Integer Programming for Learning Directed Acyclic Graphs from Continuous Data

Hasan Manzour* Simge Küçükyavuz† Hao-Hsiang Wu ‡ Ali Shojaie§

May 6, 2020

Abstract

Learning directed acyclic graphs (DAGs) from data is a challenging task both in theory and in practice, because the number of possible DAGs scales superexponentially with the number of nodes. In this paper, we study the problem of learning an optimal DAG from continuous observational data. We cast this problem in the form of a mathematical programming model that can naturally incorporate a super-structure in order to reduce the set of possible candidate DAGs. We use a negative log-likelihood score function with both ℓ_0 and ℓ_1 penalties, and propose a new mixed-integer quadratic program (MIQP), referred to as a layered network (LN) formulation. The LN formulation is a compact model that enjoys as tight an optimal continuous relaxation value as the stronger but larger formulations under a mild condition. Computational results indicate that the proposed formulation outperforms existing mathematical formulations and scales better than available algorithms that can solve the same problem with only ℓ_1 regularization. In particular, the LN formulation clearly outperforms existing methods in terms of computational time needed to find an optimal DAG in the presence of a sparse super-structure.

1 Introduction

The study of Probabilistic Graphical Models (PGMs) is an essential topic in modern artificial intelligence (Koller and Friedman 2009). A PGM is a rich framework that represents the joint probability distribution and dependency structure among a set of random variables in the form of a graph. Once learned from data or constructed from expert knowledge, PGMs can be utilized for probabilistic reasoning tasks, such as prediction; see Koller and Friedman (2009) and Lauritzen (1996) for comprehensive reviews of PGMs.

Two most common classes of PGMs are *Markov networks* (undirected graphical models) and *Bayesian networks* (directed graphical models). A Bayesian Network (BN) is a PGM in which the conditional probability relationships among random variables are represented in the form of a Directed Acyclic Graph (DAG). BNs use the richer language of directed graphs to model probabilistic influences among random variables that have clear directionality; they are widely used to model

*Department of Industrial and Systems Engineering, University of Washington (hmanzour@uw.edu).
†Industrial Engineering and Management Sciences, Northwestern University (simge@northwestern.edu).
‡Department of Management Science, National Chiao Tung University, (hhwu2@nctu.edu.tw)
§Department of Biostatistics, University of Washington (ashojaie@uw.edu).

causal relationships (Spirtes et al. 2000) and discover such relationships from observational data (Chen et al. 2019); they thus have broad applications in machine learning (Koller and Friedman 2009) and biology (Markowetz and Spang 2007, Zhang et al. 2013).

Learning BNs is a central problem in machine learning (Drton and Maathuis 2017). An essential part of this problem entails learning the DAG structure that accurately represents the (hypothetical) joint probability distribution of the BN. Although one can form the DAG based on expert knowledge, acquisition of knowledge from experts is often costly and nontrivial. Hence, there has been considerable interest in learning the DAG directly from observational data (Bartlett and Cussens 2017, Chickering 2002, Cussens et al. 2017a,b, Han et al. 2016, Park and Klabjan 2017, Raskutti and Uhler 2013, Spirtes et al. 2000).

Learning the DAG which best explains the observed data is an NP-hard problem (Chickering 1996). Despite this negative theoretical result, there has been interest in developing methods for learning DAGs in practice. There are two main approaches for learning DAGs from observational data: constraint-based and score-based. In constraint-based methods, such as the well-known PC-Algorithm (Spirtes et al. 2000), the goal is to learn a completed partially DAG (CPDAG) consistent with conditional independence relations inferred from the data. Score-based methods, including the approach in this paper, aim to find a DAG maximizing a score that measures how well the DAG fits the data.

Existing DAG learning algorithms can also be divided into methods for discrete and continuous data. Score-based methods for learning DAGs from discrete data typically involve a two-stage learning process. In stage 1, the score for each node and its candidate parent sets (CPS) is computed. The parent set of a node v, denoted by pa_v, is the set of nodes pointing to v in the DAG. In stage 2, a search algorithm is used to maximize the global score, so that the resulting graph is acyclic. Both of these stages require exponential computation time (Xiang and Kim 2013). For stage 2, there exist elegant exact algorithms based on dynamic programming (Eaton and Murphy 2007, Koivisto 2012, Koivisto and Sood 2004, Parviainen and Koivisto 2009, Perrier et al. 2008, Silander and Myllymaki 2012, Yuan et al. 2011), A* algorithm (Yuan and Malone 2013, Yuan et al. 2011), and integer-programming (Cussens 2010, 2012, Hemmecke et al. 2012, Studený and Haws 2013, Bartlett and Cussens 2013, Oates et al. 2016a,b, Bartlett and Cussens 2017, Cussens et al. 2017a,b). The A* algorithm identifies the optimal DAG by solving a shortest path problem in an implicit state-space search graph. Existing integer-programing (IP) formulations for learning DAGs from discrete data rely on identifying the best parent set, pa_v, for each node v in the network. In these formulations, each binary variable indicates whether or not a given parent set is assigned to a node. Therefore, in essence, these formulations are exponential in the number of variables ($m2^{m-1}$ binary variables for m nodes). A common practice to circumvent the exponential number of CPS is to limit the in-degree of each node (e.g., Bartlett and Cussens 2017, Cussens 2012, Cussens et al. 2017b) or prune the CPS (e.g., Correia et al. 2019, Kuipers et al. 2018). Another strategy, referred to as the hybrid approach, is to use conditional independence test to construct a super-structure of a Bayesian network (Tsamardinos et al. 2006) or obtain an ordering among nodes (Singh and Valtorta 1995) to limit the DAG space before performing the score-and-search algorithm.

A recent comprehensive empirical evaluation of A* algorithm and IP methods for discrete data (Malone et al. 2014) shows that, because of their intrinsic differences, the relative efficiency of these methods varies in different problems. In particular, state-of-the-art IP methods can solve instances up to 1,000 CPS per variable regardless of the number of nodes, whereas A* algorithm works for problems with up to 30 nodes, even with tens of thousands of CPS per node.

Exact IP methods developed for *discrete* data can also be applied to *continuous* data, since

they are compatible with arbitrary score functions. However, such an approach would result in exponentially-sized formulations, which require advanced algorithms for efficient computation. Alternatively, tailored formulations can be developed for learning DAGs for an important class of BNs where causal relations between continuous random variables are encoded using *linear structural equation models* (SEMs); see Section 2. Such formulations can be solved using off-the-shelf optimization solvers without needing advanced algorithms. To our knowledge, Park and Klabjan (2017) and Xiang and Kim (2013) provide the only exact algorithms for learning medium to large DAGs from continuous data with linear SEMs: Park and Klabjan (2017) propose an IP-based model using the topological ordering of variables; and Xiang and Kim (2013) develop an A*-lasso algorithm for learning an optimal DAG from continuous data with an ℓ_1 regularization. A*-lasso incorporates the lasso-based scoring method within dynamic programming to avoid an exponential number of parent sets and uses the A* algorithm to prune the search space of the dynamic programming method.

Given the state of existing algorithms for DAG learning from continuous data with linear SEMs, there is currently a gap between theory and computation: While statistical properties of exact algorithms can be rigorously analyzed (Loh and Bühlmann 2014, van de Geer and Bühlmann 2013), it is much harder to assess the statistical properties of heuristics (Aragam and Zhou 2015, Fu and Zhou 2013, Han et al. 2016, Zheng et al. 2018) that offer no optimality guarantees (Koivisto 2012). This gap becomes particularly noticeable in cases where the statistical model is identifiable from observational data. In this case, the optimal score from exact search algorithms is guaranteed to reveal the true underlying DAG when the sample size is large. Therefore, causal structure learning from observational data becomes feasible (Loh and Bühlmann 2014, Peters and Bühlmann 2013).

In this paper, we focus on DAG learning from continuous data, where causal relations among random variables are captured by linear structural equation models (SEMs). In this case, network edges are associated with the coefficients of regression models corresponding to linear SEMs. Consequently, the score function can be explicitly encoded as a penalized negative log-likelihood function with an appropriate choice of regularization (Park and Klabjan 2017, Shojaie and Michailidis 2010, van de Geer and Bühlmann 2013). Hence, the process of computing scores (i.e., stage 1) is completely bypassed, and a single-stage model can be formulated (Xiang and Kim 2013). Moreover, in this case, IP formulations only require a polynomial, rather than exponential, number of variables, because each variable can be defined in the space of arcs instead of parent sets. Therefore, cardinality constraints on the size of parent sets, which are used in earlier methods to reduce the search space and may lead to suboptimal solutions, are no longer necessary.

Contributions. To summarize, in developing tailored exact DAG learning methods for continuous data from linear SEMs, we make the following contributions:

- We develop a mathematical framework that can naturally incorporate prior structural knowledge, when available. Prior structural knowledge can be supplied in the form of an undirected and possibly cyclic graph (super-structure). An example is the skeleton of the DAG, obtained by removing the direction of all edges in a graph. Another example is the moral graph of the DAG, obtained by adding an edge between pairs of nodes with common children and removing the direction of all edges (Spirtes et al. 2000). The skeleton and moral graphs are particularly important cases, because they can be consistently estimated from observational data under proper assumptions (Drton and Maathuis 2017, Kalisch and Bühlmann 2007, Loh and Bühlmann 2014, Sondhi and Shojaie 2019). Such prior information limits the number of possible DAGs and improves the computational performance.

– We discuss three mathematical formulations, namely, cutting plane (CP), linear ordering (LO), and topological ordering (TO) formulations, for learning an optimal DAG, using both ℓ_0 and ℓ_1-penalized likelihood scores. We also propose a new mathematical formulation to learn an optimal DAG from continuous data, the *layered network* (LN) formulation, and establish that other DAG learning formulations entail a smaller continuous relaxation feasible region compared to that of the continuous relaxation of the LN formulation (Propositions 3 and 4). Nonetheless, all formulations attain the same optimal continuous relaxation objective function value under a mild condition (Propositions 5 and 6). Notably, the number of binary variables and constraints in the LN formulation solely depend on the number of edges in the super-structure (e.g., moral graph). Thus, the performance of the LN formulation substantially improves in the presence of a sparse super-structure. The LN formulation has a number of other advantages; it is a compact formulation in contrast to the CP formulation; its relaxation can be solved much more efficiently compared with the LO formulation; and it requires fewer binary variables and explores fewer branch-and-bound nodes than the TO formulation. Our empirical results affirm the computational advantages of the LN formulation. They also demonstrate that the LN formulation can find a graph closer to the true underlying DAG. These improvements become more noticeable in the presence of a prior super-structure (e.g., moral graph).

– We compare the IP-based method and the A*-lasso algorithm for the case of ℓ_1 regularization. As noted earlier, there is no clear winner among A*-style algorithms and IP-based models for DAG learning from discrete data (Malone et al. 2014). Thus, a wide range of approaches based on dynamic programming, A* algorithm, and IP-based models have been proposed for discrete data. For DAG learning from continuous data with complete super-structure, the LN formulation remains competitive with the state-of-art A*-lasso algorithm for small graphs, whereas it performs better for larger problems. Moreover, LN performs substantially better when a sparse super-structure is available. This is mainly because the LN formulation directly defines the variables based on the super-structure, whereas the A*-lasso algorithm cannot take advantage of the prior structural knowledge as effectively as the LN formulation.

In Section 2, we outline the necessary preliminaries, define the DAG structure learning problem, and present a general framework for the problem. Mathematical formulations for DAG learning are presented in Section 3. The strengths of different mathematical formulations are discussed in Section 4. Empirical results are presented in Sections 5 and 6. We end the paper with a summary and discussion of future research in Section 7. Proofs of technical results as well as results of additional numerical studies are presented in the Appendix.

2 Penalized DAG Estimation with Linear SEMs

The causal effect of (continuous) random variables in a DAG $\mathcal{G}_0$ can be described by SEMs that represent each variable as a (nonlinear) function of its parents. The general form of these models is given by (Pearl et al. 2009)

$$X_k = f_k\big(pa_k^{\mathcal{G}_0}, \delta_k\big), \quad k = 1, \dots, m, \tag{1}$$

where X_k is the random variable associated with node k; $pa_k^{\mathcal{G}_0}$ denotes the parents of node k in $\mathcal{G}_0$, i.e., the set of nodes with arcs pointing to node k; m is the number of nodes; and δ_k is the latent

random variables representing the unexplained variation for node k.

An important class of SEMs, defined by linear functions $f_k(\cdot)$, can be described by m linear regressions of the form

$$X_k = \sum_{j \in pa_k^{\mathcal{G}_0}} \beta_{jk} X_j + \delta_k, \quad k = 1, \ldots, m, \tag{2}$$

where β_{jk} represents the effect of node j on k for $j \in pa_k^{\mathcal{G}_0}$. In the special case where the random variables are Gaussian, Equations (1) and (2) are equivalent, in the sense that β_{jk} are coefficients of the linear regression model of X_k on X_j, $j \in pa_k^{\mathcal{G}_0}$, and $\beta_{jk} = 0$ for $j \notin pa_k^{\mathcal{G}_0}$ (Shojaie and Michailidis 2010). However, estimation procedures proposed in this paper are not limited to Gaussian random variables and apply more generally to linear SEMs (Loh and Bühlmann 2014, Shojaie and Michailidis 2010).

Let $\mathcal{M} = (V, E)$ be an undirected and possibly cyclic super-structure graph with node set $V = \{1, 2, \ldots, m\}$ and edge set $E \subseteq V \times V$. From $\mathcal{M}$, generate a bi-directional graph $\overrightarrow{\mathcal{M}} = (V, \overrightarrow{E})$ where $\overrightarrow{E} = \{(j,k), (k,j) | (j,k) \in E\}$. Throughout the paper, we refer to directed edges as *arcs* and to undirected edges as *edges*.

Consider n i.i.d. observations from the linear SEM (2). Let $\mathcal{X} = (\mathcal{X}_1, \ldots, \mathcal{X}_m)$ be the $n \times m$ data matrix with n rows representing i.i.d. samples, and m columns representing random variables. The linear SEM (2) can be compactly written as

$$\mathcal{X} = \mathcal{X} B + \Delta, \tag{3}$$

where $B = [\beta] \in \mathbb{R}^{m \times m}$ is a matrix with $\beta_{kk} = 0$ for $k = 1, \ldots, m$ and $\beta_{jk} = 0$ for all $(j,k) \notin \overrightarrow{E}$; Δ is the $n \times m$ noise matrix. More generally, B defines a directed graph $\mathcal{G}(B)$ on m nodes such that arc (j,k) appears in $\mathcal{G}(B)$ if and only if $\beta_{jk} \neq 0$.

Let σ_k^2 denote the variance of $\delta_k; k = 1, 2, , \ldots, m$. We assume that all noise variables have equal variances, i.e., $\sigma_k = \sigma$. This condition implies the identifiability of DAGs from linear SEMs with Gaussian (Peters and Bühlmann 2013) and non-Gaussian (Loh and Bühlmann 2014) noise. Under this condition, the negative log likelihood for linear SEMs with Gaussian or non-Gaussian noise is proportional to the squared loss function

$$l_n(\beta) = \frac{1}{2} \text{tr}\{(I - B)(I - B)^T S\}, \tag{4}$$

where $S = n^{-1} \mathcal{X}^T \mathcal{X}$ is the empirical covariance matrix and I is the identity matrix (Loh and Bühlmann 2014).

In practice, we are often interested in learning *sparse* DAGs. Thus, a regularization term is used to obtain a sparse estimate. For the linear SEM (2), the optimization problem corresponding to the penalized negative log-likelihood with super-structure $\mathcal{M}$ (PNL$\mathcal{M}$) for learning sparse DAGs is given by

$$\textbf{PNL}\mathcal{M} \quad \min_{B \in \mathbb{R}^{m \times m}} \quad l_n(\beta) + \lambda \phi(B), \tag{5a}$$

$$\text{s.t., } \mathcal{G}(B) \text{ induces a DAG from } \overrightarrow{\mathcal{M}}. \tag{5b}$$

The objective function (5a) consists of two parts: the quadratic loss function, $l_n(\beta)$, from (4), and the regularization term, $\phi(B)$. Popular choices for $\phi(B)$ include ℓ_1-regularization or lasso

(Tibshirani 1996), $\phi(B) = \sum_{(j,k)\in\overrightarrow{E}} |\beta_{jk}|$, and ℓ_0-regularization, $\phi(B) = \sum_{(j,k)\in\overrightarrow{E}} \mathbf{1}(\beta_{jk})$, where $\mathbf{1}(\beta_{jk}) = 1$ if $\beta_{jk} \neq 0$, and 0 otherwise. The tuning parameter λ controls the degree of regularization. The constraint (5b) stipulates that the resulting directed subgraph (digraph) has to be an induced DAG from $\overrightarrow{\mathcal{M}}$.

