**Title**
Structure Learning of Linear Bayesian Networks in High-Dimensions



**Author**
Aragam, Nikhyl Bryon

**Publication Date**
2015

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

# Structure Learning of Linear Bayesian Networks in High-Dimensions

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

**Nikhyl Bryon Aragam**

2015

Abstract of the Dissertation

# Structure Learning of Linear Bayesian Networks in High-Dimensions

by

**Nikhyl Bryon Aragam**

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2015

Professor Qing Zhou, Chair

Research into graphical models is a rapidly developing enterprise, garnering significant interest from both the statistics and machine learning communities. A parallel thread in both communities has been the study of low-dimensional structures in high-dimensional models where $p \gg n$. Recently, there has been a surge of interest in connecting these threads in order to understand the behaviour of graphical models in high-dimensions. Due to their relative simplicity, undirected models such as the Gaussian graphical model and Ising models have received most of the attention, whereas directed graphical models have received comparatively little attention. An important yet largely unresolved class of directed graphical models are Bayesian networks, or directed acyclic graphs (DAGs). These models have a wide variety of applications in aritificial intelligence, machine learning, genetics, and computer vision, but estimation of Bayesian networks in high-dimensions is not well-understood. The main focus of this dissertation is to address some fundamental questions about these models in high-dimensions.

The primary goal is to develop both algorithms and theory for estimating continuous, linear Bayesian networks, capable of handling modern high-dimensional problems. Motivated by problems from the regression literature, we show how to adapt recent work in sparse learning and nonconvex optimization to the structure

learning problem for Bayesian networks in order to estimate DAGs with several thousand nodes. We draw an explicit connection between linear Bayesian networks and so-called neighbourhood regression problems and show how this can be exploited in order to derive nonasymptotic performance bounds for penalized least squares estimators of directed graphical models.

On the algorithmic side, we develop a method for estimating Gaussian Bayesian networks based on convex reparametrization and cyclic coordinate descent. In contrast to recent methods which accelerate the learning problem by restricting the search space, we propose a method for score-based structure learning which does not restrict the search space. We do not require the existence of a so-called faithful DAG representation, and as a result, our methodology must handle the inherent nonidentifiability of the estimation problem in a novel way. On the theoretical side, we provide (a) Finite-dimensional performance guarantees for local minima of the resulting nonconvex program, and (b) A general high-dimensional framework for global minima of the nonconvex program. Both the algorithms and theory apply to a general class of regularizers, including the MCP, SCAD, $\ell_1$ and $\ell_0$ penalties. Finally, as a matter of independent interest, we provide a comprehensive comparison of our approach to several standard structure learning methods using open-source packages developed for the `R` language.

The dissertation of Nikhyl Bryon Aragam is approved.

Steve Horvath

Ying Nian Wu

Jingyi Li

Arash Ali Amini

Qing Zhou, Committee Chair

University of California, Los Angeles

2015

*To Mr. Bedwell and Mrs. Woo –*

*You'll be happy to know that I finally figured out how to add fractions.*

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

It is not possible to list every name that deserves acknowledgement here, so I will not attempt to do so. Such a list may very well be longer than the content of this dissertation. So, without naming names:

First and foremost, I thank my family and my friends—many of whom really do count as family. My best friend gets special recognition for always being there to keep me from doing stupid things, and for sometimes being there to do the stupid things with me—even from thousands of miles away.

Sincerest thanks are of course due to everyone on my committee, along with every last teacher who had to put up with me over the years. How all of you did so is beyond my understanding.

To everyone in the Statistics Department at UCLA: You taught me how it feels to be supported, encouraged, and motivated to pursue my interests.

Finally, to my tailor for making me get rid of the khakis.

# Vita

| | |
|---|---|
| 2008 | B.A. (Mathematics), University of Tennessee, Knoxville. |
| 2008-2009 | Programmer, U30 Group, Knoxville, Tennessee. |
| 2009-2010 | Teaching Assistant, Department of Mathematics, University of California, Los Angeles |
| 2010 | M.A. (Mathematics), University of California, Los Angeles. |
| 2010-2012 | Research Fellow, Department of Mathematics, University of California, Los Angeles |
| 2012-2015 | Research Fellow, Department of Statistics, University of California, Los Angeles |
| 2013-2014 | Teaching Assistant, Department of Statistics, University of California, Los Angeles |
| 2014-2015 | Teaching Assistant Coordinator, Department of Statistics, University of California, Los Angeles |

# Publications

B. Aragam and Q. Zhou. Concave penalized estimation of sparse Gaussian Bayesian networks. *The Journal of Machine Learning Research*, 16:156, 2015.

# CHAPTER 1

# Introduction

The problem of estimating Bayesian networks (BNs) has received a significant amount of attention over the past decade, with applications ranging from medicine and genetics to expert systems and artificial intelligence. The idea of using directed graphical models such as Bayesian networks to model real-world phenomena is certainly nothing new, and while the calculus of these models has been well-developed, the development of fast algorithms to accurately estimate these models in high-dimensions has been slow. The basic problem can be formulated as follows: Given observations from a probability distribution, is it possible to construct a directed acyclic graph (DAG) which decomposes the distribution into a sparse Bayesian network?

Based on observational data alone, it is well-known that there are many Bayesian networks that are consistent in the Markov sense with a given distribution. What we are interested in is finding the sparsest possible Bayesian network, estimated purely from i.i.d. observations without any experimental data. When the number of variables is small, there are many practical algorithms for solving this problem. Unfortunately, as the number of variables increases, this problem becomes notoriously difficult: the learning problem is nonconvex, NP-hard, and scales super-exponentially with the number of variables (Chickering, 1996; Chickering and Meek, 2002; Robinson, 1977). Since many realistic networks can have upwards of thousands or even tens of thousands of nodes—genetic networks being a prominent example of great importance—the development of new statistical

methods for learning the structure of Bayesian networks is critical.

In this dissertation, we propose a new score-based framework for learning the structure of Gaussian Bayesian networks from observational data. This framework is based on recent work by Fu and Zhou (2013) and van de Geer and Bühlmann (2013), who show how these ideas lead to a family of estimators with good theoretical properties and whose estimation performance is competitive with traditional approaches. Neither of these works, however, consider the computational challenges associated with high-dimensional data sets for which the dimension scales to thousands of variables, which is a key challenge in Bayesian network learning. With these computational challenges in mind, we sought to develop a score-based method that:

- Does not restrict or prune the search space in any way;
- Does not assume faithfulness;
- Does not require a known variable ordering;
- Works on observational data (i.e. without experimental interventions);
- Works effectively in high dimensions ($p \gg n$);
- Is capable of handling graphs with several thousand variables.

While various methods in the literature cover a few of these requirements, none that we are aware of simultaneously cover *all* of them. The main contribution of the present work is a fast algorithm for score-based structure learning that accomplishes precisely that, along with an appropriate high-dimensional analysis that provides non-asymptotic performance guarantees which are valid for finite sample sizes.

One of the key developments in our method is the application of modern regularization techniques, including both $\ell_1$ and concave penalties. Although $\ell_1$ regu-

larization is well-understood with attractive high-dimensional and computational properties (Bühlmann and van de Geer, 2011), as we shall see, in the context of Bayesian networks many of these advantages disappear. While our approach still allows for $\ell_1$-based penalties in practice, our results will indicate that concave penalties such as the SCAD (Fan and Li, 2001) and MCP (Zhang, 2010) offer improved performance. This is in line with recent advances in sparse learning that have highlighted the advantages of nonconvex regularization in linear and generalized linear models (Lv and Fan, 2009; Fan and Lv, 2010, 2011; Zhang and Zhang, 2012; Huang et al., 2012; Fan and Lv, 2013). Notwithstanding, both our theory and our method apply to a general class of penalties which can be chosen based on the application at hand.

In this light, our method also represents a major conceptual departure from existing methods in the literature on Bayesian networks through its deep involvement of recent developments in sparse regression, as well as using parametric modeling via structural equations as its foundation, in contrast to the more common approach using graph theory and Markov equivalence. These techniques have long been known to be useful in regression modeling, covariance estimation, matrix factorization, and image processing, but their application to Bayesian networks, as far as we can tell, is a recent development (Schmidt et al., 2007; Xiang and Kim, 2013; Fu and Zhou, 2013; Fu et al., 2014). Finally, our method offers new insights into accelerating score-based algorithms in order to compete with hybrid and constraint-based methods which, as we will show, are generally faster and more effective than traditional score-based algorithms.

## 1 Overview and motivation

Suppose

$$X = (X_1, \ldots, X_p) \sim \mathcal{N}(0, \Sigma), \tag{1.1}$$

where $X \in \mathbb{R}^p$ and the covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$ is assumed to be positive definite. Then there exist matrices $\widetilde{B}$ and $\widetilde{\Omega}$ such that the following identity holds:

$$X = \widetilde{B}^T X + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \widetilde{\Omega}). \tag{1.2}$$

In the Gaussian setting with $X \sim \mathcal{N}(0, \Sigma)$, the matrix $\widetilde{B}$ can always be interpreted as the weighted adjacency matrix of a directed acyclic graph (DAG) and the matrix $\widetilde{\Omega}$ is diagonal. This decomposition is often referred to as a *structural equation model* (SEM) for $X$, and extends well beyond the Gaussian setting considered here.

It is well-known that for any particular order of the variables in $X$ it is easy to construct such a matrix: For each $j < p$, project $X_j$ onto the linear span of $\{X_{j+1}, \ldots, X_p\}$. Unfortunately, if we re-order the variables and re-apply this procedure under the new ordering, we may end up with a completely different set of structural equations. Current understanding of this phenomenon is somewhat incomplete, which naturally leads us to consider the problem of estimating $\widetilde{B}$ given no prior assumptions or knowledge on the correct ordering. Thus, the main problem this dissertation seeks to tackle can be summarized as follows:

> *How can we estimate (statistically) and approximate (computationally) a matrix $\widetilde{B}$ satisfying* (1.2) *in a high-dimensional setting, and what kind of properties will the estimated DAG have?*

## 1.1 Previous work

The problem of inducing a directed graphical model must be contrasted with the comparatively simpler problem of estimating an undirected Gaussian graphical model, which has received much more attention particularly in the high-dimensional setting with $p \gg n$. Despite Bayesian networks continuing to rep-

resent an important and well-studied class of graphical models, the behaviour of these models in high-dimensions has received comparatively little attention.

### 1.1.1 Algorithms and computations

Traditionally, there are three main algorithmic approaches to learning Bayesian networks.

*Score-based.* In the score-based approach, a scoring function is defined over the space of DAG structures, and one searches this space for a structure that optimizes the chosen scoring function. The most commonly used scoring functions are based on the a posteriori probability of a network structure (Geiger and Heckerman, 2013), while others use minimum-description length, which is equivalent to BIC (Lam and Bacchus, 1994). In terms of implementation, the standard algorithmic approach is greedy hill-climbing (Heckerman et al., 1995), for which various improvements have been offered over the years (e.g. Chickering, 2003). Monte Carlo methods have also been used to sample network structures according to an a posteriori distribution (Ellis and Wong, 2008; Zhou, 2011).

*Constraint-based.* In the constraint-based approach, repeated conditional independence tests are used to check for the existence of edges between nodes. The idea is to search for statistical independence between variables, which indicates that an edge cannot exist in the underlying DAG structure as long as certain assumptions are satisfied. These assumptions tend to be very strong in practice, and this constitutes the main drawback of this approach. Conversely, since the tests of independence can be efficient, constraint-based approaches tend to be faster than score-based approaches. Two popular approaches in this spirit are the PC algorithm (Spirtes and Glymour, 1991; Kalisch and Bühlmann, 2007) and the MMPC algorithm (Tsamardinos et al., 2006).

*Hybrid.* In the hybrid approach, constraint-based search is used to prune the

search space (e.g. to find the skeleton or a moral graph representation), which is then used as an input to restrict a score-based search. By removing as many edges as possible in the first step, the second step can be significantly faster than unrestricted score-based searching. This technique has been shown to work well in practice by combining the advantages of score-based and constraint-based approaches (Tsamardinos et al., 2006; Gámez et al., 2011, 2012).

As previously noted, the main issue with modern approaches to structure learning is scaling algorithms to data sets of ever-increasing sizes. Tsamardinos et al. (2006) show how their hybrid MMHC algorithm scales to 5,000 variables, although the running time of 13 days left much to be desired. By assuming the underlying DAG is sparse, Kalisch and Bühlmann (2007) show how exploiting sparsity in the PC algorithm leads to significant computational gains. More recently, Gámez et al. (2012) have proposed modifications to hybrid hill-climbing that scale to 1000 or so variables. By taking advantage of distributed computation, Scutari (2014) shows how to scale constraint-based approaches to thousands of variables. Notably, none of these methods fall into the first category of score-based methods.

Moreover, in additions to these computational issues, as we will show in Section 3.3.2, existing methods tend to suffer from poor *estimation* performance as well when the data are high-dimensional. This is due to the proliferation of spurious correlations in high-dimensions (Fan and Lv, 2010), which affects both constraint-based and score-based methods. This observation underscores the fact that the algorithmic issues with learning Bayesian networks involve both *efficiency* and *accuracy*.

Finally, while the traditional approach to estimating Bayesian networks uses $\ell_0$-based penalties such as the Bayesian information criterion (BIC), Fu and Zhou (2013) recently introduced the idea of using continuous penalties via the adaptive $\ell_1$ penalty and showed that it can be competitive in practice. They combine a novel method of enforcing acyclicity with a block coordinate descent algorithm

in order to compute an $\ell_1$-penalized maximum likelihood estimator for structure learning. Their algorithm is adapted to the case of intervention data and does not exploit the underlying convexity of the Gaussian likelihood function; as a result, it cannot be used on high-dimensional data and is limited to graphs with 200 or so nodes.

### 1.1.2 Theory

In the classical low-dimensional setting, Chickering (2003) outlined a general theory for the analysis of score-based algorithms. This analysis highlighted several key properties of a scoring function that guarantee that a particular greedy algorithm is sufficient to learn the structure of a Bayesian network if $p$ stays fixed as $n \to \infty$. This local analysis makes strong use of this low-dimensional setting and unfortunately does not offer much insight into the broader high-dimensional problem.

Under a fixed or known ordering of the variables, the structure learning problem reduces to standard multiple regression, which can be estimated in high-dimensions via regularization (see Fan and Lv, 2010, for an overview). For the $\ell_1$ penalty, Shojaie and Michailidis (2010) obtained positive results for structure learning assuming a known ordering. In contrast, we will not assume any prior knowledge of the correct order of the variables, so these methods are quite different from what will be developed in the sequel.

The first truly high-dimensional analysis of an algorithm for inducing Bayesian networks was Kalisch and Bühlmann (2007), which established the consistency of the constraint-based PC algorithm in high-dimensions. More recently, van de Geer and Bühlmann (2013) provided some high-dimensional guarantees for the $\ell_0$-penalized maximum likelihood estimator in a Gaussian setting. Notably, for the main problem we are considering of structure learning, van de Geer and Bühlmann

(2013) provide no formal guarantees. Loh and Bühlmann (2014) consider general linear SEM assuming the error variances are known, but once again there are no formal guarantees for structure learning. Finally, for nonlinear additive models—which are identifiable (Peters et al., 2012)—Bühlmann et al. (2014) introduce a three-step procedure involving preliminary neighbourhood selection, order search, and sparse additive regression and show that these steps are consistent in a high-dimensional setting under strong assumptions on the model.

This multi-step approach must be contrasted with traditional score-based approaches (see Section 3). The general outline of such methods is the following:

1. Estimate an initial (undirected, directed, or partially directed) graph $\mathcal{G}_0$,
2. Search for an optimal DAG structure $\widehat{\mathcal{G}}$ subject to the constraint that $\widehat{\mathcal{G}}$ is a subgraph of $\mathcal{G}_0$.

This approach is motivated by the fact that searching for an undirected or partially directed graph in the first step can be substantially faster than searching for a DAG. Since these ideas use multiple stages, they do not apply directly to the framework developed here. In particular, to the best of our knowledge, there are no results guaranteeing consistency in *structure learning* for any score-based method under a high-dimensional scaling.

### 1.2 Challenges

The general structure learning problem for Bayesian networks is plagued by three significant issues in practice:

1. NP-hardness,
2. Nonidentifiability,
3. Nonconvexity.

The NP-hardness of the general structure learning problem was established in Chickering et al. (2004), which indicates that it is unlikely that there exists a polynomial-time algorithm for solving this problem. Furthermore, by allowing regularizers which are singular at the origin (e.g. $\ell_1$, MCP, SCAD) the scoring function will also be nondifferentiable which is an additional challenge our framework must overcome.

These challenges make structure learning for Bayesian networks especially difficult, particularly in comparison to estimating a conditional independence graph, for which the estimation problem is statistically identifiable and the parameter space is convex. This is one of the many reasons why the high-dimensional properties of DAG estimation have proven much more difficult to ascertain.

## 1.3 Connections to other methods

The structure learning problem for Bayesian networks may be interpreted within two significant schools of thought (i) Graphical models, and (ii) Structural equation models. Both interpretations are useful to keep in mind, and it is important to clarify what can and cannot be done with observational data, which we do here.

### 1.3.1 Graphical models

In Section 3.2 (Remark 3.3), we will make an explicit connection between undirected graphical models the directed graphical models such as (1.2). In the Gaussian setting, the precision matrix $\Gamma = \Sigma^{-1}$ defines an undirected conditional independence graph, which is of significant interest in many applications. The problem of estimating $\Gamma$, commonly known as covariance selection or precision matrix estimation, has a long history in the statistical literature (Dempster, 1972), with recent approaches employing regularization in various incarnations (e.g. Meinshausen and Bühlmann, 2006; Chaudhuri et al., 2007; Banerjee et al., 2008; Fried-

man et al., 2008; Ravikumar et al., 2011). A detailed survey of recent progress in this area can be found in Pourahmadi (2013). We will not pursue this connection in detail in the present work, however, a few comments are in order.

First, as noted in Section 1.2, estimating a conditional independence graph is significantly easier both theoretically and computationally. Second, our approach is also distinct from existing methods that directly regularize Cholesky factors (Huang et al., 2006; Lam and Fan, 2009), as they make implicit use of an *a priori* ordering amongst the variables. As such, the consistency theory in Lam and Fan (2009) for the sparse Cholesky decomposition does not apply directly to our method. Finally, while there are important similarities between Bayesian networks and other undirected models such as Markov random fields and Ising models, our framework has so far only been applied to the former.

Part of the justification for our framework is that it produces sparse BNs that yield good fits to the true distribution, which is tantamount to producing good estimates of the inverse covariance matrix $\Gamma$. This will be established through the theory presented in Section 1, as well as empirically via the simulations discussed in Section 3.3. Because of the significance and popularity of covariance selection methods, it would of course be interesting to compare our estimate of $\Gamma$ to the methods cited in the above discussion. As our desire is to keep the focus on estimating Bayesian networks, such comparisons are left to future work.

### 1.3.2 Structural equation modeling

Much of the work on high-dimensional Bayesian networks relies on interpreting Bayesian networks in terms of structural equation models, which dates back to Wright (1918). This provides a natural interpretation of network edges in terms of coefficients of a regression model which we outline in Section 2.1.2, and provides an explicit mapping between a covariance matrix $\Sigma$ and its partial regression

coefficients (Wright, 1921, 1934). This has important consequences in terms of identifiability (Drton et al., 2011), and has led to some interesting connections between graphical modeling, structural equations, and algebraic statistics (Drton et al., 2009).

The focus of the present work is structure estimation of Bayesian networks based on observational data, which is not to be confused with the problem of causal inference. From our perspective, the data-generation mechanism is a multivariate Gaussian distribution as in (1.1), and thus there are many linear structural equations (1.3) that may generate (1.1). Our goal is to find the most parsimonious representation of the true distribution as a set of structural equations.

Alternatively, one could view the structural equation model (1.3) as the data-generating mechanism, in which case there is a *particular* set of structural equations that we wish to estimate. This is the perspective commonly adopted in the social sciences and in public health, in which the structural equations model causal relationships between the variables. In this set-up, it is well-known that one cannot expect to recover the directionality of causal relationships based on observational data alone, and the issues of causality, confounding and identifiability take center stage. Since we are only considering observational data, our framework does not address these questions.

## 2 Background and preliminaries

We will develop our framework by using a multivariate Gaussian distribution as our starting point, which we will then decompose into a Bayesian network in order to define our estimator. Our approach is purely algebraic, relying on the uniqueness of the Cholesky decomposition in order to factorize a Gaussian distribution into a set of linear structural equations. This approach will allow us to sidestep much of the traditional calculus of Bayesian networks.

**Notation and terminology.** For a general matrix $A = (a_{ij})_{n\times p} \in \mathbb{R}^{n\times p}$, its columns will be denoted using lowercase and single subscripts, so that

$$A = [\, a_1 \,|\, \cdots \,|\, a_p \,], \quad a_i \in \mathbb{R}^n \text{ for } i = 1, \ldots, p.$$

The square brackets signal that $A$ is a matrix with $p$ columns given by $a_1, \ldots, a_p$. The support of a matrix is defined by $\mathrm{supp}(B) := \{(i,j) : \beta_{ij} \neq 0\}$ and the cardinality of a set by $|\cdot|$.

Given a permutation $\pi$, $P_\pi$ denotes the permutation operator on matrices: For any matrix $A$, $P_\pi A$ is the matrix obtained by permuting the rows and columns of $A$ according to $\pi$, so that $(P_\pi A)_{ij} = a_{\pi(i)\pi(j)}$. For any integer $m$, we define $[m] = \{1, \ldots, m\}$ and $[m]_j = [m] - \{j\}$. For a vector $v \in \mathbb{R}^m$ and a subset $S \subset [m]$, we let $v_S \in \mathbb{R}^{|S|}$ denote the restriction of $v$ to the components in $S$. For a matrix $A \in \mathbb{R}^{n\times m}$, $A_S \in \mathbb{R}^{n\times|S|}$ denotes *column-wise* restriction to the columns in $S$. The maximum and minimum eigenvalues of a matrix are denoted by $r_{\max}(A)$ and $r_{\min}(A)$, respectively.

Unless otherwise specified, $\|\cdot\|$ will alway mean the standard Euclidean norm. We will also denote the $\ell_q$ norm by $\|\cdot\|_q$ and the Frobenius norm on matrices by $\|\cdot\|_F$.

Given two quantities $X$ and $Y$, we will write $X \lesssim Y$ to mean there exists a universal constant $a > 0$ such that $X \leq aY$, and analogously for $X \gtrsim Y$. We will also make use of the following asymptotic notation:

$$\begin{aligned}
f(n) = O(g(n)) &\iff f(n) \lesssim g(n) \text{ for sufficiently large } n,\\
f(n) = \Omega(g(n)) &\iff f(n) \gtrsim g(n) \text{ for sufficiently large } n,\\
f(n) \asymp g(n) &\iff g(n) \lesssim f(n) \lesssim g(n) \text{ for sufficiently large } n.
\end{aligned}$$

As a general rule, boldface type is reserved for *random* quantities that also depend on the sample size $n$; for example a random matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$ or the response $\mathbf{y} \in \mathbb{R}^n$ in a linear model. By contrast, a fixed design matrix $Z \in \mathbb{R}^{n\times m}$

is not bold since it is nonrandom (even though it depends on $n$), and similarly for the population vector $X$ since this does not depend on the sample size. This is very convenient for highlighting where in the argument randomness arises due to sampling. For probabilistic statements, there is always an implicit assumption of some underlying probability space $(\Omega, \mathcal{B}, \mathbb{P})$. Events defined on this probability space will usually be denoted by script font, e.g. $\mathcal{E}$, $\mathcal{F}$, etc. For an event $\mathcal{F}$, when we refer to *bounding* $\mathcal{F}$ we shall interpret this to mean we are bounding the probability of $\mathcal{F}$.

Finally, in order to disambiguate terminology conventions from different fields, we lay out some preliminary definitions here. Suppose $\widehat{\theta}$ is an estimator of the parameter $\theta^*$ that may depend on some tuning parameters $\alpha$, i.e. $\widehat{\theta} = \widehat{\theta}(\alpha)$. The problem of estimating $\text{supp}(\theta^*)$ will be called *model selection* or *support recovery*. For graphical models (i.e. when $\theta^*$ is a graph), this is also known as *structure learning*. By *parameter estimation*, we refer to controlling the estimation error $\|\widehat{\theta} - \theta^*\|_q$ for some $q > 0$. Typically, we are interested in (i) *Model selection consistency*, or $\mathbb{P}(\text{supp}(\widehat{\theta}) = \text{supp}(\theta^*)) \to 1$, and (ii) *Parameter estimation consistency*, or $\|\widehat{\theta} - \theta^*\|_q \xrightarrow{P} 0$. Finally, *(tuning) parameter selection* refers to the problem of selecting $\alpha$ (note that this is also frequently referred to as model selection in the literature).

### 2.1 Classical and linear Bayesian Networks

Strictly speaking, the classical theory of Bayesian networks is not a prerequisite to our framework. For comparative purposes and completeness, however, we briefly outline some of these classical notions here in an informal manner. For a more thorough treatment of this material, see Chickering (2003) or Spirtes et al. (2000). We will then contrast this material with an alternative approach via linear structural equations.

### 2.1.1 Classical Bayesian networks

Suppose $Z = (Z_1, \ldots, Z_p)$ is an arbitrary random vector defined on some abstract probability space $(\Omega, \mathcal{B}, \mathbb{P})$, and let $G = (V, E)$ be a directed graph whose vertices $V$ are identified with the components of $Z$.

- $X_j$ is a *parent* of $X_i$ if $X_j \to X_i$, i.e. $(X_j, X_i) \in E$.
- $X_j$ is a *descendant* of $X_i$ if there exists a directed path between $X_j$ and $X_i$. The descendants of $X_j$ are denoted by $\mathrm{de}(X_j)$ and the non-descendants are denoted by $\mathrm{nd}(X_j) := V - \mathrm{de}(X_j)$.
- The collection of all parents of a node $X_i$ is denoted $\mathrm{pa}(X_i)$.
- If $G$ contains no directed cycles, then we call $G$ a *directed acyclic graph* (DAG).

*Definition* 2.1 (Bayesian network). A directed graph $G = (V, E)$ is a *Bayesian network* (BN) for $X$ if there exist parameters $\theta_1, \ldots, \theta_p$ such that

$$\mathbb{P}(Z_1, \ldots, Z_p) = \mathbb{P}(Z_1 \mid \mathrm{pa}(X_1), \theta_1)\, \mathbb{P}(Z_2 \mid \mathrm{pa}(X_2), \theta_2) \cdots \mathbb{P}(Z_p \mid \mathrm{pa}(X_p), \theta_p).$$

Denote the joint distribution of $(Z_1, \ldots, Z_p)$ by $\mu$. Alternatively, we will say that $G$ is a Bayesian network for the distribution $\mu$. This abstract definition does not give us much intuition into *what* a Bayesian network really is, or how it is useful. To see this, we need the following concept:

*Definition* 2.2 (Local Markov condition). A directed graph $G = (V, E)$ satisfies the *local Markov condition* if and only if for each $j$, $X_j$ is independent of its nondescendants given its parents in $G$, i.e. if

$$X_j \perp\!\!\!\perp \mathrm{nd}(X_j) \mid \mathrm{pa}(X_j), \quad \forall\, j = 1, \ldots, p.$$

The utility of BNs comes from the following consequence of Definition 2.1, and is sometimes taken as an alternative definition of a Bayesian network:

**Lemma 2.1.** *Any Bayesian network for $X$ satisfies the local Markov condition.*

Thus, a Bayesian network somehow "encodes" conditional independence (CI) information about the joint distribution of the random vector $X$. The CI relations implied by the local Markov condition, in turn, entail other CI relations. These relations can be completely specified by the *d-separation* criterion, which we do will not go into here.

Thus, we have two sets of CI relations: Those implied (probabilistically) by the distribution $\mu$, and those implied (graphically) by the local Markov condition applied to $G$. This leads one to wonder when these sets coincide.

*Definition* 2.3. Let $G$ be a Bayesian network for the distribution $\mu$. $G$ is called *faithful* to $\mu$ if and only if all of the CI relations implied by $G$ are implied by $\mu$, and vice versa.

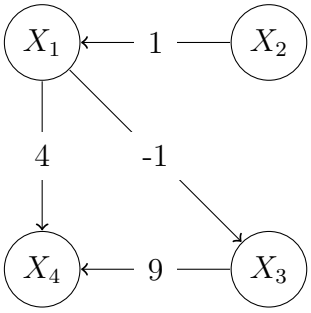
Naturally, when $G$ is faithful to $\mu$ we also say that $\mu$ is faithful to $G$. Other names for this condition in the literature include *perfect map* and *DAG isomorphism*.

Most structure learning algorithms in the literature are motivated by the existence of a faithful BN, and that this is the network one wishes to estimate. In practice, in order to ensure consistent estimation, stronger assumptions such as *strong faithfulness* are needed (Zhang and Spirtes, 2002), but recent work by Uhler et al. (2013) suggests that these assumptions are difficult to enforce in practice. Thus, it is of interest to relax this assumption wherever possible.

### 2.1.2 Linear Bayesian networks

Let $\varepsilon \in \mathbb{R}^p$ be an arbitrary random vector with independent components. Given coefficients $\widetilde{\beta}_{ij}$, we may consider the following linear structural equation model:

$$X_j = \sum_{i=1}^{p} \widetilde{\beta}_{ij} X_i + \varepsilon_j, \quad j = 1, \ldots, p. \tag{1.3}$$




$$X_1 = X_2 + \varepsilon_1$$
$$X_2 = \varepsilon_2$$
$$X_3 = -X_1 + \varepsilon_3$$
$$X_4 = 4X_1 + 9X_3 + \varepsilon_4.$$

$$\widetilde{B} = \begin{pmatrix} 0 & 0 & -1 & 4 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 1.1: An example of a linear structural equation model.

