Mixed Integer Quadratic Optimization for Learning Directed Acyclic Graphs from Continuous Data

Hasan Manzour

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Simge Küçükyavuz, Chair

Ali Shojaie, Chair

Archis Ghate

Program Authorized to Offer Degree:
Industrial and Systems Engineering

University of Washington

Abstract

Mixed Integer Quadratic Optimization for Learning Directed Acyclic Graphs from Continuous Data

Hasan Manzour

Co-Chairs of the Supervisory Committee:
Simge Küçükyavuz

Ali Shojaie

The study of probabilistic graphical models (PGMs) is an essential topic in statistics and machine learning fields. Bayesian networks (BNs), arguably one of the most central classes of PGMs, is frequently used to represent causal relations among a set of random variables in complex systems. A Bayesian network (BN) is a PGM that consists of a labeled directed acyclic graph (DAG) in which the vertices in the vertex set correspond to random variables (nodes), and the edge set prescribe a decomposition of the joint probability distribution of nodes such that the value of any node is a probabilistic function of the values of the nodes which are its parents in the DAG. The edge set encodes Markov conditions on the nodes in the sense that each node is conditionally independent of its non-descendents given its parents. While statistical properties of BNs from continuous data have been extensively studied, the development of efficient computational tools for learning an optimal DAG remains an open challenge. The goal is to learn a DAG structure that maximizes a score function. One such score metric is the posteriori probability of the DAG structure given data. Learning DAGs from observational data is a computationally difficult task, because the number of possible DAGs scales superexponentially with the number of nodes. In this work, we propose novel discrete optimization formulations for learning DAGs for continuous variables.

Learning DAG from continuous data can be cast as a mixed-integer quadratic programming (MIQP) with the objective function as the penalized negative log-likelihood (PNL) and an ℓ_0 regularization penalty subject to linear constraints. There are two key challenges: (i) imposing a set of constraints to remove cycles from a directed graph, (ii) enforcing tight bounds on the semi-continuous optimization variables corresponding to the arc weights.

We tackle the first challenge by presenting a way to remove cycles which results in a new MIQP formulation with linear constraints, referred to as a *layered network* (LN). We establish that LN is a compact formulation and the objective value of its continuous relaxation is as tight as stronger but larger formulations under a mild condition. An additional benefit of the LN formulation is that it effectively incorporates a prior structural knowledge (super-structure) in order to reduce the set of possible candidate DAGs. 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, especially in the presence of a sparse super-structure.

To model semi-continuous variables, a common practice is to use a standard "big-M constraint" in a MIQP. This is commonly modeled using a standard "big-M constraint" in the associated mixed-integer program. However, this strategy leads to a poor continuous relaxation because there is no natural upper bound for the arc weights. To circumvent this deficiency, we present a *mixed-integer second-order cone program* (MISOCP), which has tighter continuous relaxation bounds than the existing formulations based on big-M constraints – including the LN formulation. We show the promising performance of the MISOCP in terms of reducing the optimality gap when compared to the best existing optimization formulations. The performance of each formulation depends on the size and tightness of its continuous relaxation. This work highlights that the best formulation applies LN constraints to remove cycles to keep the size of the optimization problem small while using conic constraints to tighten the semi-continuous variables.

The statistical properties of PNL with ℓ_0 regularization have been studied extensively in the context of DAG structural learning. However, it is often difficult to prove optimality of a solution from an optimization point of view, and there are no results which establish the statistical properties of an approximate solution. We give an early stopping criterion under which the branch-and-bound process can be terminated early. We establish that the obtained solution under the propsoed stopping creterion asymptotically converges to the true coefficients in DAG with high probability under proper conditions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I'd like to thank my advisers Professor. Simge Küçükyavuz and Professor Ali Shojaie for their guidance over the last four years. I'd like to thank my dissertation committee Professor Archis Ghate and Professor Michael Wagner for their thoughtful questions. I'd like to thank all my good friends who have been there for me through the sunshines and rains.

DEDICATION

To my parents

Chapter 1

INTRODUCTION

With the deluge of data, the field of statistical learning has gained rapid prominence and has grown in complexity by the problems that arise in science and industry. To cope with the ever increasing complexity of statistical learning problems, the development of new optimization models and efficient algorithms becomes imperative. Traditionally, continuous optimization has received a lot of attention from the machine learning and statistics communities. However, in recent years, a body of works has emerged highlighting the importance of discrete optimization in the field of statistical learning (see, e.g., [7, 8, 9]).

The study of Probabilistic Graphical Models (PGMs) is an essential topic in statistics and machine learning fields [42]. 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, see [42, 44] 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). A node in the DAG corresponds to a random variable, and the edge set in DAG encodes the Markov condition on the nodes i.e., every node is conditionally independent of its non-descendents given its parents. DAGs play a prominent role in describing causal models [56, 68] and facilitating causal discovery from observational data [14], and have a wide range of applications in biology [48], machine learning [42, 70], and genetics [78].

Learning BNs, arguably one of the most central classes of probabilistic graphical models, is an important task in modern artificial intelligence research [42]. An essential part of this

problem entails learning the DAG structure of BN from observational data. Learning the DAG which best explains observed data is an NP-hard problem [15]. Despite this negative theoretical result, there has been interest in developing methods for learning DAGs in practice [5, 16, 23, 24, 36, 53, 61].

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 [68], 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 dissertation, aim to find a DAG that maximizes a score that measures how well the DAG fits the data. To establish the consistency of score-and-search methods, one does not require the so-called (strong) faithfulness assumption in contrast with constraint-based methods, see [58, 72].

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 set (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, see [74]. For stage 2, there exist elegant exact algorithms based on dynamic programming [28, 40, 41, 54, 57, 66, 77], A* algorithm [76, 77], and integer-programming [4, 5, 21, 22, 23, 24]. The The A* algorithm identifies the optimal DAG by solving a shortest path problem in an implicit state-space search graph. The integer-programing (IP) formulations for learning DAG from discrete data rely on identifying the best parent set (i.e, pa_v) for each node v in the network wherein 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 (i.e., $m2^{m-1}$ 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., [5, 22, 24]) or prune the CPS (e.g., [19, 43]). Another strategy, referred to as the hybrid approach, is to use conditional independence test to construct a super-structure of a Bayesian network [71]

or obtain an ordering among nodes [67] a priori to limit the DAG space before performing the score-and-search algorithm.

A comprehensive empirical evaluation of A* algorithm and IP methods for discrete data is conducted in [46]. The results show that the relative efficiency of these methods varies due to the intrinsic differences between them. 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.

In principle, exact methods developed for *discrete* data can be applied to *continuous* data, since they are compatible with arbitrary score functions. However, such an approach results in exponentially sized formulations, which require advanced algorithms for efficient computation. Alternatively, tailored formulations are possible for learning DAG from an important class of BNs where the causal relations between continuous random variables is encoded using *linear structural equation models* (SEMs), see Section 1.2. Such formulations can be solved using off-the-shelf optimization solvers without the need to implement advanced algorithms. To our knowledge, Park and Klabjan [53] and Xiang and Kim [74] provide the only exact algorithms for learning medium to large DAGs from continuous data with linear SEMs. An MIQP formulation using the topological ordering of variables is proposed in [53]. An A*-lasso algorithm for learning an optimal DAG from continuous data with an ℓ_1 regularization is developed in [74]. 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.

While statistical properties of exact algorithms can be rigorously analyzed [45, 72], it is much harder to assess the statistical properties of approximate algorithms [1, 33, 36, 81] that offer no optimality guarantees [40]. 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 with large sample size, and causal structure learning from observational data becomes feasible [45, 58].

In this work, we focus on developing new discrete optimization formulations to learn an optimal DAG structure of a BNs when the causal relations between continuous random variables can be encoded using linear structural equation models (SEMs).

1.1 Bayesian networks

A Bayesian Network (BN) provides an elegant yet intuitive graphical representation of the joint probability distribution $p(X_1, \ldots, X_m)$ across many variables $X = (X_1, \ldots, X_m)$. Since joint probability distribution contains all information about a set of random variables, it is naturally of great importance to model it. By using chain rule, one can express joint probability distribution as

$$p(X_1, \ldots, X_m) = p(X_1)p(X_2|X_1)p(X_3|X_1, X_2) \ldots p(X_{m-1}|X_1, \ldots, X_m).$$

Despite this factorization, even in the simplest case where variables are binary-valued, a joint distribution still requires the specification of $2^m - 1$ numbers, i.e., the probabilities of the 2^m different assignments of values $x_1, \ldots, x_m$. By exploiting the conditional independence among variables, one can cope with intractability by using fewer parameters and a compact expression. Random variables X and Y are conditionally independent given another random variable Z if

$$p(X = x|Y = y, Z = z) = p(X = x|Z = z) \quad \forall x, y, z.$$

This definition is equivalent to

$$p(X = x, Y = y|Z = z) = p(X = x|Z = z)p(Y = y|Z = z) \quad \forall\, x, y, z.$$

Conditional independence is fundamental to BNs. For each X_i, if we find a subset $pa_{X_i} \subseteq \{X_1, ..., X_m\}$ such that given pa_{X_i}, X_i is conditionally independent of all variables in $\{X_1, ..., X_m\} \setminus pa_{X_i}$, i.e.,

$$p(X_i|X_1, \ldots, X_m) = p(X_i|pa_{X_i}),$$

where pa_{X_i} denotes the parent set of variables for X_i, then the joint probability distribution can be concisely written as

$$p(X_1, \ldots, X_m) = \prod_{i=1}^{m} p(X_i | pa_{X_i}) \tag{1.1}$$

which reduces the number of parameters substantially. The $2^m - 1$ dimensional joint distribution is thus determined by merely $2m - 1$ parameters in a binary case.

A BN represents this factorization of the joint probability distribution with a DAG. Given a directed graph $\mathcal{G}_0 = (V, E)$ where V denotes the set of nodes and E denotes the set of arcs between the nodes in V. Each node represents a random variable. The arc set E then comprises of ordered pairs $(i, j) \in E$ that represent an arc pointing from i to j. Arcs in a BN represent direct dependencies among variables. The parents of a node X_i, i.e., pa_{X_i}, consist of all the nodes pointing at X_i. Similarly, X_i is their child. Therefore, a BN has two components: a DAG and a set of conditional probability distributions of each node X_i given its parents, i.e., $p(X_i | pa_{X_i})$. The first qualitative component is referred to as *BN structure* while the second quantitative component is referred to as the *BN parameters* [11]. This BN represents a unique joint probability distribution given by Equation (1.1).

1.2 Penalized DAG Estimation with Linear SEM

The causal effect of (continuous) random variables in a DAG $\mathcal{G}_0$ can be described by structural equation models (SEMs) that represent each variable as a (nonlinear) function of its parents. The general form of these models is given by Pearl [56]

$$X_k = f_k\Big(pa_k^{\mathcal{G}_0}, \delta_k\Big), \quad k = 1, \ldots, m, \tag{1.2}$$

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 latent random variables, δ_k represent the unexplained variation node k, $k = 1, \ldots, m$.

An important class of SEMs is defined by linear functions, $f_k(\cdot)$, which can be described

by m linear regressions of the form

$$X_k = \sum_{j \in pa_k^{\mathcal{G}_0}} \beta_{jk} X_j + \delta_k, \quad k = 1, \ldots, m, \tag{1.3}$$

where β_{jk} represents the effect of node j on k for $j \in pa_k^{\mathcal{G}_0}$. In the special case that the random variables are Gaussian, Equations (1.2) and (1.3) are equivalent. In this case, 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 $j \notin pa_k^{\mathcal{G}_0}$ [65]. However, estimation procedures proposed in this work are not limited to Gaussian random variables and apply more generally to linear SEMs [45, 65].

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 this work, we refer to directed edges as *arcs* and to undirected edges as *edges*.

Consider n i.i.d. observations from the linear SEM (1.3). 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 (1.3) can be compactly written as

$$\mathcal{X} = \mathcal{X}B + \Delta, \tag{1.4}$$

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 $\beta_{jk} \neq 0$.

The negative log likelihood for linear SEMs is proportional to

$$l(\beta; X) = \frac{n}{2} \text{tr}\{(I - B)(I - B)^T S\}, \tag{1.5}$$

where S is the empirical covariance matrix $S = \frac{1}{n}\mathcal{X}^T\mathcal{X}$; I is the identity matrix [58, 72]. The minimizer of the loss $l(\beta; X)$ provably recovers a true DAG with high probability on finite samples and is consistent for Gaussian SEM [2, 58, 72]. Notably, these results imply that the faithfulness assumption is not required in this setup [72].The statistical properties of $l(\beta; \mathcal{X})$ for non-Gaussian data in [45].

Since we are interested in learning *sparse* DAGs in practice, we add an ℓ_0 regularization term to obtain a sparse estimate. A Tikhonov regularization $\mu||\beta||^2$ for $\mu \geq 0$ could also be added to the formulation to alleviate the effect of noise in the input data [9]. Note the choice of Tikhonov regularization is not required in our setting similar to Dong et al. [27] Thus, the optimization problem corresponding to the penalized negative log-likelihood with super-structure $\mathcal{M}$ (PNL$\mathcal{M}$) for learning sparse DAGs is equivalent to

$$\textbf{PNL}\mathcal{M} \quad \min_{B \in \mathbb{R}^{m \times m}} \quad l(\beta; X) + \frac{\lambda}{2}||\beta||_0 + \frac{\mu}{2}||\beta||^2 \tag{1.6a}$$

$$\text{s.t., } \mathcal{G}(B) \text{ induces a DAG from } \overrightarrow{\mathcal{M}}, \tag{1.6b}$$

where the tuning parameters λ and μ control the degree of regularization. The constraint (1.6b) stipulates that the resulting directed subgraph has to be an induced DAG from $\overrightarrow{\mathcal{M}}$.

When the super-structure is not known, $\mathcal{M}$ is taken as a complete graph, and PNL$\mathcal{M}$ reduces to the classical PNL. The choice of ℓ_0 regularization is deliberate. Although ℓ_1 regularization is well understood with attractive high-dimensional and computational properties in regression [12], in the context of DAG structural learning many of these advantages disappear [1, 33]. Peters and Bühlmann [58] and van de Geer and Bühlmann [72] study the properties of the ℓ_0 regularization norm penalty and show that PNL estimator of a DAG has about the same number of edges as the minimal-edge I-MAP (i.e., a DAG with minimal number of edges representing the joint distribution), and establish the consistency of PNL for learning sparse DAG from Gaussian data when $\lambda = \ln(n)$ which is equivalent to the Bayesian Information Criterion (BIC) score. 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 [72]. In particular, we have

$$\sum_{j=1}^{m}\sum_{k=1}^{m}[\hat{\beta}_{jk} - \beta^0_{jk}]^2 \to O_p(\log(n)n^{-1}) \quad (n \to \infty),$$

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

where $\hat{\beta}_{jk}$ and where $\hat{\mathcal{G}}_n$ is the estimate of the true structure $\mathcal{G}^0$ with sample size n. Further, Loh and Bühlmann [45] study the statistical properties of PNL$\mathcal{M}$ for non-Gaussian data.

1.3 Exact DAG Recovery and Causal Inference

1.3.1 DAG Identifiability

Given the joint distribution $p(X_1, ..., X_m)$ from a structural equation model with DAG $\mathcal{G}_0$, it is of interest to know under which conditions we can recover the graph $\mathcal{G}_0$. Since the joint distribution $p(X_1, \ldots, X_m)$ is Markov with respect to different DAGs, there are many possible graphical models $\{\mathcal{G}, p(X_1, \ldots, X_m)\}$ for the same distribution $p(X_1, \ldots, X_m)$. Hence, the graph $\mathcal{G}_0$ is not *identifiable* in general.

The distribution $p(X_1, \ldots, X_m)$ is faithful with respect to the DAG $\mathcal{G}_0$ if each conditional independence found in $p(X_1, \ldots, X_m)$ is implied by the Markov condition. If faithfulness holds, one can ascertain the Markov equivalence graph of the true DAG $\mathcal{G}_0$ [68]. Nonetheless, the Markov equivalence class may be large. Moreover, faithfulness condition cannot be tested from data [78]. Two graphs entailing the same conditional independences cannot be distinguished from each other, because both the Markov condition and faithfulness only restrict the conditional independences in the joint distribution.

Structural equation models use a different type of restriction. A general Gaussian structural equation model is equivalent to a Gaussian graphical model $\{\mathcal{G}_0, p(X_1, X_2, \ldots, X_m)\}$ [72]. Thus, the structure $\mathcal{G}_0$ is not identifiable from $p(X_1, \ldots, X_m)$. However, it has been recently established that there are multiple cases where the DAG is identifiable: (i) if we consider linear functions and non-Gaussian noise, the underlying DAG $\mathcal{G}_0$ can exactly be recovered [64]; (ii) if one restricts the functions to be additive in the noise component and excludes the linear Gaussian case along with a few other pathological function-noise combinations, the DAG $\mathcal{G}_0$ is identifiable from $p(X_1, \ldots, X_m)$ [59]; (iii) if one consider Gaussian structural equation models where all functions are linear with normally distributed error with equal variance, then DAG $\mathcal{G}_0$ is identifiable again [58]. The identifiability results of all these cases require a condition called causal minimality which is weaker than the faithfulness condition.

1.3.2 Causal Bayesian Networks

A common approach for discovering causal structure is randomized experiments. However, randomized experiments is in many cases too expensive, time-consuming, or even impossible. Therefore, revealing causal information by analyzing purely observational data, known as causal discovery, has received a great deal of attention [68]. The task is not possible if we do not impose some assumptions that connect causal structure with statistical structure.

Spirtes et al. [68] and Pearl [56] used DAGs in causal inference. Their frameworks make it possible to systematically connect DAGs to probability distributions. A DAG is causally sufficient if all common causes of variables in the DAG are included in the DAG. Two conditions are sufficient to create a mapping between a DAG and the probabilistic dependence and independence among the variables in a probability distribution: (i) The Causal Markov Condition (CMC), (ii) the Causal Faithfulness Condition.

Definition 1 ***Causal Markov Condition (CMC)****: Given a set of variables V whose causal structure is represented by a DAG $\mathcal{G}_0$, every variable in V is probabilistically independent of its non-descendants in $\mathcal{G}_0$ given its parents in $\mathcal{G}_0$.*

Definition 2 ***Causal Faithfulness Condition (CFC)****: Given a set of variables V whose true causal DAG is $\mathcal{G}_0$, the joint probability of V, i.e., $p(v)$, is faithful to $\mathcal{G}_0$ in the sense that $p(V)$ implies no conditional independence relations not already entailed by the CMC.*

CMC states which variables will be probabilistically independent conditional on other variables, whereas CFC specifies which variables will be probabilistically dependent. The correct DAG will often be underdetermined by the probability distribution even if assuming CMC and CFC [73]

To determine the sets of variables that are probabilistically independent according to CMC, a graph concept known as d-separation is introduced. A path is d-separated by variable set U in case: (1) the path has a triple $i \to m \to j$ or $i \leftarrow m \to j$ such that m is in

U, (2) The path has a collider $i \to m \leftarrow j$ such that m is not in U and no descendant of m is in U [73].

Two variables are d-separated by U if and only if they are d-separated by U along all paths. CMC implies that all variables that are d-separated in a DAG will be conditionally independent, and CFC implies that only the variables that are d-separated in a DAG will be independent (i.e., all others will be dependent) in the corresponding probability distribution [73].

1.4 Research Scope and Outline

Given the state of existing algorithms for DAG learning from continuous data, there is currently a gap between theory and computation. This dissertation focuses on learning DAG corresponding to linear SEMs from continuous observational data.