For a complete super-structure $\mathcal{M}$, i.e. a complete undirected graph, PNL$\mathcal{M}$ reduces to the classical PNL. In this case, the consistency of *sparse* PNL for DAG learning from Gaussian data with an ℓ_0 penalty follows from an analysis similar to van de Geer and Bühlmann (2013). In particular, we have

$$pr(\widehat{\mathcal{G}}_n = \mathcal{G}^0) \to 1 \quad (n \to \infty),$$

where $\widehat{\mathcal{G}}_n$ is the estimate of the true structure $\mathcal{G}^0$. An important advantage of the PNL estimation problem is that its consistency does not require the (strong) faithfulness assumption (Peters and Bühlmann 2013, van de Geer and Bühlmann 2013).

The mathematical model (5) incorporates a super-structure $\mathcal{M}$ (e.g., moral graph) into the PNL model. When $\mathcal{M}$ is the moral graph, the consistency of sparse PNL$\mathcal{M}$ follows from the analysis in Loh and Bühlmann (2014), which studies the consistency of the following two-stage framework for estimating sparse DAGs: (1) infer the moral graph from the support of the inverse covariance matrix; and (2) choose the best-scoring induced DAG from the moral graph. The authors investigate conditions for identifiability of the underlying DAG from observational data and establish the consistency of the two-stage framework.

While PNL and PNL$\mathcal{M}$ enjoy desirable statistical properties for linear SEMs with Gaussian (van de Geer and Bühlmann 2013) and non-Gaussian (Loh and Bühlmann 2014) noise, the computational challenges associated with these problems have not been fully addressed. This paper aims to bridge this gap.

3 Mathematical Formulations

Prior to presenting mathematical formulations for solving PNL$\mathcal{M}$, we discuss a property of DAG learning from continuous data that distinguishes it from the corresponding problem for discrete data. To present this property in Proposition 1, we need a definition.

Definition 1. A tournament is a directed graph obtained by specifying a direction for each edge in the super-structure graph $\mathcal{M}$ (see Figure 1).

Proposition 1. *There exists an optimal solution $\mathcal{G}(B)$ to PNL$\mathcal{M}$ (5) with ℓ_0 (or an ℓ_1) regularization that is a cycle-free tournament.*

All proofs are given in Appendix I. Proposition 1 implies that for DAG learning from continuous variables, the search space reduces to acyclic tournament structures, where each original undirected edge in the super-structure graph is assigned a direction even though the corresponding β variable may have a zero value. This may not be the case for discrete data, because the score function for discrete data depends on the parents of each node, hence assigning directions to all edges in the super-structure graph imposes a parent set that may be suboptimal. The acyclic tournament structures is a much smaller search space when compared with the super-exponential $O\Big(m!2^{\binom{m}{2}}\Big)$ search space of DAGs for discrete variables. However, one has to also identify the optimal β parameters. The search for optimal β parameters is critical, as it further reduces the super-structure to the edges of the DAG by removing the edges with zero β coefficients.

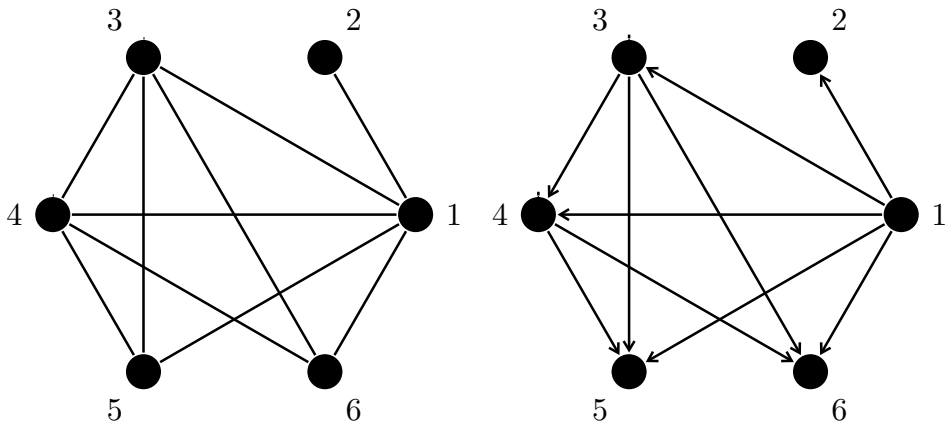


Figure 1: A super-structure graph $\mathcal{M}$ (left) and a tournament (right) with six nodes which does not contain any cycles.

A solution method based on brute force enumeration of all tournaments requires $\gamma m!$ computational time when $\mathcal{M}$ is complete, where γ denotes the computational time associated with solving PNL$\mathcal{M}$ given a known tournament structure. This is because when $\mathcal{M}$ is complete, the total number of tournaments (equivalently the total number of permutations) is $m!$. However, when $\mathcal{M}$ is incomplete, the number of tournaments is fewer than $m!$. The topological search space is $m!$ regardless of the structure of $\mathcal{M}$ and several topological orderings can correspond to the same DAG. The TO formulation by Park and Klabjan (2017) is based on this search space. In Section 3.2, we discuss a search space based on the layering of a DAG, which uniquely identifies a DAG, and propose the corresponding Layered Network (LN) formulation, which effectively utilizes the structure of $\mathcal{M}$. We first discuss existing mathematical formulations for the PNL$\mathcal{M}$ optimization problem (5) in the next section. Given the desirable statistical properties of ℓ_0 regularization (van de Geer and Bühlmann 2013) and the fact that existing mathematical formulations are given for ℓ_1 regularization, we present the formulations for ℓ_0 regularization. We outline the necessary notation below.

Indices
$V = \{1, 2, \ldots, m\}$: index set of random variables
$\mathcal{D} = \{1, 2, \ldots, n\}$: index set of samples

Input
$\mathcal{M} = (V, E)$: an undirected super-structure graph (e.g., the moral graph)
$\overrightarrow{\mathcal{M}} = (V, \overrightarrow{E})$: the bi-directional graph corresponding to the undirected graph $\mathcal{M}$
$\mathcal{X} = (\mathcal{X}_1, \ldots, \mathcal{X}_m)$, where $\mathcal{X}_v = (x_{1v}, x_{2v}, \ldots, x_{nv})^\top$ and x_{dv} denotes dth sample ($d \in \mathcal{D}$) of random variable X_v
λ : tuning parameter (penalty coefficient)

Continuous optimization variables
β_{jk}: weight of arc (j, k) representing the regression coefficients $\forall (j, k) \in \overrightarrow{E}$

Binary optimization variables

$z_{jk} = 1$ if arc (j,k) exists in a DAG; otherwise 0, $\forall (j,k) \in \overrightarrow{E}$
$g_{jk} = 1$ if $\beta_{jk} \neq 0$; otherwise 0, $\forall (j,k) \in \overrightarrow{E}$

3.1 Existing Mathematical Models

The main source of difficulty in solving PNL$\mathcal{M}$ is due to the acyclic nature of DAG imposed by the constraint in (5b). There are several ways to encode such a cycle elimination constraint, which we discuss next.

Let $\mathcal{C}$ be the set of all possible cycles and $\mathcal{C}_A \in \mathcal{C}$ be the set of arcs defining a cycle and define $F(\beta, g) := n^{-1} \sum_{k \in V} \sum_{d \in \mathcal{D}} \left(x_{dk} - \sum_{(j,k) \in \overrightarrow{E}} \beta_{jk} x_{dj} \right)^2 + \lambda \sum_{(j,k) \in \overrightarrow{E}} g_{jk}$. Then, the ℓ_0-PNL$\mathcal{M}$ model can be formulated as

$$\ell_0\textbf{-CP} \quad \min \quad F(\beta, g) \tag{6a}$$

$$-M g_{jk} \leq \beta_{jk} \leq M g_{jk}, \quad \forall (j,k) \in \overrightarrow{E}, \tag{6b}$$

$$\sum_{(j,k) \in \mathcal{C}_A} g_{jk} \leq |\mathcal{C}_A| - 1, \quad \forall \mathcal{C}_A \in \mathcal{C}, \tag{6c}$$

$$g_{jk} \in \{0,1\}, \quad \forall (j,k) \in \overrightarrow{E}. \tag{6d}$$

Following Park and Klabjan (2017), the objective function (6a) is an expanded version of $l_n(\beta)$ in PNL$\mathcal{M}$ (multiplied by $2n^{-1}$) with an ℓ_0 regularization. The constraints in (6b) stipulate that $\beta_{jk} \neq 0$ only if $g_{jk} = 1$; here M is a sufficiently large constant. The constraints in (6c) rule out all cycles. Note that for $|\mathcal{C}_A| = 2$, constraints in (6c) ensure that at most one arc exists among two nodes. The last set of constraints specifies the binary nature of the decision vector g. Note that β variables are continuous and unrestricted; however, in typical applications, they can be bounded by a finite number M. This formulation requires $|\overrightarrow{E}|$ binary variables and an exponential number of constraints. A cutting plane method (Nemhauser and Wolsey 1988) that adds the cycle elimination inequalities as needed is often used to solve this problem. We refer to this formulation as the *cutting plane* (CP) formulation.

Removing cycles in a graph bears resemblance to removing subtours in a Traveling Salesman Problem (TSP), which has been studied extensively (Cook et al. 1997). For example, similar to the cycle elimination constraint (6c) in the CP formulation, the subtour elimination constraint in the Dantzig-Fulkerson-Johnson (DFJ) formulation (Dantzig et al. 1954) is given by $\sum_{i \in Q} \sum_{j \in Q} g_{ij} \leq |Q| - 1$, $\forall Q \subsetneq \{1, \ldots, m\}$, $|Q| \geq 2$. However, this inequality is not valid for constructing DAGs, because the TSP constraint requires that there is exactly one incoming and one outgoing arc to any node.

The second formulation is based on a well-known combinatorial optimization problem, known as *linear ordering* (LO) (Grötschel et al. 1985). Given a finite set S with q elements, a linear ordering of S is a permutation $\mathcal{P} \in S_q$ where S_q denotes the set of all permutations with q elements. In the LO problem, the goal is to identify the best permutation among m nodes. The "cost" for a permutation $\mathcal{P}$ depends on the order of the elements in a pairwise fashion. Let p_j denote the order of node $j \in V$ in permutation $\mathcal{P}$. Then, for two nodes $j, k \in \{1, \ldots, m\}$, the cost is c_{jk} if the order of node j precedes the order of node k ($p_j \prec p_k$) and is c_{kj} otherwise ($p_j \succ p_k$). A binary variable w_{jk} indicates whether $p_j \prec p_k$. Because $w_{jk} + w_{kj} = 1$ and $w_{jj} = 0$, one only needs $\binom{m}{2}$ variables to cast the LO problem as an IP formulation (Grötschel et al. 1985).

The LO formulation for DAG learning from continuous data has two noticeable differences compared with the classical LO problem: (i) the objective function is quadratic, and (ii) an additional set of continuous variables, i.e., $\beta_{jk}, j, k = 1, \ldots, m$, is added. Cycles are ruled out by directly imposing the linear ordering constraints. The PNL$\mathcal{M}$ can be formulated as (7).

$$
\begin{aligned}
\ell_0\textbf{-LO} \quad \min \quad & F(\beta, g), && && \text{(7a)}\\
& -Mg_{jk} \leq \beta_{jk} \leq Mg_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(7b)}\\
& g_{jk} \leq w_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(7c)}\\
& w_{jk} + w_{kj} = 1, && \forall j, k \in V, j \neq k, && \text{(7d)}\\
& w_{ij} + w_{jk} + w_{ki} \leq 2, && \forall i, j, k \in V, i \neq j \neq k, && \text{(7e)}\\
& w_{jk} \in \{0, 1\}, && \forall j, k \in V, j \neq k, && \text{(7f)}\\
& g_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(7g)}
\end{aligned}
$$

The interpretation of constraints (7b)–(7d) is straightforward. The constraints in (7c) imply that if node j appears after node k in a linear ordering ($w_{jk} = 0$), then there should not exist an arc from j to k ($g_{jk} = 0$). The set of inequalities (7e) implies that if $p_i \prec p_j$ and $p_j \prec p_k$, then $p_i \prec p_k$. This ensures the linear ordering of nodes and removes cycles.

The third approach for ruling out cycles is to impose a set of constraints such that the nodes follow a topological ordering. A topological ordering is a linear ordering of the nodes of a graph such that the graph contains an arc (j, k) if node j appears before node k in the linear order. Define decision variables $o_{rs} \in \{0, 1\}$ for all $r, s \in \{1, \ldots, m\}$. This variable takes value 1 if topological order of node r (i.e., p_r) equals s, and 0, otherwise. If a topological ordering is known, the DAG structure can be efficiently learned in polynomial time (Shojaie and Michailidis 2010), but the problem remains challenging when the ordering is not known. The topological ordering prevents cycles in the graph. This property is used in Park and Klabjan (2017) to model the problem of learning a DAG with ℓ_1 regularization. We extend their formulation to ℓ_0 regularization. The *topological ordering* (TO) formulation is given below.

$$
\begin{aligned}
\ell_0\textbf{-TO} \quad \min \quad & F(\beta, g), && && \text{(8a)}\\
& -Mg_{jk} \leq \beta_{jk} \leq Mg_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(8b)}\\
& g_{jk} \leq z_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(8c)}\\
& z_{jk} + z_{kj} \leq 1, && \forall (j,k) \in \overrightarrow{E}, && \text{(8d)}\\
& z_{jk} - mz_{kj} \leq \sum_{s \in V} s\,(o_{ks} - o_{js}), && \forall (j,k) \in \overrightarrow{E}, && \text{(8e)}\\
& \sum_{s \in V} o_{rs} = 1, && \forall r \in V, && \text{(8f)}\\
& \sum_{r \in V} o_{rs} = 1, && \forall s \in V, && \text{(8g)}\\
& z_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}, && \text{(8h)}\\
& o_{rs} \in \{0, 1\}, && \forall r, s \in \{1, 2, \ldots, m\}, && \text{(8i)}\\
& g_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(8j)}
\end{aligned}
$$

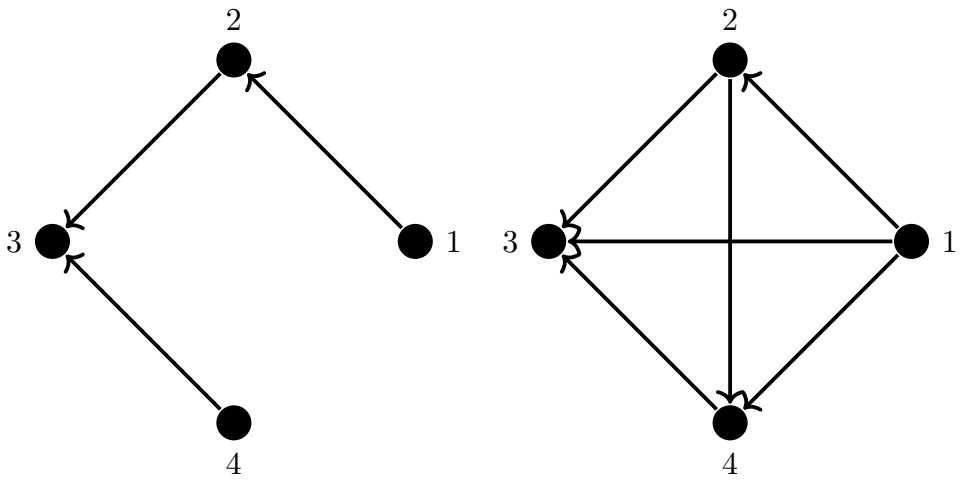


Figure 2: The role of the binary decision variables in ℓ_0 regularization: Using z (instead of g) in the objective function creates a graph similar to (b) and counts the number of arcs instead of the number of non-zero βs in (b).

In this formulation, z_{jk} is an auxiliary binary variable which takes value 1 if an arc exists from node j to node k. Recall that, $g_{jk} = 1$ if $|\beta_{jk}| > 0$. The constraints in (8c) enforce the correct link between g_{jk} and z_{jk}, i.e., g_{jk} has to take value zero if $z_{jk} = 0$. The constraints in (8d) imply that there should not exist a bi-directional arc among two nodes. This inequality can be replaced with equality (Corollary 1). The constraints in (8e) remove cycles by imposing an ordering among nodes. The set of constraints in (8f)–(8g) assigns a unique topological order to each node. The last two sets of constraints indicate the binary nature of decision variables o and z.

Corollary 1, which is a direct consequence of Proposition 1, implies that we can use $z_{jk} = 1 - z_{kj}$ for all $j < k$ and reduce the number of binary variables.

Corollary 1. *The constraints in* (8d) *can be replaced by* $z_{jk} + z_{kj} = 1,\ \forall (j,k) \in \overrightarrow{E}$.