We can write this more compactly as follows:

$$X = \widetilde{B}^T X + \varepsilon, \quad \widetilde{B} = (\widetilde{\beta}_{ij}) \in \mathbb{R}^{p \times p}.$$

An illustration of this is given in Figure 2.1.2.

When $\widetilde{B}$ is not DAG, this is referred to as a *non-recursive* or *cyclic* structural equation model. The interesting case is when $\widetilde{B}$ corresponds to a DAG: If $\widetilde{B} \in \mathbb{D}$, then $\widetilde{B}$ is automatically a Bayesian network for $\mathbb{P}(X_1, \ldots, X_p)$ with $\mathrm{pa}(X_j) = \{X_i : \widetilde{\beta}_{ij} \neq 0\}$. This is readily verified according to factorization criterion in Definition 2.1. When $X$ satisfies (1.3) for some $\widetilde{B} \in \mathbb{D}$, we call $\widetilde{B}$ a *linear Bayesian network* for $X$. This is also known as a *recursive linear structural equation* in the SEM literature.

*Example* 2.1 (Multivariate Normal). Suppose $X = (X_1, \ldots, X_p) \sim \mathcal{N}(0, \Sigma)$. Then there exist matrices $\widetilde{B}$ and $\widetilde{\Omega}$ such that:

$$X = \widetilde{B}^T X + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \widetilde{\Omega}).$$

The matrix $\widetilde{B}$ is not unique and corresponds to a (modified) Cholesky factor of $\Sigma^{-1}$ under a fixed permutation of the variables. The matrix $\widetilde{B}$ can be interpreted as the weighted adjacency matrix of a directed acyclic graph, and $\widetilde{\Omega}$ is a diagonal matrix whose nonzero entries correspond to the conditional variances of each node given its parents in $\widetilde{B}$. Thus, $\widetilde{B}$ is a Bayesian network for $\mathcal{N}(0, \Sigma)$.

## 2.2 Gaussian DAG models

We assume throughout that the data are generated from a $p$-variate Gaussian distribution (1.1) where the covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$ is positive definite. Such a model can always be written as a set of Gaussian structural equations as in (1.3) (see Dempster, 1969), where the $\varepsilon_j$ are mutually independent with $\varepsilon_j \sim \mathcal{N}(0, \widetilde{\omega}_j^2)$, $\varepsilon_j$ is independent of $\mathrm{pa}(X_j) = \{X_i : \widetilde{\beta}_{ij} \neq 0\}$, and $\widetilde{\beta}_{jj} = 0$. This decomposition is not unique, and we will let $\widetilde{B} = (\widetilde{\beta}_{ij})$ denote any matrix of coefficients that satisfies (1.3), with corresponding variance matrix $\widetilde{\Omega} = \mathrm{var}(X - \widetilde{B}^T X)$. In a slight abuse of notation, we will identify a DAG $B$ with its weighted adjacency matrix, which we will also denote by $B = (\beta_{ij})$.

The space of (weighted) DAGs will be denoted by $\mathbb{D}$; formally,

$$\mathbb{D} = \{B \in \mathbb{R}^{p\times p} : B \text{ represents a DAG}\}.$$

The space of $p \times p$ diagonal matrices with positive entries on the diagonal is denoted by $\mathbb{R}_+^p$. Thus, in full generality, we have $(\widetilde{B}, \widetilde{\Omega}) \in \mathbb{D} \times \mathbb{R}_+^p$. Since the decomposition of a normal distribution as a linear SEM as in (1.3) is not unique, we can define the following equivalence class of DAGs:

*Definition* 2.4. Given a covariance matrix $\Sigma$ and $X \sim \mathcal{N}(0, \Sigma)$, define the *equivalence class* of $\Sigma$ to be

$$\mathfrak{D}(\Sigma) = \left\{\widetilde{B} \in \mathbb{D} : \widetilde{B} \text{ satisfies (1.2) for some } \widetilde{\Omega} \in \mathbb{R}_+^p\right\}.$$

Two (or more) DAGs in $\mathfrak{D}(\Sigma)$ will be called *permutation equivalent*, or simply *equivalent*.

This definition of equivalence in terms of equivalent parametrizations is indeed different from the usual definition of *distributional* or *Markov equivalence* that is common in the Bayesian network literature. Furthermore, while it is commonplace to assume that the true underlying distribution is faithful to the DAG

$\widetilde{B}$ (Definition 2.3), we have deliberately sidestepped further consideration of this assumption since our theory does not rely on faithfulness.

*Remark* 2.1. Strictly speaking, a Gaussian Bayesian network is specified by *both* a weighted adjacency matrix $B$ and a variance matrix $\Omega$, however, we will frequently refer to a BN simply by its adjacency matrix $B$. Although it may not be explicitly mentioned, when there is any ambiguity one may assume that there is an assumed variance matrix $\Omega$ paired with $B$.

### 2.3 Permutations and equivalence

In this section we wish to exhibit the connection between equivalent DAGs as defined in Definition 2.4 and the choice of a permutation of the variables. Recall that a *topological sort* of a directed graph is an ordering on the nodes, often denoted by $\prec$, such that the existence of a directed edge $X_k \to X_j$ implies $X_k \prec X_j$ in the ordering. A directed graph has a topological sort if and only if it is acyclic, and in general such a sort need not be unique.

When describing equivalent DAGs, it is easier to interpret an ordering in terms of a permutation of the variables. Let $\mathcal{P}$ denote the collection of all permutations of the indices $\{1, \ldots, p\}$. For an arbitrary matrix $A$ and any $\pi \in \mathcal{P}$, recall that $P_\pi A$ is the matrix obtained by permuting the rows and columns of $A$ according to $\pi$. Then a DAG can be equivalently defined as any graph whose adjacency matrix $B$ admits a permutation $\pi$ such that $P_\pi B$ is strictly triangular. When the order of the nodes in $P_\pi B$ matches a topological sort of $B$, that is if $X_k \prec X_j \implies \pi^{-1}(k) < \pi^{-1}(j)$, then the matrix $P_\pi B$ will be strictly upper triangular. For our purposes, however, it will be easier to use a *lower*-triangularization, which we now describe.

A DAG $B$ will be called *compatible with the permutation* $\pi$ if $P_\pi B$ is lower-triangular, which is equivalent to saying that $X_k \to X_j$ (i.e. $X_k \prec X_j$) in $B$ implies $\pi^{-1}(k) > \pi^{-1}(j)$. Conversely, $\pi$ will also be called *compatible* with $B$. Such a

permutation $\pi$ may be obtained by simply reversing any topological sort for $B$, so that parents come *after* their children. Formally, suppose $X_1 \prec X_2 \prec \cdots \prec X_p$ is a topological sort of $B$. Then the permutation

$$\pi(i) = p - i + 1, \quad i = 1, \ldots, p,$$

is compatible with $B$. Our decision to use lower-triangular matrices is for consistency with existing literature and to allow a convenient interpretation of the matrix $B$ as the weighted adjacency matrix of a graph. This will also simplify the technical discussion below (e.g. compare equations (1.4) and (1.13)).

We will now show that this equivalence class can be specified explicitly and constructively. Let $\mathcal{P}$ denote the class of permutations on $p$ elements and $\pi \in \mathcal{P}$ be a fixed permutation. Write $\Gamma := \Sigma^{-1}$. We may use the Cholesky decomposition to write $P_\pi \Gamma$ uniquely as

$$P_\pi \Gamma = (I - L)D^{-1}(I - L)^T, \tag{1.4}$$

where $L$ is strictly lower triangular and $D$ is diagonal. Define

$$\widetilde{B}(\pi) := P_{\pi^{-1}} L,$$
$$\widetilde{\Omega}(\pi) := P_{\pi^{-1}} D.$$

**Lemma 2.2.** *$X \sim \mathcal{N}(0, \Sigma)$ if and only if $X = \widetilde{B}(\pi)^T X + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \widetilde{\Omega}(\pi))$ and hence $\mathfrak{D}(\Sigma) = \{\widetilde{B}(\pi) : \pi \in \mathcal{P}\}$.*

Therefore, without further qualification, we shall always write an arbitrary element of $\mathfrak{D}(\Sigma)$ as $\widetilde{B}(\pi)$. The question now arises: which DAG $(\widetilde{B}(\pi), \widetilde{\Omega}(\pi))$ do we want to estimate? In the presence of experimental data, one may consider issues of causality, in which case each DAG represents a different causal structure. In the absence of such data, however, we can make no such distinctions. All of the DAGs in $\mathfrak{D}(\Gamma)$ are statistically indistinguishable based on observational data

alone, so a natural objective is to estimate the DAG that most parsimoniously represents the parameter $\Gamma$ in the sense that it has the fewest number of edges. This choice can also be motivated as it represents a so-called *minimal I-map*.

Under this assumption, there is an obvious connection between our approach and the *sparse Cholesky factorization* problem: Given a symmetric, positive definite matrix $A$, find a permutation $\pi$ such that the Cholesky factor of $P_\pi A$ has the fewest number of nonzero entries possible. In the oracle setting in which we know $\Gamma$, this is exactly the same problem as finding a permutation $\pi$ such that $\widetilde{B}(\pi)$ has the fewest number of edges. This connection has been studied in much more detail in Raskutti and Uhler (2014). They show that in this oracle setting, there is an equivalence between $\ell_0$-penalized estimation and sparse Cholesky factorization. In contrast, here we seek to estimate $\Gamma$ *as well as* find a sparse permutation $\pi$, and in this sense we provide a non-oracular, computationally feasible alternative to searching across all $p!$ permutations when $p$ is large.

*Example* 2.2. Suppose the DAG $B_0$ has the structure $X_1 \to X_2 \to X_3$ with edge weights $\beta_{12} = 1$ and $\beta_{23} = 1$, and $\omega_j = 1$ for each $j$. In this case, we have

$$B_0 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \Omega_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \Gamma(B_0, \Omega_0) = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}.$$

A topological sort for $B_0$ is $X_1 \prec X_2 \prec X_3$ (i.e. $B_0$ is already sorted), but $B_0$ is lower triangularized by the permutation $\pi_0 = (3, 2, 1)$ that swaps $X_1$ and $X_3$. Thus $B_0 = \tilde{B}_0(\pi_0)$.

Now consider another DAG, defined by

$$B_1 = \begin{pmatrix} 0 & 1/2 & 1 \\ 0 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix}, \quad \Omega_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \Gamma(B_1, \Omega_1) = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}.$$

Since $\Gamma(B_1, \Omega_1) = \Gamma(B_0, \Omega_0)$, the DAG $(B_1, \Omega_1)$ is equivalent to $(B_0, \Omega_0)$. Thus, according to Lemma 2.2, there must be a permutation $\pi_1$ such that $B_1 = \tilde{B}_0(\pi_1)$ and $\Omega_1 = \tilde{\Omega}_0(\pi_1)$. Indeed, if we let $\pi_1 = (2, 3, 1)$, one can check (by (1.4)) that these identities hold. Furthermore, if we reverse the order of the variables in $\pi_1$, we obtain a topological sort for $B_1$: $X_1 \prec X_3 \prec X_2$.

This example highlights two important points: (i) For the reader familiar with Markov equivalence of DAGs, it is obvious that $B_0$ and $B_1$ are not Markov equivalent, so our definition of equivalence is indeed different; and (ii) Equivalent DAGs in the sense we have defined need not have the same number of edges. This is the primary complication our framework must manage: Amongst all the DAGs which are equivalent to the true parameter $\Gamma$, we wish to find one which has the fewest number of edges.

# 3 Framework for score-based structure learning

Score-based structure learning has a long history in the machine learning and statistical literature. In this section we provide a brief introduction to this material.

## 3.1 Score-based learning overview

The idea behind score-based learning is to define a suitable *scoring function* over the space of parameterized DAGs, which in our case is given by the product space $\mathbb{D} \times \mathbb{R}_+^p$. For consistency with the literature, we first introduce a slightly more abstract version. Let $\mathcal{G}$ denote the space of directed acyclic graphs on $p$ vertices, i.e.,

$$\mathcal{G} = \{G = (V, E) : V = X \text{ and } G \text{ is acyclic}\}.$$

The discrete space $\mathcal{G}$ should be contrasted with the continuous space $\mathbb{D}$. We also let $\Theta$ be some abstract parameter space, typically a subset of $\mathbb{R}^K$ but not

necessarily so. Then a scoring function is a function

$$\text{score} : \mathcal{G} \times \Theta \to \mathbb{R} \tag{1.5}$$

that assigns a "score" to each parameterized DAG $(G, \theta) \in \mathcal{G} \times \Theta$. Naturally, this function also depends on the data $\mathbf{X}$, however, this dependence is suppressed in the notation above. In some frameworks, one eschews the parameters $\theta$ and only allows the scoring function to depend on the graph structure $G$, however, we do not consider this more restrictive version here.

The idea is that the higher the score, the better fit the graph $G$ is to the data $\mathbf{X}$. This suggests a natural estimator to be given by

$$\widehat{B}_{\text{score}} \in \operatorname*{arg\,max}_{(G,\theta)\in\mathcal{G}\times\Theta} \text{score}(G, \theta \,|\, \mathbf{X}). \tag{1.6}$$

We must be careful, however, since $\widehat{B}_{\text{score}}$ is not necessarily well-defined. In the sequel, we assume (a) that at least one such maximizer exists, and (b) $\widehat{B}_{\text{score}}$ is defined to be any such maximizer.

Some examples of scoring functions are given below.

*Example* 3.1 (Bayesian scoring criterion). Given some prior $\mathbb{P}(G, \theta)$, define a scoring function by

$$\text{score}(G, \theta) = \log \mathbb{P}(G, \theta) + \log \mathbb{P}(\mathbf{X} \,|\, G, \theta).$$

This is the Bayesian scoring criterion (Chickering, 2003), which is naturally motivated from a Bayesian perspective. Here, $\log \mathbb{P}(\mathbf{X} \,|\, G, \theta)$ is the marginal likelihood, and the score is just the log-posterior of $(G, \theta)$ given the data $\mathbf{X}$. This is by far the most popular scoring function in the literature.

*Example* 3.2 (Gaussian networks). For continuous Gaussian networks such as those considered in this work, Geiger and Heckerman (2013) defined a suitable Bayesian scoring criterion using a Wishart prior.

*Example* 3.3 ($\ell_1$-penalized profile likelihood). Fu and Zhou (2013) introduce a scoring function based on penalized profile likelihood. Given the Gaussian likelihood $L(B, \Omega \,|\, \mathbf{X})$ (see (1.9)), compute the *profile likelihood*

$$\widetilde{L}(B \,|\, \mathbf{X}) = \min_{\Omega \in \mathbb{R}^p_+} L(B, \Omega \,|\, \mathbf{X}).$$

For some weights $w_{ij}$, define a scoring function by

$$\text{score}(B \,|\, \mathbf{X}) = \widetilde{L}(B \,|\, \mathbf{X}) + \sum_{i,j} w_{ij} |\beta_{ij}|.$$

Note that this scoring function does not explicitly depend on the variance parameters given by $\Omega$.

A basic story about score-based methods for learning Bayesian networks can be given as follows (see Chickering, 2003, for more details). Call two DAGs $G$ and $G'$ *observationally equivalent* if and only if for every distribution $\mu$ such that $G$ is a BN for $\mu$, $G'$ is also a BN for $\mu$.[1] Instead of searching across the entire space of DAGs $\mathcal{G}$, we might consider searching over equivalence classes of DAGs. Using the Bayesian scoring criterion given in Example 3.1, a greedy search procedure will always recover an equivalence class that contains the true, faithful DAG $G_0$ in the limit $n \to \infty$ (Chickering, 2003, Lemma 10). In fact, it is not necessary to use the Bayesian scoring criterion as long as the scoring function is *locally consistent*, *score-equivalent*, and *decomposable*. Unfortunately, this result depends on the dimension of the model (i.e. $p$) remaining fixed as $n \to \infty$.

Since we use the linear model (1.2) in order to represent the parameters and the graph cohesively through the weighted adjacency matrix $B$, this abstract set-up simplifies considerably. In this setting, a scoring function can equivalently be defined by

$$\text{score} : \mathbb{D} \times \mathbb{R}^p_+ \to \mathbb{R}, \tag{1.7}$$

[1] This notion of equivalence is not to be confused with the definition of permutation or parametric equivalence given in Defintion 2.4.

and this is the way we will interpret our scoring functions in the sequel. In the next section we will define concrete scoring functions that directly exploit this way of representing the learning problem.

*Remark* 3.1. Given a weighted adjacency matrix $B \in \mathbb{D}$, let $\mathrm{gr} : \mathbb{D} \to \mathcal{G}$ denote the map that sends $B$ to the graph $G = (V, E) \in \mathcal{G}$ implied by interpreting $B$ as an adjacency matrix. If we let $\Theta = \mathbb{D} \times \mathbb{R}^p_+$, then any scoring function as defined in (1.7) can equivalently be considered as a function from $\mathcal{G} \times \Theta \to \mathbb{R}$ as in (1.5) by setting $\mathrm{score}(B, \Omega) = \mathrm{score}(\mathrm{gr}(B), (B, \Omega))$.

### 3.2 Loss functions and convexity

If $\mathbf{X} = [\mathbf{x}_1 \,|\, \cdots \,|\, \mathbf{x}_p]$ is an $n \times p$ data matrix of i.i.d. observations from (1.1), then we can rewrite (1.2) as

$$\mathbf{X} = \mathbf{X}\widetilde{B} + \mathbf{E} \tag{1.8}$$

where $\mathbf{E} \in \mathbb{R}^{n \times p}$ is the (random) matrix of noise vectors whose rows are drawn i.i.d. according to $\mathcal{N}(0, \widetilde{\Omega})$. This model has $p(p-1) + p = p^2$ free parameters, which we encode through two matrices given by $(\widetilde{B}, \widetilde{\Omega})$. Recall that $\widetilde{B} = (\widetilde{\beta}_{ij})$ and $\widetilde{\Omega} = \mathrm{diag}(\widetilde{\omega}_1^2, \ldots, \widetilde{\omega}_p^2)$.

There are thus two unknown parameters in (1.3):

$$B \in \mathbb{D}, \quad \Omega \in \mathbb{R}^p_+.$$

Given $n$ i.i.d. observations of the variables $(X_1, \ldots, X_p)$, the negative log-likelihood of the data $\mathbf{X} \in \mathbb{R}^{n \times p}$ may be derived as

$$L(B, \Omega \,|\, \mathbf{X}) = \frac{n}{2} \log |\Omega| + \frac{1}{2} \operatorname{tr}\left[(\mathbf{X} - \mathbf{X}B)\Omega^{-1}(\mathbf{X} - \mathbf{X}B)^T\right] \tag{1.9}$$

$$= \sum_{j=1}^{p} \left[\frac{n}{2} \log(\omega_j^2) + \frac{1}{2\omega_j^2} \|\mathbf{x}_j - \mathbf{X}\beta_j\|^2\right]. \tag{1.10}$$

Observe that $L(B, \Omega \,|\, \mathbf{X})$ is nonconvex; this fact will play an important role in the development of our method.

Suppose $\rho_\lambda : [0, \infty) \to [0, \infty)$ is a regularizer with $\lambda \geq 0$ as a tuning parameter that controls the amount of regularization. We allow for $\rho_\lambda$ to depend on other shape parameters as well. For any $B \in \mathbb{D}$, let

$$\rho_\lambda(B) := \sum_{i=1}^{p} \sum_{j=1}^{p} \rho_\lambda(|\beta_{ij}|).$$

In order to allow our framework to be valid for a general class of penalties, we allow $\rho_\lambda$ to be arbitrary for now. The details of choosing the penalty function will be discussed in Section 3.3.

Once $\rho_\lambda$ is chosen, one may seek to find a solution to

$$\arg\min_{B,\Omega} \big\{ L(B, \Omega \,|\, \mathbf{X}) + n\rho_\lambda(B) : B \text{ is a DAG} \big\}. \tag{1.11}$$

This is just the familiar penalized maximum likelihood estimator, under the (nonconvex) constraint that $B \in \mathbb{D}$. This program has three sources of nonconvexity:

1. The parameter space $\mathbb{D} \times \mathbb{R}_+^p$ is nonconvex,
2. The negative log-likelihood is nonconvex as a function of $(B, \Omega)$,
3. The regularizer $\rho_\lambda$ may be nonconvex.

The nonconvexity of the parameter space, it turns out, is an intrinsic property of the DAG estimation problem and will prove difficult to overcome. Notwithstanding, wherever possible, we will seek to convexify this program to the fullest extent possible. This will lead us to consider two different versions of the estimator defined by (1.11):

1. In Section 1, we will show how $L(B, \Omega \,|\, \mathbf{X})$ can be reparametrized in order to obtain a convex loss function. This convexification enables us to employ an efficient coordinate descent scheme in order to approximate solutions to the resulting program.

2. In Section 2, we will use a least squares loss, which is evidently equivalent to setting $\Omega = I_p$ in (1.11). The resulting program has a nice interpretation in terms of so-called *neighbourhood regressions*, which will be exploited in Chapter 3.

Thus, in general the estimators we will consider will not be the same as (1.11).

In this light, our approach is different from existing approaches in two ways:

1. Our choice of the penalty term $\rho_\lambda(\cdot)$ is different from traditional approaches and results in a continuous optimization problem,

2. The loss function will be convex, which accelerates computations and simplifies the theory.

*Remark* 3.2. The vast majority of the literature on Bayesian networks focuses on discrete data, in contrast to our work which assumes the data are Gaussian. As the motivation for this work is to scale penalized likelihood methods for high-dimensional data, the Gaussian case is a natural starting point, as much of the high-dimensional statistical theory is tailored towards this case. Recent work has shown how to adapt our techniques to the discrete case via multi-logit regession (Fu et al., 2014). Further generalizations to more general continuous distributions remain for future work. Finally, even though our method implicitly assumes the data are Gaussian, one may naively use our algorithm on discrete data and still obtain reasonable results (see Section 4.2).

*Remark* 3.3. Thus far we have viewed the distribution $\mathcal{N}(0, \Sigma)$ as the data-generating mechanism, rewriting this in terms of $(\widetilde{B}, \widetilde{\Omega})$ by using well-known properties of the Gaussian distribution. We could just as well have gone the other way around: Given a DAG $B \in \mathbb{D}$ and variance matrix $\Omega \in \mathbb{R}^p_+$, the parameters $(B, \Omega)$ uniquely define a structural equation model as in (1.3), and this

model defines a $\mathcal{N}(0, \Sigma)$ distribution. By (1.8), we have for any $(B, \Omega)$,

$$\Sigma = (I - B)^{-T}\Omega(I - B)^{-1}, \tag{1.12}$$

and hence $\Sigma$ is uniquely determined by $(B, \Omega)$. Considering instead the inverse covariance matrix $\Gamma = \Sigma^{-1}$, we can define

$$\Gamma = \Gamma(B, \Omega) = (I - B)\Omega^{-1}(I - B)^T. \tag{1.13}$$

By using (1.13) and defining $S_n := X^T X$, the negative log-likelihood in (1.10) can be rewritten in terms of $\Gamma = \Gamma(B, \Omega)$ directly as

$$L(\Gamma \,|\, \mathbf{X}) = -\frac{n}{2} \log\det \Gamma + \frac{1}{2} \operatorname{tr}(\Gamma S_n). \tag{1.14}$$

By combining (1.10) and (1.14), we have $L(B, \Omega \,|\, \mathbf{X}) = L(\Gamma(B, \Omega) \,|\, \mathbf{X})$. This expression shows how the weighted adjacency matrix of a DAG can be considered as a reparametrization of the usual normal distribution, and gives us an explicit connection between inverse covariance estimation and DAG estimation.

### 3.3 Choice of penalty function

The standard approach in the Bayesian network literature is to use AIC or BIC to penalize overly complex models, although $\ell_1$-based methods have been slowly gaining in popularity. Traditionally, $\ell_1$ regularization is viewed as a convex relaxation of optimal $\ell_0$ regularization, which results in a convex program that is computationally efficient to solve. Unfortunately, in our situation the constraint that $B$ is a DAG is also nonconvex, so there is little hope to recover a convex program. Thus, there is nothing lost in using concave penalties, which have more attractive theoretical properties than $\ell_1$-based alternatives. We will briefly review the details here.

Fan and Li (2001) introduce the fundamental theory of concave penalized likelihood estimation and outline three principles that should guide any variable

selection procedure: unbiasedness, sparsity, and continuity. They argue that the following conditions are sufficient to guarantee that a penalized least squares estimator has these properties:

1. (Unbiasedness) $\rho'_\lambda(t) = 0$ for large $t$;
2. (Sparsity) The minimum of $t + \rho'_\lambda(t)$ is positive;
3. (Continuity) The minimum of $t + \rho'_\lambda(t)$ is attained at zero.

Condition (1) only guarantees unbiasedness for large values of the parameter; in general we cannot expect a penalized procedure to be totally unbiased. Note also that (1-3) imply that $\rho_\lambda$ must be a concave function of $t$.

In the methodological developments which follow, it will not be necessary to assume that the penalty function is concave. The theory developed in Chapter 3 will illuminate how the properties of the penalty function influence the theoretical properties of the estimator (2.3, 2.4), however, the only strict requirement on the penalty function needed for the proposed algorithm is that there exists a corresponding threshold function $S(\cdot, \lambda)$ to perform the single parameter updates (see Section 2.2 for details). Examples of common penalty functions in the literature include $\ell_1$ (or Lasso, Tibshirani, 1996), SCAD (Fan and Li, 2001) and MCP (Zhang, 2010). The SCAD penalty represents a smooth quadratic interpolation between the $\ell_1$ and $\ell_0$ penalties, and the MCP minimizes the maximum concavity by shifting the $\ell_1$ portion to the origin. See Figure 1.2 for a visual comparison of these three penalties. The key difference between the $\ell_1$ penalty and SCAD or MCP is the flat part of the penalty, which helps to reduce bias.

As the $\ell_1$ penalty does not satisfy the unbiasedness condition (Condition (1) above), it yields biased estimates in general. Allowing ourselves to be motivated by some recent developments in regression theory, we can say even more. There the assumptions required for consistency are rather strong and require a so-called

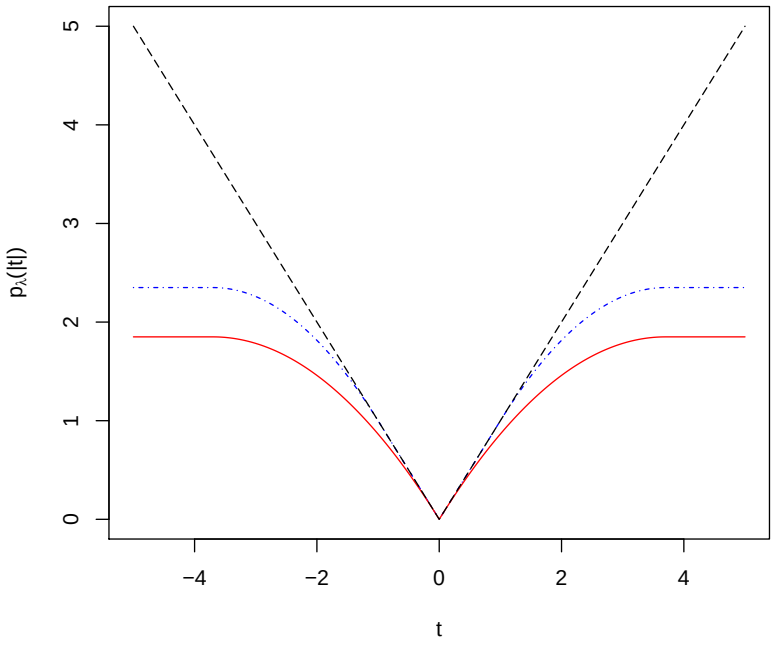



Figure 1.2: Comparison of penalty functions. The red, solid line is the minimax concave penalty (MCP), the blue dot-dashed line is the smoothly clipped absolute deviation penalty (SCAD), and the black dashed line is the $\ell_1$ or Lasso penalty. Both the MCP and SCAD represent smooth interpolations of the $\ell_1$ and $\ell_0$ penalties and hence have better statistical properties, whereas the $\ell_1$ penalty exhibits bias due to its divergence as $t \to \infty$.

*irrepresentability condition* (Zhao and Yu, 2006), also known as *neighbourhood stability* (Meinshausen and Bühlmann, 2006). The bias issues can be circumvented by employing the adaptive Lasso (Zou, 2006), an idea which has been explored in Fu and Zhou (2013). Recent theoretical analysis of regularization with concave penalties has shown that, compared to $\ell_1$ penalties, the assumptions on the data needed for consistency can be relaxed substantially. Generalizing these ideas to Bayesian network models is the main focus of Chapter 3. These theoretical results are supported by the comparisons in Chapter 2.

## 4 Summary of results obtained in this dissertation

At a high-level, the contributions from this dissertation can be summarized as follows:

1. A fast algorithm for score-based structure learning that accomplishes each of the goals set out in the introduction (Chapter 2, Section 2),

2. Extensive experiments to show that the proposed algorithm outperforms existing methods in high-dimensions (Chapter 2, Section 3),

3. A proof of asymptotic consistency in the traditional large sample set-up (Chapter 3, Section 1),

4. A new high-dimensional analysis based on neighbourhood regression (Chapter 3, Section 2),

5. Nonasymptotic bounds on:

    - The probability of false selection (Theorem 2.1)
    - The $\ell_1/\ell_2$ estimation error (Theorem 2.2),

# CHAPTER 2

# Algorithms

We begin by introducing a complete algorithm for estimating a Gaussian Bayesian network based on the model (1.2). This algorithm revolves around two core ideas:

1. A reparametrized negative log-likelihood,

2. Coordinate descent.

After introducing an estimator (Section 1) and an algorithm for approximating this estimator (Section 2), we proceed with a comparison of this method against several popular structure learning algorithms from the literature using both simulated (Section 3) and unsimulated (Section 4) networks. Theoretical considerations are deferred until Chapter 3.