In Chapter 2, we propose a new mixed-integer quadratic program (MIQP) with an ℓ_0 regularization to learn a sparse DAG from data with linear SEMs. The resulting LN formulation uses an alternative way to remove cycles. The computational performance of an MIQP usually depends on the size and tightness (strength) of its continuous relaxation. With respect to size, the LN formulation entails the fewest number of variables among existing models in the literature. In regard to the strength, we show that continuous relaxations of existing models – including LN formulation – attain the same optimal objective function value under a mild condition. To further improve the computational performance for large-scale graphs, we initially learn an undirected and possibly cyclic graph (super-structure). An important example of a super-structure is the moral graph of a DAG, because it can be consistently estimated from observational data under proper assumptions by solving a convex optimization optimization. One unique feature of LN formulation size is that its number of binary variables and constraints solely depend on the number of edges in the super-structure (e.g., moral graph). Our computational results show that LN formulation is computationally far more efficient compared to existing formulations, especially in the presence of a sparse super-structure.

In Chapter 3, we focus on a key challenge in existing MIQP formulations which is to enforce bounds on the continuous optimization variables corresponding to the arc weights in the DAG structure. This is commonly modeled using a standard "big-M constraint" in the associated MIQPs. However, this strategy leads to a poor continuous relaxation because there is no natural upper bound for the arc weights. Therefore, LN formulation – as well as all other "big-M" formulations – attain poor lower bounds. To circumvent this issue, we present a perspective formulation for DAG structural learning problem. The resulting formulation is a mixed-integer non-linear program (MINLP) which is computationally challenging to solve. Nonetheless, we present two different reformulations for this MINLP: (i) a *mixed-integer second-order conic program* (MISOCP), (ii) a *semi-infinite mixed-integer linear program* (SIMILP). These formulations have tighter continuous relaxations compared to the existing "big-M" formulations. Our experimental results demonstrate that MISOCP improves the lower bound and reduces the optimality gap when compared with the LN formulation whereas SIMILP is not computationally as competitive as LN. We further give an early stopping criterion under which the branch-and-bound process can be terminated early. We establish that the obtained solution under this stopping criterion asymptotically converges to the true coefficients in DAG with high probability under proper conditions. Our result leverages the statistical consistency results of the PNL estimate with ℓ_0 regularization along with the unique structure of branch-and-bound wherein both lower and upper bound values on the objective function are available [58, 72].

In Chapter 4, we share our conclusions and future research avenues.

Chapter 2

MIXED INTEGER QUADRATIC PROGRAMMING (MIQP)

In this chapter, we present a way to remove cycles which results in a new MIQP formulation for leaning DAG from continuous observational data. This chapter is based on Manzour et al. [47].

2.1 Relevant Work

Prior to presenting existing 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 new definition.

Definition 3 *A tournament is a directed graph obtained by specifying a direction for each edge in the super-structure graph $\mathcal{M}$ (see, Figure 2.1).*

Proposition 1 *There exists an optimal solution $\mathcal{G}(B)$ to PNL$\mathcal{M}$ (1.6) with an ℓ_0 (or an ℓ_1) regularization that is a cycle-free tournament.*

Proof. Let $(\hat{\beta}, \hat{z})$ be an optimal solution for (1.6) with an optimal objective value $F(\hat{\beta})$. Let us refer to the DAG structure corresponding to this optimal solution by $DAG(V, \hat{E}^{\rightarrow})$. Suppose that for some (j, k), we have $\hat{z}_{jk} + \hat{z}_{kj} = 0$. To prove the proposition, we construct an optimal solution which satisfies $z_{jk} + z_{kj} = 1$ for all pairs of (j, k) and meets the following conditions: (i) this corresponding DAG (tournament) is cycle free (ii) this tournament has the same objective value, i.e., $F(\hat{\beta})$, 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

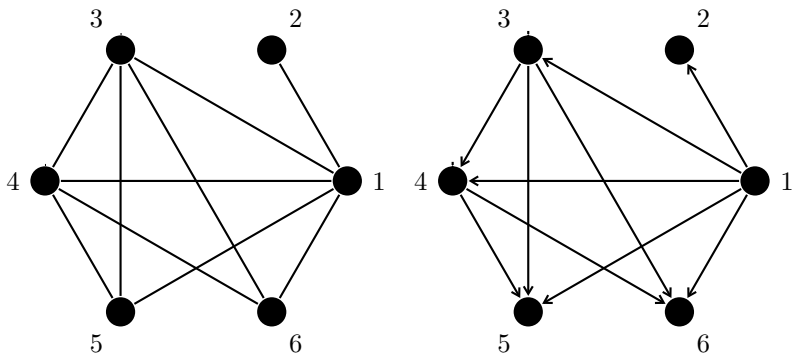


Figure 2.1: A super-structure graph $\mathcal{M}$ (left) and a tournament (right) with six nodes which does not contain any cycles.

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. □

Proposition 1 implies that for DAG learning from continuous variables, the search space reduces to acyclic tournament structures. This is a far smaller search space when compared with the super-exponential $O\left(m!2^{\binom{m}{2}}\right)$ search space of DAGs for discrete variables. However, one has to also identify the optimal β parameters simultaneously.

A solution method based on brute force enumeration of all tournaments requires $\bar{C} \times m!$ computational time when $\mathcal{M}$ is complete, where $\bar{C}$ 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 DAGs 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 is based on this search space.

In Section 2.2, 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 (1.6) in the section 2.1.1. Given the desirable statistical properties of ℓ_0 regularization [72] and the fact that existing mathematical formulations are given for ℓ_1 regularization, we present the formulations for ℓ_0 regularization.

2.1.1 Existing Mathematical Models

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, $\forall (j,k) \in \overrightarrow{E}$

In DAG structural learning, μ is assumed to be zero. We follow this convention in this chapter. In the next chapter, we further study the choice of μ.

One source of difficulty in solving PNL$\mathcal{M}$ is due to the acyclic nature of DAG. There are several ways to encode such a constraint. A popular technique for ensuring acyclicity is to use cycle elimination constraints. Let $\mathcal{C}$ be the set of all possible cycles and $\mathcal{C}_A \in \mathcal{C}$ be the set of arcs from 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

$$\ell_0\textbf{-CP} \quad \min \quad F(\beta, g) \tag{2.1a}$$

$$-M g_{jk} \leq \beta_{jk} \leq M g_{jk}, \quad \forall (j,k) \in \overrightarrow{E}, \tag{2.1b}$$

$$\sum_{(j,k) \in \mathcal{C}_A} g_{jk} \leq |\mathcal{C}_A| - 1, \quad \forall \mathcal{C}_A \in \mathcal{C}, \tag{2.1c}$$

$$g_{jk} \in \{0, 1\}, \quad \forall (j,k) \in \overrightarrow{E}. \tag{2.1d}$$

Following Park and Klabjan [53], the objective function (2.1a) is an expanded version of $l_n(\beta)$ in PNL$\mathcal{M}$ (multiplied by $2n^{-1}$) with an ℓ_0 regularization. The constraints in (2.1b) stipulate that $\beta_{jk} \neq 0$ only if $z_{jk} = 1$, where M is a sufficiently large constant. The constraints in (2.1c) rule out all cycles. Note that for $|\mathcal{C}_A| = 2$, constraints in (2.1c) 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 (e.g., [50]) 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.

Note that removing cycles in a graph bears resemblance to removing subtours in a Traveling Salesman Problem (TSP), which has been studied extensively [18]. For example, similar to the cycle elimination constraint (2.1c) in the CP formulation, the subtour elimination constraint in the Dantzig-Fulkerson-Johnson (DFJ) formulation [25] 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 a DAG, because it assumes that there is exactly one incoming and one outgoing arc to any node due to the TSP constraints.

The second formulation is based on a well-known combinatorial optimization problem, known as *linear ordering* (LO), see [35]. 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 $_j$ 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 $\binom{p}{2}$ variables to cast the LO problem as an MIQP formulation [35].

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., βs, is added. Indeed, the DAG structural learning reduces to classical LO problem once the β values are known.

The PNL$\mathcal{M}$ can then be formulated as (2.2) in which cycles are ruled out by directly imposing the linear ordering constraints.

$$
\begin{aligned}
\ell_0\textbf{-LO} \quad \min \quad & F(\beta, g), && && \text{(2.2a)}\\
& -Mg_{jk} \leq \beta_{jk} \leq Mg_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.2b)}\\
& w_{jk}+w_{kj}=1, && \forall j,k \in V, j\neq k, && \text{(2.2c)}\\
& g_{jk} \leq w_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.2d)}\\
& w_{ij}+w_{jk}+w_{ki} \leq 2, && \forall i,j,k \in V, i\neq j\neq k, && \text{(2.2e)}\\
& w_{jk} \in \{0,1\}, && \forall j,k \in V, j\neq k, && \text{(2.2f)}\\
& g_{jk} \in \{0,1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(2.2g)}
\end{aligned}
$$

The interpretation of constraints (2.2b)-(2.2c) are straightforward. The constraints in

(2.2d) imply that if node j appears after node k in a linear ordering ($w_{jk} = 0$), there should not exist an arc from j to k ($g_{jk} = 0$). The set of inequalities (2.2e) 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 to remove 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 [65], but the problem remains challenging when the ordering is not known. The topological ordering prevents cycles in the graph. This property is used by Park and Klabjan [53] 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 by

$$
\begin{aligned}
\ell_0\textbf{-TO} \quad \min \quad & F(\beta, g), && && \text{(2.3a)}\\
& -Mg_{jk} \le \beta_{jk} \le Mg_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.3b)}\\
& g_{jk} \le z_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.3c)}\\
& z_{jk} + z_{kj} \le 1, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.3d)}\\
& z_{jk} - pz_{kj} \le \sum_{s \in V} s\,(o_{ks} - o_{js}), && \forall (j,k) \in \overrightarrow{E}, && \text{(2.3e)}\\
& \sum_{s \in V} o_{rs} = 1, && \forall r \in V, && \text{(2.3f)}\\
& \sum_{r \in V} o_{rs} = 1, && \forall s \in V, && \text{(2.3g)}\\
& z_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.3h)}\\
& o_{rs} \in \{0, 1\}, && \forall\, r, s \in \{1, 2, \ldots, m\}, && \text{(2.3i)}\\
& g_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(2.3j)}
\end{aligned}
$$

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 (2.3c) enforce the

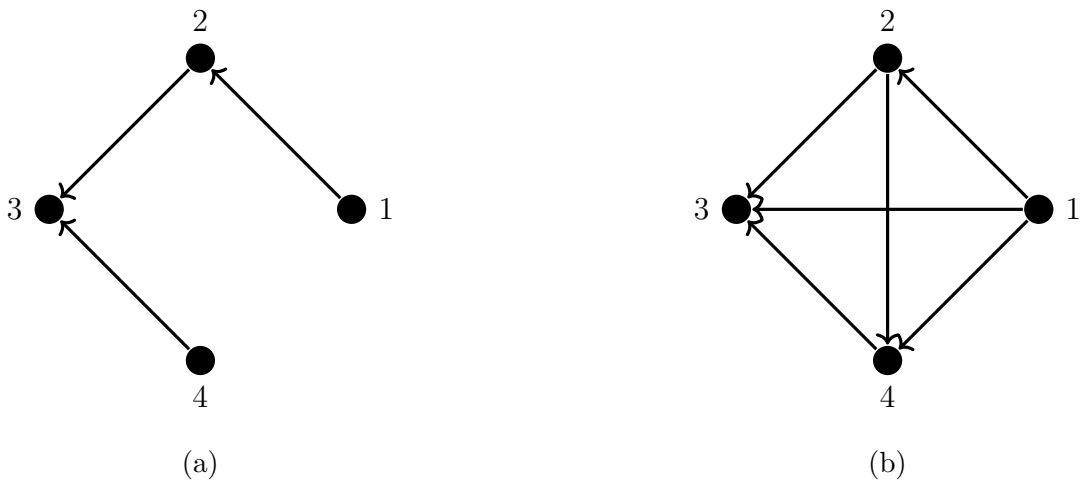


Figure 2.2: The role of the binary decision variables in ℓ_0 regularization: Using z (instead of g) in the objective function creates a graph similar to (b) and counts the number of arcs instead of the number of non-zero βs in (b).

correct link between g_{jk} and z_{jk}, i.e., g_{jk} has to take value zero if $z_{jk} = 0$. The constraints in (2.3d) 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 (2.3e) remove cycles by imposing an ordering among nodes. The set of constraints in (2.3f)-(2.3g) 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* (2.3d) *can be replaced by* $z_{jk} + z_{kj} = 1,\ \forall (j,k) \in \overrightarrow{E}$.

In constraints (2.3b)-(2.3i), both variables z and g are needed to correctly model the ℓ_0 regularization term in the objective function (see, Figure 2.2a). This is because the constraints (2.3e) 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 $\sum_{s \in V} s\,(o_{ks} - o_{is}) \geq 2$, which enforces $z_{ik} = 1$. Such a transitivity relation, however, need not hold for the decision vector g. In other words, the decision variable g_{jk} is used to keep track of the number of non-zero weights β_{jk} associated with the arc (j, k), and the decision vector z is used to remove cycles via the set of constraints in (2.3e) by creating an acyclic tournament on the super-structure $\mathcal{M}$. A tournament on super-structure $\mathcal{M}$ assigns a direction for each edge in an undirected super-structure $\mathcal{M}$. In other words, if we were to let $g_{ij} = z_{ij}$ for $(i, j) \in \overrightarrow{E}$ and hence use the decision variable z in the objective, then we would be counting the number of edges, equal to $|E|$, instead of number of non-zero β values.

2.1.2 Contributions

In this section, we develop tailored exact DAG learning methods for continuous data from linear SEMs, and 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 [68]. The skeleton and moral graphs are particularly important cases, because they can be consistently estimated from observational data under proper assumptions [39, 45]. Such prior information limits the number of possible DAGs and improves the computational performance.

- We 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 MIQP formulation and the A*-lasso algorithm for the case of ℓ_1 regularization. As noted earlier in the introduction, there is no clear winner among A*-style algorithms and IP formulations for DAG learning from discrete data [46]. Thus, a wide range of approaches based on dynamic programming, A* algorithm, and IP-based formulations have been proposed for discrete data. In contrast, 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.

2.2 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 [37]. 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 may interpret ψ_k as $\sum_{s=1}^{m} s\, o_{ks}$ for all $k \in V$, where the variables o_{ks} are as defined in the TO formulation. However, note that the notion of ψ_k is more general because ψ_k need not be integer. Figure 2.3 depicts the layered network encoding of a DAG. With this notation, our *layered network* (LN) formulation can be written as

$$\begin{aligned}
\min \quad & F(\beta, g), && && \text{(2.4a)}\\
& -Mg_{jk} \le \beta_{jk} \le Mg_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.4b)}\\
& g_{jk} \le z_{jk}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.4c)}\\
& z_{jk} + z_{kj} = 1, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.4d)}\\
& z_{jk} - (p-1) z_{kj} \le \psi_k - \psi_j, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.4e)}\\
& z_{jk} \in \{0,1\}, && \forall (j,k) \in \overrightarrow{E}, && \text{(2.4f)}\\
& 1 \le \psi_k \le m, && \forall k \in V, && \text{(2.4g)}\\
& g_{jk} \in \{0,1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(2.4h)}
\end{aligned}$$

The interpretation of the constraints (3.3b)-(2.4d) is straightforward. The constraints in (2.4e) 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 parameter $m-1$ in (2.4e). 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* (3.4) *is an optimal solution to* (1.6).

Proof. First we prove that (2.4e) removes all cycles. Suppose, for contradiction, that a cycle of size $p \ge 2$ is available and represented by $(1, 2, \ldots, p, 1)$. This implies $z_{j+1,j} =$

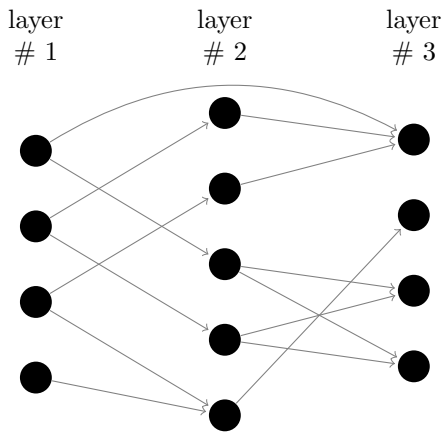


Figure 2.3: Layered Network encoding of a DAG.

0 and $z_{j,j+1} = 1$, $\forall j = \{1, \ldots, p-1\}$, and $z_{p,1} = 1, z_{1,p} = 0$. Then,

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

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 (2.4e) is always satisfied. □

The LN formulation highlights a desirable property of the layered network representation of a DAG in comparison to 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 2.3). 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 2.2a 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 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, both quantities are the same, and equal to $m!$. Otherwise, a brute-force search for finding the optimal DAG has computational time $\mathcal{L}\bar{C}$, where $\mathcal{L}$ denotes the total number of minimal layered numberings, and $\bar{C}$ is the computational complexity of solving PNL$\mathcal{M}$ given a known tournament structure.

We close this sub-section by noting that a related set of constraints (2.4e) appear in the Miller-Tucker-Zemlin (MTZ) formulation for asymmetric TSP [18, 49, 63]. In this context, constraints (2.4e), along with the TSP constraints that ensure that each node has one in-

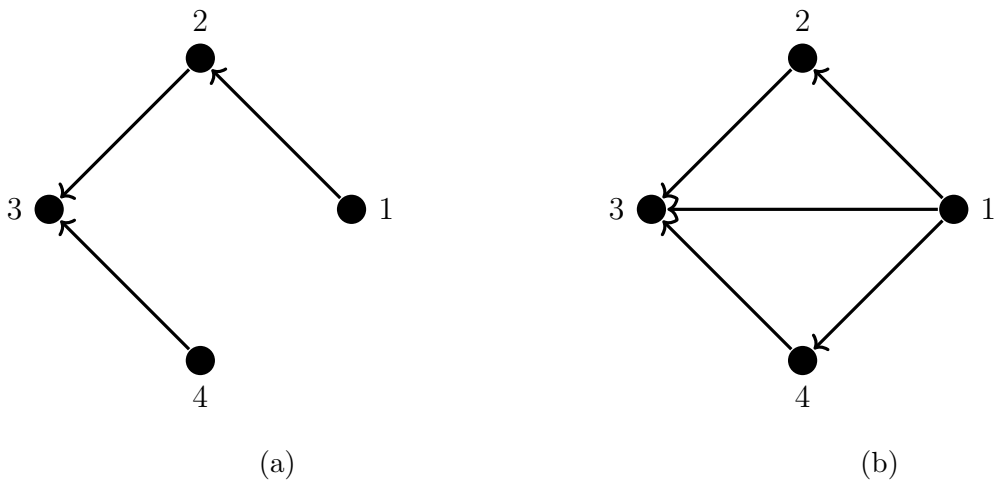


Figure 2.4: Illustration of the Layered Network formulation: (a) nodes 1, 2, 3, 4 are placed in layers 1, 2, 3, 1, respectively; (b) nodes 1, 2, 3, 4 are placed in layers 1, 2, 3, 2, respectively.

coming and one outgoing arc, result in distinct ψ values that correspond to the order 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 [6, 26]; 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 (2.4e) is introduced in [21] for learning DAG with discrete data. However, the formulation in [21] 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 [22, 23, 24]. 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.