It may be appealing to consider a simpler formulation without the z variables, by replacing all occurrences of z_{jk} with g_{jk}. However, in constraints (8b)–(8i), both variables z and g are needed to correctly model the ℓ_0 regularization term in the objective function (see Figure 2). This is because the constraints (8e) satisfy the transitivity property: if $z_{ij} = 1$ for $(i,j) \in \overrightarrow{E}$ and $z_{jk} = 1$ for $(j,k) \in \overrightarrow{E}$, then $z_{ik} = 1$ for $(i,k) \in \overrightarrow{E}$, since $z_{ij} = 1$ implies that $\sum_{s \in V} s(o_{js} - o_{is}) \geq 1$; similarly, $z_{jk} = 1$ implies $\sum_{s \in V} s\,(o_{ks} - o_{js}) \geq 1$. If we sum both inequalities, we have $\sum_{s \in V} s\,(o_{ks} - o_{is}) \geq 2$. From constraint (8e) for $(k,i) \in \overrightarrow{E}$, we then have $z_{ki} - m z_{ik} \leq \sum_{s \in V} s(o_{is} - o_{ks}) \leq -2$. Therefore, we must have $z_{ik} = 1$. However, we may have $\beta_{ik} = \beta_{ki} = 0$, and in conjunction with the ℓ_0 term in the objective, we have $g_{ik} = 0 < z_{ik}$.

3.2 A New Mathematical Model: The Layered Network (LN) Formulation

As an alternative to the existing mathematical formulations, we propose a new formulation for imposing acyclicity constraints that is motivated by the layering of nodes in DAGs (Healy and Nikolov 2002). More specifically, our formulation ensures that the resulting graph is a *layered network*, in the sense that there exists no arc from a layer v to layer u, where $u < v$. Let ψ_k be the *layer value* for node k. One feasible assignment of ψ_k is $\sum_{s=1}^{m} s\, o_{ks}$ for all $k \in V$, where the variables o_{ks} are as defined in the TO formulation. However, the layer value ψ_k is more general because ψ_k need not be integer and it need not be distinct for all k. Figure 3 depicts the layered network encoding of a DAG. With this notation, our *layered network* (LN) formulation can be written as

$$\min \quad F(\beta, g), \tag{9a}$$

$$-Mg_{jk} \le \beta_{jk} \le Mg_{jk}, \qquad \forall (j,k) \in \overrightarrow{E}, \tag{9b}$$

$$g_{jk} \le z_{jk}, \qquad \forall (j,k) \in \overrightarrow{E}, \tag{9c}$$

$$z_{jk} + z_{kj} = 1, \qquad \forall (j,k) \in \overrightarrow{E}, \tag{9d}$$

$$z_{jk} - (m-1)z_{kj} \le \psi_k - \psi_j, \quad \forall (j,k) \in \overrightarrow{E}, \tag{9e}$$

$$z_{jk} \in \{0,1\}, \qquad \forall (j,k) \in \overrightarrow{E}, \tag{9f}$$

$$1 \le \psi_k \le m, \qquad \forall k \in V, \tag{9g}$$

$$g_{jk} \in \{0,1\}, \qquad \forall (j,k) \in \overrightarrow{E}. \tag{9h}$$

The interpretation of the constraints (9b)–(9c) is straightforward. The constraints in (9e) ensure that the graph is a layered network. The last set of constraints indicates the continuous nature of the decision variable ψ and gives the tightest valid bound for ψ. It suffices to consider any real number for layer values ψ as long as layer values of any two nodes differ by at least one if there exists an arc between them. Additionally, LN uses a tighter inequality compared to TO, by replacing m with $m-1$ in (9e). This is because the difference between the layer values of two nodes can be at most $m-1$ for a DAG with m nodes. The next proposition establishes the validity of the LN formulation.

Proposition 2. *An optimal solution to* (9) *is an optimal solution to* (5).

The LN formulation highlights a desirable property of the layered network representation of a DAG in comparison with the topological ordering representation. Let us define nodes in layer 1 as the set of nodes that have no incoming arcs in the DAG, nodes in layer 2 as the set of nodes that have incoming arcs only from nodes in the layer 1, layer 3 as the set of nodes that have incoming arcs from layer 2 (and possibly layer 1), etc; see Figure 3. The *minimal layer number* of a node is the length of the *longest* directed path from any node in layer 1 to that node. For a given DAG, there is a unique minimal layer number, but not a unique topological order. As an example, Figure 2 has three valid topological orders: (i) 1,2,4,3, (ii) 1,4,2,3, and (iii) 4,1,2,3. In contrast, it has a unique layer representation, 1,2,3,1.

There is a one-to-one correspondence between minimal layer numbering and a DAG. However, the solutions of the LN formulation; i.e., ψ variables (layer values), do not necessarily correspond to the minimal layer numbering. This is because the LN formulation does not impose additional

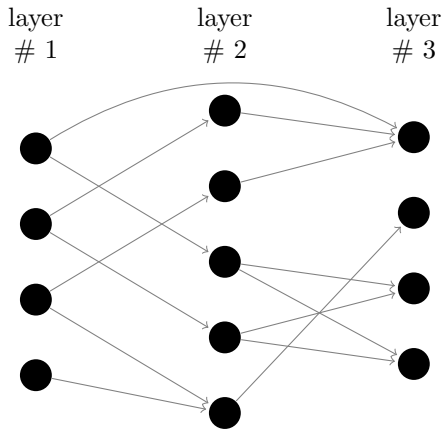


Figure 3: Layered Network encoding of a DAG.

constraints to enforce a minimal numbering and can output solutions that are not minimally numbered. However, because branch-and-bound does not branch on continuous variables, alternative (non-minimal) feasible solutions for the ψ variables do not impact the branch-and-bound process. On the contrary, we have multiple possible representations of the same DAG with topological ordering because TO is a symmetric formulation i.e., its variables can be permuted without changing the structure of the problem. Because topological ordering variables are binary, the branch-and-bound method applied to the TO formulation explores multiple identical DAGs as it branches on the topological ordering variables. This enlarges the size of the branch-and-bound tree and increases the computational burden.

Layered network representation also has an important practical implication: Using this representation, the search space can be reduced to the total number of ways we can layer a network (or equivalently the total number of possible minimal layer numberings) instead of the total number of topological orderings. When the super-structure $\mathcal{M}$ is complete, the two quantities are the same, and equal to $m!$. Otherwise, a brute-force search for finding the optimal DAG has computational time $\mathcal{L}\gamma$, where $\mathcal{L}$ denotes the total number of minimal layered numberings, and γ is the computational complexity of solving PNL$\mathcal{M}$ given a known tournament structure.

We close this section by noting that a related set of constraints (9e) appear in the Miller-Tucker-Zemlin (MTZ) formulation for asymmetric TSP (Cook et al. 1997, Miller et al. 1960, Sawik 2016). In this context, constraints (9e), along with the TSP constraints that ensure each node has one incoming and one outgoing arc, result in distinct ψ values that correspond to the order in which a node is visited in the TSP tour. In contrast, in LN formulation for DAG learning, the ψ values are not necessarily distinct and we provide their interpretation as layers of a DAG. Strong valid inequalities for MTZ formulation are proposed for asymmetric TSP (e.g., Bektaş and Gouveia 2014, Desrochers and Laporte 1991); however, they are not valid for DAG structural learning, because they are derived using the TSP constraint that there must be one incoming and one outgoing arc to every node. In addition, a set of constraints similar to (9e) is introduced in Cussens (2010) for learning DAG with discrete data. However, the formulation in Cussens (2010) requires an exponential number of variables. In more recent work on DAGs for discrete data, Cussens and colleagues have focused on a tighter formulation for removing cycles, known as cluster constraints

Table 1: The number of binary variables and the number of constraints

	Incomplete (moral) $\mathcal{M}$				Complete (moral) $\mathcal{M}$			
	CP	LO	TO	LN	CP	LO	TO	LN
# Binary Vars (ℓ_0)	$\lvert\overrightarrow{E}\rvert$	$\lvert\overrightarrow{E}\rvert + \binom{m}{2}$	$m^2 + \lvert E\rvert + \lvert\overrightarrow{E}\rvert$	$\lvert E\rvert + \lvert\overrightarrow{E}\rvert$	$2\binom{m}{2}$	$\binom{m}{2}$	$m^2 + 3\binom{m}{2}$	$3\binom{m}{2}$
# Binary Vars (ℓ_1)	$\lvert E\rvert$	$\binom{m}{2}$	$m^2 + \lvert E\rvert$	$\lvert E\rvert$	$\binom{m}{2}$	$\binom{m}{2}$	$m^2 + \binom{m}{2}$	$\binom{m}{2}$
# Constraints (both ℓ_0 and ℓ_1)	*Exp*	$2\binom{m}{3}$	$\lvert\overrightarrow{E}\rvert + 2m$	$\lvert\overrightarrow{E}\rvert$	$2\binom{m}{3}$	$2\binom{m}{3}$	$\binom{m}{2} + 2m$	$\binom{m}{2}$

(Cussens 2012, Cussens et al. 2017a,b). To represent the set of cluster constraints, variables have to be defined according to the parent set choice leading to an exponential number of variables. Thus, such a representation is not available in the space of arcs.

3.2.1 Layered Network with ℓ_1 regularization

Because of its convexity, the structure learning literature has utilized the ℓ_1-regularization for learning DAGs from continuous variables (Han et al. 2016, Park and Klabjan 2017, Shojaie and Michailidis 2010, Xiang and Kim 2013, Zheng et al. 2018). The LN formulation with ℓ_1-regularization can be written as

$$\min\ n^{-1} \sum_{i\in I} \sum_{k\in V} \left(x_{ik} - \sum_{(j,k)\in E^{\rightarrow}} \beta_{jk} x_{ij} \right)^2 + \lambda \sum_{(j,k)\in\overrightarrow{E}} |\beta_{jk}|, \tag{10a}$$

$$-Mz_{jk} \le \beta_{jk} \le Mz_{jk} \qquad \forall (j,k) \in \overrightarrow{E}, \tag{10b}$$

$$(9e) - (9g).$$

We next remark on two properties that only hold for a complete super-structure.

Remark 1. For a complete super-structure $\mathcal{M}$, $\psi_k = \sum_{j\in V\setminus k} z_{jk}\ \forall k \in V$. Thus, the LN formulations (both ℓ_0 and ℓ_1) can be encoded without ψ variables by writing (9e) as

$$z_{jk} - (m-1)z_{kj} \le \sum_{j\in V\setminus k} z_{jk} - \sum_{k\in V\setminus j} z_{kj} \quad \forall\, j,k \in V \quad j \ne k.$$

Remark 2. CP and LO formulations reduce to the same formulation for ℓ_1 regularization when the super-structure $\mathcal{M}$ is complete by letting $w_{ij} = g_{ij}$ in formulation (7) for all $(j,k) \in \overrightarrow{E}$.

An advantage of the ℓ_1-regularization for DAG learning is that all models (CP, LO, TO and LN) can be formulated without decision variables g_{jk}, since counting the number of non-zero β_{jk} is no longer necessary.

Table 1 shows the number of binary variables and the number of constraints associated with cycle prevention constraints in each model. Evidently, ℓ_0 problems require additional binary variables compared with the corresponding ℓ_1 problems. Note that the number of binary variables and constraints for the LN formulation solely depend on the number of edges in the super-structure $\mathcal{M}$. This property is particularly desirable when the super-structure $\mathcal{M}$ is sparse. The LN formulation

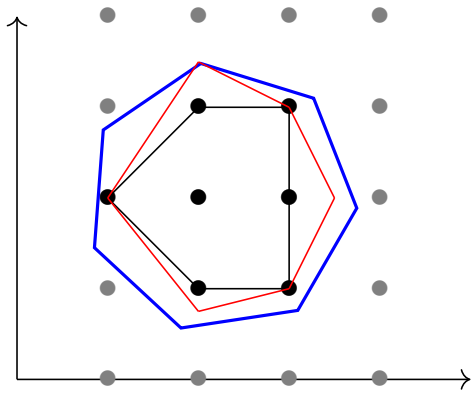


Figure 4: Continuous relaxation regions of three hypothetical IP models. The tightest model (convex hull) is represented by the black polygon strictly inside other polygons. The blue and red polygons represent valid yet weaker formulations. The black points show the feasible integer points for all three formulations.

requires the fewest number of constraints among all models. The LN formulation also requires fewer binary variables than the TO formulation. More importantly, different topological orders for the same DAG are symmetric solutions to the associated TO formulation. Consequently, branch-and-bound requires exploring multiple symmetric formulations as it branches on fractional TO variables. As for the LO formulation, the number of constraints is $O(m^3)$ which makes its continuous relaxation cumbersome to solve in the branch-and-bound process. The LN formulation is compact, whereas the CP formulation requires an exponential number of constraints for incomplete super-structure $\mathcal{M}$. The CP formulation requires fewer binary variables for ℓ_0 formulation than LN; both formulations need the least number of binary variables for ℓ_1 regularization.

In the next section, we discuss the theoretical strength of these mathematical formulations and provide a key insight on why the LN formulation performs well for learning DAGs from continuous data.

4 Continuous Relaxation

One of the fundamental concepts in IP is relaxations, wherein some or all constraints of a problem are loosened. Relaxations are often used to obtain a sequence of easier problems which can be solved efficiently yielding bounds and approximate, not necessarily feasible, solutions for the original problem. Continuous relaxation is a common relaxation obtained by relaxing the binary variables of the original mixed-integer quadratic program (MIQP) and allowing them to take real values. Continuous relaxation is at the heart of branch-and-bound methods for solving MIQPs. An important concept when comparing different MIQP formulations is the strength of their continuous relaxations.

Definition 2. A formulation A is said to be *stronger* than formulation B if $\mathcal{R}(A) \subset \mathcal{R}(B)$ where $\mathcal{R}(A)$ and $\mathcal{R}(B)$ correspond to the feasible regions of continuous relaxations of A and B, respectively.

Proposition 3. *The LO formulation is stronger than the LN formulation, that is,* $\mathcal{R}(LO) \subset \mathcal{R}(LN)$.

Proposition 4. *When the parameter* m *in* (8e) *is replaced with* $m - 1$*, the TO formulation is stronger than the LN formulation, that is,* $\mathcal{R}(TO) \subset \mathcal{R}(LN)$.

These propositions are somewhat expected because the LN formulation uses the fewest number of constraints. Hence, the continuous relaxation feasible region of the LN formulation is loosened compared with the other formulations. Related strength results are also established for MTZ formulation of TSPs; see e.g., Padberg and Sung (1991), Pataki (2003). Even though one of the constraints in the LN formulation for DAG learning is similar to MTZ formulation for TSP, the polyhedron corresponding to the LN formulation is different, and thus the strength results for LN do not directly follow from those for MTZ. We establish these results in Propositions 3 and 4.

For mixed-integer *linear* programs with linear objective functions such as TSP, it is known that stronger formulations attain better computational performance, see Öncan et al. (2009) for a comparative analysis in the TSP context. This is because, in this case, the continuous relaxations are linear programs, and there exists an optimal solution to a linear program that is an extreme point of the associated feasible set. Therefore, a stronger formulation that better approximates the convex hull of a mixed-integer linear program generally provides better lower bounds from relaxations, which in turn translates to faster convergence. In contrast, DAG structural learning for continuous variables with linear SEMs is cast as a MIQP. Due to the quadratic objective, an optimal solution may not be at an extreme point of the convex hull of feasible points. Thus, stronger formulations do not necessarily guarantee better lower bounds. The next two results justify the advantages of the LN formulation. Note that the initial continuous relaxation is obtained by relaxing all binary variables to $[0, 1]$ in each formulation.

Proposition 5. *Let* $\beta^\star_{jk}$ *denote the optimal coefficient associated with an arc* $(j, k) \in \overrightarrow{E}$ *from* (5)*. For both* ℓ_0 *and* ℓ_1 *regularizations, the initial continuous relaxations of the LN formulation attain as tight an optimal objective function value as the LO, CP, TO formulations if* $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$.

Proposition 5 states that although the LO and TO formulations are tighter than the LN formulation with respect to the feasible region of their continuous relaxations, the continuous relaxation of all models attain the same objective function value (root relaxation).

Proposition 6. *For branching on the same variable (i.e., the binary variables associated with the same arc) in the branch-and-bound process, the continuous relaxations of the LN formulation for both* ℓ_0 *and* ℓ_1 *regularizations attain as tight an optimal objective function value as LO, CP and TO, if* $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$.

Proposition 6 is at the crux of this section. It shows that not only does the tightness of the optimal objective function value of the continuous relaxation hold for the initial relaxation, but it also holds throughout the branch-and-bound process under the specified condition on M, if the same branching choices are made on the binary variables that take fractional values. Thus, the advantages of the LN formulation are due to the fact that it is a compact formulation that entails the fewest number of constraints, while attaining the same optimal objective value of continuous relaxation as tighter models.

In practice, finding a tight value for M is difficult. Our computational results show that the approach suggested in Park and Klabjan (2017) to obtain a value of M, which is explained in Section 5 and used in our computational experiments, always satisfies the condition in Proposition 6 across all generated instances.

5 Comparison of MIQP Formulations

We present numerical results comparing the proposed LN formulation with existing approaches. Experiments are performed on a cluster operating on UNIX with Intel Xeon E5-2640v4 2.4GHz. All MIQP formulations are implemented in the Python programming language. Gurobi 8.0 is used as the MIQP solver. A time limit of $50m$ (in seconds), where m denotes the number of nodes, is imposed across all experiments after which runs are aborted. Unless otherwise stated, an MIQP optimality gap of 0.001 is imposed across all experiments; the gap is calculated by $\frac{UB-LB}{UB}$ where UB denotes the objective value associated with the best feasible integer solution (incumbent) and LB represents the best obtained lower bound during the branch-and-bound process.