## 1 Reparameterized penalized maximum likelihood

The first step in our proposal will be to reparameterize the Gaussian negative log-likelihood.

### 1.1 Convex reparametrization

One of the drawbacks of the loss in (1.10) is that it is nonconvex, which complicates the minimization of the penalized loss. If we minimize (1.10) with respect to $\Omega$ and use the adaptive Lasso penalty, we obtain the estimator described in Fu and Zhou (2013) (Example 3.3). By keeping the variance parameter, however, we can

exploit a clever reparametrization of the problem, introduced in Städler et al. (2010), which leads to a convex loss.

The idea is to define new variables by $\rho_j = 1/\omega_j$ and $\phi_{ij} = \beta_{ij}/\omega_j$, which yields the reparametrized negative log-likelihood

$$L(\Phi, R \,|\, \mathbf{X}) = \sum_{j=1}^{p} \left[ -n \log(\rho_j) + \frac{1}{2} \|\rho_j \mathbf{x}_j - \mathbf{X}\phi_j\|^2 \right], \tag{2.1}$$

where $\Phi = [\phi_1 \,|\, \cdots \,|\, \phi_p] \in \mathbb{D}$ and $R = \operatorname{diag}(\rho_1, \ldots, \rho_p) \in \mathbb{R}^p_+$. The loss function in (2.1) is easily seen to be convex. Furthermore, if we interpret $\Phi$ as the adjacency matrix of a directed graph, then $\Phi$ has exactly the same edges and nonzero entries as $B$, and thus in particular $\Phi$ is acyclic if and only if $B$ is acyclic.

In analogy with the parametrization $(B, \Omega)$, define

$$\Gamma(\Phi, R) = (R - \Phi)(R - \Phi)^T, \tag{2.2}$$

which gives a formula for the inverse covariance matrix in the parametrization $(\Phi, R)$. Note that if $\Phi = \Phi(B, \Omega)$ and $R = R(B, \Omega)$, then $\Gamma(B, \Omega) = \Gamma(\Phi, R)$, and hence also $L(B, \Omega) = L(\Phi, R)$.

This reparametrization is *not* the same as the likelihood in (1.14), which is well-known to lead to a convex program (see, for instance, Boyd and Vandenberghe, 2009, §7.1). Indeed, plugging (1.13) into (1.14) leads back to (1.10), which is nonconvex in the parameters $\beta_{ij}$ and $\omega_j$. To wit, the problem is convex in $\Gamma$ but not in $(B, \Omega)$. The key insight from Städler et al. (2010) is to observe that one may recover convexity by switching to the alternate parametrization in terms of $\phi_{ij}$ and $\rho_j$. Unfortunately, the DAG constraint in (1.11) is still nonconvex. The idea behind this reparametrization is to allow our algorithm to exploit convexity wherever possible in order to reap at least *some* computational and analytical gains. As we shall see, the gains are indeed significant.

## 1.2 The estimator

We are now prepared to introduce the formal definition of the DAG estimator which will be the focus of this chapter.

Fix a regularizer $\rho_\lambda(\cdot)$. Then given

$$\begin{aligned}(\widehat{\Phi}, \widehat{R}) &:= \arg\min_{\Phi, R} \big\{ L(\Phi, R \,|\, \mathbf{X}) + n \sum_{i,j} \rho_\lambda(|\phi_{ij}|) : \Phi \text{ is a DAG} \big\} \qquad (2.3) \\ &= \arg\min_{\Phi, R} \Big\{ \sum_{j=1}^{p} \Big[ -n \log(\rho_j) + \frac{1}{2} \|\rho_j \mathbf{x}_j - \mathbf{X}\phi_j\|^2 \Big] \\ &\qquad\qquad + n \sum_{i,j} \rho_\lambda(|\phi_{ij}|) : \Phi \text{ is a DAG} \Big\},\end{aligned}$$

we define an estimator by

$$(\widehat{B}, \widehat{\Omega}) = \begin{cases} \hat{\beta}_{ij} = \hat{\phi}_{ij}/\hat{\rho}_j, & i \neq j \\ \hat{\beta}_{jj} = 0, & \\ \hat{\omega}_j^2 = 1/\hat{\rho}_j^2, & j = 1, \ldots, p \end{cases} \qquad (2.4)$$

where $\hat{\phi}_{ij}$ and $\hat{\rho}_j$ denote the respective components of $(\widehat{\Phi}, \widehat{R})$. When we wish to emphasize the estimator's dependence on $\lambda$, we shall denote it by $(\widehat{\Phi}_\lambda, \widehat{R}_\lambda)$.

There is an intuitive interpretation of the problem in (2.3): By the identity $L(\Phi, R \,|\, \mathbf{X}) = L(\Gamma(\Phi, R) \,|\, \mathbf{X})$, it is evident that the loss function for $(\Phi, R)$ is simply the negative log-likelihood of the resulting estimate of $\Gamma = \Gamma(\Phi, R)$. In this sense, we are implicitly approximating the true parameter $\Gamma$. The key ingredient, however, is the penalty term: We only penalize the edge weights $\phi_{ij}$, which has the effect of self-selecting for DAGs which are sparse. In this way, the solution to (2.3) produces a sparse Bayesian network whose distribution is close to the true, underlying distribution.

For the algorithm, the only requirement on the penalty function needed will be the existence of a corresponding threshold function $S(\cdot, \lambda)$ (see Section 2.2 for

details). For our computations we chose to use the MCP, defined for $t \geq 0$ by

$$\rho_\lambda(t;\gamma) := \lambda\left(t - \frac{t^2}{2\lambda\gamma}\right)1(t < \lambda\gamma) + \frac{\lambda^2\gamma}{2}1(t \geq \lambda\gamma) \tag{2.5}$$
$$= \begin{cases} \lambda\left(t - \frac{t^2}{2\lambda\gamma}\right), & t < \lambda\gamma, \\ \frac{\lambda^2\gamma}{2}, & t \geq \lambda\gamma. \end{cases}$$

The $\gamma$ parameter in the MCP controls the concavity of the penalty: As $\gamma \to 0$, MCP approaches the $\ell_0$ penalty and as $\gamma \to \infty$, it approaches the $\ell_1$ penalty. In the sequel we will thus refer to $\gamma$ as the *concavity parameter* and $\lambda$ as the *regularization parameter*. From the above formula, MCP is easily seen to be a quadratic spline between the origin and the $\ell_0$ penalty with a knot at $t = \lambda\gamma$. To demonstrate the differences and potential advantages of a concave penalty, we also implemented our method with the $\ell_1$ penalty, $\rho_\lambda(|t|) = \lambda|t|$.

*Remark* 1.1. For most choices of the penalty, the solution to (2.3) is *not* the same as the solution to (1.11) since we are penalizing different terms. In the original parametrization, we penalize the coefficients $\beta_{ij}$, whereas after reparametrizing we are penalizing the rescaled coefficients $\phi_{ij} = \beta_{ij}/\omega_j$. Thus we are also penalizing choices of coefficients which overfit the data, i.e., which have small $\omega_j$. A notable exception, however, occurs when $\rho_\lambda(\cdot)$ is taken to be the $\ell_0$ penalty. In this special case, the problems in (1.11) and (2.3) are the same, and thus in particular the analysis in van de Geer and Bühlmann (2013) applies.

## 2 CCDr algorithm

Both the objective function and the constraint set in (2.3) are nonconvex, which makes traditional gradient descent algorithms for performing the necessary minimization difficult to implement. One could employ naive gradient descent to find a local minimizer of (2.3), but it would still be difficult to enforce the DAG constraint. Thus, a different approach must be taken altogether. Extending the

algorithm of Fu and Zhou (2013), we employ a cyclic coordinate-descent based algorithm that relies on checking the DAG constraint at each update. By properly exploiting the sparsity of the estimates and the reparametrization (2.1), however, we will be able to perform the single parameter updates and enforce the constraint with ruthless efficiency.

## 2.1 Overview

Before outlining the technical details of implementing our algorithm, we pause to provide a high-level overview of our approach.

The idea behind cyclic coordinate descent is quite simple: Instead of minimizing the objective function over the entire parameter space simultaneously, we restrict our attention to one variable at a time, perform the minimization in that variable while holding all others constant (hereafter referred to as a *single parameter update*), and cycle through the remaining variables. This procedure is repeated until convergence. Coordinate descent is ideal in situations in which each single parameter update can be performed quickly and efficiently. For more details on the statistical perspective on coordinate descent, see Wu and Lange (2008); Friedman et al. (2007).

Moreover, due to acyclicity, we know *a priori* that the parameters $\phi_{kj}$ and $\phi_{jk}$ cannot simultaneously be nonzero for $k \neq j$. This suggests performing the minimization in blocks, minimizing over $\{\phi_{kj}, \phi_{jk}\}$ simultaneously. An immediate consequence of this is that we reduce the number of free parameters from $p^2$ to $p(p-1)/2 + p$, a substantial savings.

In order to enforce acyclicity, we use a simple heuristic: For each block $\{\phi_{kj}, \phi_{jk}\}$, we check to see if adding an edge from $X_k \to X_j$ induces a cycle in the estimated DAG. If so, we set $\phi_{kj} = 0$ and minimize with respect to $\phi_{jk}$. Alternatively, if the edge $X_j \to X_k$ induces a cycle, we set $\phi_{jk} = 0$ and minimize with respect to $\phi_{kj}$.

If neither edge induces a cycle, we minimize over both parameters simultaneously.

Before we outline the details, let us introduce some functions which will be useful in the sequel. Define

$$Q(\Phi, R) := L(\Phi, R) + \sum_{i,j} \rho_\lambda(|\phi_{ij}|) \tag{2.6}$$

to be our objective function for coordinate descent. Note that we have suppressed the dependence of the log-likelihood on the data $\mathbf{X}$ as well as the dependence of the penalty term on $n$. In fact, in the computations we may treat $n$ as fixed, so we can absorb this term into the penalty function $\rho_\lambda$. This simply amounts to rescaling the regularization parameter $\lambda$, which causes no problems in computing $(\widehat{\Phi}, \widehat{R})$. Thus solving (2.3) is equivalent to minimizing $Q$.

Now define the single-variable functions

$$Q_1(\phi_{kj}) = \frac{1}{2} \left\| \rho_j \mathbf{x}_j - \sum_{i=1}^{p} \phi_{ij} \mathbf{x}_i \right\|^2 + \rho_\lambda(|\phi_{kj}|), \tag{2.7}$$

$$Q_2(\rho_j) = -n \log \rho_j + \frac{1}{2} \left\| \rho_j \mathbf{x}_j - \sum_{i=1}^{p} \phi_{ij} \mathbf{x}_i \right\|^2. \tag{2.8}$$

The function $Q_1$ is $Q(\Phi, R)$ in (2.6) considered as a function of the single parameter $\phi_{kj}$, while holding the other $p^2 - 1$ variables fixed and ignoring terms that do not depend on $\phi_{kj}$, and $Q_2$ is the corresponding function for the parameter $\rho_j$. We express the dependence of $Q_1$ and $Q_2$ on $k$ and/or $j$ implicitly through their respective argument, $\phi_{kj}$ or $\rho_j$.

An overview of the algorithm is given in Algorithm 1. We use the notation $\phi_{kj} \Leftarrow 0$ to mean that $\phi_{kj}$ must be set to zero due to acyclicity, as outlined above. The remainder of this section is devoted to the details of implementing the above algorithm, which we call Concave penalized Coordinate Descent with reparametrization (CCDr).

## 2.2 Coordinate descent

In what follows, we assume that the data have been appropriately normalized so that each column unit norm, $\|\mathbf{x}_j\|^2 == 1$. Furthermore, although the details of the algorithm do not depend on the choice of penalty, we will focus on the MCP and $\ell_1$ penalties, as these are the methods implemented and discussed in Sections 3 and 4.

### 2.2.1 Update for $\phi_{kj}$

Mazumder et al. (2011) show that the minimum of (2.7) can be found by solving

$$\arg\min_{\beta} Q^1(\beta), \quad \text{where } Q^1(\beta) := \frac{1}{2}(\beta - \tilde{\beta})^2 + \rho_\lambda(|\beta|). \tag{2.9}$$

The solution to (2.9) is given by a so-called threshold function which is associated to each choice of penalty. For the MCP with $\gamma > 1$ this is defined by

$$S_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \le \lambda, \\ \text{sgn}(\tilde{\beta}) \left( \frac{|\tilde{\beta}| - \lambda}{1 - 1/\gamma} \right), & \lambda < |\tilde{\beta}| \le \lambda\gamma, \\ \tilde{\beta}, & |\tilde{\beta}| > \lambda\gamma. \end{cases} \tag{2.10}$$

For the $\ell_1$ penalty, we have

$$S(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \le \lambda, \\ \text{sgn}(\tilde{\beta})(|\tilde{\beta}| - \lambda), & |\tilde{\beta}| > \lambda. \end{cases} \tag{2.11}$$

To see how to convert (2.7) into (2.9), write $\mathbf{X} = (x_{ij})$ and note that

$$\begin{aligned} Q_1(\phi_{kj}) &= \frac{1}{2} \sum_{h=1}^{n} \left( \rho_j x_{hj} - \sum_{i \neq k} \phi_{ij} x_{hi} - \phi_{kj} x_{hk} \right)^2 + \rho_\lambda(|\phi_{kj}|) \\ &= \frac{1}{2} \sum_{h=1}^{n} x_{hk}^2 \left( \frac{1}{x_{hk}} r_{kj}^{(h)} - \phi_{kj} \right)^2 + \rho_\lambda(|\phi_{kj}|), \end{aligned} \tag{2.12}$$

where $r_{kj}^{(h)} := \rho_j x_{hj} - \sum_{i \neq k} \phi_{ij} x_{hi}$. Expanding the square in the last line and using $\sum_h x_{hk}^2 = 1$,

$$Q_1(\phi_{kj}) = \frac{1}{2} \left\{ \sum_{h=1}^{n} (r_{kj}^{(h)})^2 - 2\phi_{kj} \sum_{h=1}^{n} x_{hk} r_{kj}^{(h)} + \phi_{kj}^2 \right\} + \rho_\lambda(|\phi_{kj}|) \tag{2.13}$$

$$= \frac{1}{2} \left( \phi_{kj} - \sum_{h=1}^{n} x_{hk} r_{kj}^{(h)} \right)^2 + \rho_\lambda(|\phi_{kj}|) + \text{const.} \tag{2.14}$$

The constant term in (2.14) does not depend on $\phi_{kj}$ and hence does not affect the minimization of $Q_1$. Thus minimizing $Q_1(\phi_{kj})$ is equivalent to minimizing $Q^1(\beta)$ in (2.9) with $\tilde{\beta} = \sum_h x_{hk} r_{kj}^{(h)}$. Hence for MCP with $\gamma > 1$,

$$\arg\min Q_1(\phi_{kj}) = S_\gamma \left( \sum_h x_{hk} r_{kj}^{(h)}, \lambda \right), \tag{2.15}$$

and similarly for the $\ell_1$ penalty. The existence of a closed-form solution to the single parameter update for $\phi_{kj}$ is a key ingredient to our method, and is one of the reasons we chose the MCP and $\ell_1$ penalties in our comparisons. Many other penalty functions, however, allow for closed-form solutions to (2.9), and our algorithm applies for any such penalty function.

### 2.2.2 Update for $\rho_k$

The single parameter update for $\rho_j$ is straightforward to compute and is given by

$$\arg\min Q_2(\rho_j) = \frac{c + \sqrt{c^2 + 4n}}{2}, \quad \text{with } c = \sum_{i \neq j} \phi_{ij} \sum_h x_{hi} x_{hj}. \tag{2.16}$$

Since $Q_2(\rho_j)$ is a strictly convex function, this is the only minimizer.

## 2.3 Regularization paths

In practice, it is difficult to select optimal choices of the penalty parameters $(\lambda, \gamma)$ in advance. Thus it is necessary to compute several models at many discrete

choices of $(\lambda_i, \gamma_j)$, and then do parameter selection. In testing, we observed a dependence on the concavity parameter $\gamma$, however, for simplicity we will consider $\gamma$ fixed in the sequel, and postpone further study of the method's dependence on $\gamma$ to future work.

The regularization parameter $\lambda$, on the other hand, has a strong effect on the estimates. In particular, as $\lambda \to \infty$, $\widehat{\Phi}(\lambda) \to \mathbf{0}$, and as $\lambda \to 0$ we obtain the unpenalized maximum likelihood estimates. It is thus desirable to obtain a sequence of estimates $(\widehat{\Phi}_{\lambda_i}, \widehat{R}_{\lambda_i})$ for some sequence $\lambda_i > \lambda_{i+1} > 0$, $i = 0, 1, \ldots, L$. In practice, we will always choose $\lambda_0$ so that $\widehat{\Phi}(\lambda_0) = \mathbf{0}$, with successive values of $\lambda_i$ decreasing on a linear scale. One can easily check that if we use an initial guess of $\Phi^0 = \mathbf{0}$, then the choice $\lambda_0 = n^{1/2}$ ensures that the null model is a local minimizer of $Q$.

Once we have estimated a sequence of models $(\widehat{\Phi}_{\lambda_i}, \widehat{R}_{\lambda_i})$, $i = 0, 1, \ldots, L$, we must choose the best model from these $L + 1$ models. This is the parameter selection problem, and is beyond the scope of this work. The present work should be considered a "proof of concept," showing that under the right conditions, there exists a $\lambda$ that estimates the true DAG with high fidelity. The problem of correctly selecting this parameter is left for future work, but some preliminary empirical analysis is provided in Section 3.5. See Wang et al. (2007) for some positive results concerning the SCAD penalty, and Fu and Zhou (2013) for a relevant discussion of some difficulties that are idiosyncratic to structure estimation of BNs. In particular, it is worth re-emphasizing here that cross-validation is suboptimal, and should be avoided.

## 2.4 Implementation details

As presented so far, the CCDr algorithm is not particularly efficient. Fortunately, there are several computational enhancements we can exploit to greatly improve

the efficiency of the algorithm. Many of these ideas are adapted from Friedman et al. (2010), and the reader is urged to refer to this paper for an excellent introduction to coordinate descent for penalized regression problems.

In implementing the CCDr algorithm, we use warm starts and an active set of blocks as described in Friedman et al. (2010); Fu and Zhou (2013). We also use a sparse implementation of the parameter matrix $\Phi$ to speed up internal calculations. Naive recomputation of the $n$ weighted residual factors $r_{kj}^{(h)}$ for $h = 1, \ldots, n$ for every update incurs a cost of $O(np)$ operations, which is prohibitive in general, and is the main bottleneck in the algorithm. Friedman et al. (2010) observe that this calculation can be reduced to $O(p)$ operations by noting that the sum in (2.15) can be written as

$$\sum_{h=1}^{n} x_{hk} r_{kj}^{(h)} = \rho_j \langle \mathbf{x}_j, \mathbf{x}_k \rangle - \sum_{i \neq k} \phi_{ij} \langle \mathbf{x}_i, \mathbf{x}_k \rangle. \tag{2.17}$$

The inner products above do not change as the algorithm progresses, and hence can be computed once at a cost of $O(n^2 \log n)$ operations. This is a substantial improvement over several million $O(np)$ computations, which is typical for large $p$.

Similar reasoning applies to the computation of (2.16), which highlights why the repara-metrization (2.1) is useful: the single parameter update for each $\rho_j$ only requires $O(p)$ operations, compared with $O(p^2)$ required operations for the standard residual estimate for $\omega_j^2$ in the original parametrization. Since we perform $p$ of these updates in each cycle, we reduce the total number of operations per cycle from $O(p^3)$ down to $O(p^2)$, which is a substantial savings. Moreover, by leveraging sparsity, both (2.15) and (2.16) become $O(1)$ calculations when the maximum number of parents per node is bounded.

As stated, our algorithm will take a pre-specified sequence of $\lambda$-values and compute an estimate $(\widehat{\Phi}_{\lambda_i}, \widehat{R}_{\lambda_i})$ for all $L + 1$ choices of $\lambda_i$. In general, we do not know in advance what the smallest value of $\lambda$ appropriate for the data is,

and we typically choose $\lambda_L$ as some small value. Since the model complexity (in terms of the number of edges) increases as $\lambda$ decreases, more and more time is spent computing complex models for small $\lambda$. We can exploit these facts in order to avoid wasting time on computing unnecessarily complex models. As the algorithm proceeds calculating estimates for each $\lambda_i$, if the estimated number of edges $\hat{s}_i := s_{\widehat{B}(\lambda_i)}$ is too large, we know that we need not continue computing new models for smaller $\lambda$. We can justify this as follows: *either* the true model is sparse, in which case we know that complex models with $\hat{s}_i$ large can be ignored, *or* the true model is *not* sparse, in which case our algorithm is less competitive. Thus, in this sense, prior knowledge or intuition of the sparsity of the true model is needed. In practice, we implement this by halting the algorithm whenever $\hat{s}_i > \alpha p$, where $\alpha > 0$ is a pre-specified parameter. While the choice of $\alpha$ should be application driven, we will use $\alpha = 3$ unless reported otherwise. In the sequel, $\alpha$ shall be referred to as the *threshold parameter*.

### 2.5 Full algorithm

A complete, detailed description of the algorithm is given in Algorithm 2, including the implementation details discussed in the previous section. We refer to steps (1-2) of Algorithm 1 as a single "sweep" of the algorithm (i.e. performing a single parameter update for every parameter in the active set).

Finally, note that it is trivial to adapt the *SparseNet* procedure from Mazumder et al. (2011) to our algorithm in order to compute a *grid* of estimates

$$(\widehat{\Phi}_{\lambda_i,\gamma_j}, \widehat{R}_{\lambda_i,\gamma_j}), \quad i = 0, \ldots, L,\ j = 0, \ldots, J,$$

if one wishes to adjust $\gamma$ in addition to $\lambda$.

# 3 Numerical simulations

In order to assess the accuracy and efficiency of the CCDr algorithm, we compared our algorithm with four other well-known structure learning algorithms: the PC algorithm (Spirtes and Glymour, 1991), the max-min hill-climbing algorithm (MMHC; Tsamardinos et al., 2006), Greedy Equivalent Search (GES; Chickering, 2003), and standard greedy hill-climbing (HC). This selection was based on a pre-screening in which we compared the performance of several more algorithms in order to select those which showed the best performance in terms of accuracy and efficiency, and is by no means intended to be exhaustive. We were mainly interested in the accuracy and timing performance of each algorithm as a function of the model parameters $(p, s_0, n)$. Details on the implementations used and our experimental choices will be discussed in Section 3.1.

Our comparisons thus consist of two score-based methods (GES, HC), one constraint-based method (PC), and one hybrid method (MMHC). For brevity, in the ensuing discussion we will frequently refer to both PC and MMHC as constraint-based methods since both methods employ some form of constraint-based search whereas GES and HC do not. In order to compare the effects of regularization, we also compared each of these algorithms to two implementations of CCDr: One using MCP as the penalty (CCDr-MCP), and a second with the $\ell_1$ penalty (CCDr-$\ell_1$). This gives us a total of six algorithms overall. To offer a sense of scale, the experiments in this section total over $140,000$ individual DAG estimates for almost 1,000 "gold-standard" DAGs.

We begin with a comprehensive evaluation in low-dimensions ($n \geq p$) of all six algorithms using randomly generated DAGs, the main purpose of which is to show that hill-climbing and GES are significantly slower and less accurate in comparison with the other approaches. This supports our first claim that CCDr represents a clear improvement over existing score-based methods. We then move onto a

similar assessment for high-dimensional data, which will show the advantages of our method over the constraint-based methods when sample sizes are limited and the number of nodes increases. Once this has been done, we show that our method scales efficiently on graphs with up to 2000 nodes as well as discuss some issues related to parameter selection and timing. We conclude this section with some detailed discussions about our experiments.

### 3.1 Experimental set-up

All of the algorithms were implemented in the `R` language for statistical computing (R Core Team, 2014). For the PC and GES algorithms, we used the `pcalg` package (version 2.0-3, Kalisch et al., 2012), and for the MMHC and HC algorithms we used the `bnlearn` package (version 3.6, Scutari, 2010). Both packages employ efficient, optimized implementations of each algorithm, and were updated as recently as July 2014. At the time of the experiments, these were the most up-to-date publicly available versions of either package. All of the tests were performed on a late 2009 Apple iMac with a 2.66GHz Intel Core i5 processor and 4GB of RAM, running Mac OS X 10.7.5.

For all the experiments described in this section, DAGs were randomly generated according to the Erdös-Renyi model, in which edges are added independently with equal probability of inclusion. In each experiment, an array of values were chosen for each of the three main parameters: $p$, $s_0$, and $n$. For every possible combination of $(p, s_0, n)$, $N$ individual tests were then run with these parameters fixed. For each test, a DAG was randomly generated using the `pcalg` function `randomDAG` with $p$ nodes and $s_0$ expected edges, and then $n$ random samples were generated using the function `rmvDAG`, according to the structural model (1.3). For tests involving different choices of the sample size, the same DAG was used for each choice of $n$ to generate data sets of different sizes. Since the edges were selected at random, the simulated DAGs did not have *exactly* $s_0$ edges, but instead

$s_0$ edges on average. For each simulation, the nonzero coefficients $\beta_{ij}^0$ were chosen randomly and uniformly from the interval $[0.5, 2]$ and the error variances were fixed at $\omega_j^0 = 1$ for all $j$.

With the exception of HC and GES, each algorithm has a tuning parameter which strongly affects the accuracy of the final estimates. For CCDr, this is $\lambda$, which controls the amount of regularization, and for PC and MMHC it is $\alpha$, the significance level. In order to study the dependence of each algorithm on these parameters, we chose a sequence of parameters to use for each algorithm. For CCDr, we used a linear sequence of 20 values, starting from $\lambda_{\max} = n^{1/2}$. For both PC and MMHC, we used

$$\alpha \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}.$$

Our choices for $\alpha$ were motivated by the recommendations in Kalisch and Bühlmann (2007) and Tsamardinos et al. (2006), respectively, as well as by computational concerns: It was necessary to use a much smaller sequence for these algorithms since their running times are significantly longer than CCDr. Furthermore, we found that setting $\alpha < 0.0001$ results in estimates with too few edges, and setting $\alpha > 0.05$ can lead to runtimes well in excess of 24 hours.

When using the MCP, we must also select the concavity parameter $\gamma$ in addition to $\lambda$. In order to keep our experiments constrained to a reasonable size, we elected not to study the effect of this parameter in detail. Based on the extensive evaluations in Zhang (2010), we chose $\gamma = 2$, which was supported by internal tests to gauge the effect of this parameter. This value represents a fair balance between convexity ($\gamma \to \infty$) and complexity ($\gamma \to 0$). The CCDr algorithm also has three other user-specific parameters: $\varepsilon$, $M$, and $\alpha$. Based on our simulations, $\varepsilon$ and $M$ have a minimal impact on the accuracy of the estimates, and can simply be chosen to be small and large respectively. The default parameters we used in these simulations were: $\varepsilon = 10^{-4}$, $M = p^{1/2} \vee 10$, and $\alpha = 3$. Recall that in the

full algorithm (Algorithm 2), for each $\lambda_i$ there are at most $M^2 = p \vee 100$ sweeps. When $p$ is small a maximum of 100 iterations is more than enough.

*Remark* 3.1. Traditionally, the PC algorithm produces either a skeleton or a CPDAG, depending on how many phases of the algorithm are run (for the definition of a CPDAG and its relation to the PC algorithm, see Kalisch and Bühlmann, 2007). As discussed in Rütimann and Bühlmann (2009), however, it is possible to orient a DAG given its CPDAG using the function `pdag2dag` from the `pcalg` package. This works well in practice, although we found that in some cases the provided method was not able to orient the edges in the CPDAG successfully. In this case, we were able to compare skeletons but not DAGs for the PC algorithm. In the analysis, we treated this situation agnostically by ignoring such problematic estimates and entering them as missing values in the final analysis. This situation arose in less than 5% of cases, so it was not a significant issue.

### 3.2 Performance metrics

Our emphasis will be on the performance of each algorithm with respect to structure learning; that is, how well each algorithm reconstructs the DAG which is used to generate the data. Thus for every estimated structure, we compare both the final oriented DAG and its skeleton (i.e. the undirected graph that results by ignoring the directionality of the edges) to those of the true DAG. For a directed graph, we distinguish between *true edges* (or *true positives*)—edges which are estimated with the correct orientation—and *reversed edges*—edges which are in the skeleton but have the wrong direction. No such distinction can be made for the skeletons, of course. A *false positive* is any edge—regardless of directionality—which is not in the skeleton of the true graph.

We gauge the performance of the algorithms on the following metrics:

1. $P$ = number of estimated (predicted) edges,

2. $TP$ = number of true positives,

3. $R$ = number of reversed edges,

4. $FP$ = number of false positives,

5. SHD of the estimated DAG,

6. SHD of the estimated skeleton,

7. Test-data log-likelihood,

8. Test-data BIC,

9. Total and average running time in seconds.

SHD refers to the structural Hamming distance, which measures the number of edge reversals, additions, and/or removals necessary to convert an estimated graph into the true graph. This is a useful metric since it gives an absolute sense of "how far" away the estimates are from the true graph. For the precise definition of the structural Hamming distance, see Tsamardinos et al. (2006). Also, in order to compute the log-likelihood and BIC, it is necessary to estimate the parameters given the estimated structures, which we did by simple ordinary linear regression. As $p$ increases the time to compute these parameters becomes burdensome, and so comparisons of the log-likelihood and BIC were only performed for the low-dimensional experiments with $p \leq 200$. While our primary concern in these evaluations is accuracy in structure learning, these two metrics give us a sense of the implied parameter estimation consistency.