Both TSP and DAG structural learning with discrete data have a *linear objective func-*

tion. For mixed-integer *linear* programs, with linear objective functions, it is known that stronger formulations attain better computational performance, see e.g., [51] for a comparative analysis in the TSP context. This is because the problem is solved as successive linear programs, and it is known that 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 and faster convergence. In contrast, DAG structural learning for continuous variables with linear SEMs is cast as a mixed-integer quadratic optimization (MIQO) model. Due to the quadratic objective, an optimal solution may not be at an extreme point of the convex hull of feasible points and stronger formulations do not necessarily guarantee better lower bounds. Later in Propositions 5 and 6, we shed light on why LN formulation performs well for learning DAG from continuous data with Linear SEMs.

2.2.1 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 [36, 53, 65, 74, 81]. The LN formulation with ℓ_1-regularization can be written as

$$\min \frac{1}{n}\sum_{i\in I}\sum_{k\in V}(x_{ik} - \sum_{(j,k)\in E^{\rightarrow}} \beta_{jk}x_{ij})^2 + \lambda \sum_{(j,k)\in \overrightarrow{E}} |\beta_{jk}|, \tag{2.5a}$$

$$-Mz_{jk} \leq \beta_{jk} \leq Mz_{jk} \quad \forall (j,k) \in \overrightarrow{E}, \tag{2.5b}$$

$$(2.4e) - (2.4g).$$

In the next remark, we state a property which only holds 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 (2.4e) as*

$$z_{jk} - (p-1)z_{kj} \leq \sum_{j\in V\setminus k} z_{jk} - \sum_{k\in V\setminus j} z_{kj} \quad \forall\, j,k \in V \quad j \neq k.$$

Table 2.1: 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)	$\lvert\overrightarrow{E}\rvert$	$\lvert\overrightarrow{E}\rvert + \binom{p}{2}$	$p^2 + \lvert E\rvert + \lvert\overrightarrow{E}\rvert$	$\lvert E\rvert + \lvert\overrightarrow{E}\rvert$	$2\binom{p}{2}$	$\binom{p}{2}$	$p^2 + 3\binom{p}{2}$	$3\binom{p}{2}$
# Binary Vars (ℓ_1)	$\lvert E\rvert$	$\binom{p}{2}$	$p^2 + \lvert E\rvert$	$\lvert E\rvert$	$\binom{p}{2}$	$\binom{p}{2}$	$p^2 + \binom{p}{2}$	$\binom{p}{2}$
# Constraints (both ℓ_0 and ℓ_1)	*Exp*	$2\binom{p}{3}$	$\lvert\overrightarrow{E}\rvert + 2p$	$\lvert\overrightarrow{E}\rvert$	$2\binom{p}{3}$	$2\binom{p}{3}$	$\binom{p}{2} + 2p$	$\binom{p}{2}$

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 (2.2) 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 2.1 shows the number of binary variables and the number of constraints associated with cycle prevention constraints in each model. Evidently, ℓ_0 models require additional binary variables compared to the corresponding ℓ_1 models. 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 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.

2.3 Continuous Relaxation

One of the fundamental concepts in IP is relaxations, wherein some or all constraints of a problem are relaxed. Relaxations are used to obtain a sequence of easier to solve problems resulting 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 4 *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, i.e., $\mathcal{R}(LO) \subset \mathcal{R}(LN)$.*

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

Proof. 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

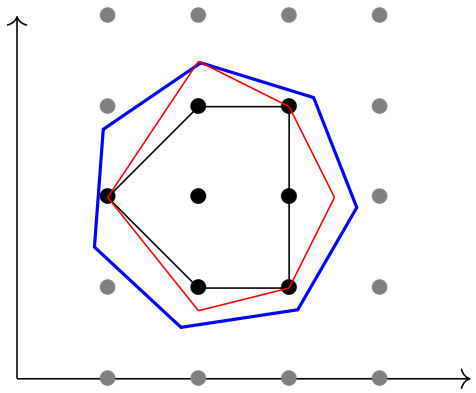


Figure 2.5: Continuous relaxation regions of three IP models. The tightest model (convex hull) is represented by the black polygon strictly inside other polygons. The blue and red polygons represent valid yet weaker formulations. The black points show the feasible integer points for all three formulations.

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 (2.2e) over $i \in V \setminus \{j,k\}$ we obtain

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

where the equivalences follow from constraints (2.2c). 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 (2.4e) 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.

□

Consider (2.2c)-(2.2e) in the linear ordering formulation given by

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

Fixing j and k for each $(j,k) \in \overrightarrow{E}$ and summing over all $i \in V$ we obtain

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

Given our mapping, $z_{jk} = w_{jk}$ for all $(j,k) \in \overrightarrow{E}$ and $\psi_j = \sum_{\ell \in V} w_{\ell j}$ 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 (2.4e) 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. □

Proposition 4 *When the parameter m in (2.3e) is replaced with $m-1$, the TO formulation is stronger than the LN formulation, that is, $\mathcal{R}(TO) \subset \mathcal{R}(LN)$.*

Proof. This proof is for TO formulation when the parameter m on (2.3e) 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 (2.3f), (2.3g), and (2.3i). 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 $\mathcal{R}(TO) \subset \mathcal{R}(LN)$.

□

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 to the other formulations. Related strength results are also established for MTZ formulation for TSPs, see e.g., [52, 55]. However, even though one of the constraints is common in LN formulation for DAG learning and MTZ formulation for TSP, because the polyhedron we study is different, we establish the strength results given in Propositions 3 and 4 for completeness.

Proposition 5 *Let $\beta^\star_{jk}$ denote the optimal coefficient associated with an arc $(j,k) \in \overrightarrow{E}$ from (1.6). 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 $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$.*

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

Part A. $\bar{F}(\beta^\star_{LO}) = \bar{F}(\beta^\star_{LN})$.

Case 1. ℓ_1 regularization

Suppose $(\beta^\star_{LN}, z^\star)$ is an optimal solution associated with the continuous relaxation of the LN formulation (3.3a)-(2.4g) and $(\beta^\star_{LO}, w^\star)$ is an optimal solution associated with continuous relaxation of the LO formulation with ℓ_1-regularization.

Given Proposition 3, we conclude that $\bar{F}(\beta^\star_{LN}) \leq \bar{F}(\beta^\star_{LO})$. We prove that $\bar{F}(\beta^\star_{LN}) \geq \bar{F}(\beta^\star_{LO})$ 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 $\bar{F}(\beta^\star_{LN}) \geq \bar{F}(\beta^\star_{LO})$.

Given an optimal solution $(\beta^\star_{LN}, z^\star)$ 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

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

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.

$|\beta_{jk}| \leq M w_{jk}, \quad \forall (j,k) \in \overrightarrow{E}$ (2.2b)

$w_{jk} + w_{kj} = 1, \quad \forall (j,k) \in \overrightarrow{E}$ (2.2c)

$w_{ij} + w_{jk} + w_{ki} \leq 2, \quad \forall (i,j),(j,k),(k,i) \in \overrightarrow{E}\ i \neq j \neq k.$ (2.2e)

The set of constraints (2.2b) is trivially satisfied because we set $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$. The set of constraints (2.2c) is trivially satisfied. The set of constraints (2.2e) 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

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

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.

$$|\beta_{jk}| \leq M g_{jk}, \quad \forall (j,k) \in \overrightarrow{E}, \text{ (2.2b)}$$
$$g_{jk} \leq w_{kj}, \quad \forall (j,k) \in \overrightarrow{E} \text{ (2.2d)}$$
$$w_{jk} + w_{kj} = 1, \quad \forall (j,k) \in \overrightarrow{E} \quad \text{(2.2c)}$$
$$w_{ij} + w_{jk} + w_{ki} \leq 2, \quad \forall i,j,k \in V, i \neq j \neq k, \quad \text{(2.2e)}$$

The set of constraints in (2.2c) is satisfied similar to ℓ_1 case. The proof that constraints (2.2b), (2.2d), and (2.2e) 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 (2.3f)-(2.3g). 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 (2.2c)-(2.2e) 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 (2.1c) 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 (2.1c). 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 (2.2c)-(2.2e) 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 $\bar{F}(\beta^{\star}_{LN}) \geq \bar{F}(\beta^{\star}_{TO})$. □

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 $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^{\star}_{jk}|$.*

Proof. 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. □

Proof. Let $(\beta^\star, g^\star)$ be the optimal solution to the continuous relaxation of MIQP$_M$ where $M \to \infty$. In such a solution, $g^\star_{jk}$ has to take value $\frac{\beta^\star_{jk}}{M}$. Otherwise, we can reduce the value of decision variable g without violating any constraints while reducing the objective function. This implies that $g^\star = \frac{\beta^\star}{M}$. Thus, the objective function reduces to ℓ_1 regularization with the coefficient $\frac{\lambda||\beta||_1}{2M}$. □

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 root relaxation, but it also holds throughout the branch-and-bound process under the specified condition on M, if the same branching choices are made. 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 [53] to obtain a value of M, which is explained in Section 2.4 and used in our computational experiments, always satisfies the condition in Proposition 6 across all generated instances.

2.4 Experiments

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 terminated. 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 (2.1c), we begin with no constraint of type (2.1c). 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 process, whenever an integer solution with cycles is obtained. The same approach is used by Park and Klabjan [53]. 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 [53]. 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 [53] 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.

2.4.1 Synthetic datasets

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, that applies 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 raw observational 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 graph, 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 raw observational 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 [39, 45]. 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 A.

2.4.2 Comparison of ℓ_0 MIQPs

Figure 2.6 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 2.6(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 moral graph. Figure 2.6(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 2.6(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 2.6(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 2.6(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 dissertation 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 2.7(a)-(b) show the SHDs for all formulations for $n = 1000$ and $n = 100$, respectively. Comparing Figure 2.7(a) with Figure 2.7(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 identifiability in our setting

[58, 72]. However, for a finite sample size, identifiability may not be guaranteed. Moreover, the appropriate choice of λ for $n = 100$ may be different than the corresponding λ for $n = 1000$.

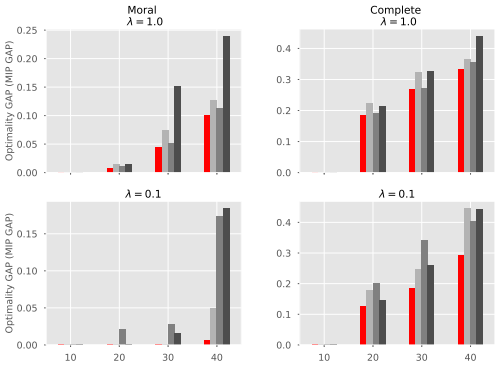
Figure 2.7(a) shows that all methods learn the true DAG with $\lambda = 0.1$, and given a moral graph for $m \in \{10, 20, 30\}$. In addition, SHD is negligible for LN and CP formulations for $m = 40$. However, we observe a substantial increase in SHD (e.g., from 0.2 to near 10 for LN) for complete graphs. 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 [53], a slight change in the objective function value could significantly alter the estimated DAG. Our results corroborate this observation.

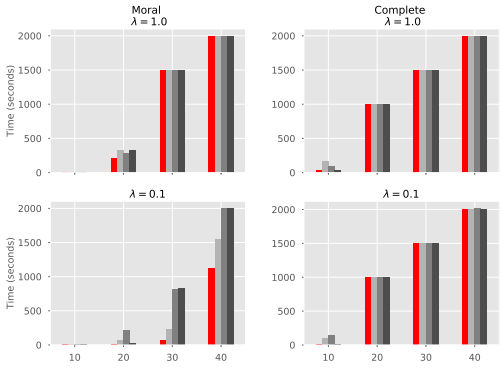
2.4.3 Comparison of ℓ_1 MIQPs

Figure 2.8 shows various average metrics across 10 random graphs for ℓ_1 regularization with $n = 1000$ samples. Figure 2.8(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 2.8(b) shows that the LN formulation requires significantly less computational time in moral instances, and in complete instances with $m \in \{10, 20\}$ compared to other methods. In complete instances, all methods hit the time limit for $m \in \{30, 40\}$. Figures 2.8(c)-(f) can be interpreted similar to the Figures 2.6(c)-(f) for ℓ_0 regularization. Similar to ℓ_0 regularization, Figures 2.8(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.

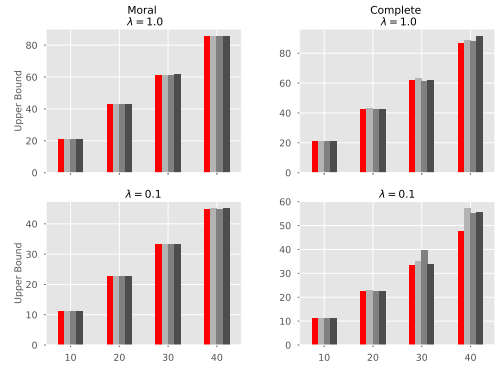
As expected, the DAG estimation accuracy with ℓ_1 regularization is inferior to the ℓ_0 regularization. This is in part due to the bias associated with the ℓ_1 regularization, which could



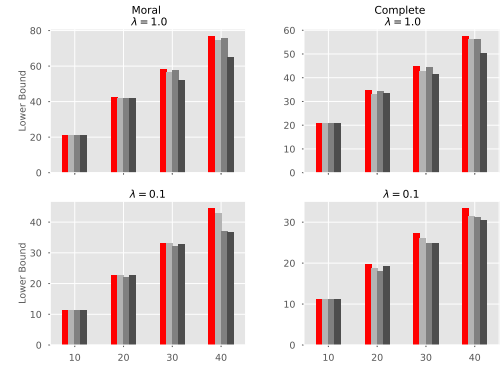
(a) Optimality GAPs for MIQPs



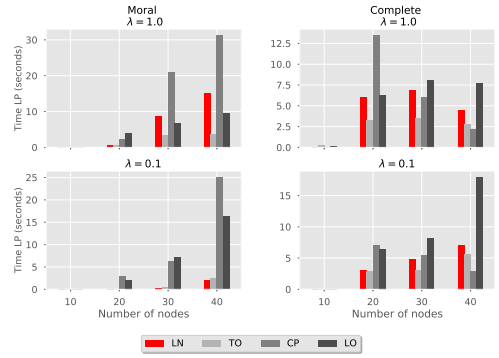
(b) Time (in seconds) for MIQPs



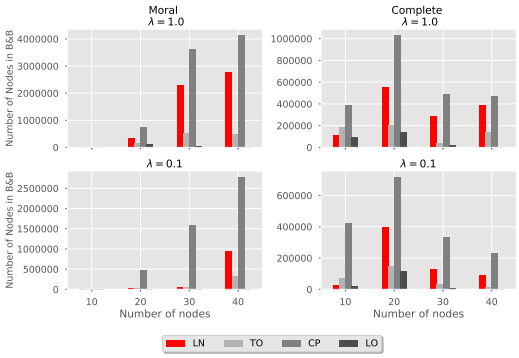
(c) Best upper bounds for MIQPs



(d) Best lower bounds for MIQPs

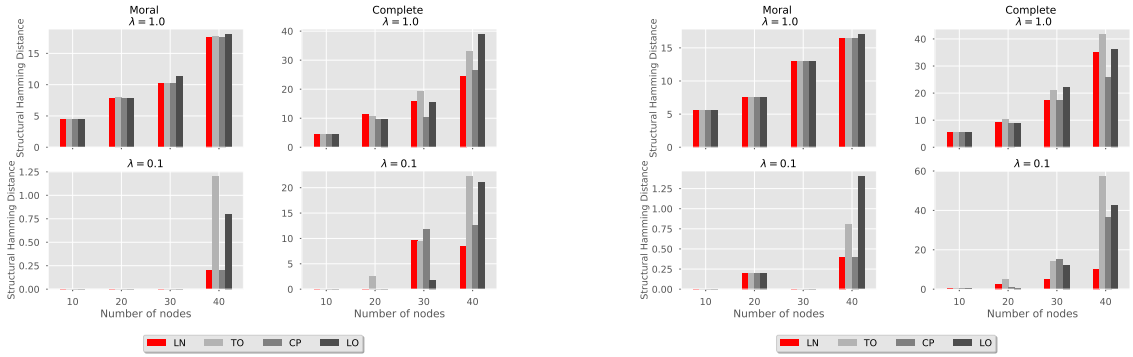


(e) Time (in seconds) for continuous root relaxation



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

Figure 2.6: Optimization-based measures for MIQPs for ℓ_0 regularization with the number of samples $n = 1000$.



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

Figure 2.7: Structural Hamming Distance (SHD) of MIQP estimates for ℓ_0 regularization.

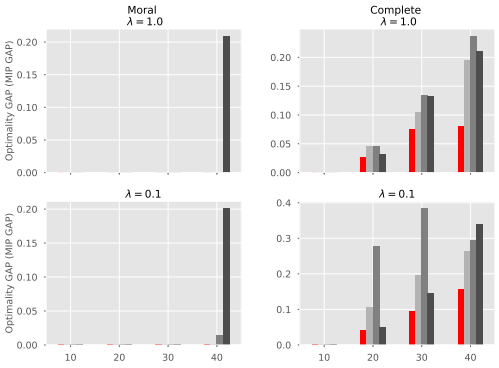
be further controlled with, for example, adaptive ℓ_1-norm regularization [82]. Nonetheless, formulations for ℓ_1 regularization require less computational time and are easier to solve than the corresponding formulations for ℓ_0 regularization.

2.4.4 Comparison with the A-lasso algorithm*

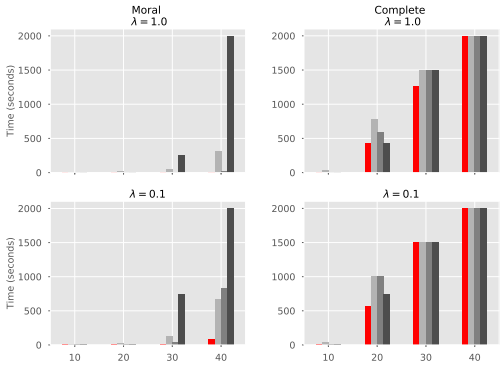
In this section, we compare the LN formulation with A*-lasso [74], using the MATLAB code made available by the authors. For this comparison, the same true DAG structures are taken from [74] and the strength of arcs (β) are chosen from $\mathcal{U}[-1,-0.1] \cup \mathcal{U}[0.1,1]$. The number of nodes in the 10 true DAGs varies from $m = 6$ to $m = 27$ (see Table 2.2). The true DAG and resulting random β coefficients are used to generate $n = 500$ samples for each column of data matrix $\mathcal{X}$.

A time limit of six hours is imposed across all experiments after which runs are terminated. 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.