For CP, instead of incorporating all constraints given by (6c), we begin with no constraint of type (6c). Given an integer solution with cycles, we detect a cycle and impose a new cycle prevention constraint to remove the detected cycle. Depth First Search (DFS) can detect a cycle in a directed graph with complexity $O(|V| + |E|)$. Gurobi Lazy Callback is used, which allows adding cycle prevention constraints in the branch-and-bound algorithm, whenever an integer solution with cycles is found. The same approach is used by Park and Klabjan (2017). Note that Gurobi solver follows a branch-and-cut implementation and adds many general-purpose and special-purpose cutting planes.

To select the M parameter in all formulations, we use the proposal of Park and Klabjan (2017). Specifically, given λ, we solve each problem without cycle prevention constraints. We then use the upper bound $M = 2 \max_{(j,k)\in\overrightarrow{E}} |\beta_{jk}|$. The results provided in Park and Klabjan (2017) computationally confirm that this approach gives a large enough value of M. We also confirmed the validity of this choice across all our test instances.

In this section we report the numerical results on synthetic datasets for the four MIQP formulations. In particular, we provide a comprehensive analysis for Erdős-Rényi graphs. In Appendix III, we also report the results for another family of random graphs, namely Barabási-Albert graphs. Results for real datasets are presented in Appendix IV. The relegation of the additional numerical results to the Appendix is mainly for brevity and consistency of computational platforms and the findings from all studies are similar.

We use the `R` package `pcalg` to generate random Erdős-Rényi graphs. Firstly, we create a DAG using `randomDAG` function and assign random arc weights (i.e., β) from a uniform distribution, $\mathcal{U}[0.1, 1]$. This *ground truth* DAG is used to assess the quality of estimates. Next, the resulting DAG and random coefficients are input to the `rmvDAG` function, which uses linear regression as the underlying model, to generate multivariate data (columns of matrix $\mathcal{X}$) with the standard normal error distribution.

We consider $m \in \{10, 20, 30, 40\}$ nodes and $n \in \{100, 1000\}$ samples. The average outgoing degree of each node, denoted by d, is set to 2. We generate 10 random graphs for each setting (m, n, d). The observed data $\mathcal{X}$ for the datasets with $n = 100$ is the same as first 100 rows of the datasets with $n = 1000$.

We consider two types of problem instances: (i) a set of instances for which the moral graph corresponding to the true DAG is available; (ii) a set of instances with a complete undirected

super-structure, i.e., assuming no prior knowledge. The first class of problems is referred to as *moral* instances, whereas the second class is called *complete* instances. The observed data $\mathcal{X}$ for moral and complete instances are the same. The function `moralize(graph)` in the `pcalg` R-package is used to generated the moral graph from the true DAG. The moral graph can also be (consistently) estimated from data using penalized estimation procedures with polynomial complexity (Kalisch and Bühlmann 2007, Loh and Bühlmann 2014). However, since the quality of the moral graph equally affects all optimization models, the true moral graph is used in our experiments.

We use the following IP-based metrics to measure the quality of a solution: Optimality gap (MIQP GAP), computation time in seconds (Time), Upper Bound (UB), Lower Bound (LB), computational time of root continuous relaxation (Time LP), and the number of explored nodes in the branch-and-bound tree.

We also evaluate the quality of the estimated DAGs by comparing them with the ground truth. To this end, we use the average structural Hamming distance (SHD), as well as true positive (TPR) and false positive rates (FPR). These criteria evaluate different aspects of the quality of the estimated DAGs: SHD counts the number of differences (addition, deletion, or arc reversal) required to transform predicted DAG to the true DAG; TPR is the number of correctly identified arcs divided by the total number of true arcs, P; FPR is the number of incorrectly identified arcs divided by the total number of negatives (non-existing arcs), N. For brevity, TPR and FPR plots are presented in Appendix II.

5.1 Comparison of ℓ_0 formulations

Figure 5 reports the average metrics across 10 random graphs for ℓ_0 formulations with $n = 1000$. The LO formulation fails to attain a reasonable solution for one graph (out of 10) with $m = 40$ and $\lambda \in \{0.1, 1\}$. This is due to the large computation time for solving its continuous relaxation. We excluded these two instances from LO results.

Figure 5(a) shows that the LN formulation outperforms other formulations in terms of the average optimality gap across all number of nodes $m \in \{10, 20, 30, 40\}$ and regularization parameters, $\lambda \in \{0.1, 1\}$. The difference becomes more pronounced for moral instances. For moral instances, the number of binary variables and constraints for LN solely depends on the size of the moral graph. Figure 5(b) also indicates that the LN formulation requires the least computational time for small instances, whereas all models hit the time limit for larger instances.

Figures 5(c)–(d) show the performance of all methods in terms of their upper and lower bounds. For easier instances (e.g., complete instances with $m \in \{10, 20\}$ and moral instances), all methods attain almost the same upper bound. Nonetheless, LN performs better in terms of improving the lower bound. For more difficult instances, LN outperforms other methods in terms of attaining a smaller upper bound (feasible solution) and a larger lower bound.

Figures 5(e)–(f) show the continuous relaxation time of all models, and the number of explored nodes in the branch-and-bound tree, respectively. The fastest computational time for the continuous relaxation is for the TO formulation followed by the LN formulation. However, the number of explored nodes provides more information about the performance of mathematical formulations. In small instances, i.e., $m = 10$, where an optimal solution is attained, the size of the branch-and-bound tree for the LN formulation is smaller than the TO formulation. This is because the TO formulation has a larger number of binary variables, leading to a larger branch-and-bound tree. On the other hand, for large instances, the number of explored nodes in the LN formulation is larger than the TO formulation. This implies that the LN formulation explores more nodes in the

branch-and-bound tree given a time limit. This may be because continuous relaxations of the LN formulation are easier to solve in comparison to the continuous relaxations of the TO formulation in the branch-and-bound process. As stated earlier, the branch-and-bound algorithm needs to explore multiple symmetric formulations in the TO formulation as it branches on fractional topological ordering variables. This degrades the performance of the TO formulation. The LO formulation is very slow because its continuous relaxation becomes cumbersome as the number of nodes, m, increases. Thus, we can see a substantial decrease in the number of explored nodes in branch-and-bound trees associated with the LO formulation. The CP formulation is implemented in a cutting-plane fashion. Hence, its number of explored nodes is not directly comparable with other formulations.

Figures 5(a)–(f) show the importance of incorporating available structural knowledge (e.g., moral graph). The average optimality gap and computational time are substantially lower for moral instances compared to complete instances. Moreover, the substantial difference in the optimality gap elucidates the importance of incorporating structural knowledge. Similar results are obtained for $n = 100$ samples; see Appendix II.

We next discuss the performance of different methods in terms of estimating the true DAG. The choice of tuning parameter λ, the number of samples n, and the quality of the best feasible solution (i.e., upper bound) influence the resulting DAG. Because our focus in this paper is on computational aspects, we fixed the values of λ for a fair comparison between the formulations, and used $\lambda = 0.1$ based on results in preliminary experiments. Thus, we focus on the impact of sample size as well as the quality of the feasible solution in the explanation of our results.

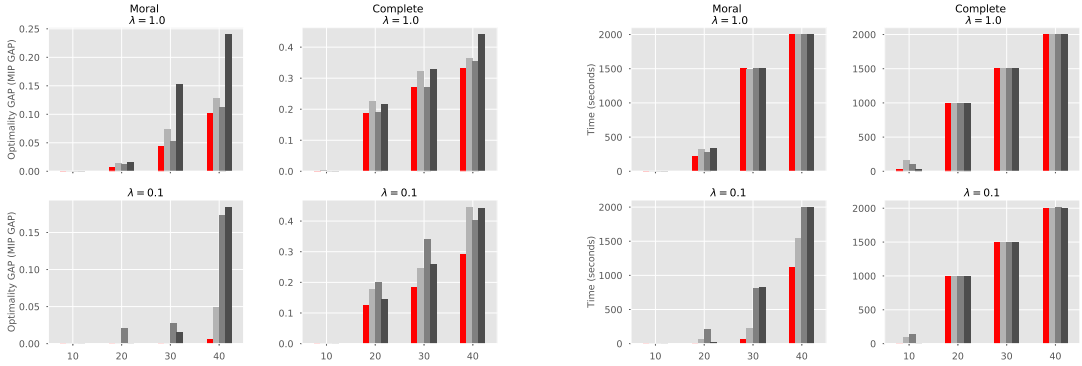
Figures 6(a)–(b) show the SHDs for all formulations for $n = 1000$ and $n = 100$, respectively. Comparing Figure 6(a) with Figure 6(b), we observe that the SHD tends to increase as the number of samples decreases. As discussed earlier, when $n \to \infty$, penalized likelihood likelihood estimate with an ℓ_0 regularization ensures consistency in our setting (Peters and Bühlmann 2013, van de Geer and Bühlmann 2013). However, this may not be guaranteed for a finite sample size. Moreover, the appropriate choices of λ for $n = 100$ and $n = 1000$ may be different.

Figure 6(a) shows that all methods learn the true DAG with $\lambda = 0.1$, and given a moral graph for $m \in \{10, 20, 30\}$. In addition, SHD is negligible for LN and CP formulations for $m = 40$. However, we observe a substantial increase in SHD (e.g., from 0.2 to near 10 for LN) for complete graphs. These figures indicate the importance of incorporating available structural knowledge (e.g., a moral graph) for better estimation of the true DAG.

While, in general, LN performs well compared with other formulations, we do not expect to see a clear dominance in terms of accuracy of DAG estimation either due to finite samples or the fact that none of the methods could attain a global optimal solution for larger instances. On the contrary, we retrieve the true DAG for smaller graphs for which optimal solutions are obtained. As pointed out in Park and Klabjan (2017), a slight change in the objective function value could significantly alter the estimated DAG. Our results corroborate this observation.

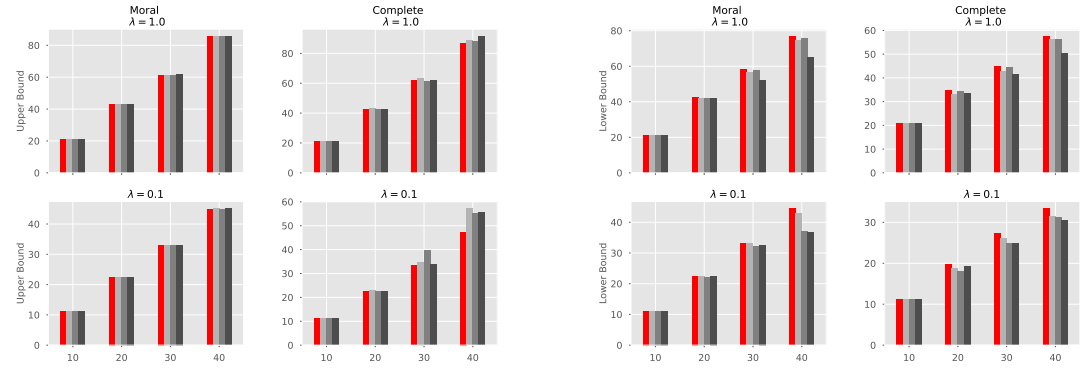
5.2 Comparison of ℓ_1 formulations

Figure 7 shows various average metrics across 10 random graphs for ℓ_1 regularization with $n = 1000$ samples. Figure 7(a) shows that the LN formulation clearly outperforms other formulations in terms of average optimality gap across all number of nodes, $m \in \{10, 20, 30, 40\}$, and regularization parameters, $\lambda \in \{0.1, 1\}$. Moreover, Figure 7(b) shows that the LN formulation requires significantly less computational time in moral instances, and in complete instances with $m \in \{10, 20\}$ compared



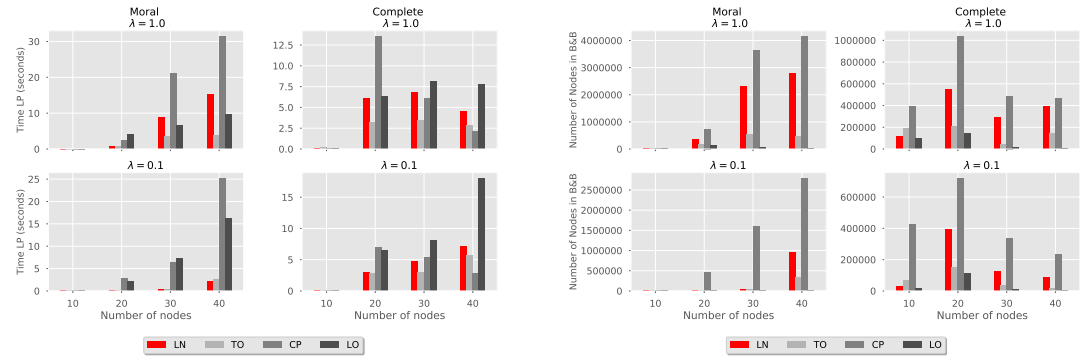
(a) Optimality GAPs for MIQPs

(b) Time (in seconds) for MIQPs



(c) Best upper bounds for MIQPs

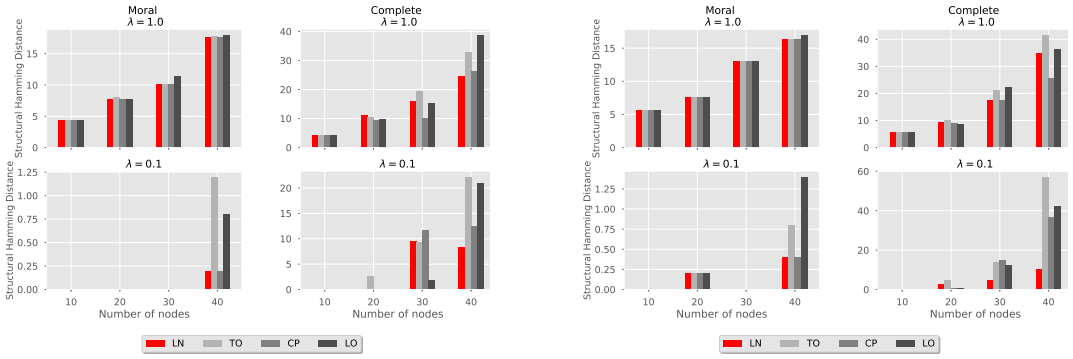
(d) Best lower bounds for MIQPs



(e) Time (in seconds) for continuous root relaxation

(f) Number of explored nodes in B&B tree

Figure 5: Optimization-based measures for MIQPs for ℓ_0 regularization with the number of samples $n = 1000$.



(a) $n = 1000$ (b) $n = 100$

Figure 6: Structural Hamming Distance (SHD) of MIQP estimates with ℓ_0 regularization.

to other methods. In complete instances, all methods hit the time limit for $m \in \{30, 40\}$. Figures 7(c)–(f) can be interpreted similar to the Figures 5(c)–(f) for ℓ_0 regularization. Similar to ℓ_0 regularization, Figures 7(a)–(b) demonstrate the importance of incorporating structural knowledge (e.g., a moral graph) for ℓ_1 regularization. Similar results are observed for $n = 100$ samples; see Appendix II.

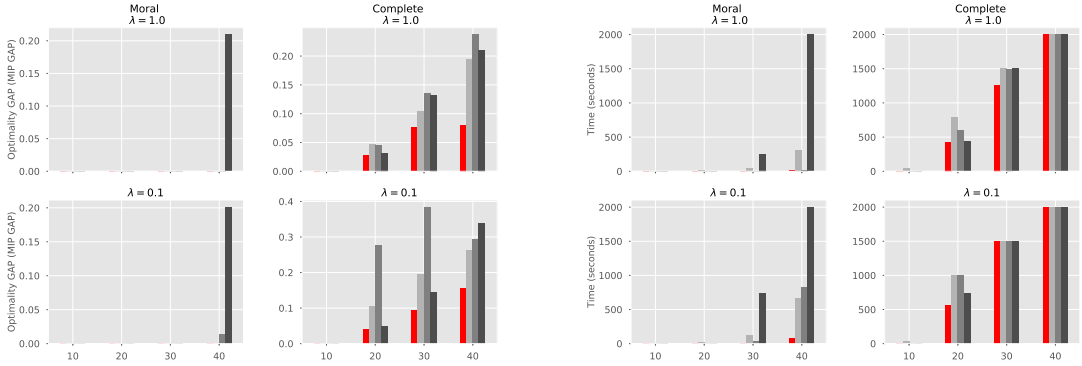
As expected, the DAG estimation accuracy with ℓ_1 regularization is inferior to the ℓ_0 regularization. This is in part due to the bias associated with the ℓ_1 regularization, which could be further controlled with, for example, adaptive ℓ_1-norm regularization (Zou 2006). Nonetheless, formulations for ℓ_1 regularization require less computational time and are easier to solve than the corresponding formulations for ℓ_0 regularization.