We will also sometimes refer to the following common normalizations of the above metrics:

1. False discovery rate (FDR) = $(R + FP)/P$,

2. True positive rate (TPR) = $TP/T$,

3. False positive rate (FPR) $= (R + FP)/F$,

Here, $T$ is number of edges in the true graph and $F = \frac{1}{2}p(p-1) - T$ is the number of edges absent from the true graph. In some literature, the complement of the false discovery rate (i.e. $1 - \text{FDR}$) is sometimes called *specificity*, while TPR is also variously called *recall* or *sensitivity*.

Finally, when comparing the timing data it is important to recall that each algorithm computes a different number of estimates: HC and GES only produce one, the implementations of PC and MMHC used here produce exactly six, and both CCDr approaches produce up to 20 estimates. Thus it is necessary to consider both the total running time for each algorithm as well as the average time per estimate, which gives a better sense of the computational complexity of each approach. In the sequel, the *total runtime* is defined as the real processor time required to run an algorithm over a full sequence of tuning parameters, and the *average runtime* is defined as the total runtime divided by the number of graphs estimated, i.e., the number of tuning parameters in the sequence.

### 3.3 Experiments on random graphs

In this section we provide detailed results comparing the performance of each algorithm on randomly generated DAGs, across a wide range of choices of $(p, s_0, n)$, using the metrics described in Section 3.2.

In order to properly compare the algorithms, a single model needed to be selected from each sequence of estimates generated by each algorithm. To keep things simple, and since we have not considered a theoretical analysis of consistent parameter selection, we simply chose the most accurate model produced by each algorithm by selecting the DAG estimate with the smallest SHD. While this may seem artificial, it provides a good assessment of the potential of each approach. This choice of parameter selection results in DAGs with somewhat low sensitivity,

but nonetheless it still provides a consistent method of comparing the performance of different algorithms. In Section 3.5 we will discuss some interesting issues related to parameter selection.

### 3.3.1 Low-dimensions

We first generated relatively small random graphs along with low-dimensional data sets according to the following settings:

- $p \in \{50, 100, 200\}$;
- $s_0/p \in \{0.2, 0.5, 1.0, 2.0\}$;
- $n/p \in \{1, 5\}$;
- Algorithms: CCDr-MCP, CCDr-$\ell_1$, GES, HC, MMHC, PC.

For all combinations of $(p, s_0, n)$, we ran $N = 50$ tests each. The result was 600 random DAGs, 1200 data sets, and 86,400 individual estimates across all six algorithms tested.

The results are shown in Tables 2.1, 2.2, and 2.3, and Figure 2.1. For each $p$, the results are averaged over all 50 tests and each value of $s_0$ and $n$. In the low-dimensional regime, it is expected that constraint-based algorithms will show good performance as the statistical tests on which they rely are more reliable and consistent when $n \geq p$. As expected, in our experiments, both PC and MMHC produced the most accurate results in this setting. This is further substantiated by the seemingly counterintuitive observation that the performance of both algorithms improves as $p$ increases; this is explained by recalling that $n$ also increases as $p$ increases, so for larger $p$ the statistical tests also have increased power.

The score-based algorithms GES and HC, on the other hand, easily perform the worst in terms of structure learning: these algorithms include far too many

| $p = 50$, $T = 46.48$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 26.50 | 22.98 | 109.83 | 113.78 | 26.46 | 26.39 |
| TP | 14.35 | 11.86 | **33.20** | 27.49 | 15.88 | 16.64 |
| R | 8.38 | **7.96** | 8.19 | 12.29 | 9.14 | 8.26 |
| FP | 3.78 | 3.15 | 68.44 | 74.00 | **1.44** | 1.48 |
| SHD (DAG) | 35.92 | 37.77 | 81.72 | 92.99 | 32.04 | **31.32** |
| SHD (skeleton) | 27.54 | 29.81 | 73.53 | 80.69 | **22.89** | 23.06 |
| TPR | 0.31 | 0.26 | **0.71** | 0.59 | 0.34 | 0.36 |
| FDR | 0.46 | 0.48 | 0.70 | 0.76 | 0.40 | **0.37** |

Table 2.1: Average estimation performance of algorithms in low-dimensions ($p = 50$).

edges and as a result obtain high sensitivity but also high false discovery rates. For example, when $p = 200$ and the simulated DAGs had 185 edges on average, both HC and GES estimate well over 500 edges, almost three times the true number, and exhibit false discovery rates greater than 70%. Notwithstanding, GES does noticeably outperform HC, which was anticipated.

Both CCDr methods fall in the middle, with CCDr-MCP outperforming CCDr-$\ell_1$ by a few edges in each case. Both methods estimate fewer edges than their score-based competitors—150 and 140 edges respectively when $p = 200$—but slightly more than the constraint-based methods, which estimate 135 edges (PC) and 129 edges (MMHC). This shows that CCDr represents a clear improvement over both GES and HC, and this is even without consideration of efficiency, which we will discuss shortly (Section 3.3.3).

The results for the test-data log-likelihood and the BIC score highlight several difficulties with existing methods which the proposed methods help to overcome. GES and HC both show higher log-likelihood than the others, and since the results

| $p = 100$, $T = 91.48$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 67.14 | 60.32 | 241.71 | 256.20 | 60.97 | 60.33 |
| TP | 36.40 | 30.85 | **74.30** | 60.24 | 39.03 | 39.85 |
| R | 18.95 | 19.87 | **12.90** | 23.16 | 18.71 | 17.33 |
| FP | 11.79 | 9.60 | 154.51 | 172.81 | 3.22 | **3.15** |
| SHD (DAG) | 66.86 | 70.23 | 171.69 | 204.05 | 55.67 | **54.78** |
| SHD (skeleton) | 47.91 | 50.36 | 158.79 | 180.88 | **36.95** | 37.45 |
| TPR | 0.40 | 0.34 | **0.81** | 0.66 | 0.43 | 0.44 |
| FDR | 0.46 | 0.49 | 0.69 | 0.76 | 0.36 | **0.34** |

Table 2.2: Average estimation performance of algorithms in low-dimensions ($p = 100$).

are computed based on *test data*, this cannot be attributed to overfitting. What's more, even though both methods produce far more edges than the others, they each only estimate roughly 3 edges per node, which is further evidence that these methods are not necessarily overfitting. Rather, going back to (1.14), we see that the log-likelihood is a function of $\Gamma$ alone, which means the test-data log-likelihood is not influenced by the accuracy of the graph structure estimated by an algorithm. This results in two distinct issues in evaluating algorithms on the basis of test-data log-likelihood:

- Even if $\|\widehat{\Gamma} - \Gamma\|_F$ is small, i.e. $\widehat{\Gamma}$ is a good estimate of the true parameter, the estimated equivalence class can still be different from the true equivalence class;
- Even if the equivalence class is correctly estimated, the chosen representation may not be the sparsest.

| $p = 200$, $T = 185.06$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 150.44 | 140.51 | 553.78 | 591.55 | 134.72 | 128.73 |
| TP | 83.60 | 73.28 | **158.38** | 127.69 | 90.74 | 89.23 |
| R | 39.05 | 42.58 | **22.35** | 45.65 | 37.59 | 34.28 |
| FP | 27.79 | 24.65 | 373.06 | 418.21 | 6.39 | **5.22** |
| SHD (DAG) | 129.24 | 136.43 | 399.74 | 475.58 | 100.70 | **96.69** |
| SHD (skeleton) | 90.19 | 93.86 | 377.39 | 429.93 | **63.12** | 65.25 |
| TPR | 0.45 | 0.40 | **0.86** | 0.69 | 0.49 | 0.48 |
| FDR | 0.44 | 0.48 | 0.71 | 0.78 | 0.33 | **0.31** |

Table 2.3: Average estimation performance of algorithms in low-dimensions ($p = 200$).

This explains why GES and HC perform the best on this metric: They do a good job of estimating $\Gamma$, as opposed to a sparse Bayesian network. By contrast, the constraint-based methods do not use the log-likelihood at all and thus exhibit the worst generalization in terms of log-likelihood. For methods which estimate approximately the same number of edges, CCDr-MCP is optimal, falling in between the score-based and constraint-based approaches (Figure 2.1). A similar discussion applies to the BIC scores, with the added complication of the BIC penalty. The fact that GES and HC still perform the best with respect to BIC—in spite of estimating far too many edges—underscores the fact that the BIC penalty is too lenient for estimating DAGs. This observation is further substantiated and discussed in more detail in Section 3.5.

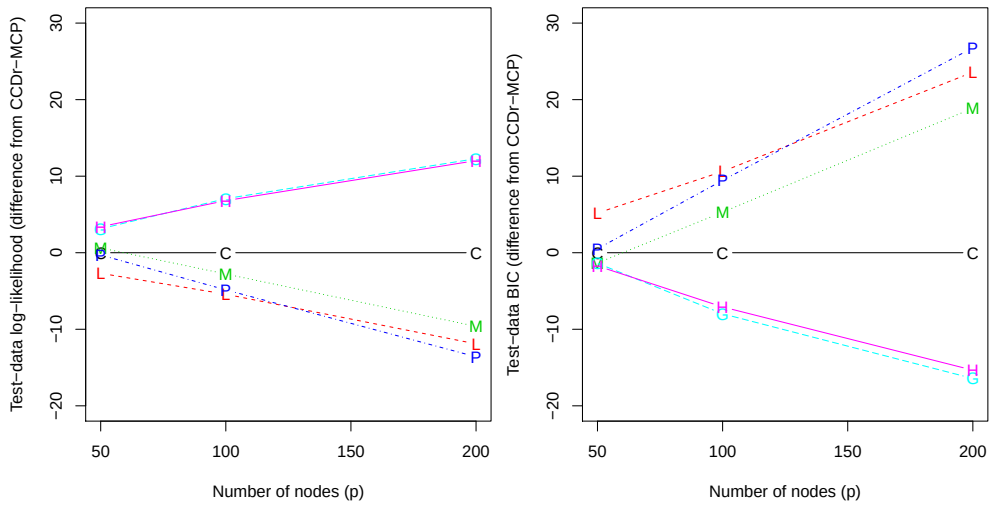



Figure 2.1: Comparison of test-data log-likelihood and BIC scores (low dimensions). The data are presented relative to the scores for CCDr-MCP. For log–likelihood, larger scores (positive values in the plot) are better; for BIC smaller scores (negative values) are better. (C = CCDr-MCP, L = CCDr-$\ell_1$, G = GES, H = HC, M = MMHC, P = PC)

### 3.3.2 High-dimensions

In this section we use the same random set-up as in the previous section, however, our focus is now on high-dimensional estimation. Both HC and GES were omitted in this experiment because of their poor performance—both in terms of accuracy and timing—in the low-dimensional setting. This allowed us to scale up the experiments to $p = 500$. In order to ensure a reasonable signal was detectable in each test, we fixed $n = 50$ for the tests. The following settings were used:

- $p \in \{100, 200, 500\}$;
- $s_0/p \in \{0.2, 0.5, 1.0, 2.0\}$;
- $n = 50$ fixed for all models;
- Algorithms: CCDr-MCP, CCDr-$\ell_1$, MMHC, PC.

For all combinations of $(p, s_0, n)$, we ran $N = 20$ tests each, resulting in 240 tests. These tests give us a better sense of the performance of the algorithms when the sample size is small relative to $p$.

The results are shown in Tables 2.4, 2.5, and 2.6. As before, the results are presented for each value of $p$, averaged over all tests and each value of $s_0$ (note that $n$ did not change in these tests). In contrast to the low-dimensional scenario in which the constraint-based methods outperform our method, in high-dimensions we begin to see the advantages of CCDr in structure learning. As $p$ increases and $n$ remains fixed, the gap between CCDr-MCP and both PC and MMHC increases. In particular, across each value of $p$, the false discovery rates for all the methods are comparable, however, the increased sensitivity (true positive rate) and lower SHD indicates that CCDr-MCP provides a higher quality reconstruction of the true network. The numbers are illuminating: when $p = 500$, for graphs which have 460 edges on average, CCDr-MCP estimates approximately 100 more edges while maintaining roughly the same false discovery rate and including 50-70 *more* true edges on average.

By comparison, CCDr-$\ell_1$ estimates fewer edges, obtaining lower sensitivity, and more closely mirrors the performance of PC and MMHC. This discrepancy in the performance of concave and $\ell_1$ regularization in high dimensions highlights the advantages of concave regularization and supports the conclusions in the literature on sparse regression. This is not altogether surprising since our framework is closely tied to the Gaussian linear model and regression analysis.

Comparing the low- and high-dimensional results when $p = 100, 200$, we also see that the CCDr methods are more robust to smaller sample sizes. When $p = 200$, for example, the net decrease in true positives between low- and high-dimensions is roughly 18 edges for CCDr-MCP, 26 edges for CCDr-$\ell_1$, 46 edges for MMHC, and 42 edges for PC. Similar patterns are observed for $p = 100$, and for other metrics as well. This confirms what we already know about constraint-

| $p = 100$, $T = 92.31$ | CCDr-MCP | CCDr-$\ell_1$ | MMHC | PC |
|---|---|---|---|---|
| P | 52.74 | 43.95 | 43.02 | 43.89 |
| TP | **27.59** | 21.48 | 23.82 | 24.12 |
| R | 16.95 | 16.29 | **16.07** | 16.19 |
| FP | 8.20 | 6.19 | **3.12** | 3.58 |
| SHD (DAG) | 72.92 | 77.03 | **71.61** | 71.76 |
| SHD (skeleton) | 55.98 | 60.74 | **55.54** | 55.58 |
| TPR | **0.30** | 0.23 | 0.26 | 0.26 |
| FDR | 0.48 | 0.51 | **0.45** | **0.45** |

Table 2.4: Average estimation performance of algorithms in high-dimensions ($p = 100$).

based methods: they are more reliable when sample sizes are large. Moreover, in spite of the fact that GES and HC were omitted from the high-dimensional experiments, we of course do not expect *improved* performance when $n$ decreases. These observations confirm our expectations that regularization can improve the performance of structure learning algorithms in high-dimensions, with concave regularization providing a noticeable improvement upon $\ell_1$ regularization.

### 3.3.3 Timing comparison

A comparison of the total and average runtimes for all the algorithms is provided by Figures 2.2 and 2.3 and Tables 3.3.3 and 3.3.3.

In low-dimensions, both GES and HC produce a single DAG estimate and take 15s and 25s, respectively, to estimate graphs with 200 nodes. This is compared with 3-5s for both CCDr-MCP and CCDr-$\ell_1$, in which time both methods compute approximately 20 estimates. Amongst all the compared methods, the

| $p = 200$, $T = 181.89$ | CCDr-MCP | CCDr-$\ell_1$ | MMHC | PC |
|---|---|---|---|---|
| P | 122.05 | 97.36 | 82.71 | 86.41 |
| TP | **65.14** | 47.40 | 44.71 | 46.70 |
| R | 35.75 | 34.89 | **31.40** | 33.17 |
| FP | 21.16 | 15.07 | 6.60 | **6.54** |
| SHD (DAG) | **137.91** | 149.56 | 143.78 | 141.72 |
| SHD (skeleton) | **102.16** | 114.67 | 112.38 | 108.55 |
| TPR | **0.36** | 0.26 | 0.25 | 0.26 |
| FDR | 0.47 | 0.51 | **0.46** | **0.46** |

Table 2.5: Average estimation performance of algorithms in high-dimensions ($p = 200$).

fastest alternative is the PC algorithm, however, the difference in timing is still roughly an order of magnitude: When $p = 200$, PC takes a little less than 4s on average for a single estimate, whereas CCDr takes approximately *one-fifth of a second* per estimate. This translates to a total runtime of less than 4s for 20 CCDr estimates—faster than the time to compute a single model, on average, for the PC algorithm. Furthermore, CCDr-MCP is slightly faster than CCDr-$\ell_1$, although the difference is small. Similar observations continue to hold in high-dimensions up to the tested limit of $p = 500$. Interestingly, both PC and MMHC are significantly faster in high-dimensions than in low-dimensions, which we suspect is due to how these algorithms scale with $n$: data sets with more samples require more time to process (see Section 3.6 for more details).

Combined with the improved performance in high-dimensions (Section 3.3.2), these results support our claim that CCDr is an improvement in both timing and accuracy over existing methods for high-dimensional data when $p \leq 500$. To see how CCDr performs when $p > 500$, we will show in the next subsection that the

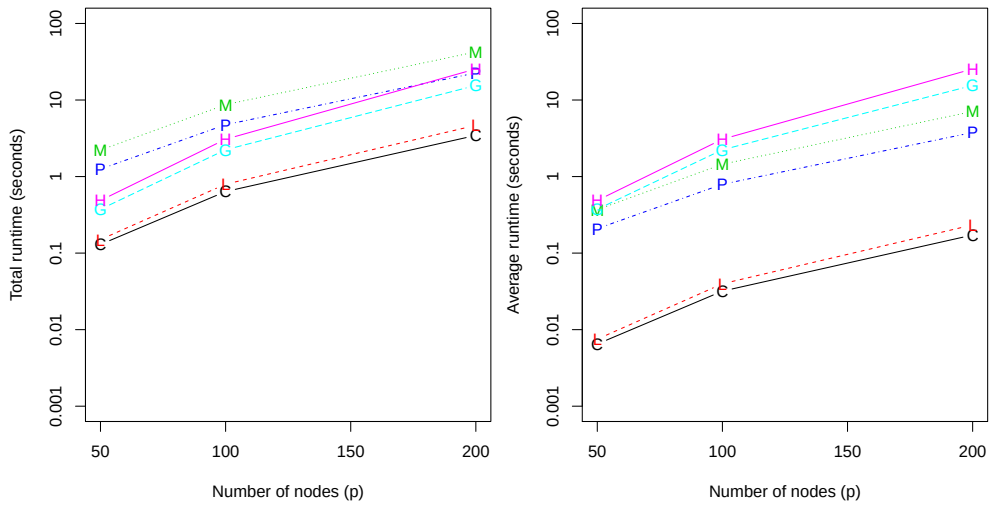



Figure 2.2: Timing comparison in low dimensions for all six algorithms (C = CCDr-MCP, L = CCDr-$\ell_1$, G = GES, H = HC, M = MMHC, P = PC).

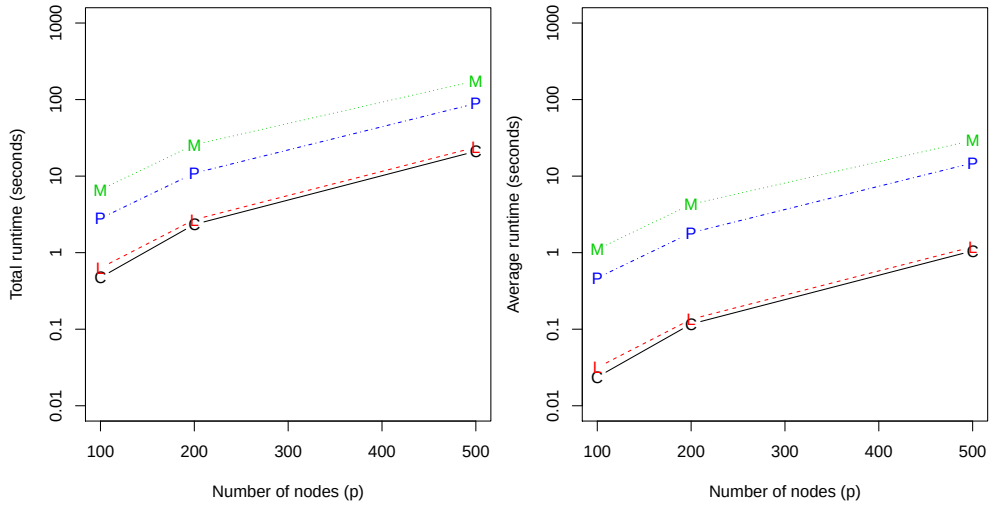



Figure 2.3: Timing comparison in high dimensions, excluding GES and HC (C = CCDr-MCP, L = CCDr-$\ell_1$, M = MMHC, P = PC).

| $p = 500$, $T = 460.21$ | CCDr-MCP | CCDr-$\ell_1$ | MMHC | PC |
|---|---|---|---|---|
| P | 319.94 | 252.56 | 195.07 | 202.64 |
| TP | **172.34** | 121.75 | 101.49 | 104.33 |
| R | 88.51 | 89.33 | **75.50** | 82.60 |
| FP | 59.09 | 41.49 | 18.09 | **15.71** |
| SHD (DAG) | **346.96** | 379.95 | 376.81 | 371.60 |
| SHD (skeleton) | **258.45** | 290.62 | 301.31 | 289.00 |
| TPR | **0.37** | 0.26 | 0.22 | 0.23 |
| FDR | **0.46** | 0.52 | 0.48 | 0.49 |

Table 2.6: Average estimation performance of algorithms in high-dimensions ($p = 500$).

CCDr algorithm scales efficiently to high-dimensional problems with thousands of variables with almost no loss in reconstruction accuracy.

### 3.4 Large graphs

The previous section offered a detailed assessment of the performance of the CCDr algorithm when $p \leq 500$. In order to test how our algorithm scales as the number of nodes increases, we ran further tests up to $p = 2000$ using CCDr-MCP. The purpose of these tests is to show how the proposed method scales as $p$ increases in terms of timing and accuracy. Since the timing is acutely dependent on the relationship between the dimension, the sparsity of the true graph, and the number of samples, we opted to compare the timing over random choices of the latter two parameters. This also gives us a sense of how the algorithm performs when faced with a more realistic scenario in which the relationship between $p$, $s_0$, and $n$ can be unpredictable. Specifically, we ran $N = 20$ tests with the following parameters:

Table 2.7: Total runtime (top) and average runtime (bottom) in seconds for all six algorithms from Section 6.3.1.

| $p$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| 50 | 0.13 | 0.15 | 0.37 | 0.49 | 2.21 | 1.25 |
| 100 | 0.64 | 0.79 | 2.24 | 3.08 | 8.70 | 4.76 |
| 200 | 3.45 | 4.67 | 15.50 | 25.47 | 42.55 | 22.59 |
| $p$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
| 50 | 0.01 | 0.01 | 0.37 | 0.49 | 0.37 | 0.21 |
| 100 | 0.03 | 0.04 | 2.24 | 3.08 | 1.45 | 0.79 |
| 200 | 0.17 | 0.23 | 15.50 | 25.47 | 7.09 | 3.77 |

- $p \in \{100, 200, 500, 1000, 1500, 2000\}$;
- $s_0/p \in \{0.2, 0.3, 0.4, \ldots, 2\}$;
- $n/p \in \{0.1, 0.2, 0.3, \ldots, 5\}$.

The parameters $s_0$ and $n$ were chosen randomly from the above sets in each test, which resulted in an average sparsity level of $s_0/p = 1.06$. The results are displayed in Table 2.9 and Figure 2.4. Since the timing of the algorithm depends crucially on the total number of models estimated, and also on the threshold parameter $\alpha$, we have plotted both the total and average runtimes for two scenarios: The time it took to estimate DAGs with up to $p$ edges, and then the full running time with the edge threshold set at $\alpha = 3$. When $p = 1000$, the total running time is just under six minutes, with an average time per model of about 20 seconds. When $p = 2000$, the total running time is just under thirty minutes, with an average time per model of about 85 seconds.

In terms of accuracy, Table 2.9 shows that the results are comparable to those in Section 3.3. Furthermore, as $p$ increases we notice that TPR increases while

Table 2.8: Total runtime (top) and average runtime (bottom) in seconds for the four algorithms from Section 6.3.2.

| $p$ | CCDr-MCP | CCDr-$\ell_1$ | MMHC | PC |
|---|---|---|---|---|
| 100 | 0.47 | 0.63 | 6.64 | 2.80 |
| 200 | 2.34 | 2.70 | 25.71 | 10.93 |
| 500 | 21.09 | 23.84 | 175.54 | 88.85 |
| $p$ | CCDr-MCP | CCDr-$\ell_1$ | MMHC | PC |
| 100 | 0.02 | 0.03 | 1.11 | 0.47 |
| 200 | 0.12 | 0.13 | 4.29 | 1.82 |
| 500 | 1.05 | 1.19 | 29.26 | 14.81 |

FDR decreases, which is likely due to the increased number of samples (on average) as $p$ increases; when $p = 100$, there were $n = 114$ samples on average vs. $n = 2260$ when $p = 2000$. Combined with the timing data in Figure 2.4, this confirms that CCDr scales efficiently in terms of both $n$ and $p$ when the underlying graph is sparse.

After these experiments in this work were completed, the performance of our method was further improved, so that the total runtime for $p = 2000$ is now less than five minutes. A comprehensive comparison of the updated implementation vs. the numbers reported here can be found in Figure 3.4. These changes were made to the underlying codebase, and *not* to the algorithm, thus the improvements were purely in terms of code efficiency. Using this updated implementation, we can report that our method has been successfully tested on graphs with up to 8000 nodes, with comparable accuracy to the results exhibited in Table 2.9. The total runtime for 20 estimates was 75 minutes, which may be compared with the 13 days reported for MMHC on a graph with $p = 5000$ in Tsamardinos et al. (2006). Regarding the internal implementation of our method, we did not make

| Number of nodes ($p$) | 100 | 200 | 500 | 1000 | 1500 | 2000 |
|---|---|---|---|---|---|---|
| Number of samples ($n$) | 114 | 190 | 520 | 1280 | 1470 | 2260 |
| T | 83.15 | 237.15 | 538.15 | 1186.35 | 1550.15 | 2057.95 |
| P | 66.15 | 191.90 | 488.30 | 1082.20 | 1434.20 | 1926.90 |
| TP | 36.15 | 111.50 | 279.80 | 636.70 | 854.25 | 1156.10 |
| R | 20.75 | 46.45 | 115.80 | 226.45 | 323.75 | 447.90 |
| FP | 9.25 | 33.95 | 92.70 | 219.05 | 256.20 | 322.90 |
| SHD (DAG) | 56.25 | 159.60 | 351.05 | 768.70 | 952.10 | 1224.75 |
| SHD (skeleton) | 35.50 | 113.15 | 235.25 | 542.25 | 628.35 | 776.85 |
| TPR | 0.43 | 0.47 | 0.52 | 0.54 | 0.55 | 0.56 |
| FDR | 0.45 | 0.42 | 0.43 | 0.41 | 0.40 | 0.40 |

Table 2.9: Average estimation performance of CCDr-MCP from Section 3.4, averaged over $N = 20$ random choices of $s_0$ and $n$ for each $p$.

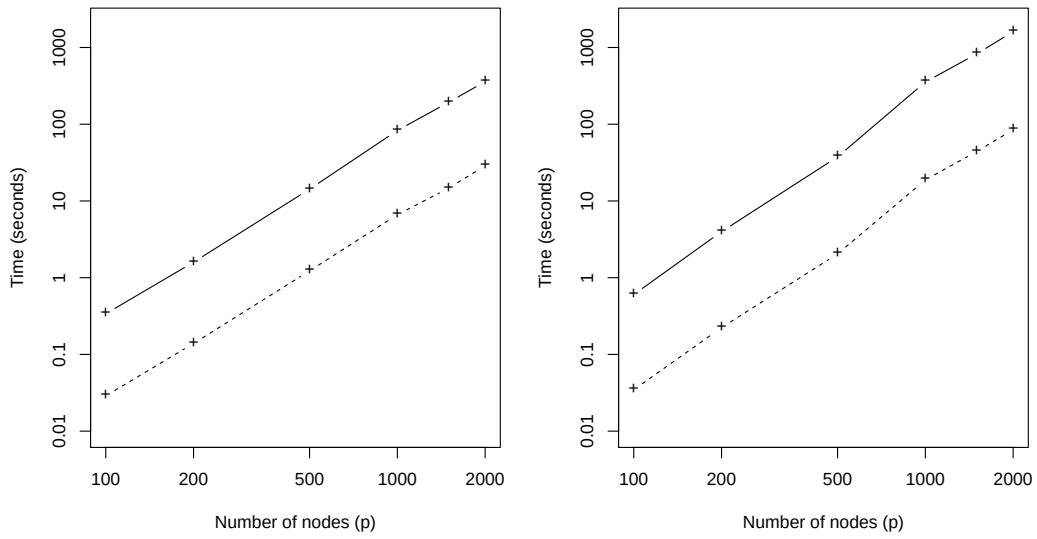



Figure 2.4: Timing data for CCDr-MCP up to $p = 2000$. The solid line is the total runtime and the dashed line is the average runtime. (left) Time to estimate graphs with at most $p$ edges; (right) Full runtime with edge threshold $\alpha = 3$.