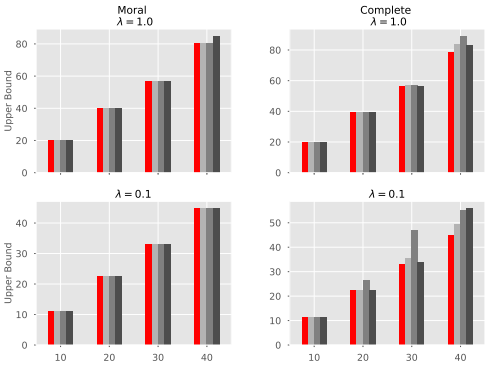
Similar to synthetic data described in Section 2.4.1, we consider two cases: (i) *moral*



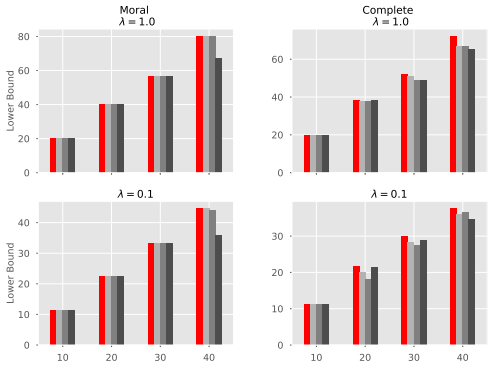
(a) Optimality GAPs for MIQPs



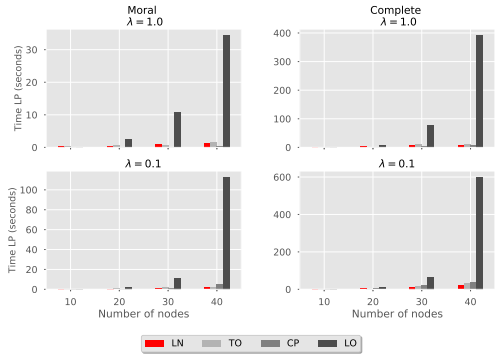
(b) Time (in seconds) for MIQPs



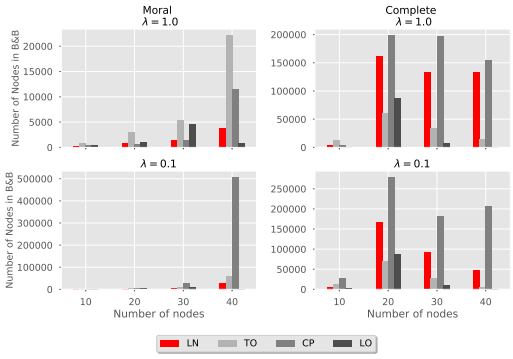
(c) Best upper bounds for MIQPs



(d) Best lower bounds for MIQPs

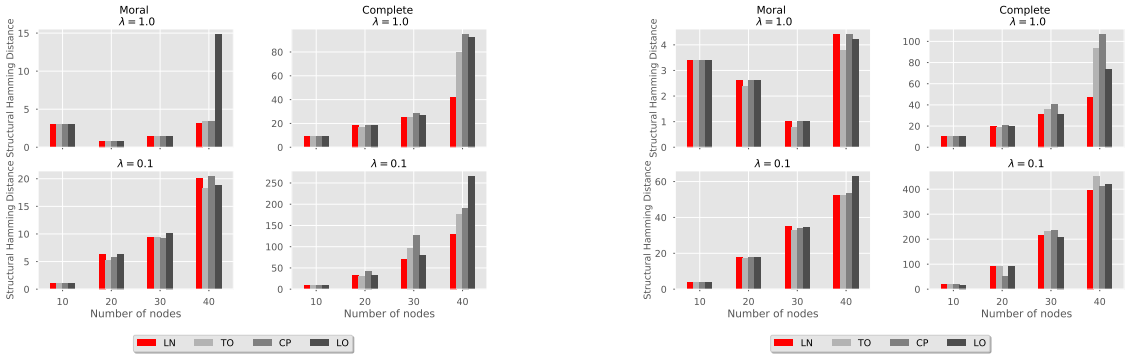


(e) Time (in seconds) for continuous root relaxation



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

Figure 2.8: 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 2.9: Structural Hamming Distance (SHD) of MIQP estimates for ℓ_1 regularization.

instances and (ii) *complete* instances. For the former case, the moral graph is constructed from the true DAG as done in Section 2.4.1. The raw observational data (i.e., $\mathcal{X}$) for moral and complete instances are the same.

We compare the LN formulation with A*-lasso using ℓ_1 regularization. Note that A*-lasso cannot solve the model with ℓ_0 regularization. Furthermore, the original A*-lasso algorithm assumes no super-structure. Therefore, to enhance its performance, we modified the MATLAB code for A*-lasso in order to incorporate the moral graph structure, when available.

In this section, our focus is to evaluate the computational performance of these approaches. We consider $\lambda = 0.1$ for our comparison. Table 2.2 shows the solution times (in seconds) of 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. For complete instances, the results highlight that for small instances A*-algorithm and LN are competitive, whereas the LN formulation outperforms A*-lasso for larger instances. 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. For moral instances, we observe significant improvement in the computational performance of the LN formulation, whereas the improvement in A*-lasso is marginal in comparison. This observation highlights the fact that dynamic programming-based approaches cannot effectively utilize the super-structure knowledge, whereas an IP-based approach, particularly the LN formulation, can significantly reduce the computational times. 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, which implies we cannot obtain a DAG from A* solution for these instances.

For DAG learning from discrete data, an IP-based model, see e.g., [38], 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 have such a limitation because variables are encoded in the space of arcs (instead of parent sets). That is why LN performs well even for complete instances (i.e., no restriction on the 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) all 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) all 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 abrupt the search with the aim of achieving a feasible solution. On the other hand, 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 which 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 [9] 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 2.10 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 whereas the rest of the time is spent to close the optimality gap by increasing the lower bound.

Table 2.2: Computational performance of LN versus A*-algorithm with ℓ_1 for $\lambda = 0.1$

			Moral $\lambda = 0.1$		Complete $\lambda = 0.1$	
Graphs (Data sets)	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	6 hrs	0.395	6 hrs	3.478
Galaxy	20	76	6 hrs	0.740	6 hrs	9.615
Insurance	27	168	6 hours	12.120	6 hrs	6 hrs (.031)
Factors	27	310	6 hours	55.961	6 hrs	6 hrs (.01)

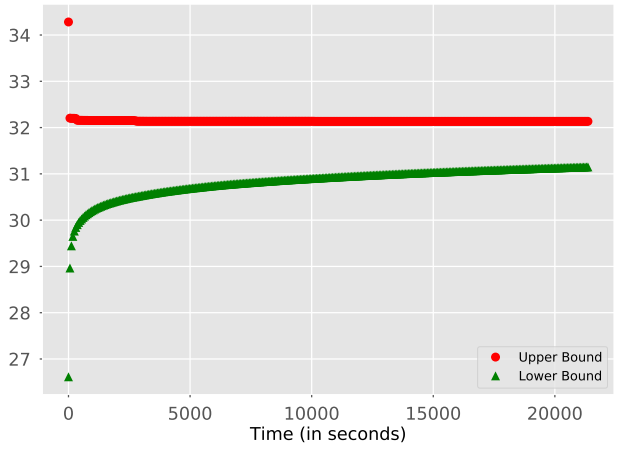


Figure 2.10: The progress of upper bound versus lower bound in the branch-and-bound tree for the LN formulation with $\lambda = 0.1$ for complete super-structure $\mathcal{M}$, for the Insurance dataset.

Chapter 3

STRENGTHENED MIXED-INTEGER QUADRATIC PROGRAMMING

In this chapter, we present another MIQP formulation for leaning DAG from continuous observational data. This chapter focuses on tightening the big-M constraints corresponding to the arc weights in the DAG structure.

3.1 Relevant Work

Let $F(\beta, g) = \frac{1}{2}\sum_{k\in V}\sum_{d\in\mathcal{D}}\left(x_{dk} - \sum_{(j,k)\in\overrightarrow{E}}\beta_{jk}x_{dj}\right)^2 + \frac{\mu}{2}\sum_{(j,k)\in\overrightarrow{E}}\beta_{jk}^2 + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}}g_{jk}$. The PNL$\mathcal{M}$ can be cast as the following optimization problem:

$$\begin{aligned}
\min \quad & F(\beta, g), && && \text{(3.1a)}\\
& \mathcal{G}(B) \text{ induces a DAG from } \overrightarrow{\mathcal{M}}, && && \text{(3.1b)}\\
& \beta_{jk}(1 - g_{jk}) = 0, && \forall (j,k) \in \overrightarrow{E}, && \text{(3.1c)}\\
& g_{jk} \in \{0, 1\}, && \forall (j,k) \in \overrightarrow{E}. && \text{(3.1d)}
\end{aligned}$$

The objective function (3.1a) is an expanded version of $l(\beta; \mathcal{X})$ in PNL$\mathcal{M}$ where we used the indicator variable g_{jk} to encode the ℓ_0 regularization. The constraints in (3.1b) rule out cycles. The constraints in (3.1c) are non-linear and stipulate that $\beta_{jk} \neq 0$ only if $g_{jk} = 1$.

There are two sources of difficulty in solving (3.1a)-(3.1d): (i) the acyclic nature of DAG imposed by the constraint in (3.1b), (ii) the set of *non-linear* constraints in (3.1c) encoding the semi-continuous variable β_{jk} which stipulates that $\beta_{jk} \neq 0$ only if there exists an arc (j, k) in $\mathcal{G}(B)$. In the previous chapter, we discussed existing formulations to address the former. Next, we present relevant studies for the latter.

3.1.1 Convex encodings of non-convex constraints (3.1c)

The nonconvexity of the set of constraints (3.1c) causes challenges in obtaining provably optimal solutions with existing optimization packages. Therefore, we consider convex representations of this set of constraints. First, we consider a linear representation of the constraints in (3.1c). Although the formulations discussed in Chapter 1 differ in their approach to rule out cycles, one major commonality among them is that they replace the non-linear constraint (3.1c) by the so called big-M constraints given by

$$-Mg_{jk} \leq \beta_{jk} \leq Mg_{jk}, \forall (j,k) \in \overrightarrow{E}, \tag{3.2}$$

for a large enough M. Unfortunately, these big-M constraints (3.2) are poor approximations of (3.1c), especially in this problem, because no natural and tight value for M exist. It is worth noting that there are a few techniques to obtain big-M parameter for sparse regression problem [8, 34]. However, these big-M parameters are often too large in practice. Further, finding a tight big-M parameter itself is a difficult problem to solve for DAG structure learning.

3.1.2 Quadratic optimization with semi-continuous variables

Problem (3.1) is a *mixed-integer convex quadratic optimization* (MICQP) problem with semi-continuous variables, a class of problems which has received a fair amount of attention from the operations research community over the last decade e.g., [29, 30, 31, 32, 34]. There has also been recent interest in leveraging these developments to solve the sparse regression with ℓ_0 regularization, see e.g., [60, 69, 27, 75, 3].

We herein review applications of MIQPs with semi-continuous variables for solving sparse regression with ℓ_0 regularization. Frangioni et al. [32] develop a so-called projected perspective relaxation method, to solve the perspective relaxation of mixed-integer nonlinear programming problems with a convex objective function and semi-continuous variables. This reformulation requires that the corresponding binary variables are not involved in other constraints. Therefore, it is suitable for ℓ_0 sparse regression whereas it cannot be applied for

DAG structure learning problem. Pilanci et al. [60] show how a broad class of ℓ_0-regularized problems, including sparse regression as a special case, can be formulated exactly as optimization problems. The authors use the Tikhonov regularization term $\frac{\mu}{2}||\beta||_2^2$ and convex analysis to construct an improved convex relaxation using the reverse Huber penalty. In a similar vein, Bertsimas and Van Parys [8] exploit the Tikhonov regularization and develop an efficient algorithm by reformulating the sparse regression mathematical formulation as a saddle-point optimization problem with an outer linear integer optimization problem and an inner dual quadratic optimization which is capable of solving high-dimensional sparse regression. Xie and Deng [75] apply the perspective formulation to the Tikhonov regularization $\frac{\mu}{2}||\beta||_2^2$ of sparse regression optimization problem with ℓ_0 regularization. The authors establish that the relaxation of perspective formulation is equivalent to the continuous relaxation of the formulation given by Bertsimas and Van Parys [8]. Dong et al. [27] propose perspective relaxation for ℓ_0 sparse regression optimization formulation and establish that the popular sparsity-inducing concave penalty function known as the reverse Huber penalty and the minimax concave penalty [79] [60] can be obtained as special cases of the perspective relaxation– thus the relaxations of formulations by Zhang et al. [79], Pilanci et al. [60], Bertsimas and Van Parys [8], Xie and Deng [75] are equivalent. The authors further pursue to obtain an optimal perspective relaxation that is no weaker than any perspective relaxation. Among the related approaches, the optimal perspective relaxation by Dong et al. [27] is the only one that does not explicitly require the use of the Tikhonov regularization, i.e., $\frac{\mu}{2}||\beta||_2^2$.

The perspective formulation, which in essence is a fractional non-linear program, can be cast either as a mixed integer second-order cone program (MISOCP) or a semi-infinite mixed integer linear programming (SIMILP). Both formulations can directly be solved by the state-of-the-art optimization packages. Nevertheless, directly solving this problem has not been assessed for sparse regression. Instead, authors solve the continuous relaxation and then use a heuristic approach (e.g., rounding techniques) to obtain an upper bound, e.g., see [3, 27]. In this paper, we directly solve the MISOCP and SIMILP formulations of sparse DAG structural learning.

3.1.3 Contributions

The main contributions of this paper are twofold.

- In spite of recent progress, a key challenge in DAG learning corresponding to linear SEMs is enforcing bounds on the arc weights. This is commonly modeled using the standard "big-M constraint" approach [47, 53]. As shown by Manzour et al. [47], this strategy leads to poor continuous relaxations for this problem, which in turn results in slow lower bound improvement in the branch-and-bound tree. In particular, Manzour et al. [47] establish that all existing big-M formulations achieve the same continuous relaxation objective function under a mild condition (see Propositions 5 and 6). To overcome this issue, we present a mixed-integer second-order cone program (MISOCP) which gives a provably tighter continuous relaxation than existing big-M formulations. This formulation can be solved by powerful state-of-the-art optimization packages. Our numerical results show the superior performance of MISOCP compared to the existing big-M formulations in terms of improving the lower bound and reducing the optimality gap.

- The statistical properties of *optimal* PNL with ℓ_0 regularization have been studied extensively [45, 72]. However, it is often difficult to prove optimality of a solution, and there are no results which establish the statistical properties of an approximate solution. In this paper, we give an early stopping criterion under which the branch-and-bound process can be terminated early. We establish that the obtained solution asymptotically converges to the true coefficients in DAG with high probability under proper conditions. Our result leverages the statistical consistency results of the PNL estimate with ℓ_0 regularization along with the unique structure of branch-and-bound wherein both lower and upper bound values on the objective function are available [58, 72].

3.2 A New Mathematical Formulation: The Mixed-integer Conic Program

In this section, we discuss mathematical formulations for learning DAG structure of a BN.

3.2.1 Big-M formulation

Consider (2.1a)-(2.1d) by substituting (2.1b) by the linear big-M constraints (3.2) and writing the objective function in a matrix form. We denote the resulting formulation by MIQP.

$$\textbf{MIQP} \quad \min \quad \frac{1}{2}\text{Tr}[(I-B)(I-B)^\top \mathcal{X}^\top \mathcal{X}] + \frac{\mu}{2}\text{Tr}[BB^\top] + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk} \tag{3.3a}$$

$$(2.1c),$$

$$-Mg_{jk} \leq \beta_{jk} \leq Mg_{jk} \quad \forall (j,k) \in \overrightarrow{E}, \tag{3.3b}$$

$$g_{jk} \in \{0,1\} \quad \forall (j,k) \in \overrightarrow{E}. \tag{3.3c}$$

Depending on which types of constraints are used instead of (2.1c) to remove cycles, as explained the previous chapter, MIQP results in different formulations: MIQP+LN, MIQP+TO, and MIQP+CP.

Proposition 7, adapted from Dong et al. [27] for DAG structure learning problem, shows that ℓ_1-regularization (i.e., $\|\beta\|_1 = \sum_{(j,k)\in\overrightarrow{E}} |\beta_{jk}|$) is a special continuous relaxation of MIQP. This motivates us to consider tighter continuous relaxation for MIQP.

Let $(\beta^\star, g^\star)$ be the optimal solution to the continuous relaxation of MIQP.

Proposition 7 *For $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$, a continuous relaxation of MIQP, where the binary variables are relaxed, is equivalent to ℓ_1-regularization with penalty parameter $\hat{\lambda} = \frac{\lambda}{2M}$.*

Proof. For $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$, the value $g^\star_{jk}$ has to be $\frac{\beta^\star_{jk}}{M}$. Otherwise, we can reduce the value of the decision variable g without violating any constraints while reducing the objective function. Note that since $M \geq \max_{(j,k)\in\overrightarrow{E}} |\beta^\star_{jk}|$, we have $\frac{g^\star_{jk}}{M} \leq 1, \forall (j,k) \in \overrightarrow{E}$. Thus, the set of constraints in (2.1c) is satisfied. To show this, we consider the set of CP constraints. In this case, the set of constraints (2.1c), i.e., $\sum_{(j,k)\in\mathcal{C}_A} \frac{\beta^\star_{jk}}{M} \leq |\mathcal{C}_A| - 1, \quad \forall \mathcal{C}_A \in \mathcal{C}$ is because

$M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^{\star}_{jk}|$. This implies that $\frac{g^{\star}_{jk}}{M}$ is the optimal solution. Thus, the objective function reduces to ℓ_1 regularization with the coefficient $\frac{\lambda}{2M}$.

Proposition 5 establishes that for $M \geq 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^{\star}_{jk}|$, the objective function value of the continuous relaxation of CP, LN and TO is equivalent. This implies that the continuous relaxations of all formulations are equivalent. This completes the proof. □

3.2.2 Perspective formulation

In this section, we present how perspective formulation can be suitably applied for DAG structure learning with ℓ_0 regularization. We further cast the problem in the form of a MISOCP and SIMILP.

We now present a derivation of the perspective formulation for MIQP. To this end, we express the objective function (3.3a) in the following way:

$$\frac{1}{2}\mathrm{Tr}[(I-B)(I-B)^{\top}\mathcal{X}^{\top}\mathcal{X}] + \frac{\mu}{2}\mathrm{Tr}[BB^{\top}] + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk} \tag{3.4a}$$

$$= \frac{1}{2}\mathrm{Tr}[(I-B-B^{\top})\mathcal{X}^{\top}\mathcal{X} + BB^{\top}(\mathcal{X}^{\top}\mathcal{X}+\mu I)] + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk}. \tag{3.4b}$$

Let $\delta \in \mathbb{R}^m_+$ be a vector and $\mu \geq 0$ a scalar such that $\mathcal{X}^{\top}\mathcal{X} + \mu I - D_{\delta} \succeq 0$, where $D_{\delta} = \mathrm{diag}(\delta_1, \ldots, \delta_m)$ and $A \succeq 0$ denotes that matrix A is positive semi-definite. By splitting the quadratic term $\mathcal{X}^{\top}\mathcal{X} + \mu I = (\mathcal{X}^{\top}\mathcal{X} + \mu I - D_{\delta}) + D_{\delta}$ in (3.4b), the objective function can be expressed as

$$\frac{1}{2}\mathrm{Tr}[(I-B-B^{\top})\mathcal{X}^{\top}\mathcal{X} + BB^{\top}(\mathcal{X}^{\top}\mathcal{X}+\mu I - D_{\delta})] + \frac{1}{2}\mathrm{Tr}(BB^{\top}D_{\delta}) + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk}. \tag{3.5}$$

Let $Q = \mathcal{X}^{\top}\mathcal{X} + \mu I - D_{\delta}$. Then, Cholesky decomposition can be applied to decompose Q as $q^{\top}q$ (note $Q \succeq 0$). As a result, $\mathrm{Tr}(BB^{\top}Q) = \mathrm{Tr}(BB^{\top}q^{\top}q) = \sum_{i=1}^{m}\sum_{j=1}^{m}(\sum_{(\ell,j)\in\overrightarrow{E}} \beta_{\ell j}q_{i\ell})^2$. The separable component can also be expressed as $\mathrm{Tr}(BB^{\top}D_{\delta}) = \sum_{j=1}^{m}\sum_{(j,k)\in\overrightarrow{E}} \delta_j\beta^2_{jk}$. Using

this notation, the objective (3.5) can be written as

$$\frac{1}{2}\mathrm{Tr}\Big[(I-B-B^\top)\mathcal{X}^\top\mathcal{X}+BB^\top Q\Big]+\frac{1}{2}\sum_{j=1}^{m}\sum_{(j,k)\in\overrightarrow{E}}\delta_j\beta_{jk}^2+\frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}}g_{jk}.$$

The Perspective Reformulation (PRef) of MIQP is then given by

$$\begin{aligned}
\textbf{PRef}\quad \min\quad & \frac{1}{2}\mathrm{Tr}\Big[(I-B-B^\top)\mathcal{X}^\top\mathcal{X}+BB^\top Q\Big]+ && \text{(3.6a)}\\
& \frac{1}{2}\sum_{j=1}^{m}\sum_{(j,k)\in\overrightarrow{E}}\delta_j\frac{\beta_{jk}^2}{g_{jk}}+\frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}}g_{jk}, &&\\
& (2.1c), (3.3b)-(3.3c). && \text{(3.6b)}
\end{aligned}$$

We remark that the objective function (3.6a) is formally undefined when some $g_{jk}=0$. More precisely, we use the convention that $\frac{\beta_{jk}^2}{g_{jk}}=0$ when $\beta_{jk}=g_{jk}=0$ and $\frac{\beta_{jk}^2}{g_{jk}}=+\infty$ when $\beta_{jk}\neq 0$ and $g_{jk}=0$, see [31]. The continuous relaxation of PRef, referred to as the Perspective Relaxation (PRel), is much stronger than the continuous relaxation of MIQP, see [60]. However, an issue with PRef is that the objective function is nonlinear due to the fractional term. There are two ways to reformulate PRef. One as a mixed-integer second-order conic program (MISOCP), and the other as a semi-infinite mixed-integer linear program (SIMILP).