6 Comparison with the A*-lasso algorithm

In this section, we compare the LN formulation with A*-lasso (Xiang and Kim 2013), using the MATLAB code made available by the authors. In Appendix IV, we show that the LN formulation outperforms the alternative formulations; therefore, the comparison in this section only focuses on LN formulation and A*-lasso.

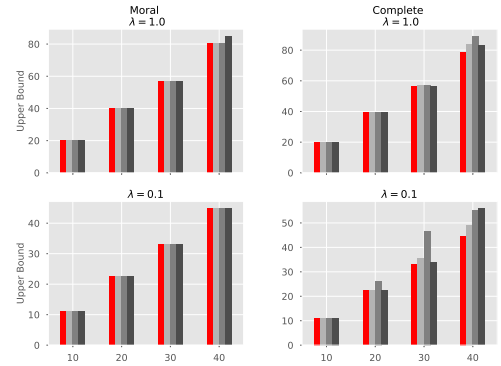
For this comparison, the true DAG structures are taken from Xiang and Kim (2013) and the strength of arcs (β) are chosen from $\mathcal{U}[-1, -0.1] \cup \mathcal{U}[0.1, 1]$. The number of nodes in the 10 true DAGs varies from $m = 6$ to $m = 27$ (see Table 2). The true DAG and the corresponding β coefficients are used to generate $n = 500$ samples for each column of data matrix $\mathcal{X}$. Similar to synthetic data described in Section 5, we consider two cases: (i) *moral* instances and (ii) *complete* instances. For the former case, the moral graph is constructed from the true DAG as done in Section 5. The observed data $\mathcal{X}$ for moral and complete instances are the same.

A time limit of six hours is imposed across all experiments after which runs are aborted. In addition, for a fairer comparison with A*-lasso, we do not impose an MIQP gap termination criterion of 0.001 for LN and use the Gurobi default optimality gap criterion of 0.0001. Moreover, we compare the LN formulation with A*-lasso using ℓ_1 regularization. Note that A*-lasso cannot solve the model with ℓ_0 regularization. Finally, the original A*-lasso algorithm assumes no super-structure.

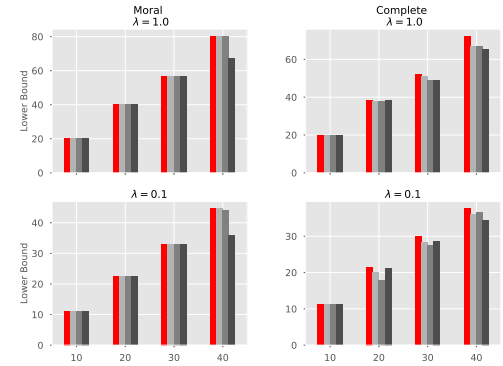


(a) Optimality GAPs for MIQPs

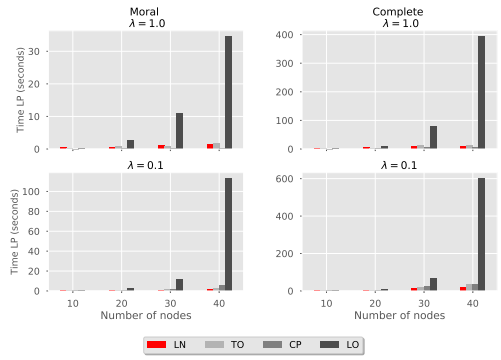
(b) Time (in seconds) for MIQPs



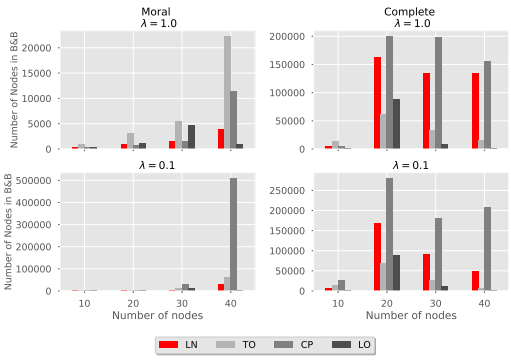
(c) Best upper bounds for MIQPs



(d) Best lower bounds for MIQPs

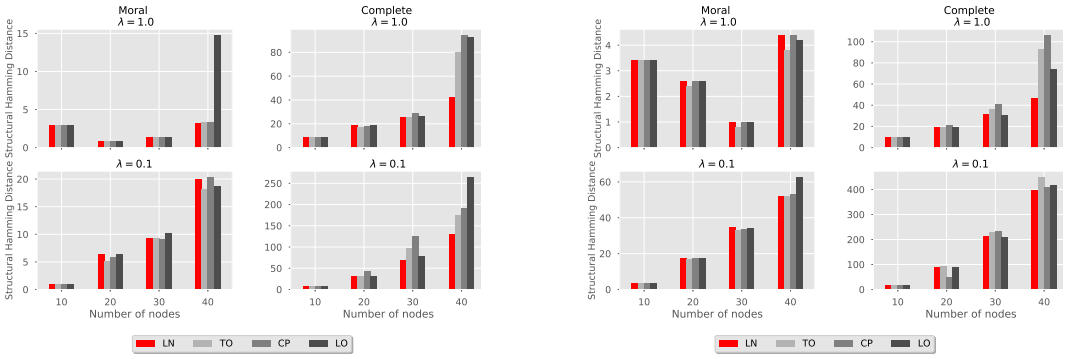


(e) Time (in seconds) for continuous root relaxation



(f) Number of explored nodes in B&B tree

Figure 7: Optimization-based measures for MIQPs for ℓ_1 regularization with the number of samples $n = 1000$.



(a) $n = 1000$ samples

(b) $n = 100$ samples

Figure 8: Structural Hamming Distance (SHD) of MIQP estimates with ℓ_1 regularization.

Therefore, to enhance its performance, we modified the MATLAB code for A*-lasso in order to incorporate the moral graph structure, when available, by only considering the edges present in the moral graph.

Since our focus here is on the computational performance of these approaches, we consider a single value of tuning parameter, $\lambda = 0.1$. Table 2 shows the solution times (in seconds) for A*-lasso versus the LN formulation for complete and moral instances. For the LN formulation, if the algorithm cannot prove optimality within the 6-hour time limit, we stop the algorithm and report, in parentheses, the optimality gap at termination. If the A*-lasso algorithm reaches the time limit before termination, then neither a feasible solution nor an optimality gap is available. We indicate this case with an "N/A" in Table 2. In complete instances, the results highlight that for small graphs A*-algorithm and LN are competitive, whereas the LN formulation outperforms A*-lasso for larger graphs. In particular, we see that the LN formulation attains the optimal solution for the Cloud and Galaxy data sets in less than one second and it obtains a feasible solution that is provably within 96.9% and 0.99% of the optimal objective value for Insurance and Factors data sets, respectively. In moral instances, we observe significant improvement in the computational performance of the LN formulation compared with the complete instances, whereas the improvement in the performance of A*-lasso is marginal in comparison; these results suggest that A*-lasso may not be able to utilize the super-structure knowledge as effectively as the LN formulation. For instance, LN's computational time for the Insurance data reduces from more than 6 hours to less than 13 seconds when the moral graph is provided. Both algorithms attain optimal solutions for the first six data sets, whereas A* does not attain a feasible solution for the last four data sets; as a result, we cannot obtain a DAG from A* solution for these instances.

For DAG learning from discrete data, an IP-based model, see e.g., Jaakkola et al. (2010), outperforms A* algorithms when a cardinality constraint on the number of the parent set for each node is imposed; A* tends to perform better if such constraints are not enforced. This is mainly because an IP-based model for discrete data requires an exponential number of variables which becomes cumbersome if such cardinality constraints are not permitted. In contrast, for DAG learning from continuous data with linear SEMs, our results show that an IP-based approach does not suffer from such a limitation because variables are encoded in the space of arcs (instead of parent sets). That is why LN performs well, even in complete instances (i.e., no restriction on the

Table 2: Computational performance of LN versus A*-algorithm with ℓ_1 regularization for $\lambda = 0.1$

			Moral		Complete	
			Time in seconds (Gap)		Time in seconds (Gap)	
Datasets)	m	$\lvert\mathcal{M}\rvert$	A*-lasso	LN	A*-lasso	LN
dsep	6	16	0.455	0.025	0.429	0.108
Asia	8	40	0.195	0.071	0.191	0.319
Bowling	9	36	0.417	0.225	0.489	0.291
Insurancesmall	15	76	2.694	1.135	3.048	0.531
Rain	14	70	51.737	0.632	69.404	3.502
Cloud	16	58	1066.08	0.426	2230.035	7.249
Funnel	18	62	N/A	0.395	N/A	3.478
Galaxy	20	76	N/A	0.740	N/A	9.615
Insurance	27	168	N/A	12.120	N/A	(.03)
Factors	27	310	N/A	55.961	N/A	(.01)

cardinality of parent set).

There are several fundamental advantages of IP-based modeling, particularly the LN formulation, compared to A*-lasso: (i) The variables in IP-based models (i.e., TO, LN, and CP) depend on the super-structure. Therefore, these IP-based models can effectively utilize the prior knowledge to reduce the search space, whereas A*-lasso cannot utilize the super-structure information as effectively. This is particularly important for the LN formulation, as the number of variables and constraints depend only on the super-structure; (ii) IP-based models can incorporate both ℓ_0 and ℓ_1 regularizations, whereas A*-lasso can solve the problem only with ℓ_1 regularization; (iii) IP-based methods in general enjoy the versatility to incorporate a wide variety of structural constraints, whereas A*-lasso and dynamic programming approaches cannot accommodate many structural assumptions. For instance, a modeler may prefer restricting the number of arcs in the DAG, this is achievable by imposing a constraint on an IP-based model, whereas one cannot impose such structural knowledge on A*-lasso algorithm; (iv) A*-lasso is based on dynamic programming; therefore, one cannot abort the search with the aim of achieving a feasible solution. In contrast, one can impose a time limit or an optimality gap tolerance to stop the search process in a branch-and-bound tree. The output is then a feasible solution to the problem, which provides an upper bound as well as a lower bound that guarantees the quality of the feasible solution; (v) algorithmic advances in integer optimization alone (such as faster continuous relaxation solution, heuristics for better upper bounds, and cutting planes for better lower bounds) have resulted in 29,000 factor speedup in solving IPs (Bertsimas et al. 2016) using a branch-and-bound process. Many of these advances have been implemented in powerful state-of-the-art optimization solvers (e.g., Gurobi), but they cannot be used in dynamic programming methods, such as A*-lasso.

Figure 9 illustrates the progress of upper bound versus lower bound in the branch-and-bound process for the Insurance dataset and highlights an important practical implication of an IP-based model: such models often attain high quality upper bounds (i.e., feasible solutions) in a short amount of time; the rest of the time is spent to close the optimality gap by increasing the lower bound.

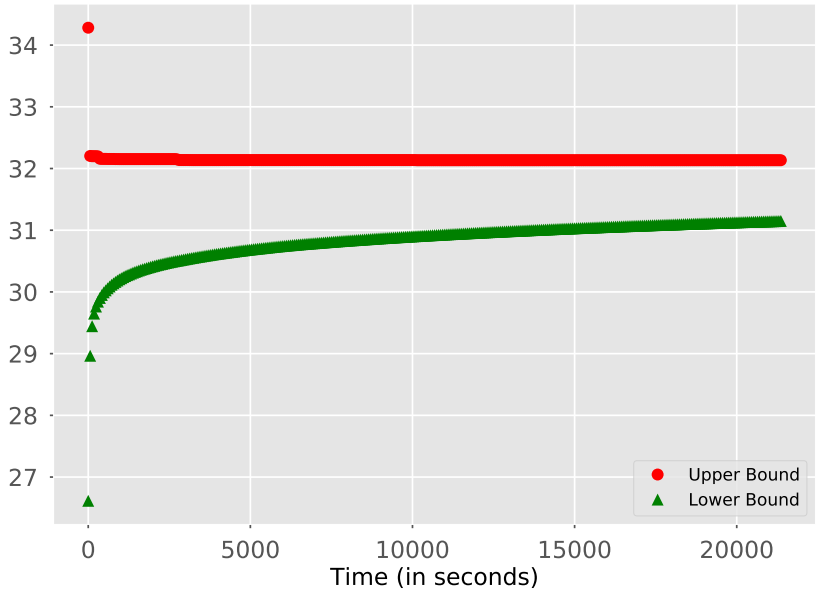


Figure 9: The progress of upper bound versus lower bound in the branch-and-bound tree for the LN formulation with $\lambda = 0.1$ for complete super-structure $\mathcal{M}$, for the Insurance dataset.

7 Conclusion

In this paper, we study the problem of learning an optimal DAG from continuous observational data using a score function, where the causal effect among the random variables is linear. We cast the problem as a mathematical program and use a negative log-likelihood score function with both ℓ_0 and ℓ_1 penalties. The mathematical programming framework can naturally incorporate a wide range of structural assumptions. For instance, it can incorporate a super-structure (e.g., skeleton or moral graph) in the form of an undirected and possibly cyclic graph. Such super-structures can be estimated from observational data. We review three mathematical formulations: cutting plane (CP), topological ordering (TO), and Linear Ordering (LO), and propose a new mixed-integer quadratic program (MIQP), referred to as the layered network (LN) formulation. We establish that the continuous relaxations of all models attain the same optimal objective function value under a mild condition. Nonetheless, the LN formulation is a compact formulation in contrast to CP, its relaxation can be solved much more efficiently compared to LO, and enjoys a fewer number of binary variables and traces a fewer number of branch-and-bound nodes than TO.

Our numerical experiments indicate that these advantages result in considerable improvement in the performance of LN compared to other MIQP formulations (CP, LO, and TO). These improvements are particularly pronounced when a sparse super-structure is available, because LN is the only formulation in which the number of constraints and binary variables solely depend on the super-structure. Our numerical experiments also demonstrate that the LN formulation has a number of advantages over the A*-lasso algorithm, especially when a sparse super-structure is available.

At least two future research avenues are worth exploring. First, one of the difficulties of estimating DAGs using mathematical programming techniques is the constraints in (6b). The big-M constraint is often very loose, which makes the convergence of branch-and-bound process slow. It is of interest to study these constraints in order to improve the lower bounds obtained from continuous relaxations. Second, in many real-world applications, the underlying DAG has special structures. For instance, the true DAG may be a polytree (Dasgupta 1999). Another example is a hierarchical structure. In that case, it is natural to learn a DAG that satisfies the hierarchy among different groups of random variables. This problem has important applications in discovering the genetic basis of complex diseases (e.g., asthma, diabetes, atherosclerosis).

Acknowledgments

We thank the two anonymous referees for their comments that improved the computational study. Simge Küçükyavuz is supported, in part, by ONR grant N00014-19-1-2321. Hao-Hsiang Wu is supported, in part, by MOST Taiwan grant 109-2222-E-009-005-MY2. Ali Shojaie is supported by NSF grant DMS-1561814 and NIH grant R01GM114029.

References

Aragam B, Zhou Q (2015) Concave penalized estimation of sparse Gaussian Bayesian networks. *Journal of Machine Learning Research* 16:2273–2328.

Bartlett M, Cussens J (2013) Advances in Bayesian network learning using integer programming. *arXiv preprint arXiv:1309.6825* .

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

Bektaş T, Gouveia L (2014) Requiem for the Miller–Tucker–Zemlin subtour elimination constraints? *European Journal of Operational Research* 236(3):820–832.

Bertsimas D, King A, Mazumder R (2016) Best subset selection via a modern optimization lens. *The Annals of Statistics* 44(2):813–852.

Chen W, Drton M, Wang YS (2019) On causal discovery with an equal-variance assumption. *Biometrika* 106(4):973–980.

Chickering DM (1996) Learning Bayesian networks is NP-complete. *Learning from data*, 121–130 (Springer).

Chickering DM (2002) Optimal structure identification with greedy search. *Journal of machine learning research* 3(Nov):507–554.

Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1997) *Combinatorial Optimization* (Wiley).

Correia AH, Cussens J, de Campos CP (2019) On pruning for score-based Bayesian network structure learning. *arXiv preprint arXiv:1905.09943* .

Cussens J (2010) Maximum likelihood pedigree reconstruction using integer programming. *WCB@ ICLP*, 8–19.

Cussens J (2012) Bayesian network learning with cutting planes. *arXiv preprint arXiv:1202.3713* .

Cussens J, Haws D, Studený M (2017a) Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming* 164(1-2):285–324.

Cussens J, Järvisalo M, Korhonen JH, Bartlett M (2017b) Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *J. Artif. Intell. Res.(JAIR)* 58:185–229.

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410.

Dasgupta S (1999) Learning polytrees. *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 134–141 (Morgan Kaufmann Publishers Inc.).

Desrochers M, Laporte G (1991) Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters* 10(1):27–36.

Drton M, Maathuis MH (2017) Structure learning in graphical modeling. *Annual Review of Statistics and Its Application* 4:365–393.

Eaton D, Murphy K (2007) Exact Bayesian structure learning from uncertain interventions. *Artificial Intelligence and Statistics*, 107–114.

Fu F, Zhou Q (2013) Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent. *Journal of the American Statistical Association* 108(501):288–300.