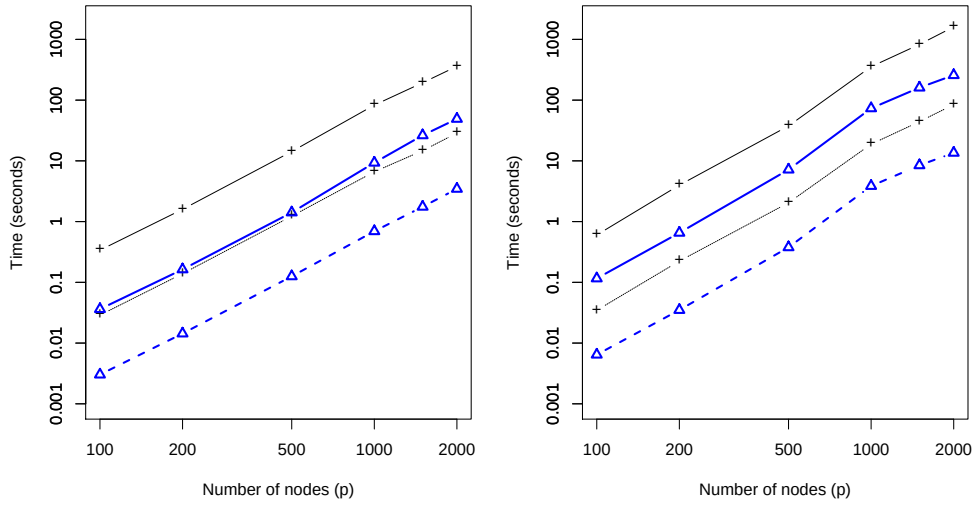



Figure 2.5: Comparison of the updated implementation of CCDr against the implementation presented in the main text. The triangles representing the new implementation are overlaid on top of Figure 5 from the main text, based on duplicating the test in Section 6.4 using the same graphs. The solid line is the total runtime and the dashed line is the average runtime. (left) Time to estimate graphs with at most $p$ edges, (right) Full runtime with edge threshold $\alpha = 3$.

use of an internal cache, memoization, or efficient data structures (i.e. besides standard vectors), all of which are common strategies used in existing methods. It stands to reason that an optimized implementation would yield even faster results. For instance, we perform the acyclicity check statically with each edge addition; one could imagine a more sophisticated strategy such as incremental topological sorting would lead to significant performance enhancements.

### 3.5 Parameter selection

Thus far, we have used the "best estimate" according to distance from the true graph, measured by SHD, in order to select models from the estimated solution paths for CCDr, MMHC, and PC. This choice provides a consistent comparison, but results in relatively sparse estimates since missing edges are penalized equally

against false positives. One of the advantages of CCDr is that it is able to estimate models with higher sensitivity much more efficiently than PC or MMHC. Alternatively, one could use empirical parameter selection techniques such as BIC or cross-validation. It has already been noted that these empirical parameter selection techniques are suboptimal in high-dimensions, particularly for graphical models. This has been previously reported in the literature, see for instance Fu and Zhou (2013). Here we briefly discuss the results of some tests to confirm this behaviour for our method.

Using both conventional BIC and the extended BIC for high-dimensional problems developed in Foygel and Drton (2010), we selected the tuning parameters for CCDr-MCP, CCDr-$\ell_1$, PC, and MMHC. The results confirm that BIC tends to select models with too many edges by insufficiently penalizing the model complexity, consistent with Figure 2.1. One may ask if all the algorithms suffer equally, and the answer is no. For the reasons already discussed, we were not able to test the performance of either PC or MMHC for $\alpha > 0.05$, which is the regime in which more edges tend to be selected. Thus, in using BIC to select the significance level, the maximum value of $\alpha = 0.05$ was over-represented. We suspect that if we had run PC and MMHC with $\alpha > 0.05$ in order to produce estimates with extraneous edges, BIC would also select these models. As a result of these limitations, in selecting models based on BIC, CCDr appeared to perform worse relative to either PC or MMHC than reported in previous sections.

To correct for this, we ran the same parameter selection test using BIC as the selection criterion, but this time restricting the set of CCDr candidates to those with at most as many edges as the most produced by either the PC algorithm or the MMHC algorithm. Using the same data as in Section 3.3.1, the results resemble those previously reported (see Table 2.10 for a sample run with $p = 200$). Across the board, graphs with more edges were selected, but the qualitative observations between CCDr and PC / MMHC remain the same.

Table 2.10: Average estimation performance of algorithms in low-dimensions using BIC as parameter selection criteria.

| $p = 200$, $T = 185.06$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 156.38 | 160.66 | 553.78 | 591.55 | 153.24 | 138.90 |
| TP | 82.45 | 76.45 | 158.38 | 127.69 | 94.96 | 90.95 |
| R | 40.74 | 45.44 | 22.35 | 45.65 | 38.00 | 35.51 |
| FP | 33.18 | 38.77 | 373.06 | 418.21 | 20.28 | 12.45 |
| SHD (DAG) | 135.79 | 147.38 | 399.74 | 475.58 | 110.38 | 102.20 |
| SHD (skeleton) | 95.05 | 101.93 | 377.39 | 429.93 | 72.38 | 69.44 |
| TPR | 0.45 | 0.41 | 0.86 | 0.69 | 0.51 | 0.49 |
| FDR | 0.47 | 0.52 | 0.71 | 0.78 | 0.38 | 0.35 |

### 3.6 Further discussion

The experiments and results described already, while providing a general overview of the performance of the algorithms tested, also raise several questions which we address briefly in this section.

While we tested a variety of sparsity levels in Section 3.3, we have not provided a detailed assessment of how the performance of the algorithms varies as the sparsity increases or decreases. An analysis of the effect of sparsity shows that the same qualitative behaviour observed in Sections 3.3.1 and 3.3.2 persists (see Figures 2.6 and 2.7). We do observe a small decrease in reconstruction accuracy for the CCDr methods when the graph is more dense ($s_0/p = 2$); improving our method when the true graph is more dense remains for future work.

For the CCDr algorithm, in order to provide a reasonable balance of complexity and efficiency in the resulting estimation problem, we fixed $\gamma = 2$. Nonetheless,

this parameter was observed to have a non-negligible effect on the results and a more in-depth study in the future would account for the effect of this parameter. Another parameter which we have not discussed is the maximum neighbourhood size in the true graph, which we controlled in our simulations by controlling the expected neighbourhood size. Keeping the neighbourhoods small is critical for keeping the running time of the PC algorithm reasonable. Further simulations in which we allowed each node to have arbitrarily many parents showed that the running time of the CCDr algorithm does not depend on this parameter. Finally, both PC and MMHC show relatively poor computational complexity with respect to the sample size $n$, with more instances requiring more time to process. Our tests indicate that the complexity of CCDr is essentially independent of $n$—the only dependence on sample size enters through the computation of the correlation matrix in the first step.

## 4 Real networks

While the random set-up in the previous section provided a convenient setting to test many random structures quickly and efficiently, random graphs may not be good representatives of realistic network structures. For this reason, we augmented these experiments with tests on real network structures, using both simulated and scientific (unsimulated) data. Our first experiment uses network structures from the Bayesian Network Repository,[1] a standardized collection of networks which is commonly used as a benchmark for structure learning methods, as well as a simulated scale-free network. In order to assess the impact of these methods on actual scientific data, we also compare the performance of the algorithms on the well-known flow cytometry data set (Sachs et al., 2005).

[1] The original repository can be found at: `http://www.cs.huji.ac.il/site/labs/compbio/Repository/`.

### 4.1 Bayesian network repository

All of the networks examined in this experiment were loaded using the `bnlearn` package.[2] We then used the graph structures to generate data according to a structural equation model, as in the previous section. Furthermore, in order to keep the focus on high-dimensional estimation, we fixed the number of samples at $n = 50$, which narrowed the choice of networks to those that satisfy $p > 50$. Seven such network structures were tested, to which we added one randomly generated scale-free structure with 200 nodes. The scale-free network was created using the `igraph` package. For each network, we generated random coefficients in the interval $[0.5, 1]$ for each edge as before and generated a single random data set with unit variances for testing. This procedure was replicated $N = 50$ times, and the number of true positives and false positives were tracked for each algorithm. We also increased the length of the regularization path used for the CCDr methods to 50 estimates while keeping both PC and MMHC fixed at six estimates for each graph. Based on the results in the previous section—particularly with respect to timing—both HC and GES were excluded from these tests.

We have already observed in Section 3.5 how traditional parameter selection techniques such as BIC and cross-validation perform poorly. For this reason, we chose to present the results graphically by their ROC curves in order to compare the true positive rate against the false positive rate as a function of the tuning parameters. The resulting ROC curves are displayed in Figure 2.8.

In terms of reconstruction accuracy, with only one exception, we see that the CCDr methods perform as well or better than the other methods in these experiments. Consistent with the previously reported experiments on random graphs, the CCDr methods tend to show higher sensitivity with comparable false positive rates in high dimensions. In some cases the improvements are dramatic—for in-

[2] A mirror of the repository used by the `bnlearn` package can be found at: `http://www.bnlearn.com/bnrepository/`.

stance, `pathfinder`, `scalefree`, and `pigs`. The one exception is the `win95pts` network, in which the PC algorithm attains slightly higher sensitivity and lower FDR compared with the CCDr methods as well as MMHC. These results further highlight the tradeoffs in learning between each approach and confirm the patterns observed previously in the literature: constraint-based methods tend to miss edges in the true skeleton, resulting in lower false discovery rates and lower sensitivity, whereas regularization tends to increase overall sensitivity with the risk of higher false positive rates if the amount of regularization is not calibrated properly.

More interesting is the comparison between CCDr-MCP and CCDr-$\ell_1$. Compared with the simulation results in Section 3, there is a more pronounced difference between the performance of concave vs $\ell_1$ regularization, with the former outperforming the latter. This is most visible in the `hailfinder` and `pigs` networks, where both methods show comparable sensitivity but CCDr-MCP exhibits lower false positive rates. The only network in which $\ell_1$ regularization is preferable is `pathfinder`, where CCDr-$\ell_1$ obtains higher sensitivity later in the solution path.

Consistent with the previous experiments, however, the main advantages of CCDr come in the form of efficiency: Figure 4.1 contains a comparison of runtime for each network and method. Unlike in the previous experiments, for these experiments the estimated solution path for the CCDr methods was 2.5 times longer, with up to 50 estimates per solution path. Notwithstanding, the CCDr methods were consistently the fastest. For example, using PC and MMHC, the `pathfinder` network with $p = 135$ nodes took 110x and 150x longer on average per estimate to compute, respectively. At the other end of the spectrum, the hardest graph to reconstruct was the `pigs` network, which took 39s for CCDr-MCP, 29s for CCDr-$\ell_1$, 71s for PC, and 147s for MMHC. In both cases CCDr-MCP easily did the best job reconstructing the true networks.

## 4.2 Application to real data

We analyzed the well-known flow cytometry data set, generated by Sachs et al. (2005), which has been previously analyzed by Fu and Zhou (2013); Shojaie and Michailidis (2010); Friedman et al. (2008) among others. The data set contains $n = 7466$ measurements of $p = 11$ continuous variables corresponding to proteins and phospholipids in human immune system cells. The underlying network, constructed through a careful series of biological experiments, has $s_0 = 20$ edges, and represents a gold-standard for comparison currently accepted by the biology community. Hereafter, we regard this consensus network as the true network in order to assess the algorithms. While this data set is hardly high-dimensional, it represents one of the few continuous data sets for which we have oracular knowledge of the true underlying DAG *as well as* real data from which to infer the true structure.

The original data set contains a mixture of both observational and experimental data. Since the methods presented here assume the data are normally distributed, we first tested the original continuous variables for normality, and much as expected the data were highly non-normal. To correct for this, we applied a logarithm transform, which produced variables that were much closer to Gaussian. This data set was used for our tests on continuous data.

We also analyzed a discretized version of the data set containing $n = 5400$ measurements, created by transforming the continuous data into three nonnegative levels which correspond to *high*, *medium*, and *low*, so that magnitudes were partially preserved (Sachs et al., 2005). This data set is especially interesting for a number of reasons. First, it represents a test of model misspecification: Our method was developed for continuous data, but nothing prevents us from naively feeding this data set into the algorithm. By treating the three levels as numeric values (*high* $= 2$, *medium* $= 1$, *low* $= 0$), we can compute the correlation matrix

and proceed with the second and third steps in Algorithm 2. Since the data are clearly not Gaussian, the results of this test give us a sense of how well our method performs on discrete, non-Gaussian data. Second, as a result of postprocessing to clean up the data as well as the discretization itself, it is much less noisy than the original data set, which provides an interesting side-by-side comparison.

A few changes were made to the set-up used in previous experiments. First, since the number of variables was small, it was feasible to run the constraint-based methods on a longer sequence of significance levels. Thus, we used a sequence of 10 levels:

$$\alpha \in \{10^{-6}, 5 \times 10^{-6}, 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 0.01, 0.05\}.$$

Furthermore, in a majority of the tests we ran, the PC algorithm was unable to orient all the edges in the final step, leading to a partially directed graph (formally a CPDAG, see Remark 3.1). As a result, we had to modify our metrics to allow for undirected edges. We did this favourably for the PC algorithm by counting an undirected edge as a true edge as long as the same edge exists in the skeleton of the true graph. Any edge that was successfully oriented by the PC algorithm was treated as a directed edge. Finally, we split each data set in half in order to obtain a testing data set on which to compute the log-likelihood of the estimated models. Since the PC algorithm was not able to estimate DAGs, log-likelihood scores could not be computed for the continuous data set.

Tables 2.11 and 2.12 summarize the results for a sample run, which are indicative of the general behaviour when different random splits are tested. Instead of selecting the best estimates as in Section 3.3, we chose estimates with comparable numbers of edges, selected to match the true graph as closely as possible with $s_0 = 20$. The results for CCDr-MCP are visualized in Figure 2.10. Both GES and HC consistently estimated too many edges, which matches the behaviour observed in Section 3.3.1. For the continuous data set, CCDr-MCP and MMHC perform the

best with almost identical metrics, while for the discrete data set CCDr-MCP is clearly optimal with fewer false positives and smaller SHD across the board. This indicates that even though this method was developed with continuous Gaussian data in mind, it can still be applied to discrete data with reasonable results.

Due to the small size of the graph with only $p = 11$ nodes, the differences in timing are largely negligible, taking fractions of a second to complete. Because of this, the processor time is subject to fluctuations in low-level bottlenecks most likely unrelated to the core algorithms themselves, and so we do not report exact times here. At a high level we did observe that HC and GES show much improved performance relative to PC and MMHC, however, the CCDr methods are still consistently the fastest.

# 5 Summary

We have introduced a general penalized likelihood framework for estimating sparse Bayesian networks, along with a fast algorithm that is easily implemented on a personal computer and have shown that our approach accurately estimates networks with 2000 nodes while scaling efficiently to handle networks with up to 8000 nodes. The proposed method is compatible with high-dimensional data where $p \gg n$, and outperforms many existing methods in both speed and accuracy in this regime. Tests on real networks have validated the performance and applicability of this method in a variety of domains.

Our focus has been on structure recovery, which is closely related to statistical inference and should not be confused with the complementary problem of *prediction*. For this reason, the metrics we employed require knowledge of the true underlying graph. Alternatively, one could inquire into the predictive performance and generalizability of learning methods, in which case metrics such as the prediction loss and test-data likelihood can be assessed without prior knowl-

edge of the true graph. Indeed, our simulations indicate that existing score-based methods such as GES may perform better with respect to such predictive metrics. We have already discussed in Section 3.3.1 why this may be, and it remains for future work to study this phenomenon in more depth.

The central theme of exploiting convexity to solve nonconvex problems is an intriguing prospect for the development of new algorithms in statistics and machine learning. Indeed, the main difficulties with nonconvex regularization are computational in nature. Although recent progress has broken this barrier in the case of least squares regression, to our knowledge the algorithm presented here is one of the first to approximate this type of nonconvex optimization problem when $p$ is in the thousands. Moreover, since our method revolves around a continuous optimization problem, we avoid approaches that rely on individual edge additions and removals, which are intrinsically discrete. As a result, future advances in nonconvex optimization will directly affect how we solve the maximum likelihood problem presented here.

# 6 `R` package

The `R` code associated with the CCDr algorithm has been assembled into an open-source package, which can be downloaded online. It has been integrated with popular packages for performing estimation and inference in Bayesian networks, including `bnlearn` and `pcalg`, as well as with common graph facilities such as `graph` and `igraph`. The package has been tested on graphs with up to 8000 nodes, although, we hope to improve this further.

The code can be downloaded at the following link:

`https://www.github.com/itsrainingdata/ccdr`

The following simple example exhibits how the package could be used. We first

generate an i.i.d. (canonical) Gaussian matrix, and then run the main method `ccdr.run` from the package:

```
### Specify dimensions
nn <- 20
pp <- 100

### Generate a random Gaussian matrix
dat <- matrix(rnorm(nn * pp), nrow = nn)

### Run the ccdr algorithm
ccdr.path <- ccdr.run(data = dat)

### Display the results
print(ccdr.path)
```

More complex examples, including generating data according to a specified linear structural equation model, are included in the online help files at the URL above.

**Algorithm 1** CCDr Algorithm

*Input:* Initial estimates $(\Phi^0, R^0)$; penalty parameters $(\lambda, \gamma)$; error tolerance $\varepsilon > 0$; maximum number of iterations $M$.

1. Cycle through $\rho_j$ for $j = 1, \ldots, p$, minimizing $Q_2$ with respect to $\rho_j$ at each step.

2. Cycle through the $p(p-1)/2$ blocks $\{\phi_{kj}, \phi_{jk}\}$ for $j, k = 1, \ldots, p$, $j \neq k$, minimizing with respect to each block:

   (a) If $\phi_{kj} \Leftarrow 0$, then minimize $Q_1$ with respect to $\phi_{jk}$ and set $(\phi_{kj}, \phi_{jk}) = (0, \phi^*_{jk})$, where $\phi^*_{jk} = \arg\min Q_1(\phi_{jk})$;

   (b) If $\phi_{jk} \Leftarrow 0$, then minimize $Q_1$ with respect to $\phi_{kj}$ and set $(\phi_{kj}, \phi_{jk}) = (\phi^*_{kj}, 0)$, where $\phi^*_{kj} = \arg\min Q_1(\phi_{kj})$;

   (c) If neither 2(a) nor 2(b) applies, then choose the update which leads to a smaller value of $Q$.

3. Repeat steps 1 and 2 $l$ times, until either $\max_{j,k} |\phi^{(l-1)}_{kj} - \phi^{(l)}_{kj}| < \varepsilon$ or $l > M$.

4. Transform the final estimates $(\widehat{\Phi}, \widehat{R})$ back to the original parameter space $(\widehat{B}, \widehat{\Omega})$ (see equation (2.4)) and output these values.

**Algorithm 2** Full CCDr Algorithm

*Input:* Initial estimates $(\Phi_0^0, R_0^0)$; sequence of regularization parameters $\lambda_0 > \lambda_1 > \cdots > \lambda_L$; concavity parameter $\gamma > 1$; error tolerance $\varepsilon > 0$.

1. Normalize the data so that $\|\mathbf{x}_j\|^2 = 1$ and compute the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ for all $i, j = 1, \ldots, p$.
2. For each $\lambda_i$:
    1. If $i > 0$, set $(\Phi_i^0, R_i^0) = (\widehat{\Phi}(\lambda_{i-1}), \widehat{R}(\lambda_{i-1}))$.
    2. Perform a full sweep of all parameters using $(\Phi_i^0, R_i^0)$ as initial values, and identify the active set.
    3. Sweep over the active set $l$ times, until either $\max_{j,k} |\phi_{kj}^{(l-1)} - \phi_{kj}^{(l)}| < \varepsilon$ or $l > M$.
    4. Repeat (2-3) $m$ times (using the current estimates as initial values) until the active set does not change, or $m > M$.
    5. If $\hat{s}_i > \alpha p$, then halt the algorithm. If not, continue by computing $(\widehat{\Phi}(\lambda_{i+1}), \widehat{R}(\lambda_{i+1}))$.
3. Transform the final estimates $(\widehat{\Phi}(\lambda_i), \widehat{R}(\lambda_i))$ back to the original parameter space $(\widehat{B}(\lambda_i), \widehat{\Omega}(\lambda_i))$ (see equation (2.4)) and output these values.

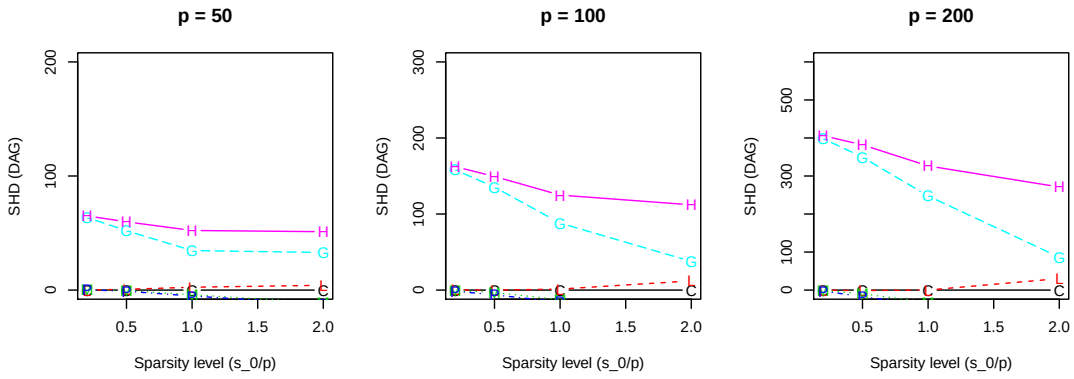



Figure 2.6: Effect of sparsity on SHD in low dimensions for all six algorithms (C = CCDr-MCP, L = CCDr-$\ell_1$, G = GES, H = HC, M = MMHC, P = PC).

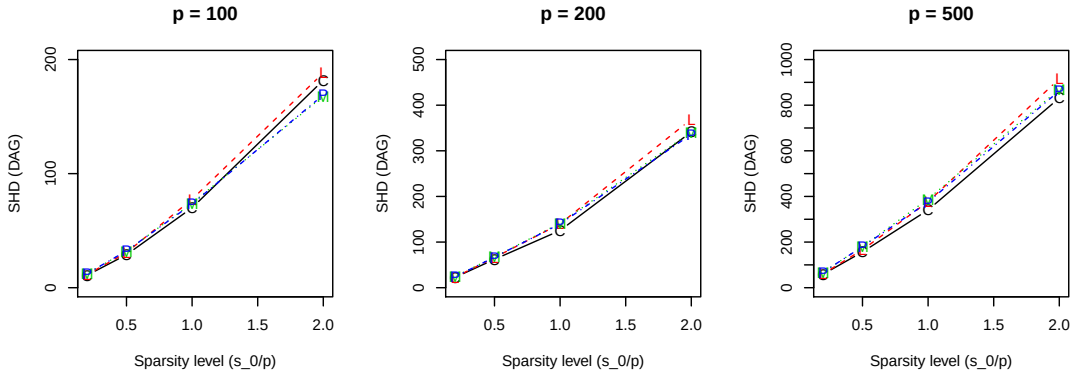



Figure 2.7: Effect of sparsity on SHD in high dimensions, excluding GES and HC (C = CCDr-MCP, L = CCDr-$\ell_1$, M = MMHC, P = PC).

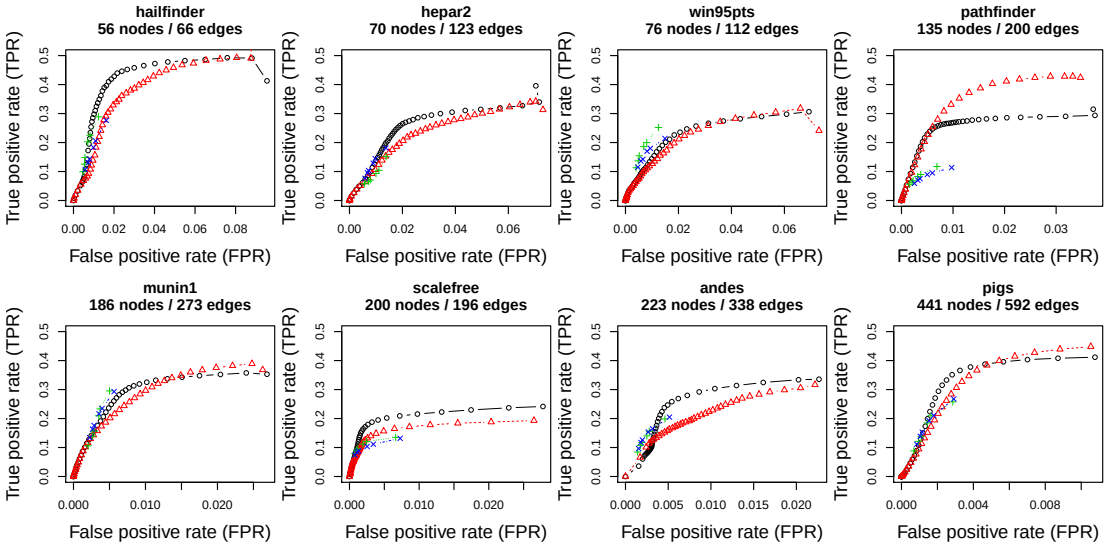



Figure 2.8: ROC curves for real networks (black ◯ = CCDr-MCP, red △ = CCDr-$\ell_1$, blue × = MMHC, green + = PC).

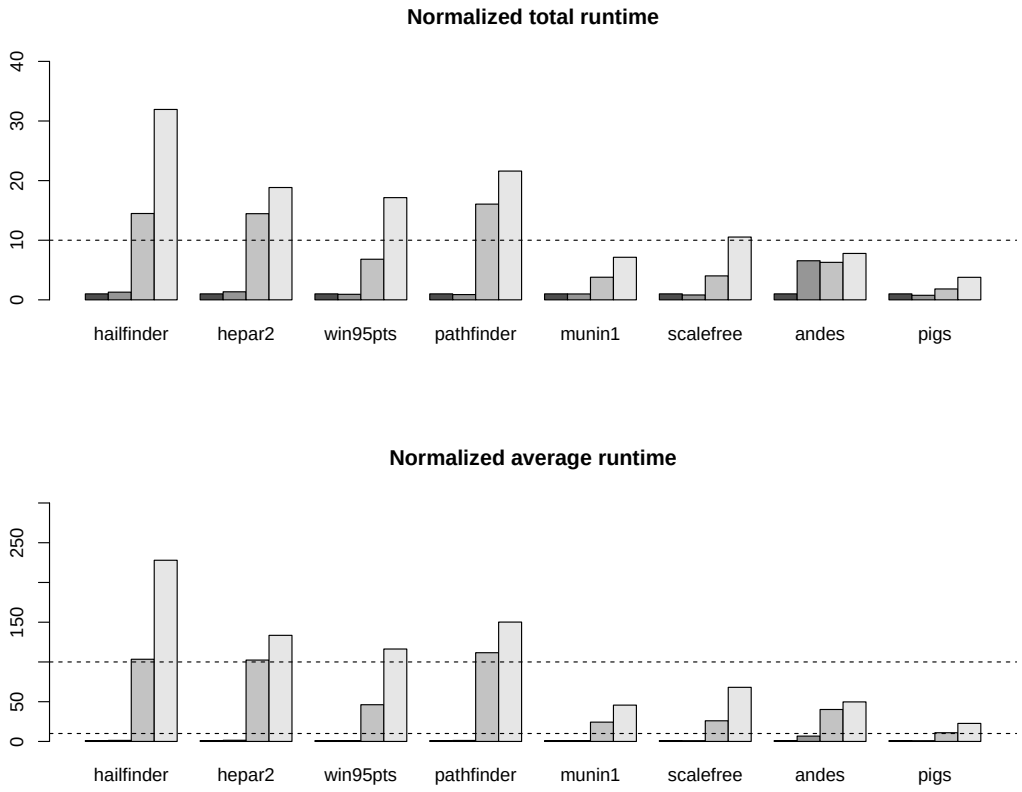



Figure 2.9: Comparison of the total and average runtime for the four algorithms tested in Section 7.1, normalized by the respective runtimes for CCDr-MCP. From left to right (also, darkest to lightest), the bars represent CCDr-MCP, CCDr-$\ell_1$, PC, MMHC.

| $p = 11$, $T = 20$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 20 | 20 | 41 | 38 | 20 | 20 |
| TP | 7 | 7 | 9 | **10** | 7 | 7 |
| R | 2 | **1** | 7 | 6 | 2 | 2 |
| FP | **11** | 12 | 25 | 22 | **11** | **11** |
| SHD (DAG) | **24** | 25 | 36 | 32 | **24** | 25 |
| SHD (skeleton) | **22** | 24 | 29 | 26 | **22** | **22** |
| Test Log-likelihood | -2.05 | -2.19 | **-0.34** | -1.09 | -2.03 | — |

Table 2.11: Structure estimation performance for all algorithms using the log–transformed continuous cytometry data.

| $p = 11$, $T = 20$ | CCDr-MCP | CCDr-$\ell_1$ | GES | HC | MMHC | PC |
|---|---|---|---|---|---|---|
| P | 20 | 20 | 43 | 35 | 20 | 20 |
| TP | 6 | 3 | **13** | 7 | 3 | 6 |
| R | 5 | 6 | 4 | 7 | 5 | **2** |
| FP | **9** | 11 | 26 | 21 | 12 | 12 |
| SHD (DAG) | **23** | 28 | 33 | 34 | 29 | 26 |
| SHD (skeleton) | **18** | 22 | 29 | 27 | 24 | 24 |
| Test Log-likelihood | -0.68 | -1.86 | **-0.10** | 0.18 | -2.32 | -2.01 |

Table 2.12: Structure estimation performance for all algorithms using discretized cytometry data.

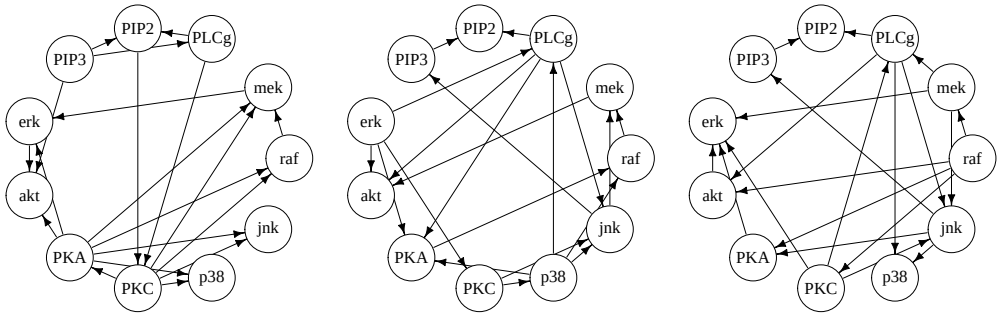



Figure 2.10: Comparison of the consensus network (left) against the DAGs estimated by the CCDr-MCP algorithm for both data sets: (middle) Log-transformed continuous data set; (right) Discretized data set.