Mixed integer second-order conic program

We introduce additional variables s_{jk} to represent β_{jk}^2. Then, the MISOCP formulation is given by

$$\textbf{MISOCP} \quad \min \quad \frac{1}{2}\text{Tr}\Big[(I - B - B^\top)\mathcal{X}^\top\mathcal{X} + BB^\top Q\Big] + \tag{3.7a}$$
$$\frac{1}{2}\sum_{j=1}^{m}\sum_{(j,k)\in\overrightarrow{E}}\delta_j s_{jk} + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk},$$
$$s_{jk}g_{jk} \geq \beta_{jk}^2 \quad (j,k) \in \overrightarrow{E}, \tag{3.7b}$$
$$0 \leq s_{jk} \leq M^2 g_{jk} \quad (j,k) \in \overrightarrow{E}, \tag{3.7c}$$
$$(2.1c), (3.3b) - (3.3c).$$

where the second-order conic constraints (3.7b) imply that $\beta_{jk} \neq 0$ only when $z_{jk} = 1$. The constraints in (3.7b) are conic representable because they can be written in the form of $\sqrt{4\beta_{jk}^2 + (s_{jk} - g_{jk})^2} \leq s_{jk} + g_{jk}$. The set of constraints (3.7c) is valid since $\beta_{jk} \leq M g_{jk}$ implies $\beta_{jk}^2 \leq M^2 g_{jk}^2 = M^2 g_{jk}^2$ and $g_{jk}^2 = g_{jk}$ for $g \in \{0,1\}$. The set of constraints in (3.7c) is not required, yet they improve the computational efficiency especially when we restrict the big-M value. Xie and Deng [75] report similar behavior for sparse regression. When we relax $g \in \{0,1\}$ and let it be $[0,1]$, we obtain the continuous relaxation of MISOCP. Let us denote the feasible region of continuous relaxation of MISOCP and MIQP by $\mathcal{R}$MISOCP and $\mathcal{R}$MIQP, and the objective function values by OFV($\mathcal{R}$MISOCP) and OFV($\mathcal{R}$MIQP), respectively. For a more general problem than ours, Cui et al. [20] give a detailed proof establishing that the feasible region of the former is contained in the feasible region of latter i.e., $\mathcal{R}\text{MISOCP} \subset \mathcal{R}MIQP$. This implies that OFV($\mathcal{R}$MISOCP) $\gneqq$ OFV($\mathcal{R}$MIQP) where OFV denotes the objective function value..

Semi-infinite mixed-integer integer linear program

An alternative approach to reformulate PRef is via *perspective cuts* developed by Frangioni and Gentile [29, 30]. To apply perspective cuts properly, we use the reformulation idea first

proposed in [29] by introducing dummy decision matrix D to distinguish the separable and non-separable part of the objective function and add the additional constraint $d = \beta$ where d_{jk} is (j,k) element of matrix D and β is the decision variable in the optimization problem.

Applying perspective cuts, the problem MIQP can be reformulated as a SIMILP as

$$\begin{aligned}
\textbf{MISILP} \quad \min \quad & \frac{1}{2}\mathrm{Tr}\Big[(I - B - B^\top)\mathcal{X}^\top\mathcal{X} + DD^\top Q\Big] + \\
& \frac{1}{2}\sum_{j=1}^{m}\sum_{(j,k)\in\overrightarrow{E}}\delta_j v_{jk} + \frac{\lambda}{2}\sum_{(j,k)\in\overrightarrow{E}} g_{jk}, && \text{(3.8a)}\\
& d_{jk} = \beta_{jk} \quad (j,k) \in \overrightarrow{E}, && \text{(3.8b)}\\
& v_{jk} \geq 2\bar{\beta}_{jk}\beta_{jk} - \bar{\beta}_{jk}^2 g_{jk} \quad \forall \bar{\beta}_{jk} \in [-M, M] \quad \forall (j,k) \in \overrightarrow{E}, && \text{(3.8c)}\\
& \text{(3.3b)} - \text{(3.3c)}, && \text{(3.8d)}\\
& v_{jk} \geq 0, \quad (j,k) \in \overrightarrow{E}. && \text{(3.8e)}
\end{aligned}$$

The set of constraints in (3.8c) are known as perspective cuts. Note that there are infinitely many such constraints. Although this problem cannot be solved directly, it is suitable for a delayed cut generation approach whereby a (small) finite subset of (3.8c) cuts are maintained, the current solution $(\beta^\star, g^\star, v^\star)$ of the relaxation is obtained, and all the violated inequalities for the relaxation solution are added for $\bar{\beta}_{jk} = \frac{\beta^\star_{jk}}{g^\star_{jk}}$ (assuming $\frac{0}{0} = 0$) are added. This process is repeated until termination criteria are met. This procedure can be implemented using `Callback` function available by off-the-shelf solvers such as Gurobi or CPLEX.

3.2.3 Selecting δ

In MISOCP and MISILP, one important question is how to identify a valid δ. A natural choice is $\mathrm{diag}(\delta) = (\lambda_{\min} - \epsilon)e$ where λ_{min} is the minimum eigenvalue value of $\mathcal{X}^\top\mathcal{X}$, $\epsilon > 0$ is a sufficiently small number to avoid to numerical instability of estimating eigenvalues, and e is the all-one column vector. The issue with this approach is that if $\lambda_{\min} = 0$, then $\mathrm{diag}(\delta)$ becomes a trivial 0 matrix. Frangioni and Gentile [30] present an effective way to obtain a

valid δ by solving the following semidefinite program (SDP) as

$$\{\max \sum_{i \in V} \delta_i | \mathcal{X}^\top \mathcal{X} + \mu I - diag(\delta) \succeq 0, \delta_i \geq 0\}. \tag{3.9a}$$

The above formulation can attain a non-zero D_δ even if $\lambda_{\min} = 0$. Numerical results by Frangioni and Gentile [30] show that this method favorably compares with the minimum eigenvalue approach. Zheng et al. [80] propose an SDP approach which obtains D_δ such that the continuous relaxation corresponding to the SOCP formulation of perspective formulation is as tight as possible.

Similar to Dong et al. [27], our formulation does not require adding a Tikhonov regularization. However, if $\mu = 0$ then PRef is effective when $\mathcal{X}^\top \mathcal{X}$ is sufficiently diagonally dominant. Note even if the minimum eigenvalue is zero, it does not imply that diag(δ) is a zero matrix because (3.9a) could still give a solution which is a non-zero matrix. If diag(δ) turns out to be a zero matrix, then MISOCP formulation reduces to the big-M formulation. When $n \geq m$ and each row of $\mathcal{X}$ is independent, then $\mathcal{X}^\top \mathcal{X}$ is guaranteed to be a positive semi-definite matrix [27]. On the other hand, when $n < m$, $\mathcal{X}^\top \mathcal{X}$ is not full-rank. Therefore, a sufficiently large μ should be provided to make $\mathcal{X}^\top \mathcal{X} + \mu I \succeq 0$.

From a computational standpoint, we expect that for larger μ values the performance of MISOCP improves because the set of conic constraints (3.7b) tightens the formulation and increases the term $\frac{1}{2} \sum_{j=1}^{m} \sum_{(j,k) \in \overrightarrow{E}} \delta_j \frac{\beta_{jk}^2}{g_{jk}}$ in the objective function of $\mathcal{R}$MISOCP. Our numerical results demonstrate this behavior clearly.

3.3 Exact recovery of DAG Structure

In this section, we first briefly review the exact recovery in sparse regression.

3.3.1 Exact recovery for sparse regression

Given a random $\{y_i, x_{i1}, \ldots, x_{im}\}_{i=1}^n$, of n statistical units, a linear regression model assumes that the relationship between the n dependent variables $y = \{y_i\}_{i=1}^n$ and the data matrix $\mathcal{X} = \{x_{i1}, \ldots, x_{im}\}_{i=1}^n$ in the form $y = \mathcal{X}^\top \beta^0 + \epsilon$, where $\epsilon \in \mathbb{R}^n$ is a noise vector with Gaussian

distribution, $\epsilon = \mathcal{N}(0, \sigma^2 I_{n\times n})$. Given the pair $(y, \mathcal{X})$, our aim is to estimate the unknown regression vector $\beta^0 \in \mathbb{R}^m$.

In a regime where $m \gg n$, it is straightforward to see that without further constraints on β^0, the statistical model $y = \mathcal{X}\beta^0 + \delta$ is not identifiable i.e., there exist many vectors $\beta^\star$ that are consistent with the observations y and $\mathcal{X}$ even in a noiseless setting where $\delta = 0$.

In the following, we review the conditions required for exact recovery of β^0 in sparse linear regression. Interested reader can refer to [13, 17, 62] for further details.

Define the set $\mathcal{C}(S;\alpha) := \{\tau \in \mathbb{R}^m \mid \|\tau_S^c\|_1 \leq \alpha\|\tau_S\|_1\}$ for a given subset $S \subset \{1, ..., m\}$ and constant $\alpha \geq 1$. For a given sparsity index $k \leq m$, the matrix $\mathcal{X}$ satisfies the *restricted nullspace (RN) condition of order* k if $\text{null}(\mathcal{X}) \cap \mathcal{C}(S;1) = \{0\}$ for all subsets S of cardinality k, see [17, 62].

In the noiseless setting, the basis pursuit estimate [13, 62] exactly recovers any vector k-sparse vector β^0 if and only if $\mathcal{X}$ meets the restricted nullspace property of order k.

In the noisy setting, exact recovery of β^0 is not possible. Instead, a more natural criterion is to bound the ℓ_2-error between β^0 and an estimate $\beta^\star$ i.e., $\|\beta^\star - \beta^0\|_2^2$. The least restrictive condition to establish such a bound is the restricted eigenvalue (RE) condition [10]. Raskutti et al. [62] propose an equivalent condition to the RE condition which is describe in the next definition [10].

Definition. The $m \times m$ sample covariance matrix $\frac{\mathcal{X}^\top\mathcal{X}}{n}$ satisfies the restricted eigenvalue (RE) condition over S with parameters $(\alpha, \gamma) \in [1, \infty) \times [0, \infty)$ if we have

$$\frac{1}{n}\theta^\top\mathcal{X}^\top\mathcal{X}\theta = \frac{1}{n}\|\mathcal{X}\theta\|_2^2 \geq \gamma^2\|\theta\|_2^2, \quad \forall\theta \in \mathcal{C}(S;\alpha).$$

If this condition is true for all subsets S with cardinality k, then $\frac{\mathcal{X}^\top\mathcal{X}}{n}$ satisfies a *restricted eigenvalue condition of order* k with parameters (α, γ).

The RE condition is essential because it provides guarantees on the ℓ_2-error of any lasso estimate of $\beta^\star$. Raskutti et al. [62] present the following result which we use in the next section to establish our main result.

Proposition 8 *(Raskutti et al. [62]) Suppose that* $\frac{\mathcal{X}^\top \mathcal{X}}{n}$ *meets the RE condition of order* k *with parameters* (α, γ)*. Then, for any Gaussian random variable design* $\mathcal{X} \in \mathbb{R}^{n \times m}$ *with i.i.d* $\mathcal{N}(0, \Sigma)$ *rows, there are universal positive constants* c, c' *such that*

$$\frac{\|\mathcal{X}v\|_2^2}{n} \geq \left\{ \frac{\gamma}{4} - 9(1+\alpha)\rho(\Sigma)\sqrt{\frac{k \ln(m)}{n}} \right\} \|v\|_2^2, \quad \forall v \in \mathbb{R}^m \tag{3.10}$$

with probability at least $1 - c'e^{-cn}$ where $\rho^2(\Sigma) = \max_{\{j=1,\ldots,m\}} \Sigma_{jj}$.

3.3.2 Statement of main result

Despite the desirable properties of ℓ_1 regularization in high-dimensional sparse regression [12, 62], many of these properties disappear in DAG structural learning [1, 33]. Peters and Bühlmann [58] and van de Geer and Bühlmann [72] study the properties of an optimal solution to PNL with an ℓ_0 regularization norm penalty, i.e., $\beta^\star$. They establish that ℓ_2-error bound, $\|\beta^0 - \beta^\star\|_2^2 \asymp O(\frac{\ln(n)}{n} s_0)$ where s_0 is the number of arcs in the true DAG.

In this section, we establish conditions to ensure the asymptotic consistency of an approximate solution, $\hat{\beta}$, to the true values of parameters, β^0, i.e., $\|\hat{\beta} - \beta^0\| \asymp O(\frac{\ln(n)}{n} s_0)$. This result is obtained by leveraging an important property of branch-and-bound process for integer programming that provides both lower and upper bounds on the objective function upon early stopping as well as the consistency results of PNL estimate with ℓ_0 regularization.

Let LB and UB denote the lower and upper bound obtained from solving (1.6) with $\mu = 0$ under an early stopping criterion (i.e., the obtained solution is not necessarily optimal), respectively. We define the absolute difference between upper bound and lower bound as GAP i.e., $GAP = |UB - LB|$. Let $\hat{\mathcal{G}}, \hat{\beta}$ denote the structure of DAG and coefficients of arcs from optimization model (1.6) under this early stopping condition with sample size n and regularization parameter λ. Let $\mathcal{G}^\star, \beta^\star$ denote the DAG structure and coefficients of arcs obtained from the optimal solution of (1.6) without Tikhonov regularization ($\mu = 0$), and $\mathcal{G}^0, \beta^0$ represent the DAG structure and coefficient of arcs corresponding to the true DAG. We denote the number of arcs in $\hat{\mathcal{G}}$, $\mathcal{G}^0$, and $\hat{G}$ by $\hat{s}$, s_0, and $\hat{s}$, respectively. The score

value in (1.6a) of each solution is denoted by $Score(\phi)$ where $\phi \in \{\beta^\star, \hat{\beta}, \beta^0\}$. We first state Theorem 5.1 in [72] in the next proposition.

Proposition 9 *(Theorem 5.1 by van de Geer and Bühlmann [72]) Assume conditions 3.1 and 3.2 and conditions 5.1 and 5.2 in [72]. Let* $\alpha_0 := \min\{\frac{4}{m}, 0.05\}$. *Then for* $\lambda = O(\frac{\ln(n)}{n} s_0)$, *it holds with probability at least* $1 - \alpha_0$, *that*

$$\|\beta^\star - \beta^0\|_2^2 + \lambda s^\star = O(\lambda s_0).$$

Next, we present our main result.

Proposition 10 *Suppose that* $\frac{\mathcal{X}^\top \mathcal{X}}{n}$ *satisfies the RE condition of order* k *with parameters* (α, γ) *and conditions 3.1 and 3.2 and conditions 5.1 and 5.2 in [72] hold. Let* $\alpha_0 = \min\{\frac{4}{m}, 0.05\}$. *Then, for any Gaussian random variable design* $\mathcal{X} \in \mathbb{R}^{n \times m}$ *with i.i.d* $\mathcal{N}(0, \Sigma)$ *rows and* $\lambda = \frac{\ln(n)}{n}$, *the branch-and-bound process can be terminated early with stopping criterion* $GAP \asymp O(\frac{\ln(n)}{n} s_0)$. *This results* $\|\hat{\beta} - \beta^0\|_2^2 \asymp O(\frac{\ln(n)}{n} s_0)$ *with probability* $\min\{1 - \alpha_0, 1 - c' e^{-cn}\}$.

Proof.

Let $\sigma_{\max}(\mathcal{X})$ be the maximal singular value and $\lambda_{\max}(\mathcal{X}^T \mathcal{X})$ be the maximal eigenvalue, and $\lambda_1 c_1 = \lambda$. Then

$$\begin{aligned}
\|\hat{\beta} - \beta^0\|_2^2 &\le c_1 \|\mathcal{X}(\hat{\beta} - \beta^0)\|_2^2 \le c_1 \|\mathcal{X}(\hat{\beta} - \beta^0)\|_2^2 + \lambda \hat{s} && \text{(3.11a)} \\
&= c_1 \|\mathcal{X}(\hat{\beta} - \beta^\star + \beta^\star - \beta^0)\|_2^2 + \lambda \hat{s} - \lambda s^\star + \lambda s^\star \\
&\le c_1 \|\mathcal{X}(\hat{\beta} - \beta^\star)\|_2^2 + c_1 \|\mathcal{X}(\beta^\star - \beta^0)\|_2^2 + \lambda(\hat{s} - s^\star) + \lambda s^\star \\
&= c_1 \|\mathcal{X}(\hat{\beta} - \beta^\star)\|_2^2 + \lambda(\hat{s} - s^\star) + c_1 \|\mathcal{X}(\beta^\star - \beta^0)\|_2^2 + \lambda s^\star \\
&= c_1 \|\mathcal{X}(\hat{\beta} - \beta^\star)\|_2^2 + \lambda_1 c_1(\hat{s} - s^\star) + c_1 \|\mathcal{X}(\beta^\star - \beta^0)\|_2^2 + \lambda s^\star \\
&\le c_1 |Score(\hat{\beta}) - Score(\beta^\star)| + c_1 \|\mathcal{X}(\beta^\star - \beta^0)\|_2^2 + \lambda s^\star \\
&\le c_1 |Score(\hat{\beta}) - Score(\beta^\star)| + \underbrace{c_1 \sigma_{max}(\mathcal{X}) \|\beta^\star - \beta^0\|_2^2 + \lambda s^\star}_{O(\lambda s_0)} && \text{(3.11b)} \\
&\le c_1 GAP + O(\lambda s_0). && \text{(3.11c)}
\end{aligned}$$

Therefore, $GAP \asymp O(\lambda s^0)$ implies $\|\hat{\beta} - \beta^0\|_2^2 \asymp O(\frac{\ln(n)}{n} s_0)$ □

Inequality (3.11a) holds because by assumption $\frac{1}{n}\mathcal{X}^\top\mathcal{X}$ satisfies the RE of order k with parameters (α, γ). Therefore, the inequality (3.11a) holds by substituting $\beta^\star - \beta^0$ in (3.10). In this case, $c_1 = n^{-1}\left\{\frac{\gamma}{4} - 9(1+\alpha)\rho(\Sigma)\sqrt{\frac{k\ln(m)}{n}}\right\}^{-1}$. Note if we are not in the $m \gg n$ regime, then we can define $c_1 = \|(\mathcal{X}^\top\mathcal{X})^{-1}\mathcal{X}^\top\|_2^2$. The inequality (3.11b) holds because for a matrix $\mathcal{X}$ and vector B, we have $\|\mathcal{X}B\|_2^2 \leq \|\mathcal{X}\|\|B\|_2^2$ where $\|\mathcal{X}\|$ denotes the spectral norm of $\mathcal{X}$ and we have $\|\mathcal{X}\| = \sigma_{\max}(\mathcal{X}) = \sqrt{\lambda_{\max}(\mathcal{X}^T\mathcal{X})}$. The second term in (3.11b) is because of the result by (3.3.2). The inequality (3.11c) holds because $Score(\hat{\beta})$ and $Score(\beta^\star)$ lie between LB and UB based on the definition of GAP.