Grötschel M, Jünger M, Reinelt G (1985) On the acyclic subgraph polytope. *Mathematical Programming* 33(1):28–42.

Han SW, Chen G, Cheon MS, Zhong H (2016) Estimation of directed acyclic graphs through two-stage adaptive lasso for gene network inference. *Journal of the American Statistical Association* 111(515):1004–1019.

Healy P, Nikolov NS (2002) A branch-and-cut approach to the directed acyclic graph layering problem. *International Symposium on Graph Drawing*, 98–109 (Springer).

Hemmecke R, Lindner S, Studenỳ M (2012) Characteristic imsets for learning Bayesian network structure. *International Journal of Approximate Reasoning* 53(9):1336–1349.

Jaakkola T, Sontag D, Globerson A, Meila M (2010) Learning Bayesian network structure using LP relaxations. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 358–365.

Kalisch M, Bühlmann P (2007) Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research* 8(Mar):613–636.

Koivisto M (2012) Advances in exact Bayesian structure discovery in Bayesian networks. *arXiv preprint arXiv:1206.6828* .

Koivisto M, Sood K (2004) Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research* 5(May):549–573.

Koller D, Friedman N (2009) *Probabilistic graphical models: principles and techniques* (MIT press).

Kuipers J, Suter P, Moffa G (2018) Efficient structure learning and sampling of bayesian networks. *arXiv preprint arXiv:1803.07859* .

Lauritzen SL (1996) *Graphical Models* (Oxford University Press), ISBN 0-19-852219-3.

Loh PL, Bühlmann P (2014) High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research* 15(1):3065–3105.

Malone B, Kangas K, Järvisalo M, Koivisto M, Myllymäki P (2014) Predicting the hardness of learning Bayesian networks. *AAAI*, 2460–2466.

Markowetz F, Spang R (2007) Inferring cellular networks–a review. *BMC Bioinformatics* 8(6):S5.

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. *Journal of the ACM* 7(4):326–329.

Nemhauser GL, Wolsey LA (1988) Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin* 20:8–12.

Oates CJ, Smith JQ, Mukherjee S (2016a) Estimating causal structure using conditional dag models. *Journal of Machine Learning Research* 17(54):1–23, URL http://jmlr.org/papers/v17/14-479.html.

Oates CJ, Smith JQ, Mukherjee S, Cussens J (2016b) Exact estimation of multiple directed acyclic graphs. *Statistics and Computing* 26(4):797–811.

Öncan T, Altınel İK, Laporte G (2009) A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research* 36(3):637–654.

Padberg M, Sung TY (1991) An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming* 52(1-3):315–357.

Park YW, Klabjan D (2017) Bayesian network learning via topological order. *Journal of Machine Learning Research* 18(99):1–32, URL http://jmlr.org/papers/v18/17-033.html.

Parviainen P, Koivisto M (2009) Exact structure discovery in Bayesian networks with less space. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 436–443 (AUAI Press).

Pataki G (2003) Teaching integer programming formulations using the traveling salesman problem. *SIAM Review* 45(1):116–123.

Pearl J, et al. (2009) Causal inference in statistics: An overview. *Statistics surveys* 3:96–146.

Perrier E, Imoto S, Miyano S (2008) Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research* 9(Oct):2251–2286.

Peters J, Bühlmann P (2013) Identifiability of Gaussian structural equation models with equal error variances. *Biometrika* 101(1):219–228.

Raskutti G, Uhler C (2013) Learning directed acyclic graphs based on sparsest permutations. *arXiv preprint arXiv:1307.0366* .

Sachs K, Perez O, Pe'er D, Lauffenburger DA, Nolan GP (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science* 308(5721):523–529.

Sawik T (2016) A note on the Miller-Tucker-Zemlin model for the asymmetric traveling salesman problem. *Bulletin of the Polish Academy of Sciences Technical Sciences* 64(3):517–520.

Shojaie A, Michailidis G (2010) Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. *Biometrika* 97(3):519–538.

Silander T, Myllymaki P (2012) A simple approach for finding the globally optimal Bayesian network structure. *arXiv preprint arXiv:1206.6875* .

Singh M, Valtorta M (1995) Construction of Bayesian network structures from data: a brief survey and an efficient algorithm. *International Journal of Approximate Reasoning* 12(2):111–131.

Sondhi A, Shojaie A (2019) The reduced pc-algorithm: Improved causal structure learning in large random networks. *Journal of Machine Learning Research* 20(164):1–31.

Spirtes P, Glymour CN, Scheines R (2000) *Causation, prediction, and search* (MIT press).

Studenỳ M, Haws DC (2013) On polyhedral approximations of polytopes for learning Bayesian networks. *Journal of Algebraic Statistics* 4(1).

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Tsamardinos I, Brown LE, Aliferis CF (2006) The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning* 65(1):31–78.

van de Geer S, Bühlmann P (2013) ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics* 41(2):536–567.

Xiang J, Kim S (2013) A* lasso for learning a sparse Bayesian network structure for continuous variables. *Advances in Neural Information Processing Systems*, 2418–2426.

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

Yuan C, Malone B, Wu X (2011) Learning optimal Bayesian networks using A* search. *IJCAI Proceedings*, volume 22(3), 2186–2191.

Zhang B, Gaiteri C, Bodea LG, Wang Z, McElwee J, Podtelezhnikov AA, Zhang C, Xie T, Tran L, Dobrin R, et al. (2013) Integrated systems approach identifies genetic nodes and networks in late-onset Alzheimer's disease. *Cell* 153(3):707–720.

Zheng X, Aragam B, Ravikumar P, Xing EP (2018) DAGs with NO TEARS: smooth optimization for structure learning. *arXiv preprint arXiv:1803.01422* .

Zheng X, Dan C, Aragam B, Ravikumar P, Xing EP (2019) Learning sparse nonparametric dags. *arXiv preprint arXiv:11909.13189* .

Zou H (2006) The adaptive lasso and its oracle properties. *Journal of the American Statistical Association* 101(476):1418–1429.

Appendix I: Proofs

PROOF OF PROPOSITION 1

Let $\hat{\beta}$ be an optimal solution for (5) with an optimal objective value $\hat{F}(\hat{\beta})$. Let $\hat{z}_{ij} = 1$ if $\hat{\beta}_{ij} \neq 0$ for $(i,j) \in \mathcal{M}$ and $\hat{z}_{ij} = 0$ otherwise. The vector $\hat{z}$ is used to encode all the arcs in an optimal DAG. Let us refer to the DAG structure corresponding to this optimal solution by $DAG(V, \hat{E}^{\rightarrow})$. Suppose that for some $(j,k) \in \mathcal{M}$, we have $(j,k), (k,j) \notin \hat{E}^{\rightarrow}$, i.e., $\hat{z}_{jk} + \hat{z}_{kj} = 0$ (because $\hat{\beta}_{jk} = \hat{\beta}_{kj} = 0$). In other words, some edges in the super-structure graph do not appear in the optimal DAG. To prove the proposition, we construct an alternative optimal solution, β, with a corresponding direction assignment z, that satisfies $z_{jk} + z_{kj} = 1$ for all pairs of $(j,k) \in \mathcal{M}$ and meets the following conditions: (i) this corresponding DAG (tournament) is cycle free (ii) this tournament has the same objective value, $\hat{F}(\hat{\beta})$, as an optimal DAG.

Select a pair of nodes, say $p, q \in \mathcal{M}, p \neq q$ from $DAG(V, \hat{E}^{\rightarrow})$ for which $\hat{z}_{pq} + \hat{z}_{qp} = 0$. If there is a directed path from p to q (respectively q to p), then we can add the following arc (p,q) (respectively, (q,p)). This arc does not create a cycle in the graph. If there is no directed path between p and q, we can add an arc in either direction. In all cases, set β value corresponding to the added arc to zero. We repeat this process for all pairs of nodes with $\hat{z}_{pq} + \hat{z}_{qp} = 0$. We can add such arcs without creating any cycle. This is because if we cannot add an arc in either direction, it implies that we should have a directed path from p to q and a directed path from q to p in graph $DAG(V, \hat{E}^{\rightarrow})$ which is a contradiction because it implies a directed cycle in an optimal DAG. Note that in each step, we maintain a DAG. Hence, by induction we conclude that condition (i) is satisfied. The pair of nodes can be chosen arbitrarily. Since in the constructed solution we set β for the added arcs as zero, the objective value does not change. This satisfies condition (ii), and completes the proof. □

PROOF OF PROPOSITION 2

First we prove that (9e) removes all cycles. Suppose, for contradiction, that a cycle of size $p \geq 2$ is available and represented by $(1, 2, \ldots, p, 1)$. This implies $z_{j+1,j} = 0$ and $z_{j,j+1} = 1$, $\forall j = \{1, \ldots, p-$

1}, and $z_{p,1} = 1, z_{1,p} = 0$. Then,

$$\begin{aligned}
1 &= z_{12} - mz_{21} \leq \psi_2 - \psi_1, \\
1 &= z_{23} - mz_{32} \leq \psi_3 - \psi_2, \\
&\vdots \\
1 &= z_{p-1,p} - mz_{p,p-1} \leq \psi_p - \psi_1, \\
1 &= z_{p,1} - mz_{1,p} \leq \psi_1 - \psi_p.
\end{aligned}$$

We sum the above inequalities and conclude $p \leq 0$, a contradiction.

To complete the proof, we also need to prove that any DAG is feasible for the LN formulation. To this end, we know that each DAG has a topological ordering. For all the existing arcs in the DAG, substitute $z_{jk} = 1$ and assign a topological ordering number to the variables $\psi_k, k \in V$ in LN. Then, the set of constraints in (9e) is always satisfied. □

PROOF OF PROPOSITION 3

The LO formulation is in w-space whereas LN is in (z, ψ)-space. Hence, we first construct a mapping between these decision variables.

Given a feasible solution w_{jk} for all $j, k \in V$, $j \neq k$ in the LO formulation, we define $z_{jk} = w_{jk}$, $(j,k) \in \overrightarrow{E}$ and $\psi_j = \sum_{\ell \in V \setminus \{j\}} w_{\ell j}$, $j \in V$. Let $\mathbf{1}(x \geq 0)$ be a function which takes value 1 if $x \geq 0$ and 0 otherwise. Given z_{jk} for all $(j,k) \in \overrightarrow{E}$ and ψ_j for $j \in V$ in the LN formulation, we map $w_{jk} = z_{jk}$ for all $(j,k) \in \overrightarrow{E}$ and $w_{jk} = \mathbf{1}(\psi_k - \psi_j + 1 \geq 0)$ for all $(j,k) \notin \overrightarrow{E}$. Note that w-space is defined for all pair of nodes whereas z-space is defined for the set of arcs in $\overrightarrow{E}$. In every correct mapping, we have $w_{jk} = z_{jk}$, $\forall (j,k) \in \overrightarrow{E}$.

Fixing j and k for each $(j,k) \in \overrightarrow{E}$ and summing the left hand-side of inequalities (7e) over $i \in V \setminus \{j,k\}$ we obtain

$$\begin{aligned}
&(m-2)w_{jk} + \sum_{i \in V \setminus \{k,j\}} w_{ki} + \sum_{i \in V \setminus \{j,k\}} w_{ij} \leq m-2 \\
\equiv\ &(m-2)w_{jk} + \sum_{i \in V \setminus \{k,j\}} w_{ki} + \sum_{i \in V \setminus \{j,k\}} (1 - w_{ji}) \leq m-2 \\
\equiv\ &mw_{jk} - 1 + \sum_{i \in V \setminus \{k\}} w_{ki} - \sum_{i \in V \setminus \{j\}} w_{ji} \leq m-2 \\
\equiv\ &mw_{jk} - 1 - \sum_{i \in V \setminus \{k\}} w_{ik} + \sum_{i \in V \setminus \{j\}} w_{ij} \leq m-2 \\
\equiv\ &mw_{jk} - m + 1 \leq \sum_{i \in V \setminus \{k\}} w_{ik} - \sum_{i \in V \setminus \{j\}} w_{ij},
\end{aligned}$$

where the equivalences follow from constraints (7d). Given our mapping, $z_{jk} = w_{jk}$ for all $(j,k) \in \overrightarrow{E}$ and $\psi_j = \sum_{\ell \in V \setminus \{j\}} w_{\ell j}$ for $j \in V$, the above set of constraints can be written as

$$z_{jk} - (m-1)z_{kj} \leq \psi_k - \psi_j \quad \forall\, (j,k) \in \overrightarrow{E},$$

which satisfies (9e) in the LN formulation. This implies $\mathcal{R}(LO) \subseteq \mathcal{R}(LN)$ for ℓ_1-regularization. For ℓ_0-regularization, we need to add that $g_{jk} \leq w_{jk} = z_{jk}, \forall (j,k) \in E^{\rightarrow}$. This implies $\mathcal{R}(LO) \subseteq \mathcal{R}(LN)$ for ℓ_0 regularization.

To show strict containment, we give a point that is feasible to the LN formulation that cannot be mapped to any feasible point in the LO formulation. Consider $m = 3$, $z_{13} = 1, z_{31} = 0, z_{32} = 0.5 + \epsilon, z_{23} = 0.5 - \epsilon, z_{12} = z_{21} = 0.5$, $\psi = (1, 2, 2)$, for $0 < \epsilon < \frac{1}{6}$, with an appropriate choice of β. It is easy to check that this is a feasible solution to the LN formulation. Because we must have $w_{ij} = z_{ij}, \forall (i,j) \in \overrightarrow{E}$, we have $w_{13} + w_{32} + w_{21} > 2$. Therefore, the corresponding point is infeasible to the LO formulation and this completes the proof.

Note that the given feasible point to the continuous relaxation of LN is not a feasible point for the continuous relaxation of the MTZ formulation for TSPs because we do not have the constraint that there is on incoming and one outgoing arc to each node. Hence the strength results do not immediately follow from similar results established for TSP.

□

PROOF OF PROPOSITION 4

This proof is for TO formulation when the parameter m on (8e) is replaced with $m - 1$.

In the TO formulation, define the term $\sum_{s \in V} so_{ks}$ as ψ_k and the term $\sum_{s \in V} so_{js}$ as ψ_j. Further, remove the set of constraints in (8f), (8g), and (8i). This implies that $\mathcal{R}(TO) \subseteq \mathcal{R}(LN)$. To see strict containment, consider the point described in the proof of Proposition 3, which is feasible to the LN formulation. For this point, there can be no feasible assignment of the decision matrix o such that $\psi_j = \sum_{s \in V} so_{js}$, hence $\mathcal{R}(TO) \subset \mathcal{R}(LN)$.

□

PROOF OF PROPOSITION 5

Let $\bar{F}(\beta_X^\star)$ denote the optimal objective value associated with the continuous relaxation of model $X \in \{CO, LO, LN\}$.

Part A. $\bar{F}(\beta_{LO}^\star) = \bar{F}(\beta_{LN}^\star)$.

Case 1. ℓ_1 regularization

Suppose $(\beta_{LN}^\star, z^\star)$ is an optimal solution associated with the continuous relaxation of the LN formulation (9a)-(9g) and $(\beta_{LO}^\star, w^\star)$ is an optimal solution associated with continuous relaxation of the LO formulation with ℓ_1-regularization.

Given Proposition 3, we conclude that $\bar{F}(\beta_{LN}^\star) \leq \bar{F}(\beta_{LO}^\star)$. We prove that $\bar{F}(\beta_{LN}^\star) \geq \bar{F}(\beta_{LO}^\star)$ also holds in an optimal solution. To this end, we map an optimal solution in continuous relaxation of the ℓ_1-LN formulation to a feasible solution in continuous relaxation of the ℓ_1-LO formulation with the same objective function. This implies $\bar{F}(\beta_{LN}^\star) \geq \bar{F}(\beta_{LO}^\star)$.

Given an optimal solution $(\beta_{LN}^\star, z^\star)$ to the continuous relaxation of the ℓ_1-LN formulation, we construct a feasible solution (β_{LO}, w) to the continuous relaxation of the LO formulation as

$$w_{jk} = w_{kj} = \begin{cases} \frac{1}{2}, & z_{jk}^\star > 0, \, (i,j) \in \overrightarrow{E}, \\ 0, & \text{otherwise}, \end{cases}$$

and let $\beta_{LO} = \beta_{LN}^*$.

We now show that this mapping is always valid for the ℓ_1-LO formulation. Recall that for the ℓ_1 regularization, we do not have the decision vector g in the formulation. Three set of constraints have to be satisfied in the ℓ_1-LO formulation.

$|\beta_{jk}| \leq M w_{jk}, \quad \forall (j,k) \in \overrightarrow{E}$ (7b)

$w_{jk} + w_{kj} = 1, \quad \forall (j,k) \in \overrightarrow{E}$ (7d)

$w_{ij} + w_{jk} + w_{ki} \leq 2, \quad \forall (i,j),(j,k),(k,i) \in \overrightarrow{E}\ i \neq j \neq k.$ (7e)

The set of constraints (7b) is trivially satisfied because we set $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$. The set of constraints (7d) is trivially satisfied. The set of constraints (7e) is satisfied because the left hand side of inequality can take at most $\frac{3}{2}$ given this mapping. Therefore, $\bar{F}(\beta^\star_{LO}) \leq \bar{F}(\beta_{LO}) = \bar{F}(\beta^\star_{LN})$. This completes this part of the proof.