# CHAPTER 3

# Theory

We now turn to assessing the statistical properties of score-based estimators. This chapter consists of two substantive components: (i) An finite-dimensional analysis of local minimizers of the program considered in Chapter 2, and (ii) A high-dimensional analysis of global minimizers under a least squares loss function.

The analysis of local maximizers is motivated by the seminal work of Fan and Li (2001), which establishes the existence of consistent local maximizers for a general class of penalized maximum likelihood estimators under nonconvex regularization. Our analysis extends these results in several ways for the model (1.2):

1. We must modify the analysis in order to accommodate the nonidentifiability of the parameters $(\widetilde{B}, \widetilde{\Omega})$,

2. We show that amongst the entire equivalence class $\mathfrak{D}(\Sigma)$, it is possible to calibrate the regularizer $\rho_\lambda$ so that the sparsest equivalent DAG is selected by our estimator.

This local analysis is covered in Section 1.

By contrast, the analysis of global minimizers is completely different and relies on interpreting the problem in terms of what we term an *abstract neighbourhood regression problem*. This allows us to draw on existing results from high-dimensional statistics. In the end, we will show that under a least squares loss and in a high-dimensional setting with $p \gg n$, there is a score-based estimator $\widehat{B}$ such that:

1. $\widehat{B}$ provably recovers the correct graphical model (i.e. $\text{supp}(\widehat{B}) = \text{supp}(\widetilde{B})$),

2. $\widehat{B}$ consistently estimates the edge weights (i.e. $\|\widehat{B} - \widetilde{B}\|_F \xrightarrow{P} 0$).

Along the way, we establish some novel results regarding neighbourhood estimation in directed graphical models.

# 1 Local minimizers

In this section, we consider the reparamterized estimator defined in Section 1.2 by (2.3, 2.4). Since this program is nonconvex, we must be careful when discussing "solutions" to (2.3). The estimator is defined to be the global minimum of the penalized loss, but theoretical guarantees are generally only available for local minimizers, and the results in this section will be no expection. Later, in Section 2, we will provide guarantees for global minimizers under least squares loss.

## 1.1 Nonidentifiability and sparsity

Based on observational data alone, the inverse covariance matrix $\Gamma$ is identifiable, but a DAG $(\widetilde{B}, \widetilde{\Omega})$ satisfying (1.2) is not. The usual theory of maximum likelihood estimation assumes identifiability, but it is possible to derive similar optimality results when the true parameter is nonidentifiable (see for instance Redner, 1981).

When the model is identifiable, one establishes the existence of a consistent local minimizer for the true parameter, which is unique (e.g as in Fan and Li, 2001). It turns out that even if the model is nonidentifiable, we can still obtain a consistent local minimizer for each equivalent parameter. As long as there are finitely many equivalent parameters, these minimizers are unique to each parameter. In particular, in the context of DAG estimation, there are up to $p!$ equivalent parameters in the equivalence class $\mathfrak{D}$ (Lemma 2.2). Thus we have a finite collection of local minimizers that serve as "candidates" for the global minimum; the question that remains is which one of these minimizers does our estimator

produce?

Each equivalent parameter has the same likelihood, so the only quantity we have to distinguish these minimizers is the penalty term. Our theory will show that by properly controlling the amount of regularization, it is possible to distinguish the *sparsest* DAGs in $\mathfrak{D}$ in the sense that they will each have strictly smaller penalized loss than their competitors. Moreover, this analysis can be transferred over to the *empirical* local minimizers, so that the sparsest local minimizer has the smallest penalized loss. Because of nonconvexity, however, it is hard to guarantee that these minimizers are the *only* local minimizers, and hence that the sparsest DAGs are the global minimizers. The simulations from Chapter 2 give us good empirical evidence that our estimator indeed approximates the sparsest DAG representation of $\Gamma$, as opposed to another DAG with many more edges.

The remainder of this section undertakes the details of this analysis. To stay consistent with the literature, instead of minimizing the penalized loss (2.3) we will maximize the penalized log-likelihood, which is of course only a technical distinction. We begin with a discussion of the technical results and assumptions which establish the existence of consistent local maximizers before stating our main result in Section 1.4. We also briefly discuss the high-dimensional scenario in which $p$ is allowed to depend on $n$.

*Remark* 1.1. For some classes of models, including nonlinear and non-Gaussian models, the DAG estimation problem considered here is known to be identifiable based on observational data alone (Shimizu et al., 2006; Peters et al., 2012), and some methods have been developed to estimate such models (Hyvärinen et al., 2010). In contrast to these developments, the main technical difficulty in our analysis is the nonidentifiability of the general Gaussian model.

## 1.2 Formal preliminaries

Conceptually this theory is quite simple: we have a function $F$ on $\mathbb{R}^{p^2}$ which we would like to maximize over the subspace $\mathbb{D} \times \mathbb{R}^p_+$. In the ensuing theoretical analysis, it will be easier to work with a single parameter vector $\nu$ (vs. the two matrices $\Phi$ and $R$), so we first transform our parameter space in this way without any loss of generality. In order to properly specify a topology for this space, and to ensure that the translation between our statistical model for $(B, \Omega)$ and the mathematical model for $\nu$ is coherent, we carefully outline the mathematical set-up here.

Given a DAG $(B, \Omega)$, consider the reparametrization $(\Phi, R)$ given by

$$\Phi = B\Omega^{-1/2} \tag{3.1}$$

$$R = \Omega^{-1/2}. \tag{3.2}$$

This is of course just the matrix version of the reparametrization that leads to (2.1). Now define the following function which maps $(\Phi, R) \in \mathbb{R}^{p\times p} \times \mathbb{R}^{p\times p}$ into $\mathbb{R}^{p^2}$:

$$\nu(\Phi, R) = \mathrm{vec}(U) = (u_1, \ldots, u_p), \quad \text{where } U = [u_1 \mid \ldots \mid u_p] = R + \Phi.$$

Recall that $\Phi$ has zeroes on the diagonal and $R$ is a diagonal matrix, so that the sum $U := R + \Phi$ has the same number of nonzero entries as $R$ and $\Phi$ separately. Furthermore, the sparsity pattern of the off-diagonal elements of $U$ exactly matches that of $\Phi$.

In the proofs, when there is no confusion we will simply write $\nu = U = (\Phi, R) = (B, \Omega)$ to mean that these are all equivalent representations of the same DAG in various parametrizations. We will also refer to $\nu$ simply as a DAG. Mathematically, we will work with $\nu$, however, our results should always be interpreted in terms of the original model $(B, \Omega)$.

The parameter space of is formally defined as follows:

$$\mathbb{V} := \left\{ \nu = \nu(\Phi, R) \in \mathbb{R}^{p^2} : \Phi \in \mathbb{D},\ R \in \mathbb{R}_+^p \right\}.$$

This space inherits its topology from the ambient space $\mathbb{R}^{p^2}$, and it is this space on which we wish to maximize the function $F(\nu) = \ell_n(\nu) - n\rho_{\lambda_n}(\nu)$.

### 1.3 Existence of local solutions

The true distribution is uniquely defined by its inverse covariance matrix, $\Gamma$. By equation (2.2), given $(\widehat{\Phi}, \widehat{R})$ we may consider the resulting estimate of the inverse covariance matrix $\widehat{\Gamma} = \Gamma(\widehat{\Phi}, \widehat{R})$. By analogy, for any DAG $\nu \in \mathbb{R}^{p^2}$, we may define in the obvious way the matrix $\Gamma(\nu)$. Thus the parameter $\nu$ is simply another parametrization of the normal distribution: For any $\Gamma$, there exists $\nu \in \mathbb{V}$ such that $\Gamma = \Gamma(\nu)$. Let $\mathfrak{D} = \mathfrak{D}(\Gamma) = \{\nu \in \mathbb{R}^{p^2} : \Gamma(\nu) = \Gamma\}$. We will denote an arbitrary element of $\mathfrak{D}$ by $\widetilde{\nu}$ and a minimal-edge DAG in $\mathfrak{D}$ by $\nu^*$.

As is customary, we denote the support set of a vector by $\mathrm{supp}(\nu) := \{j : \nu_j \neq 0\}$, and likewise for matrices $\mathrm{supp}(B) := \{(i,j) : \beta_{ij} \neq 0\}$. Let $\ell_n(\nu \,|\, \mathbf{X})$ be the unpenalized log-likelihood of the parameter vector $\nu$ and define

$$\rho_\lambda(\nu) = \sum_{i \neq j} \rho_\lambda(|u_{ij}|), \tag{3.3}$$

where $u_{ij}$ denote the elements of $U$. Note that we are penalizing only the off-diagonal elements of $U$, which correspond to the elements of $\Phi$. Now let

$$F(\nu) := \ell_n(\nu \,|\, \mathbf{X}) - n\, \rho_{\lambda_n}(\nu). \tag{3.4}$$

We are interested in maximizing $F$ over $\mathbb{V}$.

For any $\widetilde{\nu} \in \mathfrak{D}$ which represents a DAG $(\Phi_0, R_0) = ((\phi^0_{ij}), (\rho^0_j))$ as described above, define two sequences which depend on the choice of penalty $\rho_\lambda$:

$$a_n(\widetilde{\nu}) := \max\{|\rho'_{\lambda_n}(|\phi^0_{ij}|)| : \phi^0_{ij} \neq 0\}, \tag{3.5}$$

$$b_n(\widetilde{\nu}) := \max\{|\rho''_{\lambda_n}(|\phi^0_{ij}|)| : \phi^0_{ij} \neq 0\}. \tag{3.6}$$

When it is clear from context, the dependence of $a_n$ and $b_n$ on $\widetilde{\nu}$ will be suppressed. Finally, let $\tau(\lambda) := \sup_t \rho_\lambda(t)$, which may be infinite. For the MCP we have $\tau(\lambda) = \lambda^2\gamma/2$ and for the $\ell_1$ penalty $\tau(\lambda) = +\infty$.

The following result, which is similar in spirit to Theorem 2 of Fu and Zhou (2013), guarantees the existence of a consistent local maximizer:

*Theorem* 1.1. Fix $p \geq 1$. If there exists $\widetilde{\nu} \in \mathfrak{D}$ with $b_n(\widetilde{\nu}) \to 0$, then there is a local maximizer $\widehat{\nu}_n$ of $F(\nu)$ such that

$$\|\widehat{\nu}_n - \widetilde{\nu}\| = O_P(n^{-1/2} + a_n(\widetilde{\nu})).$$

When $a_n = O(n^{-1/2})$, we obtain a $n^{1/2}$-consistent estimator of $\widetilde{\nu}$. Note that by Lemma 2.2, if $\widetilde{\nu} \in \mathfrak{D}$ then $\widetilde{\nu} = (\widetilde{B}(\pi), \widetilde{\Omega}(\pi))$ for some permutation $\pi$. For this reason, in the sequel we shall refer to the local maximizer $\widehat{\nu}_n$ as the *$\pi$-local maximizer* of $F$ for the permutation $\pi$. This theorem says that as long as the curvature of the penalty at $(\widetilde{B}(\pi), \widetilde{\Omega}(\pi))$ tends to zero, the penalized likelihood has a $\pi$-local maximizer that converges to $(\widetilde{B}(\pi), \widetilde{\Omega}(\pi))$ as $n \to \infty$.

Under additional assumptions on the penalty function, we may further strengthen this result to include consistency in structure estimation when $p$ remains fixed:

*Theorem* 1.2. Assume that the penalty function satisfies

$$\liminf_{n\to\infty} \liminf_{t\to 0^+} \rho'_{\lambda_n}(t)/\lambda_n > 0. \tag{3.7}$$

Assume further that $\widetilde{\nu} \in \mathfrak{D}$ satisfies $a_n(\widetilde{\nu}) = O(n^{-1/2})$, $b_n(\widetilde{\nu}) \to 0$, and let $\widehat{\nu}_n$ be a $\pi$-local maximizer from Theorem 1.1. If $\lambda_n \to 0$ and $\lambda_n n^{1/2} \to \infty$, then

$$P(\mathrm{supp}(\widehat{\nu}_n) = \mathrm{supp}(\widetilde{\nu})) \to 1. \tag{3.8}$$

In fact, this follows immediately from Theorem 1.1 above and Theorem 2 in Fan and Li (2001).

We must be careful in interpreting these theorems correctly: They do not imply necessarily that the estimator defined in (2.3, 2.4) is consistent. These

theorems simply show that under the right conditions, there is a local maximizer of $F$ that is consistent. It remains to establish that the global maximizer of $F$ is indeed one of these local maximizers.

*Remark* 1.2. If we assume that the conditions of Theorems 1.1 and 1.2 hold for *all* $\widetilde{\nu} \in \mathfrak{D}$, then we can conclude that every equivalent DAG has a $\pi$-local maximizer that selects the correct sparse structure. This is trivial since we assume $p$ to be fixed as $n \to \infty$, which allows us to bound the probabilities over all $p!$ choices of $\widetilde{\nu}$ simultaneously. Since the number of equivalent DAGs grows super-exponentially as $p$ increases, bounding these probabilities when $p = p_n$ grows with $n$ is the main obstacle to achieving useful results in high-dimensions.

The proofs of these two theorems are found in the appendix. In the course of the proofs, we will need the following lemma:

**Lemma 1.1.** *If $B_1 \neq B_2$ are DAGs that have a common topological sort, then for any choices of $\Omega_1$ and $\Omega_2$, we have $\Gamma(B_1, \Omega_1) \neq \Gamma(B_2, \Omega_2)$. A similar result holds in the parametrization $(\Phi, R)$.*

The assumption that two DAGs have a common topological sort is equivalent to each DAG being compatible with the same permutation $\pi$. The following lemma shows that the $\widetilde{\nu}$ are isolated, which guarantees that $\pi$-local maximizers do not cluster around multiple $\widetilde{\nu}$. For any $\varepsilon > 0$, we denote the $\varepsilon$-neighbourhood of $\widetilde{\nu}$ in $\mathbb{V}$ by $B(\widetilde{\nu}, \varepsilon) := \{\nu \in \mathbb{V} : \|\nu - \widetilde{\nu}\| < \varepsilon\}$.

**Lemma 1.2.** *For any positive definite $\Gamma$ there exists $\varepsilon > 0$ such that $\mathfrak{D} \cap B(\widetilde{\nu}, \varepsilon) = \{\widetilde{\nu}\}$ for any $\widetilde{\nu} \in \mathfrak{D}$.*

The proofs of these lemmas are also found in the appendix.

### 1.4 The main result

We will now significantly strengthen Theorems 1.1 and 1.2 by showing that, under a concave penalty, a sparsest DAG $\nu^* \in \mathfrak{D}$ maximizes the penalized likelihood amongst all the possible equivalent representations of the covariance matrix $\Gamma$. Under the assumptions of Theorem 1.1, there is a $\pi$-local maximizer $\widehat{\nu}_n^*$ of $F(\nu)$ such that $\|\widehat{\nu}_n^* - \nu^*\| = O_P(n^{-1/2} + a_n(\nu^*))$. Ideally, when $\widetilde{\nu}$ has more edges than $\nu^*$, we would like these $\pi$-local maximizers to satisfy $F(\widehat{\nu}_n^*) > F(\widehat{\nu}_n)$ with high probability.

Intuitively, when $a_n(\widetilde{\nu}) = b_n(\widetilde{\nu}) = 0$, all of the nonzero coefficients lie in the flat part of the penalty where $\rho'_{\lambda_n}(|\phi_{ij}^0|) = \rho''_{\lambda_n}(|\phi_{ij}^0|) = 0$. When this happens, the penalty "acts" like the $\ell_0$ penalty by penalizing all of the coefficients equally by the amount $\tau(\lambda_n)$, and any DAG with more edges than $\nu^*$ will see a heavier penalty. In order to quantify "how close" $\widetilde{\nu}$ is to lying in the flat part of the penalty, we define

$$c_n(\widetilde{\nu}) := \min\{\rho_{\lambda_n}(|\phi_{ij}^0|) : \phi_{ij}^0 \neq 0\}.$$

When $c_n(\widetilde{\nu}) = \tau(\lambda_n)$, the penalty mimics the $\ell_0$ penalty, and since the likelihood $\ell_n(\widetilde{\nu} \,|\, X)$ is constant for all $\widetilde{\nu}$, we would then have

$$\rho_{\lambda_n}(\nu^*) < \rho_{\lambda_n}(\widetilde{\nu}) \iff \ell_n(\nu^* \,|\, \mathbf{X}) - n\,\rho_{\lambda_n}(\nu^*) > \ell_n(\widetilde{\nu} \,|\, \mathbf{X}) - n\,\rho_{\lambda_n}(\widetilde{\nu}).$$

One would hope that for local maximizers $\widehat{\nu}_n$ that are sufficiently close to the $\widetilde{\nu}$, the continuity of $F$ would guarantee that this intuition persists. As long as the amount of regularization grows fast enough, this is precisely the case:

*Theorem* 1.3. Suppose that $\rho_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $\rho_\lambda(0) = 0$. Assume further that the conditions for Theorem 1.2 hold for all $\widetilde{\nu} \in \mathfrak{D}$. Recall that $\tau(\lambda_n) := \sup_t \rho_{\lambda_n}(t)$. If

1. $c_n(\widetilde{\nu}) = \tau(\lambda_n) + O(n^{-1/2})$ for all $\widetilde{\nu} \in \mathfrak{D}$,

2. $\limsup_n \tau(\lambda_n) < \infty$,

3. $\tau(\lambda_n) n^{1/2} \to \infty$,

then for any DAG $\widetilde{\nu} \in \mathfrak{D}$ with strictly more edges than $\nu^*$, $P(F(\widehat{\nu}_n^*) > F(\widehat{\nu}_n)) \to 1$ as $n \to \infty$.

The restriction to $\widetilde{\nu}$ with strictly more edges than $\nu^*$ is necessary since $\nu^*$ may not be unique in general. Theorem 1.3 essentially answers the question of which DAG in the true equivalence class $\mathfrak{D}$ our estimator approximates. As we have discussed, there is a subtle technicality in which it is possible that there are *other* maximizers of $F(\nu)$ besides the $\pi$-local maximizers, but this is unlikely in practice.

These theorems provide general technical statements which can be used when weaker assumptions are necessary. By imposing all the conditions in Theorems 1.1, 1.2, and 1.3 uniformly, we can combine all of the results in order to characterize the behaviour of the estimates in terms of the parametrization $(\widehat{B}, \widehat{\Omega})$ given by (2.4). Before stating the main theorem, we will need some notation to distinguish $\pi$-local maximizers. Assuming the conditions of Theorem 1.1 hold for all $\pi$, denote the collection of $\pi$-local maximizers by $\mathcal{M}_n$. Continuing our notation from the previous section, we also let $(B^*, \Omega^*)$ denote any graph in $\mathfrak{D}$ with the fewest number of edges, and let $(\widehat{B}^*, \widehat{\Omega}^*)$ be the corresponding $\pi$-local maximizer. Recall that given a DAG estimate $(\widehat{B}, \widehat{\Omega})$, we define $\widehat{\Gamma} = \Gamma(\widehat{B}, \widehat{\Omega})$.

*Theorem* 1.4. Suppose that $\rho_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $\rho_\lambda(0) = 0$. Fix $p \geq 1$ and assume that the penalty function satisfies

$$\liminf_{n\to\infty} \liminf_{t\to 0^+} \rho'_{\lambda_n}(t)/\lambda_n > 0.$$

Assume further that $a_n(\widetilde{\nu}) = O(n^{-1/2})$, $b_n(\widetilde{\nu}) \to 0$, and $c_n(\widetilde{\nu}) = \tau(\lambda_n) + O(n^{-1/2})$ for each DAG in $\mathfrak{D}$. If $\lambda_n \to 0$, $\lambda_n n^{1/2} \to \infty$, $\limsup_n \tau(\lambda_n) < \infty$, and $\tau(\lambda_n) n^{1/2} \to \infty$, then for any permutation $\pi$, there is a local maximizer $(\widehat{B}, \widehat{\Omega})$ of $F$ such that

1. $\|\widehat{B} - \tilde{B}_0(\pi)\|_F + \|\widehat{\Omega} - \tilde{\Omega}_0(\pi)\|_F = O_P(n^{-1/2})$,

2. $P(\text{supp}(\widehat{B}) = \text{supp}(\tilde{B}_0(\pi))) \to 1$,

3. $\|\widehat{\Gamma} - \Gamma\|_F = O_P(n^{-1/2})$.

Furthermore,

$$P\left(F(\widehat{B}^*, \widehat{\Omega}^*) = \max_{(\widehat{B},\widehat{\Omega})\in\mathcal{M}_n} F(\widehat{B}, \widehat{\Omega})\right) \to 1.$$

The proof of Theorem 1.4 is immediate from the properties of the Frobenius norm and Theorems 1.1, 1.2, and 1.3.

*Remark* 1.3. Using an adaptive $\ell_1$ penalty, Fu and Zhou (2013) first obtained results similar to Theorems 1.1 and 1.2. These results assume a weakened form of faithfulness, however, and require experimental data with interventions in order to guarantee identifiability of the true causal DAG. The results here generalize this theory to observational data without needing faithfulness. The keys to this generalization are the notion of parametric equivalence in Definition 2.4 (as opposed to Markov equivalence) and the use of a concave penalty to rule out equivalent DAGs with too many edges. The role of concavity is highlighted by the observation that convex penalties cannot satisfy the conditions for Theorem 1.3.

## 1.5 Discussion of the assumptions

The general theme behind the theory described in the previous sections is that as long as the penalty is chosen cleverly enough, there will be a consistent local maximizer for the constrained penalized likelihood problem (2.3). We pause now to discuss these conditions more carefully, and show that they can always be satisfied.

The parameters $a_n(\widetilde{\nu})$ and $b_n(\widetilde{\nu})$ measure respectively the maximum slope and concavity of the penalty function, and the conditions on these terms are derived directly from Fan and Li (2001). The idea is that as long as the concavity of the

penalty is overcome by the local convexity of the log-likelihood function, our intuition from classical maximum likelihood theory continues to hold true. In order to simultaneously guarantee consistency in parameter estimation and structure learning, it is necessary that these parameters vanish asymptotically.

Furthermore, the assumptions on $a_n$ and $b_n$ in Theorems 1.1 and 1.2 highlight the advantages of concave regularization over $\ell_1$ regularization. In particular, the $\ell_1$ penalty trivially satisfies $b_n \to 0$, but cannot simultaneously satisfy $a_n(\widetilde{\nu}) = \lambda_n = O(n^{-1/2})$ and $\lambda_n n^{1/2} \to \infty$. Thus, for the $\ell_1$ penalty, we may apply Theorem 1.1 to obtain a local maximizer which is consistent in *parameter estimation*, but we cannot guarantee structure estimation consistency through Theorem 1.2. In contrast, these conditions are easily satisfied by a concave penalty; in particular they are satisfied when $\rho_\lambda$ is the MCP. These observations were first made in Fan and Li (2001).

The conditions on $\tau(\lambda_n)$ in Theorem 1.3 are more interesting. When the true parameter is identifiable, there is no concern about dominating the penalized likelihood for nonsparse parameters. Since our set-up is decidedly nonidentifiable—there are up to $p!$ choices of the "true" graph—it is essential to control the growth of the penalty, and more specifically, how the penalty grows at the various equivalent DAGs $\widetilde{\nu} \in \mathfrak{D}$. As long as this grows at the right rate, nonsparse graphs will see the penalty term dominate, and as a result the sparsest graph $(B^*, \Omega^*)$ emerges as the best estimate of the true graph. Since $\tau(\lambda_n) = +\infty$ for any convex penalty, Theorem 1.3 along with the remainder of this discussion do not apply to $\ell_1$ regularization.

In order to quantify the behaviour of the penalty, we need to control the growth of two different quantities: the maximum penalty $\tau(\lambda_n)$, and the rate of convergence of $c_n(\widetilde{\nu})$. By rate of convergence, we refer to the fact that the assumptions on $a_n(\widetilde{\nu})$ and $b_n(\widetilde{\nu})$ alone require that $c_n(\widetilde{\nu}) = \tau(\lambda_n) + o(1)$, or equivalently $\rho_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n) + o(1)$ whenever $\phi^0_{ij} \neq 0$. The stronger assumption that

$c_n(\widetilde{\nu}) = \tau(\lambda_n) + O(n^{-1/2})$ in Theorem 1.3 shows that it is not enough that this convergence occurs at an arbitrary rate. One may think of this as a requirement on the zeroth-order convergence of $\rho_{\lambda_n}$, in contrast to the first- and second-order convergence required by Theorems 1.1 and 1.2. In practice, it is sufficient to have $c_n(\widetilde{\nu}) = \tau(\lambda_n)$ for sufficiently large $n$, and hence also $a_n = b_n = 0$.

Of course, none of this is relevant if we cannot construct a penalty which satisfies all of these conditions simultaneously along with associated regularization parameters $\lambda_n$. When the penalty is chosen to be the MCP, all of the conditions required for Theorem 1.4 are satisfied as long as

$$\limsup_n \lambda_n \gamma_n < \min_{\widetilde{\nu} \in \mathfrak{D}} \min\{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\} \quad \text{and} \quad \lambda_n = O(n^{-\alpha}),\ 0 < \alpha < 1/2. \tag{3.9}$$

*Remark* 1.4. To better understand the conditions on $\tau(\lambda_n)$ in Theorems 1.3 and 1.4, it is instructive to consider the simplified case in which the penalty factors as $\rho_{\lambda_n}(t) = \lambda_n \rho(t)$ for some function $\rho(t)$ (not to be confused with the parameters $\rho_j$ in our model). In this case, the penalty is bounded as long as $\lim_{t\to\infty} \rho(t) < \infty$ and the conditions on $\tau(\lambda_n)$ in Theorem 1.3 reduce to $\limsup_n \lambda_n < \infty$ and $\lambda_n n^{1/2} \to \infty$. When $\lambda_n \to 0$, these conditions are simply the assumptions in Theorem 1.2. Thus, the extra conditions on $\tau(\lambda_n)$ in Theorems 1.3 and 1.4 are redundant when the penalty factors in this way.

*Example* 1.1. Although the usual formula for the MCP does not satisfy the factorization property in Remark 1.4, we may reparametrize it so that it does. To do this, define a new penalty by

$$\overline{p}_\lambda(t;\delta) := \lambda \left( t - \frac{t^2}{2\delta} \right) 1(t < \delta) + \frac{\lambda\delta}{2} 1(t \geq \delta), \qquad t \geq 0.$$

Then $\overline{p}_\lambda(t;\delta) = \lambda \cdot \overline{p}_{\lambda=1}(t;\delta)$, and by choosing $\delta = \lambda\gamma$ we may recover the usual formula for the MCP given by (2.5). Furthermore, the condition in (3.9) becomes

$$\limsup_n \delta_n < \min_{\widetilde{\nu} \in \mathfrak{D}} \min\{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\},$$

which is independent of $\lambda_n$.

## 1.6 Technical proofs

Here we provide proofs of all the main results from Section 1.

### 1.6.1 Proof of Theorem 1.1

We begin by formalizing some of the background material on the Cholesky decomposition used in Section 2.3, which will also be used in the proof of Lemma 1.1. First recall the following standard result:

**Lemma 1.3.** *For any symmetric positive definite matrix $A \in \mathbb{R}^{p\times p}$ and permutation $\pi \in \mathcal{P}$, the Cholesky decomposition $A = LDL^T$ satisfies*

$$P_\pi A = (P_\pi L)(P_\pi D)(P_\pi L)^T,$$

*where $L$ is lower triangular and $D$ is a diagonal matrix.*

Now suppose $\Gamma$ is given and use the Cholesky decomposition to write $\Gamma = \Gamma(L, D)$ as in (1.4). Then, taking $A = \Gamma(L, D)$ in Lemma 1.3, we obtain $P_\pi\Gamma(L, D) = \Gamma(P_\pi L, P_\pi D)$. Alternatively, suppose $(B, \Omega) \in \mathfrak{D}(\Gamma)$ and suppose $\pi \in \mathcal{P}$ is compatible with $(B, \Omega)$. Since $P_\pi B$ is lower-triangular, by taking $A = \Gamma(P_\pi B, P_\pi\Omega)$, we may similarly deduce

$$P_{\pi^{-1}}\Gamma(P_\pi B, P_\pi\Omega) = \Gamma(B, \Omega) \implies \Gamma(P_\pi B, P_\pi\Omega) = P_\pi\Gamma(B, \Omega).$$

This proves the following lemma, which will be useful:

**Lemma 1.4.** *Let $(B, \Omega)$ be a DAG. For any permutation $\pi \in \mathcal{P}$ that is compatible with $(B, \Omega)$, we have*

$$P_\pi\Gamma(B, \Omega) = \Gamma(P_\pi B, P_\pi\Omega).$$

We now prove Lemma 1.1, which will be used in the proof of Theorem 1.1.

*Proof of Lemma 1.1.* We only prove this for the original parameterization $(B, \Omega)$; the reparameterized case is similar.