The value for s_0 is not known; however, one can find an upper bound by using $s_0 \leq 2s_m$ where s_m denotes the number of edges in the moral graph.

3.4 Experiments

In this section, we report numerical results comparing the MIQP, MISOCP, and SIMILP formulations. Experiments are performed on a cluster operating on UNIX with Intel Xeon E5-2640v4 2.4GHz. All formulations are implemented in the Python programming language. Gurobi 8.1 is used as the solver. Unless otherwise stated, a time limit of $50m$ (in seconds), where m denotes the number of nodes, and an MIQP optimality gap of 0.01 are imposed across all experiments after which runs are terminated. The optimality gap is calculated by $\frac{UB(X)-LB(X)}{UB(X)}$ where UB(X) denotes the objective value associated with the best feasible integer solution (incumbent) and LB(X) represents the best obtained lower bound during the branch-and-bound process for the formulation $X \in \{\text{MIQP}, \text{SIMILP}, \text{MISOCP}\}$.

We assume $\lambda = \ln(n)$ which corresponds to the BIC score and let $\mu = 0$ unless otherwise stated. The classical PNL$\mathcal{M}$ for DAG structural learning assumes $\mu = 0$ [2, 53, 65, 72]. However, sparse linear regression has been studied with $\mu \geq 0$ [3, 8, 27]. Due to computational instability, if the minimum eigenvalue of $\mathcal{X}^\top\mathcal{X}$ is a negligible negative number, one can set μ as the absolute value of the smallest eigenvalue of $\mathcal{X}^\top\mathcal{X}$ to ensure $\mathcal{X}^\top\mathcal{X} + \mu I$ is

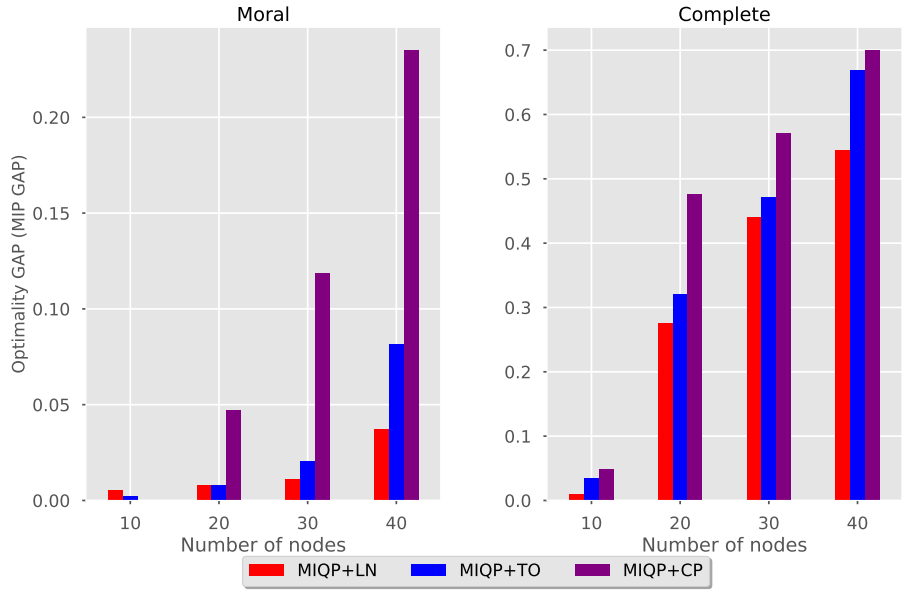
a positive semi-definite matrix. In our instances, the minimum eigenvalue of $\mathcal{X}^\top\mathcal{X}$ across all instances is 3.26 and the maximum eigenvalue is 14.21. To select the big-M parameter, M, in all formulations we use the proposal of Park and Klabjan [53]. Specifically, given λ, we solve each problem without cycle prevention constraints and obtain β^0. We then use the upper bound $M = 2 \max_{(j,k)\in\vec{E}} |\beta^0_{jk}|$. Although this value does not guarantee an upper bound for M, the results provided in [53, 47] computationally confirm that this approach gives a large enough value of M.

We use the following MIQP-based metrics to measure the quality of a solution: optimality gap (MIQP GAP), computation time in seconds (Time), Upper Bound (UB), Lower Bound (LB), objective function value (OFV) of the continuous relaxation, and the number of explored nodes in the branch-and-bound tree (# BB)

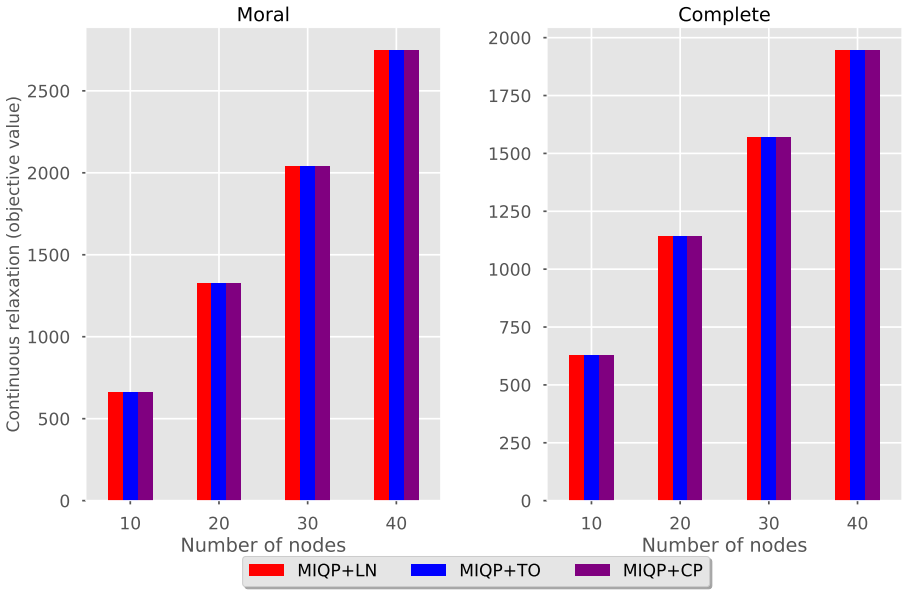
3.4.1 Comparison of MIQP formulations

We first test the MIQP formulations on synthetic dataset as explained in 2.4.1. For the set of constraints (2.1c), we use LN, TO, and CP constraints discussed in the previous chapter resulting in three formulations denoted as MIQP+LN, MIQP+TO, MIQP+CP, respectively. In this setting, the objective function is not normalized (i.e., we multiplied the objective with n). Figure 3.1a demonstrates that MIQP+LN outperforms other formulations in terms of optimality gap. Figure 3.1b shows that all MIQP formulations attain the same continuous relaxation objective function value (see, Proposition 5).

The main reason for the promising performance of MIQP+LN formulation can be attributed to three factors: (1) MIQP+LN has fewer binary variables and constraints than MIQP+TO, (2) MIQP+LN is a compact (polynomial-sized) formulation in contrast to MIQP+CP which has an exponential number of constraints.



(a) Optimality GAPs



(b) Time (in seconds)

Figure 3.1: Optimization-based measures of MIQP formulations with $n = 100$, $\mu = 0$, and $\lambda = \ln(n)$.

3.4.2 Comparison of MISOCP formulations

We next experiment with MISOCP formulations. For the set of constraints in (2.1c), we use LN, TO, and CP constraints discussed in previous chapter resulting in three formulations denoted as MISOCP+LN, MISOCP+TO, MISOCP+CP, respectively. MISOCP+TO fails to find a feasible solution for several complete instances with the number of nodes 30 and 40. For sparse instances, the optimality gap for MISOCP+TO are 0.000 and 0.021 for the number of nodes 10 and 20 respectively; for complete instances, the optimality gap for MISOCP+TO formulation are 0.009 and 0.272 for the number of nodes 10 and 20, respectively. The MISCOP+TO formulation fails to find a feasible solution for instances with 30 and 40 nodes, see Table 3.1. Moreover, Table 3.1 illustrates that MISOCP+LN performs better that MISOCP+TO for even small instances (i.e., 10 and 20 nodes).

Table 3.1: Optimality gaps between MISOCP+TO and MISOCP+LN formulations

	Sparse		Complete	
m	MISOCP + TO	MISOCP + LN	MISOCP + TO	MISOCP + LN
10	0.000	0.000	0.009	0.008
20	0.021	0.006	0.272	0.195
30	-	0.010	-	0.195
40	-	0.042	-	0.436

"-" denotes that no feasible solution, i.e., UB, obtained, so optimality gap cannot be computed.

For MISOCP+CP, 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 obtained. The same approach is used by [53].

Note that Gurobi solver follows a branch-and-cut implementation and adds many general-purpose and special-purpose cutting planes.

Figures 3.2a and 3.2b show that MISOCP+LN outperforms MIQP+CP in terms of optimality gap and computational time. In addition, MISOCP+LN attains better upper and lower bounds that MIQP+CP (see, Figures 3.2c and 3.2d). We do not illustrate the MISOCP+TO results in Figure 3.2.

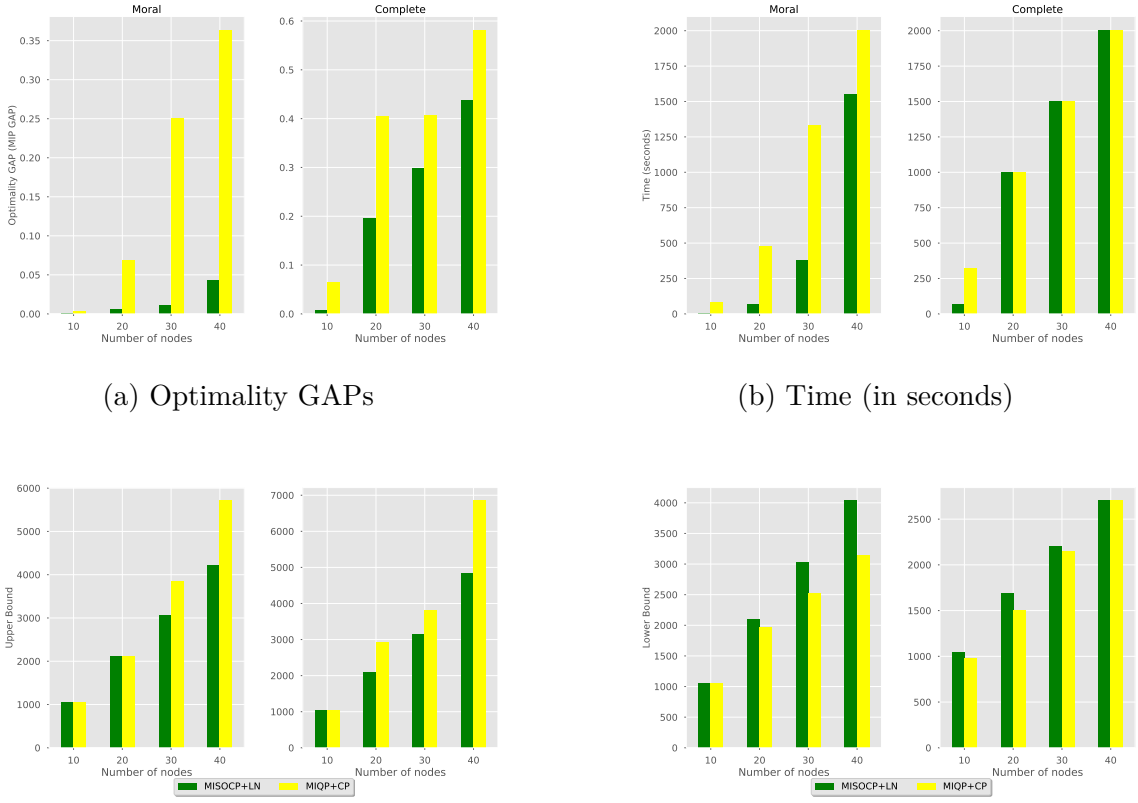
MISOCP+CP requires the solution of a second-order cone program (SOCP) after each cut, which reduces its computational efficiency and results in higher optimality gaps than MISOCP+LN. MISOCP+TO requires many binary variables which makes the problem very inefficient when the network becomes denser and larger.

3.4.3 Comparison of MISOCP versus SIMILP

Our computational experiments show that SIMILP generally performs poorly when compared to MISOCP+LN as well as MIQP+LN in terms of optimality gap, upper bound, and computational time. We report the results for SIMILP+LN, MISOCP+LN, and MIQP+LN formulations in Figure 3.3.

Figures 3.3a and 3.3b show the optimality gaps and computational times for these three formulations. Figures 3.3c and 3.3d demonstrate that SIMILP+LN attains competitive lower bounds with other two formulations. In particular, for complete instances with large number of nodes, SIMILP+LN attains better lower bounds than MIQP+LN. Nonetheless, SIMILP+LN fails to obtain good upper bounds. Therefore, the optimality gap increases drastically for SIMILP+LN.

The reason for poor performance of SIMILP+LN might be because state-of-the-art optimization packages (e.g., Gurobi, CPLEX) use many heuristics to obtain a good feasible solution (i.e., upper bound) at the root note for a compact formulation. In contrast, SIMILP is not a compact formulation, and we build the SIMILP gradually by adding new cuts. Hence, many of those built-in heuristic features may not be as effect as heuristic solutions applied on a compact formulation. Moreover, the optimization solvers capable of solving



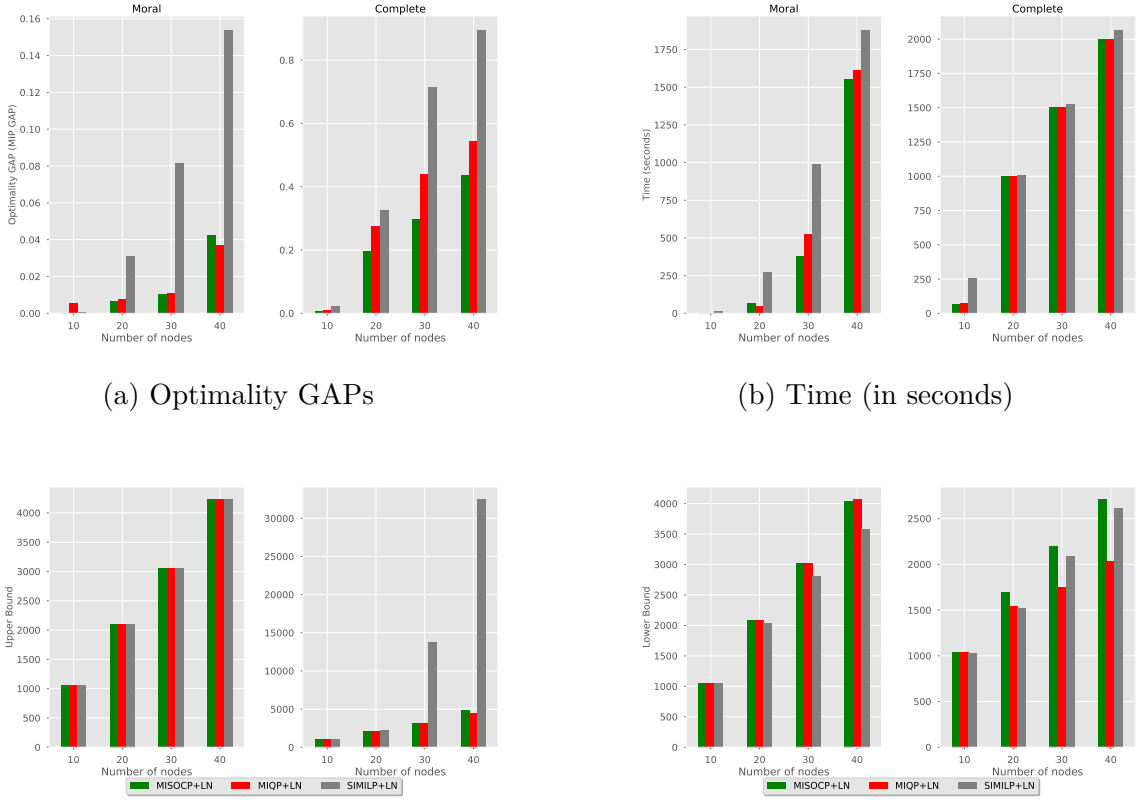
(a) Optimality GAPs

(b) Time (in seconds)

(c) Best upper bounds

(d) Best lower bounds

Figure 3.2: Optimization-based measures of MISOCP versus MIQP formulations with $n = 100$, $\mu = 0$, and $\lambda = \ln(n)$.



(a) Optimality GAPs

(b) Time (in seconds)

(c) Best upper bounds

(d) Best lower bounds

Figure 3.3: Optimization-based measures of MISOCP+LN, MIQP+LN, and SIMILP+LN formulations with $n = 100$, $\mu = 0$, and $\lambda = \ln(n)$.

MISOCP formulations have witnessed noticeable improvement due to theoretical developments in this field. In particular, Gurobi reports 20% and 38% faster solvers for their v8 and v8.1, respectively. In particular, Gurobi v8.1 reports over 4 times faster solution times than CPLEX for solving MISOCP on their benchmark instances.

3.4.4 Comparison of MISOCP versus MIQP formulations

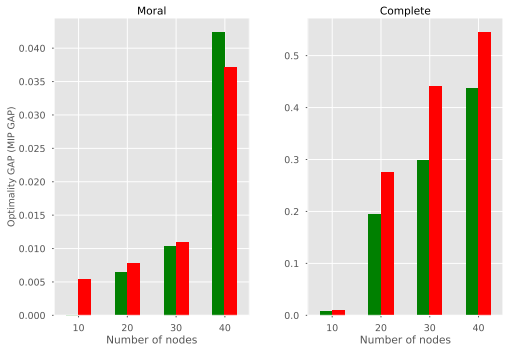
In this section, we compare the two top performing methods: MISOCP+LN and MIQP+LN. Figures 3.4a and 3.4b show that the MISOCP+LN performs better than MIQP+LN in terms of the average optimality gap across all number of nodes $m \in \{10, 20, 30, 40\}$. The only exception is $m = 40$ for sparse graphs for which MIQP+LN performs better than MISOCP+LN. Nonetheless, we observe that MISOCP+LN clearly outperforms MIQP+LN for complete instances which are more difficult to solve.

Figures 3.4c and 3.4d show the performance of both formulations in terms of the resulting upper and lower bounds on the objective function. We observe that MISOCP+LN attains better lower bounds especially for complete instances. However, MISOCP+LN cannot always obtain a better upper bound. In other words, MISOCP+LN is more effective in improving the lower bound instead of the upper bound. This is expected because MISOCP+LN attains tighter relaxation which is more advantageous in terms of improving the lower bound.