Case 2. ℓ_0 regularization

Suppose $(\beta^\star_{LN}, g^\star_{LN}, z^\star)$ is an optimal solution associated with a continuous relaxation of the ℓ_0-LN formulation, and $(\beta^\star_{LO}, g^\star_{LO}, w^\star)$ is an optimal solution associated with a continuous relaxation of the ℓ_0-LO formulation.

Given Proposition 3, $\bar{F}(\beta^\star_{LN}, g^\star_{LN}) \leq \bar{F}(\beta^\star_{LO}, g^\star_{LO})$. We now prove that, in an optimal solution $\bar{F}(\beta^\star_{LN}, g^\star_{LN}) \geq \bar{F}(\beta^\star_{LO}, g^\star_{LO})$ also holds.

Given an optimal solution $(\beta^\star_{LN}, g^\star_{LN}, z^\star)$ for the continuous relaxation of the ℓ_0-LN formulation, we construct a feasible solution (β_{LO}, g_{LO}, w) for the continuous relaxation of the ℓ_0-LO formulation as

$$w_{jk} = w_{kj} = \begin{cases} \frac{1}{2}, & z^\star_{jk} > 0,\ (j,k) \in \overrightarrow{E}, \\ 0, & otherwise, \end{cases}$$

and let $\beta_{LO} = \beta^*_{LN}$ and $g_{LO} = g^\star_{LN}$.

We now show that this mapping is always valid for the LO formulation. Four sets of constraints have to be satisfied in the LO formulation.

$|\beta_{jk}| \leq M g_{jk}, \quad \forall (j,k) \in \overrightarrow{E}$, (7b)

$g_{jk} \leq w_{kj}, \quad \forall (j,k) \in \overrightarrow{E}$ (7c)

$w_{jk} + w_{kj} = 1, \quad \forall (j,k) \in \overrightarrow{E}$ (7d)

$w_{ij} + w_{jk} + w_{ki} \leq 2, \quad \forall i,j,k \in V, i \neq j \neq k,$ (7e)

The set of constraints in (7d) is satisfied similar to ℓ_1 case. The proof that constraints (7b), (7c), and (7e) are also met is more involved. If the ℓ_0-LN formulation attains a solution for which $g^\star_{ij} = g^\star_{jk} = g^\star_{ki} = 1$ (we dropped subscript LN), then our mapping leads to an infeasible solution to the ℓ_0-LO formulation, because it forces $w_{ij} + w_{jk} + w_{ki} \geq 2$ for ℓ_0-LO. Next we show that this will not be the case and that our mapping is valid. To this end, we show that in an optimal solution for the continuous relaxation of ℓ_0-LN, we always have $g^\star_{jk} \leq \frac{1}{2}, \forall (j,k) \in \overrightarrow{E}$. Note that in the LN formulation, we have $|\beta_{jk}| \leq M g_{jk}, \forall (j,k) \in E^{\rightarrow}$ and $g_{jk} \leq z_{jk}$ (we dropped the subscript LN). Suppose that $g^\star_{jk} \geq \frac{1}{2}$. In this case, the objective function forces $g^\star_{jk}$ to be at most $\frac{1}{2}$. Note that the objective function can reduce $g^\star_{jk}$ up to $\frac{1}{2}$ and decreases the regularization term without any increase on the loss function. This is because $\beta_{jk} \leq M g_{jk}, \forall (j,k) \in E^{\rightarrow}$ can be replaced by $\beta_{jk} \leq M\frac{1}{2}, \forall (j,k) \in E^{\rightarrow}$ without any restriction on β because $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$. Therefore, our mapping is valid and implies that $\bar{F}(\beta^\star_{LO}, g^\star_{LO}) \leq \bar{F}(\beta_{LO}, g_{LO}) = \bar{F}(\beta^\star_{LN}, g^\star_{LN})$.

Part B. $\bar{F}(\beta^\star_{TO}) = \bar{F}(\beta^\star_{LN})$.

Case 1. ℓ_1 regularization

Given Proposition 4, we conclude that $F(\beta^\star_{LN}) \leq F(\beta^\star_{TO})$. We now prove that $\bar{F}(\beta^\star_{LN}) \geq \bar{F}(\beta^\star_{TO})$ also holds in an optimal solution. We map an optimal solution in continuous relaxation of the ℓ_1-LN formulation to a feasible solution in continuous relaxation of the ℓ_1-TO formulation with the same objective value. This implies $\bar{F}(\beta^\star_{LN}) \geq \bar{F}(\beta^\star_{TO})$.

Next we construct a feasible solution (β_{TO}, z_{TO}, o) to the TO formulation. Given an optimal solution $(\beta^\star_{LN}, z^\star_{LN}, \psi^\star)$ for a continuous relaxation of the LN formulation, rank the $\psi^\star_j$ in non-descending order. Ties between ψ values can be broken arbitrarily in this ranking. Then, for each variable $j \in \{1, \ldots, m\}$, $o_{jr} = 1$ where r denotes the rank of ψ_j in a non-descending order (the first element is ranked 0). This mapping satisfies the two assignment constraints (8f)-(8g). Let $\beta_{TO} = \beta^*_{LN}$, $z_{TO} = z^*_{LN}$. This gives a feasible solution for the TO formulation. Thus, $\bar{F}(\beta^\star_{LN}) \geq \bar{F}(\beta^\star_{TO})$.

Case 2. ℓ_0 regularization

Define $g^\star_{LN} = g_{TO}$. The rest of the proof is similar to the previous proofs.

Part C. $\bar{F}(\beta^\star_{CP}) = \bar{F}(\beta^\star_{LN})$.

To prove this, we first prove the following Lemma.

Lemma. *The LO formulation is at least as strong as the CP formulation, that is* $\mathcal{R}(LO) \subseteq \mathcal{R}(CP)$.

Proof. Consider (7d)-(7e) in the linear ordering formulation given by

$$\begin{aligned} &w_{jk} + w_{kj} = 1 \quad \forall\ (j,k) \in \overrightarrow{E}, \\ &w_{ij} + w_{jk} + w_{ki} \leq 2 \quad \forall (i,j),(j,k),(k,i) \in \overrightarrow{E}. \end{aligned}$$

Consider the set of constraints in CP given by (6c) as

$$\sum_{(j,k) \in \mathcal{C}_A} g_{jk} \leq |\mathcal{C}_A| - 1 \quad \forall \mathcal{C}_A \in \mathcal{C},$$

Consider the same mapping as in Proposition 3. Select an arbitrary constraint from (6c). Without loss of generality, consider $g_{1,2} + g_{2,3} + \cdots + g_{p-1,p} + g_{p,1} \leq p - 1$. We arrange the terms in (7d)-(7e) as

$$\begin{aligned} &w_{1,2} + w_{2,3} + w_{3,1} \leq 2 \\ &w_{1,3} + w_{3,4} + w_{4,1} \leq 2 \\ &w_{1,4} + w_{4,5} + w_{5,1} \leq 2 \\ &\vdots \\ &w_{1,p-2} + w_{p-2,p-1} + w_{p-1,1} \leq 2 \\ &w_{1,p-1} + w_{p-1,p} + w_{p,1} \leq 2 \end{aligned}$$

Summing the above set of inequalities and substituting $w_{ij} = 1 - w_{ji}$, when appropriate, gives $w_{1,2} + w_{2,3} + w_{3,4} + \cdots + w_{p,1} \leq p - 1$. Thus, we conclude that $\mathcal{R}(LO) \subseteq \mathcal{R}(CP)$. □

Given this lemma and Proposition 3, we conclude that $\bar{F}(\beta^{\star}_{LN}) \geq \bar{F}(\beta^{\star}_{TO})$. □

PROOF OF PROPOSITION 6

This is a generalization of Proposition 5. Suppose we have branched on variable w_{jk} in the LO formulation or correspondingly on variable z_{jk} in the LN formulation. If $w_{jk} = 0$, then $\beta_{jk} = 0$. In this case, it is as if we now need to solve the original model with $(j,k) \notin \overrightarrow{E}$. Thus, Proposition 5 (Part A) implies that both models attain the same continuous relaxation. On the other hand, if $w_{jk} = 1$ in LO (or correspondingly $z_{jk} = 1$ in LN), we define our mapping as

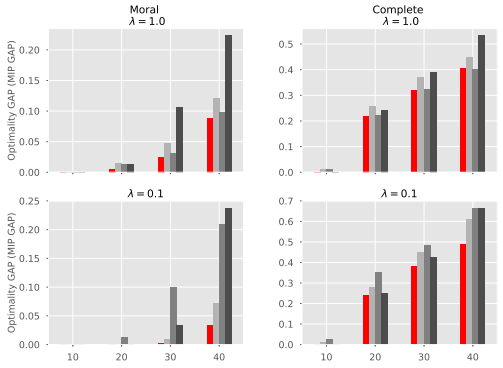
$$w_{jk} = w_{kj} = \begin{cases} \frac{1}{2}, & z^{\star}_{jk} > 0,\ (j,k) \in \overrightarrow{E}, \\ 1, & (j,k) \in \mathcal{B} \\ 0, & \text{otherwise}, \end{cases}$$

where $\mathcal{B}$ is the set of (j,k) for which $z_{jk} = 1$. The rest of the proof follows from Proposition 5 (Part A).

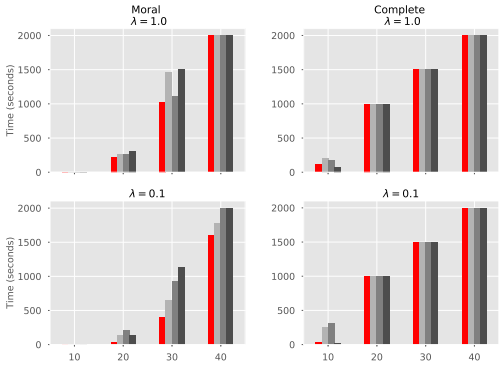
The proofs for the TO and CP formulations are almost identical with Proposition 5. Suppose we have branched on variable z_{jk} on any of the models (CP, LO, TO). If $z_{jk} = 0$, then $\beta_{jk} = 0$. In this case, it is as if we now need to solve the original model with $(j,k) \notin \overrightarrow{E}$. If $z_{jk} = 1$, one defines the mapping in Proposition 5. The proof follows from Proposition 5 parts B and C, respectively. □

Appendix II: Numerical Results for $n = 100$

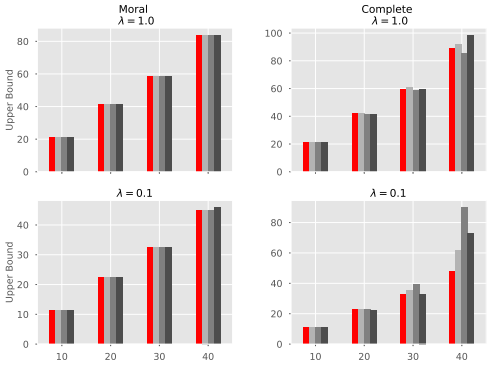
In this section, we report the numerical results for $n = 100$ samples. In particular, Figures 10 and 12 summarize various optimization criteria for ℓ_0 and ℓ_1 formulations when $n = 100$, while Figures 11 and 13 summarize the graph recovery measures for these formulations.



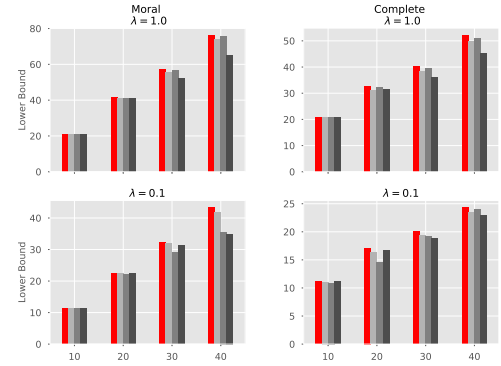
(a) Optimality GAPs for MIQPs



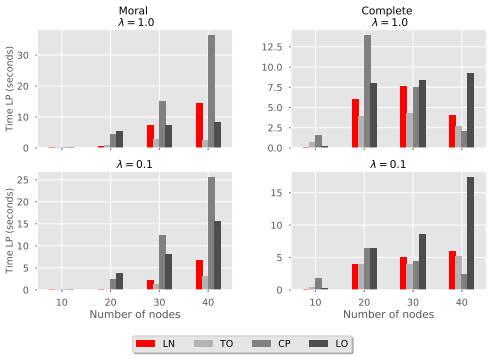
(b) Time (in seconds) for MIQPs



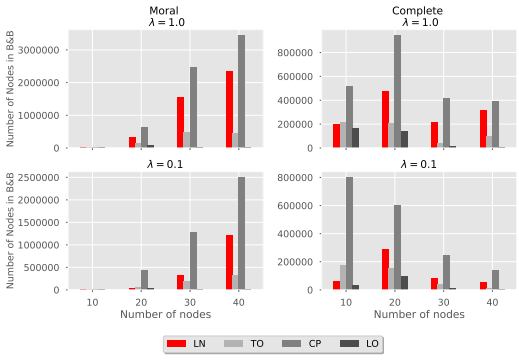
(c) Best upper bounds for MIQPs



(d) Best lower bounds for MIQPs

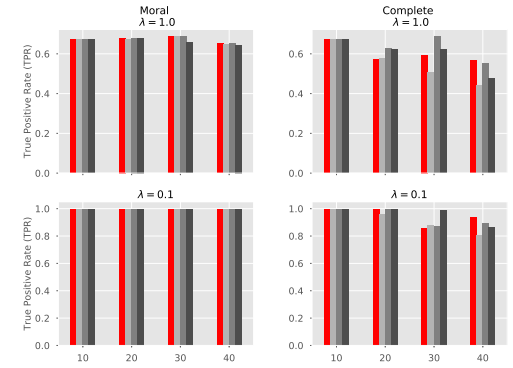


(e) Time (in seconds) for continuous root relaxation

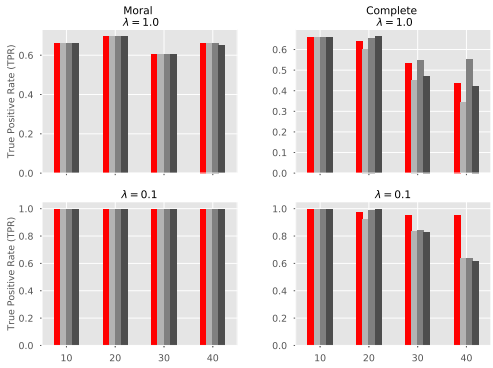


(f) Number of explored nodes in B&B tree

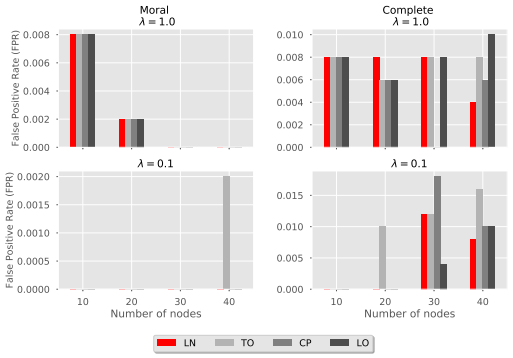
Figure 10: Optimization-based measures for MIQPs for ℓ_0 model with number of samples $n = 100$.



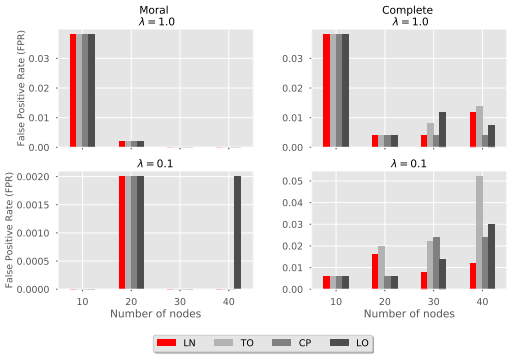
(a) True Positive Rate (TPR) for MIQPs



(b) True Positive Rate(TPR) for MIQPs

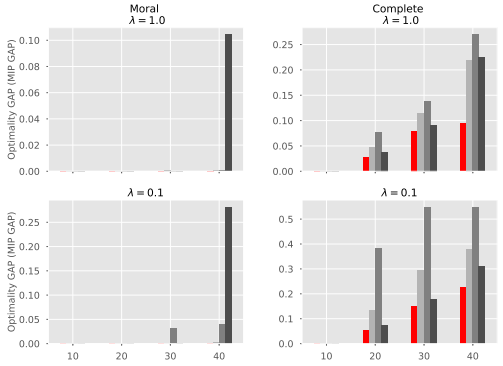


(c) False Positive Rate(FPR) for MIQPs

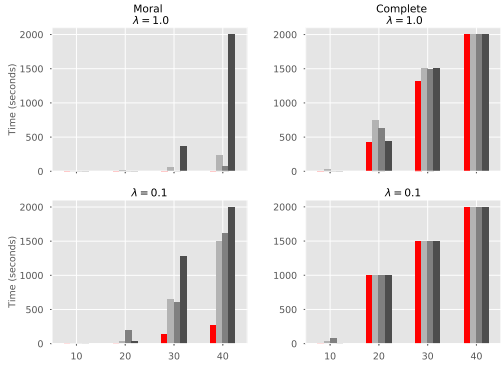


(d) False Positive Rate(FPR) for MIQPs

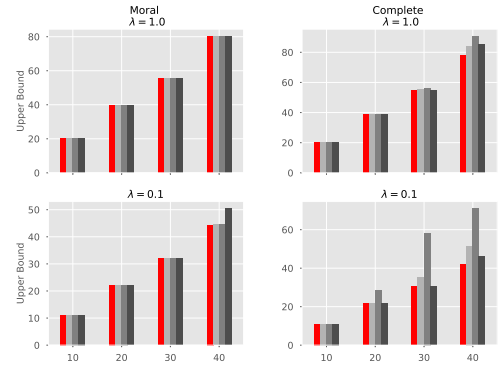
Figure 11: Graph metrics for the MIQPs for ℓ_0 regularization. Plots a, c (left) show graph metrics with the number of samples $n = 1000$ and plots b, d (right) show the graph metrics with the number of samples $n = 100$.



