.693	.488	.050	.759	.464	.067
BF	.837	.336	.083	.732	.381	.034	.781	.357	.048
LPH	.823	.437	.250	.860	.608	.078	.841	.509	.119

2-hour time limit for BIP. We do not report results for GBAP as it takes over 2.5 days to solve the first instance. We do not get as small $\Delta(BIC)$ values as before, but still get excellent fit values.

Table 12: Average scores and running time (minutes) for sparse randomly generated 20-node graphs

Metric	Orig.	ABIC	LPH
Avg. BIC	-35724	-35777	-35752
Avg. Δ(BIC)		52.9	28.2
Std. Δ(BIC)		38.8	16.9
Avg. RMSEA	.006	.017	.009
Std. RMSEA	.008	.008	.009
Avg. F-Score		.883	.862
Avg. Time		771	129

Table 13: Scores for Sparse Randomly Generated 20-Node Graphs

Orig.	ABIC	LPH
-36382.9	-36404.3	-36394.9
-35268.6	-35283.7	-35316.6
-35486.2	-35588.8	-35520.8
-35815.2	-35928.7	-35825.1
-34442.9	-34546.6	-34493.4
-35975.9	-35986.9	-36013.8
-36418.8	-36454.4	-36427.9
-36408.5	-36463.7	-36455.7
-35320.4	-35331.1	-35325.6
-35721.3	-35781.5	-35749.0
-35724.1	-35777.0	-35752.3