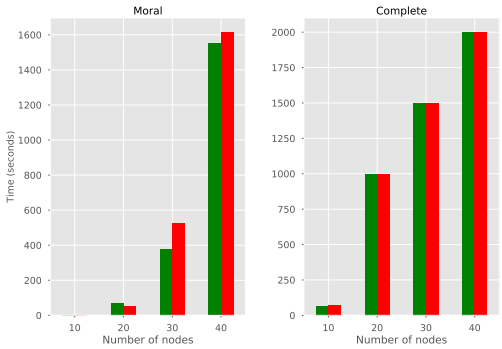
Figures 3.4e and 3.4f show that MISOCP+LN achieves noticeably higher continuous relaxation values and uses fewer branch-and-bound nodes than MIQP+LN.

3.4.5 Analysis on λ, μ, and M

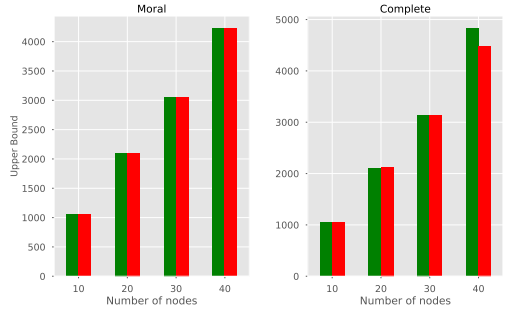
So far, we have experimented with a setting which is more in favor of MIQP+LN instead of MISOCP+LN formulation because we fixed $\mu = 0$. We now experiment on different values for λ, μ, and M to assess the effects of these parameters on the performance of MISOCP+LN and MIQP+LN. First, we change $\lambda \in \{\ln(n), 2\ln(n), 4\ln(n)\}$ while keeping the values of μ and M the same (i.e., $\mu = 0$ and $M = 2 \max_{(j,k)\in\overrightarrow{E}} |\beta^{\star}_{jk}|$). Table 3.2 shows that as λ increases, MISOCP+LN consistently performs better than MIQP+LN in terms of optimality gap, computational time, number of branch and bound nodes, and continuous relaxation objective function. Indeed, the difference becomes even more pronounced for more difficult cases (i.e., complete instances). For instance, for $\lambda = 2\ln(n) = 18.4$, the optimality gap reduces from 0.465 to 0.374, an over 24% improvement.



(a) Optimality GAPs

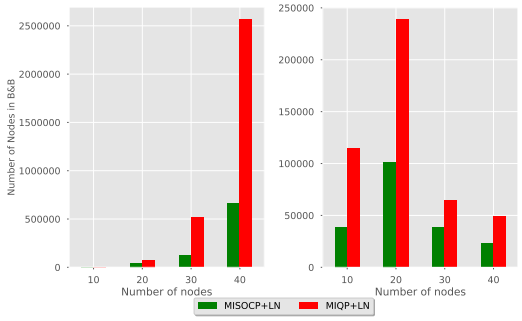


(b) Time (in seconds)

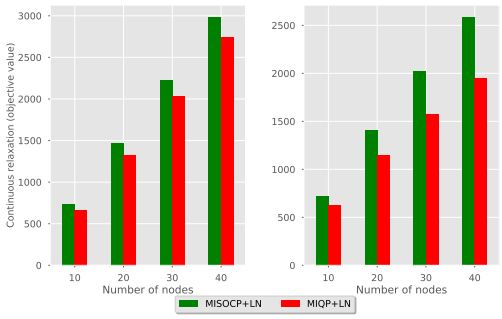


(c) Best upper bounds

(d) Best lower bounds



(e) Number of Branch and Bound nodes



(f) Continuous relaxation objective function

Figure 3.4: Optimization-based measures of MISOP+LN, MIQP+LN formulations with $n = 100$, $\mu = 0$, and $\lambda = \ln(n)$.

We next vary $\mu \in \{0, \ln(n), 2\ln(n)\}$ while keeping the values of $\lambda = \ln(n)$ and M the same as before. Table 3.3 demonstrates that for all instances with $\mu > 0$, MISOCP+LN outperforms MIQP+LN. For instance, for $m = 40$ and $\mu = 18.4$, MISOCP+LN improves the optimality gap from 0.445 to 0.366, and over 21% improvement. The reason for this improvement is that $\mu > 0$ makes the matrix more diagonally dominant; therefore, it makes the conic constraints more effective in tightening the formulation and obtaining a better optimality gap.

Finally, we study the influence of the big-M parameter. Instead of a coefficient $\gamma = 2$ in [53] proposal, we experiment with $M = \gamma \max_{(j,k)\in\vec{E}} |\beta^0_{jk}|$ for $\gamma \in \{2, 5, 10\}$ in Table 3.4, where $|\beta^0_{jk}|$ denotes the optimal solution of each optimization problem without the constraints to remove cycles. The larger the big-M parameter, the smaller the effectiveness of both models. However, MISOCP+LN tightens the formulation using the conic constraints whereas MIQP+LN does not have any means to tighten the formulation instead of big-M constraints which have poor relaxation. For $M > 2 \max_{(j,k)\in\vec{E}} |\beta^\star_{jk}|$, MISOCP+LN outperforms MIQP+LN in all measures.

Table 3.2: Computational results for different values of λ

		Sparse								Complete							
Instances		GAP		Time		# nodes		Relaxation OFV		GAP		Time		# nodes		Relaxation OFV	
m	λ	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP
10	4.6	*	*	3	2	1306	3715	738.7	664.9	*	*	65	74	38850	114433	724.4	629.3
10	9.2	*	*	4	3	1116	2936	784.6	693.5	*	*	31	39	15736	55543	772.5	662.2
10	18.4	*	*	3	2	1269	2457	857.0	747.5	*	*	26	29	18223	41197	844.5	720.2
20	4.6	*	*	69	51	46513	76261	1474.2	1325.8	**0.195**	0.275	1000	1000	101509	238765	1404.9	1144.5
20	9.2	*	*	26	27	10695	31458	1589.6	1406.8	**0.152**	0.250	1000	1000	152206	274514	1526.9	1238.6
20	18.4	*	*	24	36	9574	33788	1763.7	1552.7	**0.113**	0.208	944	1000	159789	277687	1697.1	1395.0
30	4.6	**0.010**	0.011	378	527	121358	514979	2230.1	2037.7	**0.298**	0.441	1500	1500	38474	64240	2024.0	1569.7
30	9.2	*	*	104	291	33371	248190	2392.4	2168.5	**0.239**	0.395	1500	1500	59034	71475	2217.5	1741.5
30	18.4	*	*	48	74	15649	57909	2608.3	2383.8	**0.215**	0.318	1500	1500	74952	96586	2449.2	2006.9
40	4.6	0.042	**0.037**	1551	1615	664496	2565247	2979.3	2748.6	**0.436**	0.545	2000	2000	23083	49050	2582.0	1946.3
40	9.2	**0.024**	0.036	1125	1336	353256	1347702	3200.7	2923.5	**0.397**	0.473	2000	2000	29279	73917	2869.9	2216.9
40	18.4	**0.024**	0.035	1099	1375	434648	1137666	3521.8	3225.4	**0.374**	0.465	2000	2000	31298	60697	3240.1	2633.1

* indicates that the problem is solved to the optimality tolerance.

Better MIQP GAP are in bold.

Table 3.3: Computational results for different values of μ

		Sparse								Complete							
Instances		GAP		Time		# nodes		Relaxation OFV		GAP		Time		# nodes		Relaxation OFV	
m	μ	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP
10	0	*	*	3	2	1306	3715	738.7	664.9	*	*	65	74	38850	114433	724.4	629.3
10	4.6	*	*	4	2	1043	2758	802.0	708.5	*	*	69	72	38778	119825	789.3	675.7
10	9.2	*	*	4	2	1067	2231	858.0	748.1	*	*	72	74	36326	114383	843.2	712.3
20	0	*	*	69	51	46513	76261	1474.2	1325.8	**0.195**	0.2752	1000	1000	101509	238765	1404.9	1144.5
20	4.6	*	*	45	45	15111	55302	1604.1	1426.5	**0.1666**	0.2416	1000	1000	102467	249490	1551.7	1267.1
20	9.2	*	*	43	55	15384	62297	1716.8	1515.7	**0.1422**	0.2228	1000	1000	94360	258194	1668.3	1355.1
30	0	**0.010**	0.011	378	527	121358	514979	2230.1	2037.7	**0.298**	0.4408	1500	1500	38474	64240	2024.0	1569.7
30	4.6	**0.008**	0.011	310	392	76668	358544	2432.5	2187.7	**0.2368**	0.387	1500	1500	45473	69258	2286.4	1788.5
30	9.2	**0.009**	0.010	67	377	12410	320632	2612.6	2311.4	**0.2092**	0.3666	1500	1500	41241	68661	2484.3	1915.7
40	0	0.042	**0.037**	1551	1615	664496	2565247	2979.3	2748.6	**0.4364**	0.5452	2000	2000	23083	49050	2582.0	1946.3
40	4.6	**0.027**	0.029	1331	1620	422654	1303301	3281.6	2972.8	**0.3538**	0.4708	2000	2000	13209	30995	2985.4	2261.3
40	9.2	**0.020**	0.028	870	1507	239214	1762210	3575.4	3165.3	**0.3668**	0.4454	2000	2000	13884	54638	3321.7	2468.7

* indicates that the problem is solved to the optimality tolerance.

Better MIQP GAP are in bold.

Table 3.4: Computational results for different values of γ

		Sparse								Complete							
Instances		GAP		Time		# nodes		Relaxation OFV		GAP		Time		# nodes		Relaxation OFV	
m	γ	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP	MISOCP	MIQP
10	2	*	*	3	2	1306	3715	738.7	664.9	*	*	65	74	38850	114433	724.4	629.3
10	5	*	*	5	2	1433	3026	717.9	647.1	*	*	81	82	42675	130112	705.1	607.8
10	10	*	*	5	2	1523	2564	712.5	641.1	*	*	74	100	35576	174085	699.8	600.3
20	2	*	*	69	51	46513	76261	1474.2	1325.8	**0.195**	0.275	1000	1000	101509	238765	1404.9	1144.5
20	5	*	*	103	156	65951	209595	1438.2	1274.2	**0.211**	0.308	1000	1000	97940	225050	1375.3	1080.9
20	10	*	*	215	207	150250	349335	1427.7	1256.6	**0.230**	0.310	1000	1000	90864	257998	1366.3	1058.2
30	2	**0.010**	0.011	378	527	121358	514979	2230.1	2037.7	**0.298**	0.441	1500	1500	38474	64240	2024.0	1569.7
30	5	**0.011**	0.014	571	620	164852	527847	2173.9	1950.3	**0.336**	0.474	1501	1500	33120	64339	1969.4	1448.4
30	10	0.024	**0.014**	630	638	202635	585234	2156.5	1919.6	**0.349**	0.480	1500	1500	30579	77100	1951.2	1404.0
40	2	0.042	**0.037**	1551	1615	664496	2565247	2979.3	2748.6	**0.436**	0.545	2000	2000	23083	49050	2582.0	1946.3
40	5	**0.045**	0.047	1643	1634	638323	1347868	2895.6	2635.0	**0.579**	0.580	2000	2000	12076	30858	2488.0	1751.7
40	10	**0.056**	0.057	1639	1632	599281	1584187	2869.2	2595.6	**0.585**	0.594	2000	2000	11847	30222	2456.1	1679.6

* indicates that the problem is solved to the optimality tolerance.

Better MIQP GAP are in bold.

3.4.6 Structural Hamming Distance

In this section, we evaluate the quality of estimated DAGs obtained from MISOCP+LN by comparing with the ground truth DAG. To this end, we use the average structural Hamming distance (SHD) which counts the number of differences (addition, deletion, or arc reversal) required to transform predicted DAG to the true DAG. We run our experiments with MIQP GAP $\leq \epsilon$ where $\epsilon \in \{0.01, \frac{\ln n}{n}\}$ as our termination criterion. A large time limit of $250m$ is imposed to ensure that the termination is due to MIQP GAP. Nonetheless, we may not still hit the time limit for difficult instances. Table 3.5 shows the numerical results for the average SHD for MISOCP across 10 runs for sparse instances. In parenthesis, we specify the standard deviation of SHD numbers. As for complete instances, MISOCP+LN cannot hit the MIQP GAP of 0.01 or $\frac{\ln(n)}{n}$ given the time limit. Hence, we observe almost identical average SHD measure for both stopping conditions.

Table 3.5 indicates that the average SHD for sparse instances are very similar with $\epsilon = \frac{\ln n}{n}$ when compared with our default MIQP GAP of 0.01 in practice. Note that a lower MIQP GAP does not necessarily lead to a better SHD score. To achieve a consistent estimate, we can set the MIQP GAP as $\frac{\ln(n)}{n}$ as n increases, see Proposition 10. From a computational standpoint, we can observe that by changing the stopping criterion from 0.01 to $\frac{\ln(n)}{n}$, we reduce the computational time about 46% whereas the SHD numbers for both stopping criteria are very close to each other.

Table 3.5: SHD results for early stopping with $\lambda = \ln(n)$ and MIQP GAP $\leq \epsilon$ for sparse instances

		$\epsilon = 0.01$			$\epsilon = \frac{\ln(n)}{n}$			
m	$\lvert\mathcal{M}\rvert$	Time	MIQP GAP	SHD (std)	Time	MIQP GAP	SHD (std)	Improvement
10	19	3.3	0.000	0.0 (0.00)	3.2	0.017	0.3 (0.58)	3%
20	58	96.9	0.007	1.6 (1.73)	29.67	0.038	2.0 (2.08)	69%
30	109	2603.2	0.012	3.6 (3.06)	95.33	0.045	3.3 (3.51)	96%
40	138	7988.4	0.035	3.3 (2.52)	6503.73	0.047	2.6 (1.53)	18%

Chapter 4
CONCLUSION AND FUTURE WORK

In this work, 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 penalized negative log-likelihood score function with both ℓ_0 and ℓ_1 regularizations. 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) formulation, 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.

We tackle another issue in DAG structural learning problem. That is, to enforce bounds on the continuous optimization variables corresponding to the arc weights in the DAG structure. This is commonly modeled using a standard "big-M constraint" in the associated MIQPs. However, this strategy leads to a poor continuous relaxation because there is no natural upper bound for the arc weights. Therefore, LN formulation – as well as all other "big-M" formulations – attain poor lower bounds. To cope with this difficulty, we present a perspective formulation for DAG structural learning problem. The resulting formulation

is a mixed-integer non-linear program (MINLP) which becomes computationally challenging to solve. Nonetheless, we give two different reformulations for this MINLP: (i) a *mixed-integer second-order conic program* (MISOCP), (ii) a *semi-infinite mixed-integer linear program* (SIMILP). We establish that both formulations have tighter continuous relaxations compared to the existing "big-M" formulations. Our experimental results demonstrate that MISOCP improves the lower bound and reduces the optimality gap compared to the LN formulation whereas SIMILP is not computationally as competitive as LN. We also establish an early stopping criterion under which the branch-and-bound process can be terminated early. We show that the obtained solution asymptotically converges to the true coefficients in DAG with high probability under proper conditions.

As for future research direction, it is of interest to develop more efficient techniques to solve large-scale problems. One way to approach this challenge is to use the continuous relaxation of MISOCP to obtain an initial solution and a lower bound for this problem. Then, this solution can be modified to obtain an upper bound. To obtain an optimality guarantee for such a solutions remains an open challenge.

In many real-world applications, the random variables possess hierarchical structures. Hence, it is naturally of great interest to learn a DAG such that it satisfies the hierarchy among different groups of random variables. This problem has important applications in discovering the genetic basis of common genetic diseases (e.g., asthma, diabetes, atherosclerosis).

Lastly, data quality varies based on the nature of the collected data. It is very common that observational data is inaccurate or there is missing data. It is of natural importance to develop robust DAG structural learning that directly addresses errors and incompleteness in the data.

BIBLIOGRAPHY

[1] Bryon Aragam and Qing Zhou. Concave penalized estimation of sparse Gaussian Bayesian networks. *Journal of Machine Learning Research*, 16:2273–2328, 2015.

[2] Bryon Aragam, Arash A Amini, and Qing Zhou. Learning directed acyclic graphs with penalized neighbourhood regression. *arXiv preprint arXiv:1511.08963*, 2015.

[3] Alper Atamturk and Andres Gomez. Rank-one convexification for sparse regression. *arXiv preprint arXiv:1901.10334*, 2019.

[4] Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. *arXiv preprint arXiv:1309.6825*, 2013.

[5] Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

[6] Tolga Bektaş and Luis Gouveia. Requiem for the Miller–Tucker–Zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832, 2014.

[7] Dimitris Bertsimas and Angela King. Logistic regression: From art to science. *Statistical Science*, 32(3):367–384, 2017.

[8] Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *arXiv preprint arXiv:1709.10029*, 2017.

[9] Dimitris Bertsimas, Angela King, Rahul Mazumder, et al. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[10] Peter J Bickel, Ya'acov Ritov, Alexandre B Tsybakov, et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[11] Concha Bielza and Pedro Larrañaga. Bayesian networks in neuroscience: a survey. *Frontiers in computational neuroscience*, 8:131, 2014.

[12] Peter Bühlmann and Sara A van de Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer, 2011.

[13] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[14] Wenyu Chen, Mathias Drton, and Y Samuel Wang. On causal discovery with equal variance assumption. *arXiv preprint arXiv:1807.03419*, 2018.

[15] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from data*, pages 121–130. Springer, 1996.

[16] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554, 2002.

[17] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *Journal of the American Mathematical Society*, 22(1):211–231, 2009.

[18] William J. Cook, William H Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Wiley, 1997.

[19] Alvaro HC Correia, James Cussens, and Cassio P de Campos. On pruning for score-based bayesian network structure learning. *arXiv preprint arXiv:1905.09943*, 2019.

[20] XT Cui, XJ Zheng, SS Zhu, and XL Sun. Convex relaxations and MIQCQP reformulations for a class of cardinality-constrained portfolio selection problems. *Journal of Global Optimization*, 56(4):1409–1423, 2013.

[21] James Cussens. Maximum likelihood pedigree reconstruction using integer programming. In *WCB@ ICLP*, pages 8–19, 2010.

[22] James Cussens. Bayesian network learning with cutting planes. *arXiv preprint arXiv:1202.3713*, 2012.

[23] James Cussens, David Haws, and Milan Studenỳ. Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*, 164(1-2):285–324, 2017.

[24] James Cussens, Matti Järvisalo, Janne H Korhonen, and Mark Bartlett. Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *J. Artif. Intell. Res.(JAIR)*, 58:185–229, 2017.

[25] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

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

[27] Hongbo Dong, Kun Chen, and Jeff Linderoth. Regularization vs. relaxation: A conic optimization perspective of statistical variable selection. *arXiv preprint arXiv:1510.06083*, 2015.

[28] Daniel Eaton and Kevin Murphy. Exact Bayesian structure learning from uncertain interventions. In *Artificial Intelligence and Statistics*, pages 107–114, 2007.

[29] Antonio Frangioni and Claudio Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2):225–236, 2006.

[30] Antonio Frangioni and Claudio Gentile. SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. *Operations Research Letters*, 35(2):181–185, 2007.

[31] Antonio Frangioni and Claudio Gentile. A computational comparison of reformulations of the perspective relaxation: SOCP vs. cutting planes. *Operations Research Letters*, 37(3):206–210, 2009.

[32] Antonio Frangioni, Claudio Gentile, Enrico Grande, and Andrea Pacifici. Projected perspective reformulations with applications in design problems. *Operations Research*, 59(5):1225–1232, 2011.