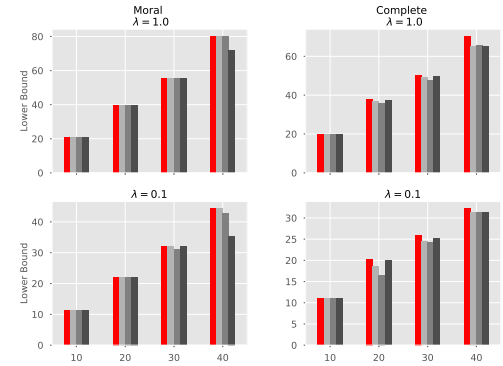
(a) Optimality GAPs for MIQPs



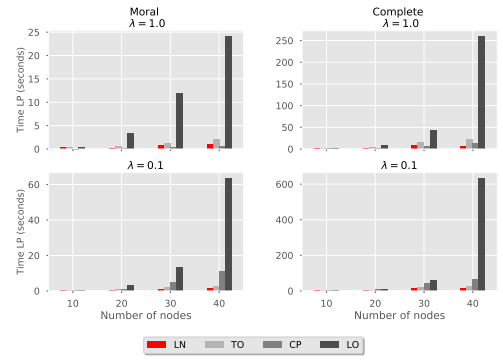
(b) Time (in seconds) for MIQPs



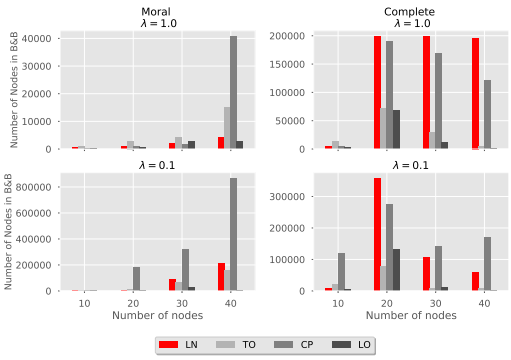
(c) Best obtained upper bounds for MIQPs



(d) Best obtained lower bounds for MIQPs

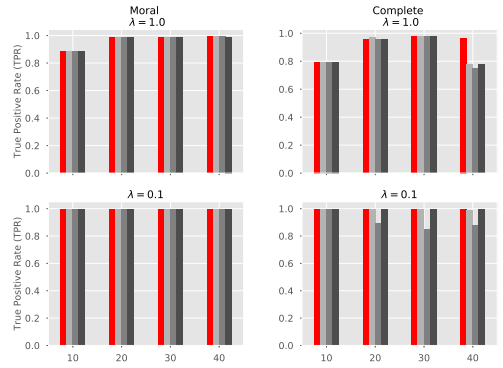


(e) Time (in seconds) for continuous root relaxation

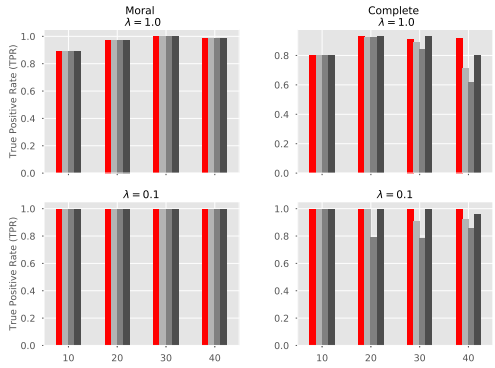


(f) Number of explored nodes in B&B tree

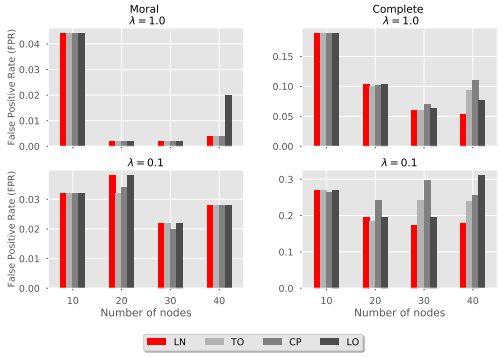
Figure 12: Optimization-based measures for MIQPs for ℓ_1 model with the number of samples $n = 100$.



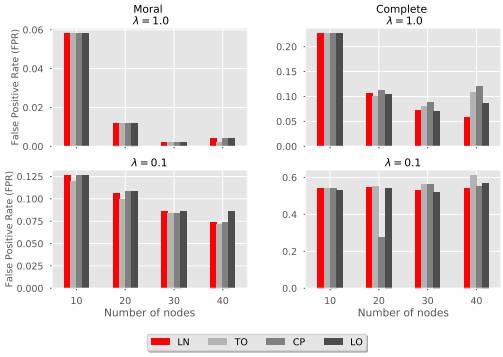
(a) True Positive Rate (TPR) for MIQPs



(b) True Positive Rate (TPR) for MIQPs



(c) False Positive Rate (FPR) for MIQPs



(d) False Positive Rate (FPR) for MIQPs

Figure 13: Graph metrics for MIQPs for ℓ_1 regularization. Plots a, c (left) show graph metrics with the number of samples $n = 1000$. Plots b, d (right) show the graph metrics with the number of samples $n = 100$.

Appendix III: Results for Synthetic Barabási-Albert Graphs

In this section, we report the numerical results comparing the four MIQP formulations (LN, CP, LO and TO) on additional datasets, namely on Barabási-Albert random graphs. These additional experiments are performed on a Windows 10 operating system with an Intel Core i7-8750H 2.2 GHz CPU, 8 GB DRAM using Python 3.6 with Gurobi 9.0.1 Optimizer. We use the `R` package `pcalg` to generate random Barabási-Albert graphs and follow the data generation scheme of Section 5 to set all parameters.

In Table 3, we compare the performance of four formulations for graphs with m nodes, using $n = 100$ or 1000 samples and $\lambda = 0.1$ or 1. We observe that, in general, the instances are easier to solve for larger λ. Moreover, the solution time is not affected significantly by the sample size n. This is because the size of the formulations, in terms of variables and constraints, does not depend on n. We also note that LN is able to solve more instances to optimality within the time limit, and when it stops early due to the time limit, it finds a solution with smaller optimality gap than other approaches. CP is competitive with LN in moral instances; however, the performance of CP suffers in complete instances. Both LO and TO are dominated by LN for non-trivial instances. Therefore, we conclude that the superior performance of the LN formulation persists in the Barabási-Albert graphs we consider in this study.

Appendix IV: Computations with real datasets

In this section, we compare the performance of the MIQP formulations on publicly available datasets. As in Appendix III, these experiments are performed on a Windows 10 operating system with an Intel Core i7-8750H 2.2 GHz CPU, 8 GB DRAM using Python 3.6 with Gurobi 9.0.1 Optimizer.

First, we consider the DAG structures used in Xiang and Kim (2013). The strength of arcs for these DAGs (β) are chosen from $\mathcal{U}[-1,-0.1]\cup\mathcal{U}[0.1,1]$. The number of nodes in the 10 true DAGs varies from $m = 6$ to $m = 27$ (see Table 2). The true DAG and the corresponding β coefficients are used to generate $n = 500$ samples for each column of data matrix $\mathcal{X}$.

The results are summarized in Table 4. In these experiments, we continue to observe that the LN formulation performs the best in all cases: It is able to solve almost all instances to optimality (except for the Insurance and Factors datasets with a complete super-structure graph). In contrast, the alternative formulations either require longer solution times, or they are unable to prove optimality and terminate with a positive optimality gap. Also the number of branch-and-bound nodes explored is generally smaller for the LN formulation, especially when the competing formulations are not able to prove optimality within the time limit. The results also highlight the importance of using the super-structure information, which drastically reduces the solution time for all formulations.

In our final experiment, we evaluate the performance of the proposed LN formulation using a real immunology dataset from Sachs et al. (2005). For this data, the "true graph" is available and can be used to evaluate the quality of the estimated DAGs, using e.g., the structural Hamming distance (SHD). The dataset is thus commonly used as a benchmark for probabilistic graphical models.

Given the advantage of LN over other MIQP formulations in the first experiments, for this analysis we compare LN with a state-of-the-art heuristic approach, namely the NOTEARS method of Zheng et al. (2018), as well as the DAG-MLP method of Zheng et al. (2019). Moreover, given

Table 3: Computational performance of four models with ℓ_1 regularization for Barabási-Albert graphs. Column 'Time(u)', reports the solution time, in seconds, for instances that solve within the time limit of 1800 seconds. If not all three replications solve with in the time limit, the number of unsolved instances are reported in parentheses (u). In the case of unsolved instances, the optimality gap at early termination is reported in the 'Gap' column; otherwise, a dash is used to indicate that all instances solved to optimality. Finally the 'Node' column reports the number of branch-and-bound nodes explored.

				LN			CP			LO			TO		
Type	m	n	λ	Time(u)	Node	Gap	Time(u)	Node	Gap	Time(u)	Node	Gap	Time(u)	Node	Gap
Complete	10	100	0.1	≤ 1	3582	-	2	27562	-	≤ 1	1320	-	3	7892	-
			1	≤ 1	682	-	≤ 1	1258	-	≤ 1	375	-	≤ 1	2768	-
		1000	0.1	≤ 1	2132	-	≤ 1	13874	-	≤ 1	713		≤ 1	3075	-
			1	≤ 1	1234	-	≤ 1	1770	-	≤ 1	753	-	≤ 1	2144	-
	20	100	0.1	64	153386	-	(3)	1757227	0.21	200	201386	-	(3)	1053144	-
			1	7	18841	-	50	257779	-	13	9152	-	26(1)	236035	0.01
		1000	0.1	8	22151	-	(3)	4599235	0.15	30	22104	-	712(1)	368691	0.02
			1	2	5101	-	19	97833	-	7	4254	-	115	53389	-
	30	100	0.1	(3)	870570	0.08	(3)	790817	0.51	(3)	121471	0.11	(3)	223021	0.14
			1	27	19335	-	44(1)	310485	0.03	205	14944	-	255(1)	97673	0.02
		1000	0.1	591(1)	639781	0.05	(3)	636426	0.41	1261(2)	163892	0.07	(3)	347455	0.08
			1	14	12506	-	266	200700	-	180	15569	-	299(1)	105648	0.02
	40	100	0.1	(3)	215322	0.20	(3)	472420	0.55	(3)	14536	0.22	(3)	76940	0.27
			1	377(2)	353422	0.03	(3)	604940	0.11	(3)	17401	0.05	(3)	52701	0.07
		1000	0.1	(3)	506888	0.10	(3)	487558	0.54	(3)	14583	0.17	(3)	60748	0.20
			1	561(1)	260258	0.02	(3)	636698	0.07	1770(2)	21799	0.04	(3)	47215	0.04
Moral	10	100	0.1	≤ 1	108	-	≤ 1	79	-	≤ 1	187	-	≤ 1	260	-
			1	≤ 1	51	-	≤ 1	66	-	≤ 1	47	-	≤ 1	145	-
		1000	0.1	≤ 1	72	-	≤ 1	70	-	≤ 1	156	-	≤ 1	189	-
			1	≤ 1	67	-	≤ 1	59	-	≤ 1	65	-	≤ 1	134	-
	20	100	0.1	≤ 1	1159	-	≤ 1	1535	-	3	3046	-	≤ 1	1623	-
			1	≤ 1	399	-	≤ 1	285	-	≤ 1	618	-	≤ 1	739	-
		1000	0.1	≤ 1	1250	-	≤ 1	1287	-	4	3922	-	≤ 1	1791	-
			1	≤ 1	334	-	≤ 1	246	-	≤ 1	439	-	≤ 1	747	-
	30	100	0.1	2	4948	-	≤ 1	6227	-	76	8348	-	7	7872	-
			1	≤ 1	320	-	≤ 1	320	-	8	910	-	2	1574	-
		1000	0.1	2	7545	-	≤ 1	7015	-	100	12525	-	6	7756	-
			1	≤ 1	543	-	≤ 1	503	-	13	1741	-	2	1030	-
	40	100	0.1	5	16864	-	24	297219	-	851(2)	17843	0.07	30	29158	-
			1	≤ 1	1384	-	≤ 1	918	-	655	2416	-	5	1977	-
		1000	0.1	5	21625	-	3	30293	-	1070(1)	27236	0.06	57	37978	-
			1	≤ 1	1090	-	≤ 1	812	-	419	2568	-	6	2225	-

the clear benefits of sparse super-structures, we use the available data to estimate the moral graph before learning the DAG itself.

Using the ℓ_0 regularization with $\lambda = 0.1$ and $\lambda = 1$, the DAG estimated by the LN formulation has SHD scores of 9 and 18, respectively, compared to the true graph. These SHD scores are very similar to those obtained using the ℓ_1 regularization with the same tuning parameters: 10 and 18, respectively for $\lambda = 0.1$ and $\lambda = 1$. In all cases, we are able to find an optimal solution within two seconds for this dataset.

Comparing the above SHD scores with the SHD scores for NOTEARS (22) and DAG-MLP (16) suggests that when complemented with a sparse super-structure, the LN formulation can significantly improve the quality of DAG estimation. In contrast, when using a complete super-structure, we obtain SHD scores of 34 and 46 for ℓ_0 and ℓ_1 regularization, respectively, for either λ values. This observation corroborates our previous findings and further highlights the potential benefits of the LN formulation, which can take full advantage of the sparse super-structure information.

Table 4: Computational performance of four MIQP models in DAGs from publicly available datasets considered in Xiang and Kim (2013). All results are obtained using ℓ_1 regularization with $\lambda = 0.1$. Column 'Time (Gap)', shows the solution time, in seconds, if the solution is found within the time limit of 1800 seconds; otherwise, the optimality gap at termination is shown in parentheses. Column 'Nodes' reports the number of branch-and-bound nodes explored.

		LN		CP		LO		TO	
Type	Dataset	Time (Gap)	Nodes	Time (Gap	Nodes	Time (Gap)	Nodes	Time (Gap)	Nodes
Complete	Dsep	≤ 1	192	≤ 1	223	≤ 1	77	≤ 1	326
	Asia	≤ 1	471	≤ 1	1886	≤ 1	437	≤ 1	2790
	Bowling	≤ 1	1111	≤ 1	5453	≤ 1	534	≤ 1	2378
	InsuranceSmall	12	59984	(0.24)	8261369	14	42838	(0.01)	1638607
	Rain	4	17581	(0.26)	10216440	6	22650	140	187871
	Cloud	20	98587	(0.07)	6469596	25	72945	(0.05)	1741647
	Funnel	17	68885	(0.19)	4769778	29	45933	(0.01)	1883745
	Galaxy	232	602837	(0.28)	2733622	895	974323	(0.08)	1367859
	Insurance	(0.23)	1634182	(0.89)	533270	(0.20)	254867	(0.22)	523575
	Factors	(0.12)	1470945	(0.81)	519826	(0.11)	252565	(0.13)	463644
Moral	Dsep	≤ 1	14	≤ 1	16	≤ 1	14	≤ 1	40
	Asia	≤ 1	46	≤ 1	55	≤ 1	107	≤ 1	62
	Bowling	≤ 1	158	≤ 1	460	≤ 1	141	≤ 1	246
	InsuranceSmall	≤ 1	2097	17	5516	≤ 1	1881	3	7294
	Rain	≤ 1	1045	≤ 1	1657	≤ 1	750	≤ 1	1797
	Cloud	≤ 1	750	≤ 1	551	≤ 1	400	≤ 1	2282
	Funnel	≤ 1	587	≤ 1	584	≤ 1	1327	≤ 1	1318
	Galaxy	≤ 1	1418	≤ 1	2117	2	5033	≤ 1	2557
	Insurance	5	20486	194	2243034	116	31638	94	148391
	Factors	35	257338	612	6530445	792	223435	(0.04)	2536166