Since $B_1$ and $B_2$ have a common topological sort, there is a permutation $\pi$ of the vertices that orders $B_1$ and $B_2$ simultaneously, so that $P_\pi B_1$ and $P_\pi B_2$ are both strictly lower triangular. Suppose then that $\Gamma(B_1, \Omega_1) = \Gamma(B_2, \Omega_2) := \tilde{\Gamma}$, so that (using Lemma 1.4 above)

$$\begin{aligned}
&P_\pi \Gamma(B_1, \Omega_1) = P_\pi \Gamma(B_2, \Omega_2) \\
\iff &\Gamma(P_\pi B_1, P_\pi \Omega_1) = \Gamma(P_\pi B_2, P_\pi \Omega_2) \\
\iff &(I - P_\pi B_1)(P_\pi \Omega_1)^{-1}(I - P_\pi B_1)^T = (I - P_\pi B_2)(P_\pi \Omega_2)^{-1}(I - P_\pi B_2)^T.
\end{aligned}$$

The last expression is equal to $P_\pi \tilde{\Gamma}$, which is a symmetric positive definite matrix. By the uniqueness of the Cholesky factorization, we must have

$$\begin{aligned}
I - P_\pi B_1 &= I - P_\pi B_2 \\
(P_\pi \Omega_1)^{-1} &= (P_\pi \Omega_2)^{-1},
\end{aligned}$$

which implies

$$B_1 = B_2, \quad \Omega_1 = \Omega_2.$$

Since $B_1$ was assumed to be distinct from $B_2$, this contradiction establishes the desired result. □

*Proof of Theorem 1.1.* Suppose $\widetilde{\nu} \in \mathfrak{D}$ with $b_n(\widetilde{\nu}) \to 0$. It suffices to check Conditions (A)-(C) from Fan and Li (2001), which are simply the standard regularity conditions for asymptotic efficiency of ordinary maximum likelihood estimates. Model identifiability is not an issue since the same analysis can be carried out for any equivalent parameter (see Section 1.1). Since the densities $f(\cdot \,|\, \nu)$ are Gaussian, the only condition that needs to be checked is that the Fisher information

is positive definite at $\widetilde{\nu}$ restricted to the DAG space $\mathcal{D}$. Theorem 1.1 will then follow immediately from Theorem 1 in Fan and Li (2001).

Let $I(\widetilde{\nu})$ denote the usual Fisher information matrix at this point; we will show that $I(\widetilde{\nu})$ is positive definite. Since $f$ is always a Gaussian density, it will suffice to show that $f(\cdot \,|\, \nu) \neq f(\cdot \,|\, \widetilde{\nu})$ for $\nu$ in a sufficiently small neighbourhood of $\widetilde{\nu}$.

Now suppose $\nu = (\Phi, R)$ is in an arbitrarily small neighbourhood of $\widetilde{\nu} = (\Phi_0, R_0)$. Then it must hold that $\phi_{ij} \neq 0$ whenever $\phi^0_{ij} \neq 0$. Indeed, otherwise

$$\|\Phi - \Phi_0\|^2 \geq (\phi_{ij} - \phi^0_{ij})^2 = |\phi^0_{ij}|^2.$$

Thus, $\phi^0_{ij} \neq 0$ implies $\phi_{ij} \neq 0$, or $i \to j$ in $\Phi_0$ implies $i \to j$ in any DAG close to $\Phi_0$. In particular, $\Phi$ contains all the edges (including orientation) in $\Phi_0$, with the possible addition of extra edges. That is, $\Phi_0$ is a subgraph of $\Phi$. It follows that there is an ordering of the vertices that is compatible with $\Phi$ and $\Phi_0$ simultaneously. Since $\Phi \neq \Phi_0$, it follows from Lemma 1.1 that $\Gamma(\nu) \neq \Gamma(\widetilde{\nu})$, whence $f(\cdot \,|\, \nu) \neq f(\cdot \,|\, \widetilde{\nu})$. □

*Proof of Lemma 1.2.* Note that Lemma 2.2 implies that the equivalence class $\mathfrak{D}$ is finite. Set $\varepsilon = \min_{\widetilde{\nu} \in \mathfrak{D}} \min_{i,j} \{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\} > 0$. Then if $\|\Phi - \Phi_0\| \leq \|\nu - \widetilde{\nu}\| < \varepsilon$, the arguments in the proof of Theorem 1.1 guarantee the existence of an ordering that is compatible with $\Phi$ and $\Phi_0$, and the result follows from Lemma 1.1. □

### 1.6.2 Proof of Theorem 1.3

Instead of directly proving Theorem 1.3, we will prove a slightly more general statement under weaker assumptions. Theorem 1.3 will then follow as a special case.

The following technical lemmas ensure that the objective function $F(\nu)$ is well-behaved with respect to taking limits. The first is a standard application of the uniform law of large numbers (see, for example, Ferguson, 1996, §16) and the

second is a direct consequence of concavity.

**Lemma 1.5.** *Fix $\widetilde{\nu}$ and suppose $\nu_n$ is a sequence with $\|\nu_n - \widetilde{\nu}\| = o(1)$. If the empirical log-likelihood $\ell_n(\nu)$ is continuous for all $n$, then*

$$P\left(\lim_{n\to\infty}\frac{1}{n}\ell_n(\nu_n) = \lim_{n\to\infty}\frac{1}{n}\ell_n(\widetilde{\nu})\right) = 1.$$

**Lemma 1.6.** *Suppose that $\rho_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $\rho_\lambda(0) = 0$. If $\limsup_n \tau(\lambda_n) < \infty$, then for any $x_0 > 0$ there exists a constant $C$, depending only on $x_0$, such that*

$$|\rho_{\lambda_n}(x) - \rho_{\lambda_n}(x_0)| \leq C|x - x_0| \quad \textit{for all } x \geq 0 \textit{ and all } n.$$

Recall that $f(n) = \omega(g(n)) \iff g(n) = o(f(n))$, that is, for every $C > 0$,

$$f(n) \geq Cg(n) \quad \text{for all large } n.$$

As in Section 1, we use $\widehat{\nu}_n$ and $\widehat{\nu}_n^*$ to denote the local maximizers close to $\widetilde{\nu}$ and $\nu^*$, respectively, whose existence is guaranteed by Theorem 1.1.

*Theorem* 1.5. Suppose that $\rho_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $\rho_\lambda(0) = 0$. Let $\widetilde{\nu} \in \mathfrak{D}$ be a DAG with strictly more edges than $\nu^*$. Assume further that the conditions for Theorem 1.2 hold for both $\widetilde{\nu}$ and $\nu^*$. If

1. $c_n(\nu^*) = \tau(\lambda_n) + O(n^{-1/2})$ and $c_n(\widetilde{\nu}) = \tau(\lambda_n) + O(n^{-1/2})$,
2. $\limsup_n \tau(\lambda_n) < \infty$,
3. $\tau(\lambda_n) = \omega(n^{-1/2})$,

then for every $\varepsilon > 0$,

$$P\left(\ell_n(\widehat{\nu}_n^*) - n\,\rho_{\lambda_n}(\widehat{\nu}_n^*) > \ell_n(\widehat{\nu}_n) - n\,\rho_{\lambda_n}(\widehat{\nu}_n)\right) \geq 1 - \varepsilon \quad \text{for sufficiently large } n.$$

*Proof.* Since we assume Theorem 1.2 holds for both $\widetilde{\nu}$ and $\nu^*$, we may assume without loss of generality that $\operatorname{supp}(\widehat{\nu}_n^*) = \operatorname{supp}(\nu^*)$ and $\operatorname{supp}(\widehat{\nu}_n) = \operatorname{supp}(\widetilde{\nu})$.

Since $\ell_n$ is continuous for each $n$, $\|\widehat{\nu}_n - \widetilde{\nu}\| = O_P(n^{-1/2})$, and $\|\widehat{\nu}_n^* - \nu^*\| = O_P(n^{-1/2})$, Lemma 1.5 implies that

$$\frac{1}{n}(\ell_n(\widehat{\nu}_n) - \ell_n(\widehat{\nu}_n^*)) \to 0$$

almost surely. It is easy to show that in fact $n^{-1}(\ell_n(\widehat{\nu}_n) - \ell_n(\widehat{\nu}_n^*)) = O_P(n^{-1/2})$.

It will suffice to show that for any $\varepsilon > 0$, there exists an $N$ such that for all $n > N$, we have

$$P\left(\rho_{\lambda_n}(\widehat{\nu}_n) - \rho_{\lambda_n}(\widehat{\nu}_n^*) - \frac{1}{n}(\ell_n(\widehat{\nu}_n) - \ell_n(\widehat{\nu}_n^*)) > 0\right) \geq 1 - \varepsilon.$$

Given $\varepsilon > 0$, there exists $M > 0$ such that

$$P\left(\frac{1}{n}(\ell_n(\widehat{\nu}_n) - \ell_n(\widehat{\nu}_n^*)) \leq Mn^{-1/2}\right) \geq 1 - \varepsilon,$$

so that it suffices to check that $\rho_{\lambda_n}(\widehat{\nu}_n) - \rho_{\lambda_n}(\widehat{\nu}_n^*) > Mn^{-1/2}$ for sufficiently large $n$.

Lemma 1.6 implies that for each $\phi_{ij}^0 \neq 0$,

$$|\rho_{\lambda_n}(\widehat{\phi}_{ij}^0) - \rho_{\lambda_n}(\phi_{ij}^0)| \leq C|\widehat{\phi}_{ij}^0 - \phi_{ij}^0| = O(n^{-1/2}),$$

and similarly for all $\phi_{ij}^* \neq 0$. Thus we can write $\rho_{\lambda_n}(\widehat{\nu}_n) = \rho_{\lambda_n}(\widetilde{\nu}) + O_P(n^{-1/2})$ and similarly for $\widehat{\nu}^*$. It thus suffices to show that

$$\rho_{\lambda_n}(\widetilde{\nu}) - \rho_{\lambda_n}(\nu^*) = \omega(n^{-1/2}).$$

Now, using Condition 1,

$$\begin{aligned}
\rho_{\lambda_n}(\widetilde{\nu}) - \rho_{\lambda_n}(\nu^*) &= \sum_{\phi_{ij}^0 \neq 0} \rho_{\lambda_n}(|\phi_{ij}^0|) - \sum_{\phi_{ij}^* \neq 0} \rho_{\lambda_n}(|\phi_{ij}^*|) \\
&\geq s_0 c_n(\widetilde{\nu}) - s^* \tau(\lambda_n) + s^* \tau(\lambda_n) - \sum_{\phi_{ij}^* \neq 0} \rho_{\lambda_n}(|\phi_{ij}^*|) \\
&= (s_0 - s^*)\tau(\lambda_n) + O(n^{-1/2}) + \sum_{\phi_{ij}^* \neq 0} (\tau(\lambda_n) - \rho_{\lambda_n}(\phi_{ij}^*)) \\
&\geq (s_0 - s^*)\tau(\lambda_n) + O(n^{-1/2}).
\end{aligned}$$

Since $\tau(\lambda_n) = \omega(n^{-1/2})$ (Condition 3), it follows that $\rho_{\lambda_n}(\widetilde{\nu}) - \rho_{\lambda_n}(\nu^*) \geq \omega(n^{-1/2})$, from which the claim follows. □

*Proof of Theorem 1.3.* Condition 3 in Theorem 1.5 is equivalent to $\tau(\lambda_n)/n^{-1/2} \to \infty$, and Theorem 1.3 follows as a special case since the equivalence class $\mathfrak{D}(\Sigma)$ is finite. □

## 2 Global minimizers

In the previous section we studied the properties of local minimizers to the program (2.3). Due to reparameterization, this program is *not* equivalent to (1.11), although similar arguments may be used to derive comparable results for the parameterization $(B, \Omega)$. We will now turn our attention to global minimizers.

Instead of the program (1.11), we will consider a program defined via penalized least squares. Define an objective function

$$Q(B) = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}B\|_F^2 + \rho_\lambda(B), \tag{3.10}$$

and corresponding estimator

$$\widehat{B} \in \operatorname*{arg\,min}_{B \in \mathbb{D}} Q(B). \tag{3.11}$$

This is a constrained penalized least squares estimator for $\widetilde{B}$, and note furthermore that

$$Q(B) \propto \frac{1}{n} L(B, I \,|\, \mathbf{X}) + \rho_\lambda(B),$$

where $L(\,\cdot\,,\,\cdot\,|\,\mathbf{X})$ is defined as in (1.9).

The reasons for considering a least squares loss will become more clear in the course of the proof, however, at a high-level, this estimator has the following nice properties:

- It is equivalent to the PMLE estimator (1.11) if we were to hold $\Omega$ constant (e.g. if we knew the "correct" value of $\Omega$ in advance),
- The CCDr algorithm can be used to approximate $\widehat{B}_\lambda$ simply by setting $\rho_j = 1$ and skipping the updates for these parameters,
- It can be interpreted by and decomposed into a collection of "neighbourhood" regression problems.

Our main result will be to establish the consistency of $\widehat{B}$ in both model selection and parameter estimation in a high-dimensional setting under no assumptions on the ordering. Our results allow for arbitrary dependence between the components of $X$ as well as encompassing a general class of regularizers, including the MCP, SCAD, and $\ell_1$.

## 2.1 Background and preliminaries

In this section we provide some intuition behind the estimator (3.11) and the choice of regularizer $\rho_\lambda$.

### 2.1.1 Global and restricted minimizers

Recall that $\mathbb{D}$ is the space of $p \times p$ real matrices that represent DAGs when interpreted as weighted adjacency matrices. For each permutation $\pi \in \mathcal{P}$, define a subset of $\mathbb{D}$ by

$$\mathbb{D}[\pi] = \{B \in \mathbb{D} : P_\pi B \text{ is lower triangular}\}.$$

It turns out that for any fixed permutation $\pi$ the structure of the DAGs in $\mathbb{D}[\pi]$ is very rigid and can be described exactly. A DAG $B = [\,\beta_1 \,|\, \cdots \,|\, \beta_p\,] \in \mathbb{D}$ is in $\mathbb{D}[\pi]$ if and if only for each $j = 1, \ldots, p$ it holds that $\text{supp}(\beta_j) \subset S_j(\pi)$, where

$$S_j(\pi) := \{k : \pi^{-1}(k) > \pi^{-1}(j)\}. \tag{3.12}$$

In other words, for each node $X_j$, the permutation $\pi$ defines a unique set of candidate parents, given by $S_j(\pi)$, and if $B \in \mathbb{D}[\pi]$, then the parent sets of $B$ must come from $S_j(\pi)$. Moreover, note that $\mathbb{D} = \cup_\pi \mathbb{D}[\pi]$ and $\widetilde{B}(\pi) \in \mathbb{D}[\pi]$ for every $\pi$.

Recall the objective function defined in (3.10) and the associated estimator (3.11). The spaces $\mathbb{D}[\pi]$ help give us some intuition into what is going on with the estimator $\widehat{B}$. In fact, suppose we had oracular knowledge of some "optimal" or "best" permutation $\pi_0$ in advance. Then we can ignore any DAG that is not consistent with $\pi_0$, which is tantamount to restricting our search space to $\mathbb{D}[\pi_0]$. Thus, estimating the "oracle DAG" $\widetilde{B}(\pi_0)$ reduces to a simple autoregressive model, which is equivalent to solving

$$\widehat{B}(\pi_0) \in \underset{B \in \mathbb{D}[\pi_0]}{\arg\min}\, Q(B).$$

This motivates the following general definition:

*Definition* 2.1. A *restricted global minimizer* of $Q$ is a solution to the restricted problem

$$\widehat{B}(\pi) \in \underset{B \in \mathbb{D}[\pi]}{\arg\min}\, Q(B).$$

The columns of $\widehat{B}(\pi)$ will be denoted $\widehat{\beta}_j(\pi)$, $j = 1, \ldots, p$..

Obviously, $\widehat{B}$ must also be a restricted global minimizer. The problem is that we do not know in advance which permutation this corresponds to, i.e. for which $\pi$ we have $\widehat{B} = \widehat{B}(\pi)$. This leads us to consider the collection of all such permutations, which is formalized by the following definition.

*Definition* 2.2. The *collection of estimated permutations*, denoted by $\widehat{\mathcal{P}}$, is defined by

$$\widehat{\mathcal{P}} = \underset{\pi \in \mathcal{P}}{\arg\min}\, Q(\widehat{B}(\pi)).$$

An arbitrary element of $\widehat{\mathcal{P}}$ will be denoted by $\widehat{\pi}$.

Intuitively, we can think of $\widehat{\mathcal{P}}$ as the set of permutations $\widehat{\pi}$ such that $P_{\widehat{\pi}}\widehat{B}$ is lower-triangular, or equivalently $\widehat{B} \in \mathbb{D}[\widehat{\pi}]$. Since $\widehat{B}$ depends stochastically on the random matrix $\mathbf{X}$, the permutation $\widehat{\pi}$ is a random quantity as well. This randomness will play a pivotal role in complicating the arguments which appear in the sequel.

### 2.1.2 Choice of regularizer

It remains to decide on a choice of regularizer $\rho_\lambda$. One of the key features of this analysis is to provide some relevant insight into how $\rho_\lambda$ affects the solution to (3.11), so we avoid specifying a particular regularizer in advance. Instead, we make the following minimal assumptions on $\rho_\lambda$:

*Condition* 2.1. The regularizer $\rho_\lambda : [0, \infty) \to [0, \infty)$ is concave, nondecreasing, right-differentiable at zero with $0 < \rho'_\lambda(0+) < \infty$ and satisfies $\rho_\lambda(0) = 0$.

This condition allow for most concave penalties, including the MCP and SCAD, along with the more familiar $\ell_1$ penalty.

### 2.1.3 Abstract neighbourhood regression

We start with some formal definitions. Everything defined in this subsection are *population* level quantities that depend on $\Sigma$, but not on the sample $\mathbf{X}$. Recall the Gaussian SEM setup (1.2). We will use $\mathbb{R}^{[p]_j}$ to denote real-valued vectors indexed by $[p]_j$.

*Definition* 2.3. For any $j \in [p]$ and $S \subset [p]_j$, let

$$\begin{aligned}
\beta_j(S) &:= \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p,\, \operatorname{supp}(\beta) \subset S} \mathbb{E}\big[X_j - \beta^T X\big]^2, \\
m_j(S) &:= \operatorname{supp}(\beta_j(S)), \\
\varepsilon_j(S) &:= X_j - \beta_j(S)^T X.
\end{aligned}$$

We call $\beta_j(S)$ the SEM coefficients for variable $j$ regressed on variables $S$, and

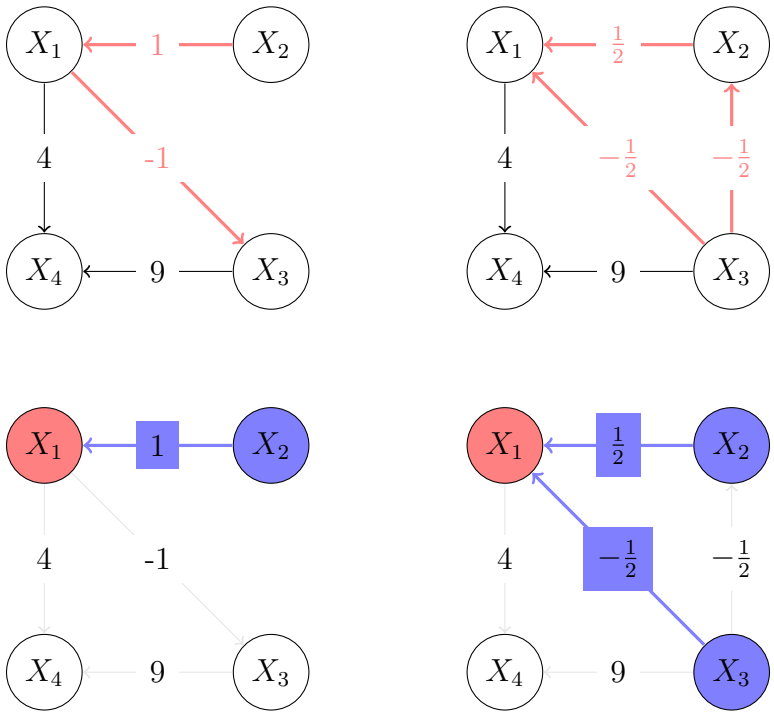



Figure 3.1: (Top) Two equivalent DAGs with different edge weights, as indicated by the edges highlighted in red. (Bottom) An illustration of neighbourhood regression for the target node $X_1$. Nodes in a neighbourhood of $X_1$ are highlighted in blue.

$\varepsilon_j(S)$ the corresponding error (or noise). This is illustrated in Figure 2.1.3.

*Definition* 2.4. For any $S \subset [p]_j$, define a collection of subsets by

$$\mathcal{T}_j(S) := \{T \subset [p]_j : \ \beta_j(T) = \beta_j(S)\} = \{T \subset [p]_j : \ m_j(T) = m_j(S)\}.$$

If $T \in \mathcal{T}_j(S)$, we call $T$ an *invariant set of* $S$ *for* $j$, or $S$*-invariant* for short.

In other words, for any $j$, $\mathcal{T}_j(S)$ is the collection of candidate sets $T \subset [p]_j$ such that the projection of $X_j$ onto $T$ is invariant and equal to $m_j(S)$.

Recall the definition of $S_j(\pi)$ in (3.12). We can now make explicit the connec-

tion between $\beta_j(S)$, $\widetilde{\beta}_j(\pi)$, and related quantities:

$$\widetilde{\beta}_j(\pi) = \beta_j(S_j(\pi)), \quad \operatorname{supp}(\widetilde{\beta}_j(\pi)) = m_j(S_j(\pi)),$$
$$\varepsilon_j(\pi) = \varepsilon_j(S_j(\pi)), \quad \widetilde{\omega}_j^2(\pi) = \operatorname{var}(\varepsilon_j(\pi)).$$

It is straightforward to verify each of these identities from the definition of $\widetilde{\beta}_j(\pi)$ as the $j$th column of $\widetilde{B}(\pi)$.

The following lemma illustrates a crucial property of invariant sets:

**Lemma 2.1.** $T_1, T_2 \in \mathcal{T}_j(S) \implies T_1 \cup T_2 \in \mathcal{T}_j(S)$.

In other words, if two neighbourhood share the same support, the union of the these neighbourhoods must also have the same support. This motivates the following definition:

*Definition* 2.5. The unique largest element of $\mathcal{T}_j(S)$ shall be denoted by $M_j(S)$. Formally,

$$M_j(S) := \bigcup \mathcal{T}_j(S) = \bigcup \{T \subset [p]_j : \ \beta_j(T) = \beta_j(S)\}.$$

The following lemma, which is used repeatedly in the sequel without further mention, justifies the name "$S$-invariant set":

**Lemma 2.2.** *For any $S \subset [p]_j$, we have the following identities for any $T \in \mathcal{T}_j(S)$:*

$$\beta_j(m_j(S)) = \beta_j(S) = \beta_j(T) = \beta_j(M_j(S)) \tag{3.13}$$
$$m_j(m_j(S)) = m_j(S) = m_j(T) = m_j(M_j(S)) \tag{3.14}$$
$$M_j(m_j(S)) = M_j(S) = M_j(T) = M_j(M_j(S)) \tag{3.15}$$
$$\varepsilon_j(m_j(S)) = \varepsilon_j(S) = \varepsilon_j(T) = \varepsilon_j(M_j(S)). \tag{3.16}$$

Finally, define

$$m_j(\Sigma) := \{m_j(S) : \ S \subset [p]_j\}, \tag{3.17}$$
$$M_j(\Sigma) := \{M_j(S) : \ S \subset [p]_j\}. \tag{3.18}$$

By Definition 2.8, $|m_j(S)| \leq d$ for all $j$ and $S$, which implies that $|m_j(\Sigma)| \leq p\binom{p}{d} = p^{O(d)}$. Since there are $2^{p-1}$ subsets of $[p]_j$, the idea is that in general the cardinalities of $m_j(\Sigma)$ and $M_j(\Sigma)$ are much smaller than the cardinality of $2^{[p]_j}$.

### 2.1.4 Model selection exponents

Now that we have defined the relevant population-level quantities, we turn to the problem of estimating these quantities via penalized least squares. We start with some fairly abstract notions in order to allow a general set-up.

*Definition* 2.6. Suppose we are given $y \in \mathbb{R}^n$ and $Z \in \mathbb{R}^{n \times m}$. Let $S \subset [m]$ and consider the set,

$$\widehat{\Theta}_\lambda(y, Z; S) := \underset{\theta \in \mathbb{R}^m,\, \mathrm{supp}(\theta) \subset S}{\arg\min} \frac{1}{2n}\|y - Z\theta\|_2^2 + \rho_\lambda(\theta) \tag{3.19}$$

i.e., the set of global minimizers of the regularized, support-restricted, least-squares problem above. Let $\widehat{\Theta}_\lambda(y, Z) := \widehat{\Theta}_\lambda(y, Z; [m])$ correspond to the case where there is no support restriction.

Note that for the purposes of this abstract definition $y$ is considered a fixed quantity and may bear no relation to the matrix $Z$. Of course, in practice we are interested the case where $y$ and $Z$ are linked via a linear model:

*Example* 2.1. Consider the linear regression problem $\mathbf{y} = Z\theta^* + \mathbf{w}$, where $\mathbf{y} \in \mathbb{R}^n$, $Z \in \mathbb{R}^{n \times m}$, $\theta^* \in \mathbb{R}^m$ and $\mathbf{w} \sim \mathcal{N}_n(0, \sigma^2 I_n)$. Then $\widehat{\Theta}_\lambda(\mathbf{y}, Z)$ is the collection of penalized least squares estimators for $\theta^*$ in the classical linear regression set-up.

The support-restricted version $\widehat{\Theta}_\lambda(\mathbf{y}, Z; S)$ allows us to properly define an *abstract neighbourhood regression problem* for some node $X_j$: A subset $S \subset [p]_j$ defines a neighbourhood of "candidate" regressors for $X_j$, and the neighbourhood regression problem for this node is given by $\widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$. Intuitively, it is the problem of estimating the usual regression problem given the data $(y, Z) = (\mathbf{x}_j, \mathbf{X}_S)$.

We will associate to each abstract neighbourhood regression problem an "exponent", which is a fundamental quantity that measures how difficult the problem is. Given some $n \times m$ matrix $Z$ and signal vector $\theta^*$, define a set of "bad" noise vectors as follows:

$$A(Z, \theta^*; S) := \Big\{ w \in \mathbb{R}^n : \mathrm{supp}(\widehat{\theta}) \neq \mathrm{supp}(\theta^*) \; \exists \, \widehat{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z; S) \Big\}.$$

For a random vector $\mathbf{w} \in \mathbb{R}^n$ (e.g. $\mathbf{w} \sim \mathcal{N}_n(0, \sigma^2 I_n)$), we then have the following event:

$$\mathcal{A}(\mathbf{w}, Z, \theta^*; S) := \Big\{ \mathbf{w} \in A(Z, \theta^*; S) \Big\}. \tag{3.20}$$

As usual we use the shorthand $\mathcal{A}(\mathbf{w}, Z, \theta^*) = \mathcal{A}(\mathbf{w}, Z, \theta^*; [m])$.

*Definition* 2.7. Define the (regression) *model selection exponent* for a vector $\theta^*$ as

$$\Phi_\lambda(Z, \theta^*, \sigma^2) := -\log \mathbb{P}\left[\mathcal{A}(\mathbf{w}, Z, \theta^*)\right], \quad \mathbf{w} \sim \mathcal{N}_n(0, \sigma^2 I_n).$$

A larger exponent corresponds to a better model selection performance.

These model selection exponents will allow us to write out explicit, non-asymptotic upper bounds on the model selection failure of $\widehat{B}$ in terms of the model selection failure of each neighbourhood regression, which we will pursue in the next subsection.

## 2.2 Main results

Our main results will be divided into two substantive components: (i) Bounds on the probability of false selection, and (ii) Bounds on the estimation error $\|\widehat{B} - \widetilde{B}(\widehat{\pi})\|_r$. The latter result can be applied in a very general setting under fairly weak assumptions, whereas the former result for model selection requires a more abstract set-up with (possibly) stronger assumptions. We will ultimately show that all of our conclusions can hold simultaneously in practical settings of interest.

### 2.2.1 Assumptions

Recall the definition of the equivalence class $\mathfrak{D}(\Sigma)$ in Definition 2.4. Our first assumption simply ensures that $\mathfrak{D}(\Sigma)$ is well-defined.

*Condition* 2.2 (Nondegeneracy). $\Sigma$ is positive definite, i.e. $r_{\min}(\Sigma) > 0$, and

$$\operatorname{var}(X_j) \leq \sigma^2_{\max} < \infty, \quad j = 1, \ldots, p.$$

Naturally, our results will depend on two quantities that are familiar from the regression literature: The sparsity level and the signal strength. The difference now is that instead of quantifying these for a single, "true" parameter, we need to consider the entire equivalence class $\mathfrak{D}(\Sigma)$.

The $j$th column of $\widetilde{B}(\pi)$ will be denoted by $\widetilde{\beta}_j(\pi)$. For any $A = (a_{ij}) \in \mathbb{R}^{p\times p}$, let $\tau_*(A) := \min\{|a_{ij}| : a_{ij} \neq 0\}$.

*Definition* 2.8. For any $\Sigma$, let

$$d = d(\Sigma) := \sup_{1\leq j\leq p} \sup_{\pi\in\mathcal{P}} \|\widetilde{\beta}_j(\pi)\|_0, \tag{3.21}$$

$$\tau_* = \tau_*(\Sigma) := \inf_{\pi\in\mathcal{P}} \tau_*(\widetilde{B}(\pi)). \tag{3.22}$$

In other words, $d$ is the size of the largest parent set within any DAG in $\mathfrak{D}(\Sigma)$, and $\tau_*(\Sigma) \in (0, \infty)$ is the smallest nonzero coefficient in absolute value of any DAG in $\mathfrak{D}(\Sigma)$. The parameter $d$ measures the relative sparsity of $\Sigma$ in terms of its equivalence class, and $\tau_*$ measures the minimum signal strength in terms of its SEM coefficients.