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

[34] Andrés Gómez and O Prokopyev. A mixed-integer fractional optimization approach to best subset selection. *Optimization-online*, 2018.

[35] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42, 1985.

[36] Sung Won Han, Gong Chen, Myun-Seok Cheon, and Hua Zhong. Estimation of directed acyclic graphs through two-stage adaptive lasso for gene network inference. *Journal of the American Statistical Association*, 111(515):1004–1019, 2016.

[37] Patrick Healy and Nikola S Nikolov. A branch-and-cut approach to the directed acyclic graph layering problem. In *International Symposium on Graph Drawing*, pages 98–109. Springer, 2002.

[38] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 358–365, 2010.

[39] Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8(Mar):613–636, 2007.

[40] Mikko Koivisto. Advances in exact Bayesian structure discovery in Bayesian networks. *arXiv preprint arXiv:1206.6828*, 2012.

[41] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5(May):549–573, 2004.

[42] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[43] Jack Kuipers, Polina Suter, and Giusi Moffa. Efficient structure learning and sampling of bayesian networks. *arXiv preprint arXiv:1803.07859*, 2018.

[44] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996. ISBN 0-19-852219-3.

[45] Po-Ling Loh and Peter Bühlmann. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research*, 15(1): 3065–3105, 2014.

[46] Brandon Malone, Kustaa Kangas, Matti Järvisalo, Mikko Koivisto, and Petri Myllymäki. Predicting the hardness of learning Bayesian networks. In *AAAI*, pages 2460–2466, 2014.

[47] Hasan Manzour, Simge Küçükyavuz, and Ali Shojaie. Integer programming for learning directed acyclic graphs from continuous data. *arXiv preprint arXiv:1904.10574*, 2019.

[48] Florian Markowetz and Rainer Spang. Inferring cellular networks–a review. *BMC bioinformatics*, 8(6):S5, 2007.

[49] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[50] George L Nemhauser and Laurence A Wolsey. 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, 1988.

[51] Temel Öncan, İ Kuban Altınel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

[52] Manfred Padberg and Ting-Yi Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1-3):315–357, 1991.

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

[54] Pekka Parviainen and Mikko Koivisto. Exact structure discovery in Bayesian networks with less space. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 436–443. AUAI Press, 2009.

[55] Gábor Pataki. Teaching integer programming formulations using the traveling salesman problem. *SIAM review*, 45(1):116–123, 2003.

[56] Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

[57] Eric Perrier, Seiya Imoto, and Satoru Miyano. Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research*, 9(Oct):2251–2286, 2008.

[58] Jonas Peters and Peter Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2013.

[59] Jonas Peters, Joris Mooij, Dominik Janzing, and Bernhard Schölkopf. Identifiability of causal graphs using functional models. *arXiv preprint arXiv:1202.3757*, 2012.

[60] Mert Pilanci, Martin J Wainwright, and Laurent El Ghaoui. Sparse learning via boolean relaxations. *Mathematical Programming*, 151(1):63–87, 2015.

[61] Garvesh Raskutti and Caroline Uhler. Learning directed acyclic graphs based on sparsest permutations. *arXiv preprint arXiv:1307.0366*, 2013.

[62] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Restricted eigenvalue properties for correlated gaussian designs. *Journal of Machine Learning Research*, 11(Aug):2241–2259, 2010.

[63] T Sawik. 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, 2016.

[64] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7 (Oct):2003–2030, 2006.

[65] Ali Shojaie and George Michailidis. Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. *Biometrika*, 97(3):519–538, 2010.

[66] Tomi Silander and Petri Myllymaki. A simple approach for finding the globally optimal Bayesian network structure. *arXiv preprint arXiv:1206.6875*, 2012.

[67] Moninder Singh and Marco Valtorta. Construction of Bayesian network structures from data: a brief survey and an efficient algorithm. *International journal of approximate reasoning*, 12(2):111–131, 1995.

[68] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT press, 2000.

[69] AA Tarkhan, F Farzaneh, and BH Khalaj. Efficient suboptimal transmit antenna selection for mimo relay channels. In *2011 International Symposium on Computer Networks and Distributed Systems (CNDS)*, pages 45–48. IEEE, 2011.

[70] AA Tarkhan, F Farzaneh, and BH Khalaj. Mutual coupling and correlation based suboptimal antenna subset selection in mimo systems. In *2011 International Symposium on Computer Networks and Distributed Systems (CNDS)*, pages 12–16. IEEE, 2011.

[71] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[72] Sara van de Geer and Peter Bühlmann. ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics*, 41(2):536–567, 2013.

[73] Naftali Weinberger. Faithfulness, coordination and causal coincidences. *Erkenntnis*, 83 (2):113–133, 2018.

[74] Jing Xiang and Seyoung Kim. A* lasso for learning a sparse Bayesian network structure for continuous variables. In *Advances in Neural Information Processing Systems*, pages 2418–2426, 2013.

[75] Weijun Xie and Xinwei Deng. The ccp selector: Scalable algorithms for sparse ridge regression from chance-constrained programming. *arXiv preprint arXiv:1806.03756*, 2018.

[76] Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

[77] Changhe Yuan, Brandon Malone, and Xiaojian Wu. Learning optimal Bayesian networks using A* search. In *IJCAI proceedings-international joint conference on artificial intelligence*, volume 22, page 2186, 2011.

[78] Bin Zhang, Chris Gaiteri, Liviu-Gabriel Bodea, Zhi Wang, Joshua McElwee, Alexei A Podtelezhnikov, Chunsheng Zhang, Tao Xie, Linh Tran, Radu Dobrin, et al. Integrated systems approach identifies genetic nodes and networks in late-onset Alzheimer's disease. *Cell*, 153(3):707–720, 2013.

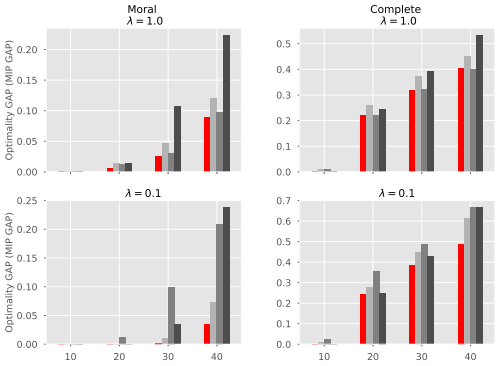
[79] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[80] Xiaojin Zheng, Xiaoling Sun, and Duan Li. Improving the performance of MIQP solvers for quadratic programs with cardinality and minimum threshold constraints: A semidefinite program approach. *INFORMS Journal on Computing*, 26(4):690–703, 2014.

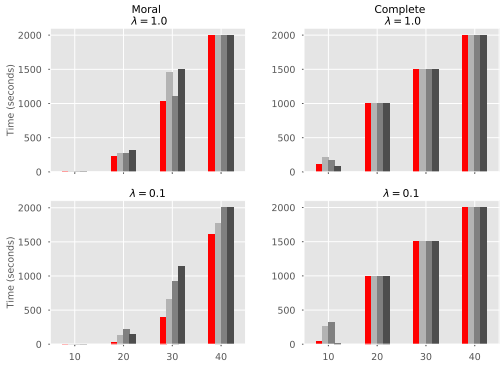
[81] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. DAGs with NO TEARS: smooth optimization for structure learning. *arXiv preprint arXiv:1803.01422*, 2018.

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

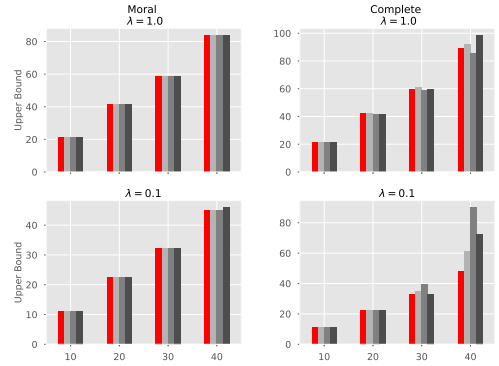
Appendix A



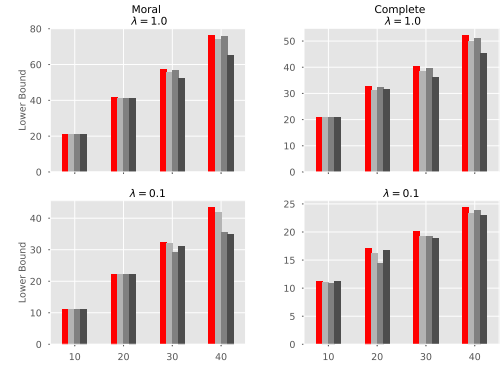
(a) Optimality GAPs for MIQPs



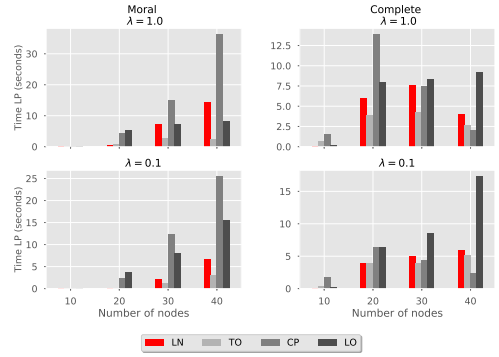
(b) Time (in seconds) for MIQPs



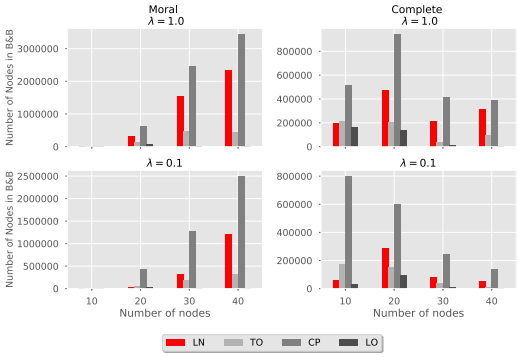
(c) Best upper bounds for MIQPs



(d) Best lower bounds for MIQPs

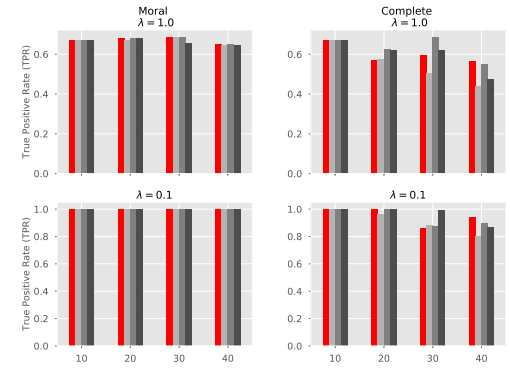


(e) Time for continuous root relaxation

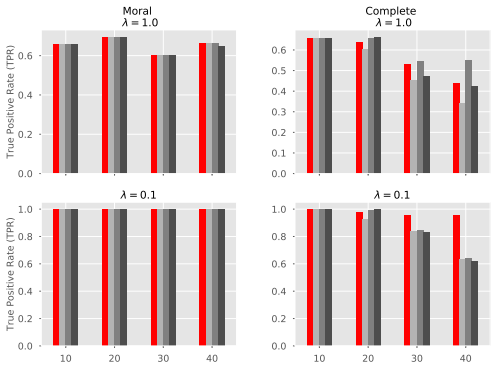


(f) Number of explored nodes in B&B tree

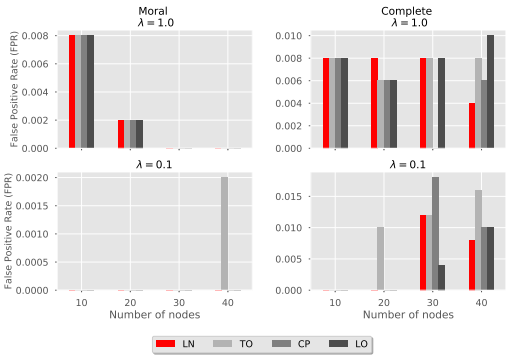
Figure A.1: Optimization-based measures for MIQPs for ℓ_0 model with number of samples $n = 100$.



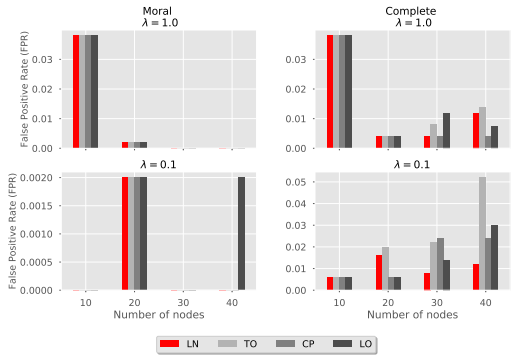
(a) True Positive Rate (TPR) for MIQPs



(b) True Positive Rate(TPR) for MIQPs

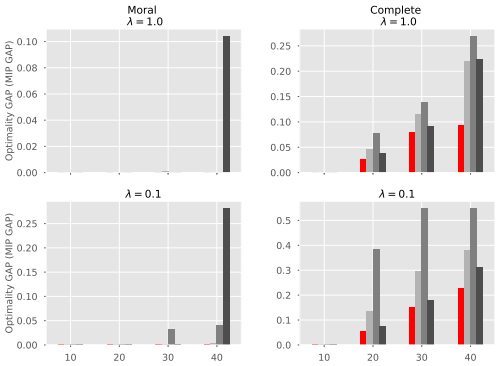


(c) False Positive Rate(FPR) for MIQPs

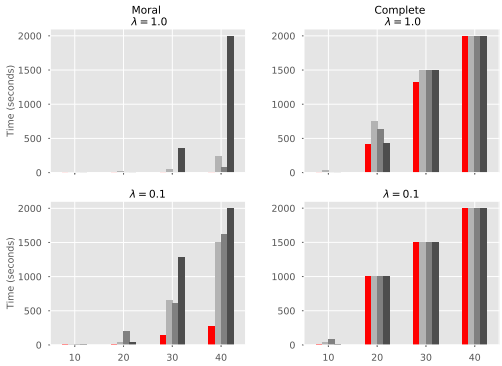


(d) False Positive Rate(FPR) for MIQPs

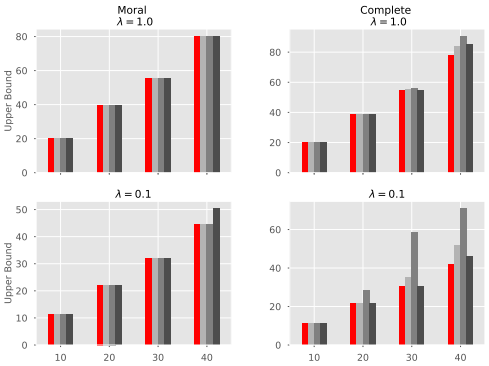
Figure A.2: Graph metrics for the MIQPs for ℓ_0 regularization. Plots a, c (left) show graph metrics with the number of samples $n = 1000$ and plots b, d (right) show the graph metrics with the number of samples $n = 100$.



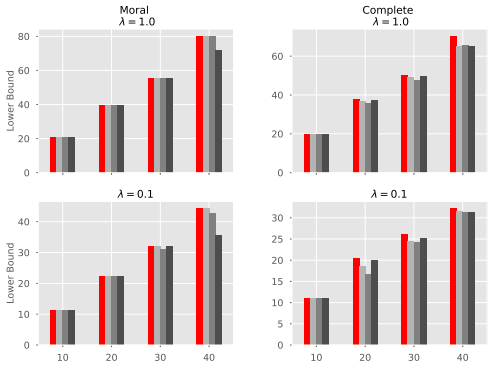
(a) Optimality GAPs for MIQPs



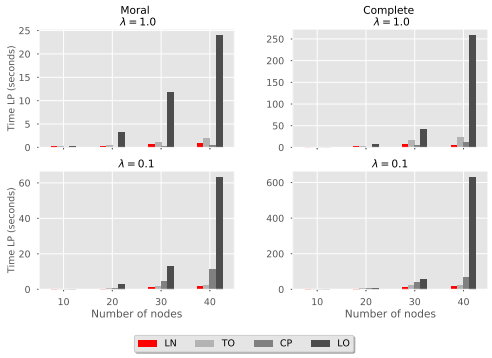
(b) Time (in seconds) for MIQPs



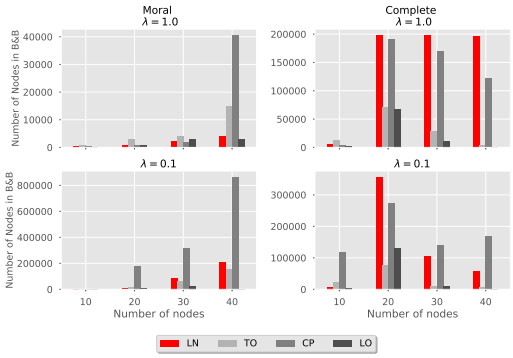
(c) Best obtained upper bounds for MIQPs



(d) Best obtained lower bounds for MIQPs

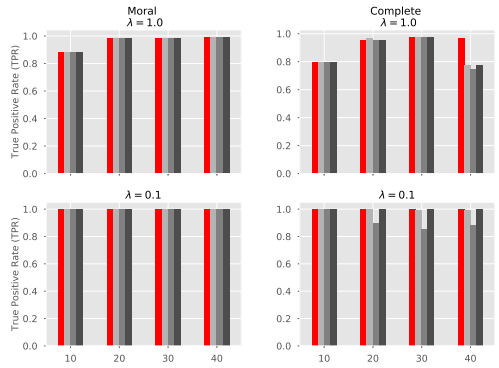


(e) Time (in seconds) for continuous root relaxation

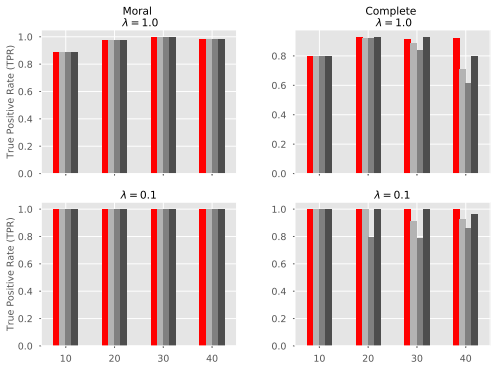


(f) Number of explored nodes in B&B tree

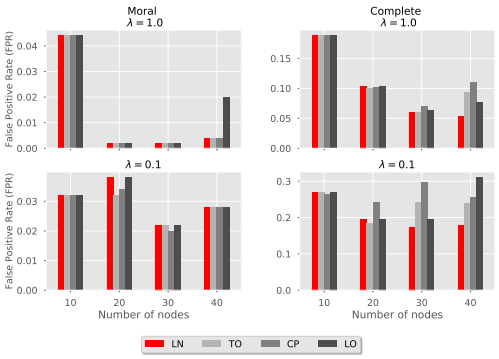
Figure A.3: Optimization-based measures for MIQPs for ℓ_1 model with the number of samples $n = 100$.



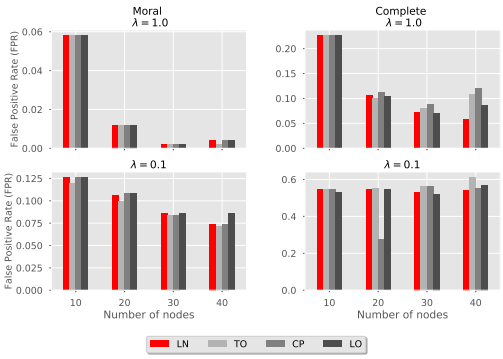
(a) True Positive Rate (TPR) for MIQPs



(b) True Positive Rate (TPR) for MIQPs



(c) False Positive Rate (FPR) for MIQPs



(d) False Positive Rate (FPR) for MIQPs

Figure A.4: 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 $n = 100$.