### 2.2.2 Support recovery

Before stating our first main result, we need to assume something about the behaviour of the exponents $\Phi_\lambda$:

*Condition* 2.3. For any $Z$, $\theta^*$, and $\lambda \geq 0$, if $\sigma^2 \leq \sigma_0^2$ then $\Phi_\lambda(Z, \theta^*, \sigma^2) \geq \Phi_\lambda(Z, \theta^*, \sigma_0^2)$.

This condition simply says that increasing the variance never makes the problem easier, which is a mild requirement. Finally, in order to simplify our upper bounds, define

$$\Psi_\lambda(\mathbf{X}, \sigma^2_{\max}) := \inf_{\substack{\|\theta\|_0 \le d(\Sigma) \\ \tau_*(\theta) \ge \tau_*(\Sigma)}} \Phi_\lambda(\mathbf{X},\, \theta,\, \sigma^2_{\max}). \tag{3.23}$$

We have the following general result:

*Theorem* 2.1. Assume Conditions 2.2 and 2.3. Then

$$\mathbb{P}\left( \operatorname{supp}(\widehat{B}(\pi)) \neq \operatorname{supp}(\widetilde{B}(\pi)),\ \exists\, \pi \in \mathcal{P} \right) \ \le\ p\binom{p}{d} \mathbb{E} e^{-\Psi_\lambda(\mathbf{X}, \sigma^2_{\max})}.$$

*Remark* 2.1. Since Theorem 2.1 provides uniform control of the probability of false selection for all permutations $\pi \in \mathcal{P}$, we obtain model selection consistency for the global minimizer $\widehat{B}$ as an immediate corollary by applying this result to $\widetilde{B}(\widehat{\pi}) = \widehat{B}$.

*Remark* 2.2. Since

$$p\binom{p}{d} \exp(-\Psi_\lambda(\mathbf{X}, \sigma^2_{\max})) \le \exp(2d \log p - \Psi_\lambda(\mathbf{X}, \sigma^2_{\max})),$$

it follows that $\psi_\lambda(\mathbf{X}, \Sigma) \gtrsim d \log p$ is a sufficient condition for the probability in Theorem 2.1 to vanish asymptotically.

### 2.2.3 Deviation bounds

In analogy with (3.23), define

$$\psi_\lambda(\mathbf{X}, \sigma^2_{\max}) = \psi_\lambda(\mathbf{X}, \sigma^2_{\max}; \delta) := \Phi_\lambda(\mathbf{X},\, 0,\, \sigma^2_{\max}/\delta^2) \tag{3.24}$$

This quantity is related to a so-called *Gaussian width condition* (see Definition 2.10).

*Theorem* 2.2. Assume Conditions 2.1, 2.2, and 2.3. Suppose $\mathbf{X} \overset{\text{iid}}{\sim} \mathcal{N}_p(0, \Sigma)$ and choose $\delta \in (0, 1)$. Define $\xi(\delta) = (1 + \delta)/(1 - \delta)$, and suppose

$$n > c'' \frac{\sigma^2_{\max}(1 + \xi)^2}{r_{\min}(\Sigma)} d \log p,$$

Then with probability $1 - c' \exp(-cn) - p\binom{p}{d}\mathbb{E}\exp(-\psi_\lambda(\mathbf{X}, \sigma^2_{\max}; \delta))$,

$$\sup_{j,S} \frac{\|\widehat{\beta}_j(S) - \beta_j(S)\|_2}{\|\beta_j(S)\|_0^{1/2}} \leq \frac{2\,\xi}{r_{\min}(\Sigma)} \rho'_\lambda(0+), \tag{3.25}$$

$$\sup_{j,S} \frac{\|\widehat{\beta}_j(S) - \beta_j(S)\|_1}{\|\beta_j(S)\|_0} \leq \frac{2\,\xi(1+\xi)}{r_{\min}(\Sigma)} \rho'_\lambda(0+). \tag{3.26}$$

As a consequence, with the same probability,

$$\|\widehat{B} - \widetilde{B}(\widehat{\pi})\|_2 \leq \frac{\xi}{r_{\min}(\Sigma)} \rho'_\lambda(0+) \|\widetilde{B}(\widehat{\pi})\|_0^{1/2}, \tag{3.27}$$

$$\|\widehat{B} - \widetilde{B}(\widehat{\pi})\|_1 \leq \frac{\xi(1+\xi)}{r_{\min}(\Sigma)} \rho'_\lambda(0+) \|\widetilde{B}(\widehat{\pi})\|_0. \tag{3.28}$$

It is worth emphasizing that Theorem 2.2 holds whenever $\Sigma$ is positive definite, and that no additional assumptions on the population-level parameters are needed. The overview of the proof technique for this theorem is found in the next section, with all the technical proofs are delayed until Section 2.4.

## 2.3 Overview of proofs

The proof of Theorems 2.1 and 2.2 will be broken down into three main steps. First, we establish some basic properties of the objective function and the probability space in order to provide a uniform bound on the probability of false selection for any neighbourhood problem. Then, we will derive an independent result that gives a deviation bound for a *fixed* design regression problem (Theorem 2.3). Using similar arguments from the first step, we will then show that this result can be applied uniformly for all neighbourhood problems, which yields our claimed deviation bound.

### 2.3.1 Proof of Theorem 2.1

We begin by controlling the probability of the event

$$\mathcal{B} := \{\text{supp}(\widehat{B}(\pi)) \neq \text{supp}(\widetilde{B}(\pi)) \,\exists\, \pi \in \mathcal{P}\}. \tag{3.29}$$

We will do this by reducing the analysis of $\widehat{B}(\pi)$ to a series of abstract neighbourhood regression problems. There are two key steps: (i) Showing that each estimator $\widehat{B}(\pi)$ is equivalent to solving a series of $p$ regression problems given by $\widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$, and (ii) Controlling the total number of sets $S$ that need to be considered.

The first step is justified by the following result:

**Lemma 2.3.** *Suppose* $\mathbf{X} \overset{iid}{\sim} \mathcal{N}_p(0, \Sigma)$ *and* $\lambda \geq 0$. *Then the following statements are true:*

(a) *For any* $j$ *and* $\pi \in \mathcal{P}$, $\boldsymbol{\varepsilon}_j(\pi) \perp\!\!\!\perp \mathbf{X}_{S_j(\pi)}$,

(b) *A matrix* $\widehat{B}(\pi) \in \mathbb{D}$ *is a* $\pi$*-global minimizer if and only if* $\widehat{\beta}_j(\pi) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S_j(\pi))$ *for each* $j = 1, \ldots, p$,

(c) $\widehat{B} = \widehat{B}(\widehat{\pi})$ *is a global minimizer of* $Q(B)$ *if and only if* $\widehat{\beta}_j(\widehat{\pi}) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S_j(\widehat{\pi}))$ *for each* $j = 1, \ldots, p$ *and* $\widehat{\pi} \in \widehat{\mathcal{P}}$.

This lemma allows us to formally establish the equivalence between the DAG problem for $\widehat{B}(\pi)$ and neighbourhood regression: In order to construct $\widehat{B}(\pi)$, it suffices to solve a neighbourhood regression problem for each column of $\widehat{B}(\pi)$, which is defined formally by $\widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S_j(\pi))$.

As a consequence of Lemma 2.3, we have

$$\mathcal{B} = \bigcup_{j=1}^{p} \{\text{supp}(\widehat{\beta}_j(\pi)) \neq \text{supp}(\widetilde{\beta}_j(\pi)) \, \exists \, \pi \in \mathcal{P}\}.$$

Since there are $p!$ total permutations, in principle the event $\{\text{supp}(\widehat{\beta}_j(\pi)) \neq \text{supp}(\widetilde{\beta}_j(\pi)) \, \exists \, \pi \in \mathcal{P}\}$ involves controlling a superexponential number of estimators, which seems hopeless. The key will be to reduce this by invoking the sets $M_j(T)$ introduced in Definition 2.5.

The details of this reduction rely crucially on the properties of what we call a *monotonic class*. While the details of this abstraction are left to Section 2.4.1,

we will highlight some of the key concepts here. To motivate this, we introduce the following result, which says that model selection properties of the $S$-restricted estimators are monotone with respect to those sets $S$ that contain the true support.

**Lemma 2.4.** *Suppose $Z \in \mathbb{R}^{n \times m}$ is a fixed matrix and consider the regression problem $\mathbf{y} = Z\theta^* + \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}_n(0, \sigma^2 I_n)$. If $\operatorname{supp}(\theta^*) \subset S \subset T$, then we have the following inclusion:*

$$A(Z, \theta^*; S) \subset A(Z, \theta^*; T). \tag{3.30}$$

*In particular, we have*

$$\mathcal{A}(\mathbf{w}, Z, \theta^*; S) \subset \mathcal{A}(\mathbf{w}, Z, \theta^*; T). \tag{3.31}$$

Intuitively, for a fixed support, the set of "bad" noise vectors for the larger problem involving $T$ is at least as big as the set of "bad" noise vectors for the smaller problem involving $S$.

Define the following "neighbourhood" estimators:

*Definition* 2.9. For any $S \subset [p]_j$, let

$$\widehat{\beta}_j(S) \in \underset{\beta \in \mathbb{R}^p,\ \operatorname{supp}(\beta) \subset S}{\arg\min} \frac{1}{2n} \|\mathbf{x}_j - \mathbf{X}\beta\|_2^2 + \rho_\lambda(\beta)$$

or equivalently $\widehat{\beta}_j(S) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$ as defined in (3.19).

We are interested in the model selection failure of $\widehat{\beta}_j(S)$ for $\beta_j(S)$, which can be stated as

$$\Big\{ \operatorname{supp}(\widehat{\beta}_j(S)) \neq \operatorname{supp}(\beta_j(S)) \quad \text{for some } \widehat{\beta}_j(S) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S) \Big\} = \mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}, \beta_j(S); S)$$

in the notation introduced in (3.20). The next result, which is an immediate corollary to Lemma 2.4, encapsulates the notion of monotonicity that is used throughout the rest of the proof:

**Corollary 2.5.** *Suppose* $\mathbf{X} \overset{iid}{\sim} \mathcal{N}_p(0, \Sigma)$. *For any* $S \subset [p]_j$, *we have*

$$\mathcal{A}\Big(\boldsymbol{\varepsilon}_j(S), \mathbf{X}, \beta_j(S); S\Big) \subset \mathcal{A}\Big(\boldsymbol{\varepsilon}_j(M_j(S)), \mathbf{X}, \beta_j(M_j(S)); M_j(S)\Big). \tag{3.32}$$

In other words, to control the neighbourhood problem for some set $S$, it suffices to control the strictly harder problem given by $M_j(S)$.

For any neighbourhood $S \subset [p]_j$, define a neighbhourhood exponent by

$$\Xi_j(S) := \Phi_\lambda(\mathbf{X}_S,\, [\beta_j(S)]_S,\, \omega_j^2(S)). \tag{3.33}$$

Note that we must restrict the SEM coefficients $\beta_j(S)$ to the subset $S$ in order for this exponent to be well-defined. Since $\mathrm{supp}(\beta_j(S)) \subset S$, this does not change anything. We have the following general result:

**Proposition 2.6.** *Under Conditions 2.2 and 2.3, we have*

$$\mathbb{P}\left(\mathrm{supp}(\widehat{\beta}_j(S)) \neq \mathrm{supp}(\beta_j(S)),\ \exists\, j \in [p],\ S \subset [p]_j\right) \ \leq \sum_{T \in m_j(\Sigma)} \mathbb{E}e^{-\Xi_j(M_j(T))},$$

*where* $m_j(\Sigma)$ *is defined by* (3.17) *and* $\Xi_j(\,\cdot\,)$ *is defined by* (3.33).

*Remark* 2.3. Proposition 2.6 says that in order to control the probability of false selection uniformly for all nodes $j$ and all neighbourhoods $S$, it suffices to control a much smaller class of problems given by the neighbhourhoods $M_j(T)$ for each *support set* $T \in m_j(\Sigma)$. For sparse DAGs, the number of "true" supports is much smaller than the total number of neighbourhoods, and this is the heart of our argument.

We can now prove Theorem 2.1:

*Proof of Theorem 2.1.* By Lemma 2.4 and Condition 2.3,

$$\begin{aligned}\Xi_j(M_j(T)) &\geq \Phi_\lambda(\mathbf{X},\, \beta_j(T),\, \omega_j^2(T)) \\ &\geq \Phi_\lambda(\mathbf{X},\, \beta_j(T),\, \sigma_{\max}^2).\end{aligned}$$

Recalling the definitions of $d(\Sigma)$ and $\tau_*(\Sigma)$ (Definition 2.8), we have $\|\beta_j(S)\|_0 \leq d(\Sigma)$ and $\tau_*(\beta_j(S)) \geq \tau_*(\Sigma)$. Thus the previous expression combined with (3.23) implies:

$$\Xi_j(M_j(T)) \geq \Psi_\lambda(\mathbf{X}, \sigma^2_{\max}) \tag{3.34}$$

Combining Proposition 2.6 and (3.34),

$$\begin{aligned}
&\mathbb{P}\left(\operatorname{supp}(\widehat{\beta}_j(S)) \neq \operatorname{supp}(\beta_j(S)),\ \exists\, j \in [p],\ S \subset [p]_j\right) \\
&\qquad \leq \sum_{j=1}^{p} \sum_{T \in m_j(\Sigma)} \mathbb{E} \exp(-\Xi_j(M_j(T)) \\
&\qquad \leq \sum_{j=1}^{p} \sum_{T \in m_j(\Sigma)} \mathbb{E} \exp(-\Psi_\lambda(\mathbf{X}, \sigma^2_{\max})).
\end{aligned} \tag{3.35}$$

Since there are at most $\binom{p}{d}$ subsets in $m_j(\Sigma)$ (Definition 2.3),

$$\sum_{j=1}^{p} \sum_{T \in m_j(\Sigma)} \mathbb{E} \exp(-\Psi_\lambda(\mathbf{X}, \sigma^2_{\max})) \leq p \binom{p}{d} \mathbb{E} \exp(-\Psi_\lambda(\mathbf{X}, \sigma^2_{\max})).$$

In order to complete the proof, note that Lemma 2.3 implies

$$\begin{aligned}
&\left\{ \operatorname{supp}(\widehat{B}(\pi)) \neq \operatorname{supp}(\widetilde{B}(\pi)),\ \exists\, \pi \in \mathcal{P} \right\} \\
&\qquad = \left\{ \operatorname{supp}(\widehat{\beta}_j(\pi)) \neq \operatorname{supp}(\widetilde{\beta}_j(\pi)),\ \exists\, \pi \in \mathcal{P},\ j \in [p] \right\} \\
&\qquad = \left\{ \operatorname{supp}(\widehat{\beta}_j(S)) \neq \operatorname{supp}(\beta_j(S)),\ \exists\, S \subset [p]_j,\ j \in [p] \right\}.
\end{aligned}$$

In the last line we used $\widetilde{\beta}_j(\pi) = \beta_j(S_j(\pi))$. Combined with (3.35), this gives the desired result. □

### 2.3.2 Proof of Theorem 2.2

In order to establish uniform control over the probability of false selection for all possible neighbourhood regression problems in the previous section, we relied on a monotonicity property (cf. Lemma 2.4) of model selection. Unfortunately, this

may not hold for weaker types of consistency. In order to bound the deviations $\|\widehat{B}(\pi) - \widetilde{B}(\pi)\|_r$ $(r = 1, 2)$, we will use a modified argument that invokes a different kind of monotone class.

We start by establishing a general bound on the $\ell_r$ $(r = 1, 2)$ estimation errors for a fixed design regression problem with a general regularizer $\rho_\lambda$. The objective here is to derive general conditions under which we can guarantee such bounds hold for a fixed design problem, and then show that conditions hold uniformly for all neighbourhood problems. The conditions we will need are familiar from the literature: A *Gaussian width condition* and a *restricted eigenvalue condition*.

For the rest of this subsection, we let $Z \in \mathbb{R}^{n \times m}$ and $w \in \mathbb{R}^n$ be a fixed matrix and fixed vector, respectively.

*Definition* 2.10 (Gaussian width condition). We say that the *Gaussian width condition* holds for $(w, Z)$ at $\lambda$ if there is a numerical constant $\delta \in (0, 1)$ such that

$$\frac{1}{n}|\langle w, Zu\rangle| \leq \delta\left[\frac{1}{2n}\|Zu\|^2 + \rho_\lambda(u)\right], \ \forall u \in \mathbb{R}^m,$$

in which case we write $(w, Z) \in \mathrm{GW}_{\rho_\lambda}(\delta)$.

For any set $A \subset [m]$ and $\xi > 0$, define the following "cone":

$$C_{\rho_\lambda}(A, \xi) := \{u \in \mathbb{R}^m : \rho_\lambda(u_{A^c}) \leq \xi \rho_\lambda(u_A)\}.$$

Note that this definition depends on the ambient dimension $m$; when we wish to emphasize this dependence we will write $C^m_{\rho_\lambda}(A, \xi)$.

*Definition* 2.11 (Restricted eigenvalues). The *generalized restricted eigenvalue (RE) constant* of $Z$ with respect to $\rho_\lambda$ over the subset $A$ is defined as

$$\phi^2_{\rho_\lambda}(Z, A) := \inf\left\{\frac{\|Zu\|_2^2}{n\|u\|_2^2} : u \in C_{\rho_\lambda}(A, \xi),\ u \neq 0\right\}. \tag{3.36}$$

In the sequel, we will suppress the dependence of various quantities on $\lambda$ and $\xi$, writing $\rho = \rho_\lambda$, $\mathrm{GW}_\rho(\delta) = \mathrm{GW}_{\rho_\lambda}(\delta)$, $C_\rho(A) = C^m_{\rho_\lambda}(A, \xi)$, and $\phi^2_\rho(Z, A) =$

$\phi^2_{\rho_\lambda}(Z, A)$ when no confusion should arise. The special case of $\rho_\lambda(\cdot) = \lambda\|\cdot\|_1$ will be denoted $C_1(A) := C_{\lambda\|\cdot\|_1}(A)$ with corresponding restricted eigenvalue $\phi^2_1(Z, A) = \phi^2_{\lambda\|\cdot\|_1}(Z, A)$. The usual restricted eigenvalue is then equivalent to the special case $\phi^2_1(Z, A)$ (Bickel et al. (2009)).

Consider the abstract linear regression set up given by

$$y = Z\theta^* + w,$$

where $\theta^* \in \mathbb{R}^m$ and we define $S^* = \text{supp}(\theta^*)$. The following general result establishes that the two conditions $(w, Z) \in \text{GW}_\rho(\delta)$ and $\phi^2_\rho(Z, S^*) > 0$ are sufficient to bound the deviation $\widehat{\theta} - \theta^*$:

*Theorem* 2.3. Assume that $\rho_\lambda$ satisfies Condition 2.1 and $(w, Z) \in \text{GW}_\rho(\delta)$ for some $\delta \in (0, 1)$. Let $\xi = \xi(\delta) = (1 + \delta)/(1 - \delta)$ and $\phi^2 := \phi^2_\rho(Z, S^*)$. Then any $\widehat{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z)$ satisfies

$$\|\widehat{\theta} - \theta^*\|_2 \leq \frac{2\,\xi}{\phi^2}\,\rho'_\lambda(0+)\|\theta^*\|_0^{1/2}, \tag{3.37}$$

$$\|\widehat{\theta} - \theta^*\|_1 \leq \frac{2\,\xi(1+\xi)}{\phi^2}\,\rho'_\lambda(0+)\|\theta^*\|_0. \tag{3.38}$$

The proof of Theorem 2.3 is left to the Appendix.

In order to extend this result to the neighbourhood problems for $\Sigma$, we first provide a uniform bound on the restricted eigenvalues $\phi^2_{\rho_\lambda}(\mathbf{X}_S, m_j(S))$ in terms of the smallest eigenvalue of $\Sigma$:

**Proposition 2.7.** *Assume $\Sigma \in \mathbb{R}^{p\times p}$ is positive definite and satisfies Condition 2.2. For any $\xi > 0$, there exist universal constants $c, c', c''$, such that if*

$$n > c''\frac{\sigma^2_{\max}(1+\xi)^2}{r_{\min}(\Sigma)}d(\Sigma)\log p$$

*then with probability at least $1 - c'\exp(-cn)$, any $\mathbf{X} \overset{iid}{\sim} \mathcal{N}_p(0, \Sigma)$ satisfies*

$$\inf_{1\leq j\leq p}\inf_{S\subset[p]_j}\inf_{\substack{A\subset S\\ |A|\leq d}}\phi^2_\rho(\mathbf{X}_S, A) \geq r_{\min}(\Sigma) > 0.$$

This gives us complete control over the restricted eigenvalues for each neighbourhood in terms of the universal quantity $r_{\min}(\Sigma)$.

The next step is to show that with high probability, $(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S) \in \mathrm{GW}_\rho(\delta)$ for all $j$ and $S$ and some universal constant $\delta \in (0,1)$. First, let us define a collection of events $\mathcal{E}_S(\delta,\lambda) = \mathcal{E}_S(\delta,\lambda;j)$ by

$$\mathcal{E}_S(\delta,\lambda)^c := \left\{ \frac{1}{n}|\langle \boldsymbol{\varepsilon}_j(S), \mathbf{X}_S u\rangle| < \delta\left[\frac{1}{2n}\|\mathbf{X}_S u\|^2 + \rho_\lambda(u)\right] \ \forall u \neq 0 \right\}. \tag{3.39}$$

This simply says that $(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S)$ satisfies the Gaussian width condition with inequality replaced by a strict inequality. The next result shows how this event is really a model selection problem in disguise:

**Lemma 2.8.** *For any $j$ and $S \subset [p]_j$, we have*

$$\mathcal{E}_S(\delta,\lambda) = \mathcal{A}\Big(\boldsymbol{\varepsilon}_j(S)/\delta, \mathbf{X}, 0; S\Big),$$

*where $\mathcal{A}(\,\cdot\,,\,\cdot\,,\,\cdot\,;S)$ is defined by* (3.20).

That is, the Gaussian width condition is equivalent to requiring that the neighbourhood regression problem for $j$ and $S$ is model selection consistent when the *true coefficients are all set to zero* and the noise variance is proportionally inflated by a factor of $1/\delta^2$. Zhang and Zhang (2012) have referred to this property as "null-consistency".

An immediate consequence is that the events $\mathcal{E}_S(\delta,\lambda)$ are monotonic in the sense that they obey Lemma 3.32 when $\beta_j(S)$ is replaced with the zero vector. In analogy with (3.23), define

$$\psi_\lambda(\mathbf{X}, \sigma^2_{\max}) = \psi_\lambda(\mathbf{X}, \sigma^2_{\max}; \delta) := \Phi_\lambda(\mathbf{X},\, 0,\, \sigma^2_{\max}/\delta^2) \tag{3.40}$$

A similar monotonicity argument as used in the proof of Theorem 2.1 then yields the following:

**Proposition 2.9.** *Define an event by*

$$\mathcal{E}(\delta, \lambda) = \bigcap_{j=1}^{p} \bigcap_{S \in m_j(\Sigma)} \mathcal{E}_S(\delta, \lambda; j).$$

*Then for any* $\delta \in (0,1)$ *and* $\lambda \geq 0$,

$$\Pr(\mathcal{E}(\delta, \lambda)) \geq 1 - p\binom{p}{d}\mathbb{E}e^{-\psi_\lambda(\mathbf{X}, \sigma^2_{\max}; \delta)}.$$

Together, Propositions 2.7 and 2.9 show that we have uniform control over both the restricted eigenvalues and the Gaussian widths for the neighbourhood problems $\widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$. Thus, with high probability, we can apply Theorem 2.3 to each of these problems in order to obtain uniform bounds on the deviations $\beta_j(S) - \widehat{\beta}_j(S)$ across all $j \in [p]$ and $S \subset [p]_j$. We can now prove our second main result:

*Proof of Theorem 2.2.* Define two events by

$$\begin{aligned}
\mathcal{E}(\delta, \lambda) &= \left\{ (\boldsymbol{\varepsilon}_j(\pi), \mathbf{X}_{S_j(\pi)}) \in \mathrm{GW}_{\rho_\lambda}(\delta),\ j \in [p],\ \pi \in \mathcal{P} \right\} \\
&= \bigcap_{j=1}^{p} \bigcap_{S \in m_j(\Sigma)} \mathcal{E}_S(\delta, \lambda; j), \qquad (3.41) \\
\mathcal{R}(\delta) &= \left\{ \phi^2_\rho(\mathbf{X}_S, m_j(S)) \geq r_{\min}(\Sigma) > 0,\ \forall j \in [p], S \subset [p]_j \right\}. \qquad (3.42)
\end{aligned}$$

Propositions 2.7 and 2.9 guarantee that

$$\mathbb{P}\left(\mathcal{E}(\delta, \lambda) \cap \mathcal{R}(\delta)\right) \geq 1 - c' \exp(-cn) - p\binom{p}{d}\mathbb{E}\exp(-\psi_\lambda(\mathbf{X}, \sigma^2_{\max}; \delta)).$$

Thus, it suffices to deduce (3.25) and (3.26) whenever we are on the event $\mathcal{E}(\delta, \lambda) \cap \mathcal{R}(\delta)$. But this follows from Lemma 2.10 below. □

**Lemma 2.10.** *Fix* $j$ *and* $S \subset [p]_j$. *Assume that* $\rho_\lambda$ *satisfies Condition 2.1, and* $(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S) \in \mathrm{GW}_{\rho_\lambda}(\delta)$. *Let* $\xi = \xi(\delta) = (1+\delta)/(1-\delta)$ *and*

$$\phi^2_j(S) = \phi^2_{\rho_\lambda}(\mathbf{X}_S, m_j(S)).$$

*Then any* $\widehat{\beta}_j(S) \in \widehat{\Theta}_\lambda(\mathbf{X}\beta_j(S) + \boldsymbol{\varepsilon}_j(\pi), \mathbf{X})$ *satisfies*

$$\|\widehat{\beta}_j(S) - \beta_j(S)\|_2 \leq \frac{2\,\xi}{\phi_j^2(S)} \cdot \rho'_\lambda(0+)\|\beta_j(S)\|_0^{1/2}, \tag{3.43}$$

$$\|\widehat{\beta}_j(S) - \beta_j(S)\|_1 \leq \frac{2\,\xi(1+\xi)}{\phi_j^2(S)} \cdot \rho'_\lambda(0+)\|\beta_j(S)\|_0. \tag{3.44}$$

*Proof.* Apply Theorem 2.3 to $\widehat{\theta} \in \widehat{\Theta}_\lambda(\mathbf{X}_S[\beta_j(S)]_S + \boldsymbol{\varepsilon}_j(\pi), \mathbf{X}_S)$ and note that $\|\widehat{\beta}_j(S) - \beta_j(S)\|_r = \|\widehat{\theta} - [\beta_j(S)]_S\|_r$ for $r = 1, 2$. □

## 2.4 Technical proofs

In this section we state and prove all of the technical results needed for the proofs in the previous subsection.

### 2.4.1 General monotonicity and consequences.

It turns out that Corollary 2.5 is precisely the property we need in order to exploit the sparsity of $\mathfrak{D}$. This property is so important that we first prove a general result regarding classes of events with this property.

For the remainder of this section, we consider $j \in [p]$ to be fixed. Suppose that

$$\mathcal{F} = \left\{\mathcal{F}_S : S \subset [p]_j\right\}$$

is a class of events for each subset of $[p]_j$. Intuitively, $\mathcal{F}_S$ represents some event that measures the quality (or lack thereof) of the neighbhourhood regression estimators $\widehat{\beta}_j(S) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$. We have the following definition:

*Definition* 2.12. The class $\mathcal{F}$ is called *monotonic over invariant sets*, or simply *monotonic* for short, if the following inclusion holds for any $S \subset [p]_j$:

$$\mathcal{F}_S \subset \mathcal{F}_{M_j(S)}.$$

The name "monotonic over invariant sets" follows from the inclusion

$$\mathcal{F}_T \subset \mathcal{F}_{M_j(S)} \text{ for any } T \in \mathcal{T}_j(S), \tag{3.45}$$

which is an immediate consequence of Definition 2.12.

**Lemma 2.11.** *Suppose that the class* $\mathcal{F} = \{\mathcal{F}(S) : S \subset [p]_j\}$ *is monotonic over invariant sets. Then we have the following identities:*

$$\bigcup_{T \in \mathcal{T}_j(S)} \mathcal{F}_T = \mathcal{F}_{M_j(S)} \text{ for any } S \subset [p]_j, \tag{3.46}$$

$$\bigcup_{S \subset [p]_j} \mathcal{F}_S = \bigcup_{T \in M_j(\Sigma)} \mathcal{F}_T. \tag{3.47}$$

*Proof.* The first identity follows from (3.45) and the fact that $M_j(S) \in \mathcal{T}_j(S)$ for every $S \subset [p]_j$. The second identity follows from the first by taking the union over all subsets $S \subset [p]_j$. □

When interpreted in terms of neighbourhood regression, this abstract result has some important consequences. Consider the special case where each $\mathcal{F}_S$ corresponds to some property of the neighbourhood regression problem for $S$, given by $\widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$. For example, $\mathcal{F}_S$ could be the event that conditional on $\mathbf{X}_S$, there is some $\widehat{\theta} \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S)$ such that $\text{supp}(\widehat{\theta}) \neq \text{supp}(\beta_j(S))$. This is in fact the event $\mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S, (\beta_j(S))_S)$ defined in (3.20). Equation (3.46) says that in order to *simultaneously* control the probability of false selection for *any* neighbourhood problem in $\mathcal{T}_j(S)$, it suffices to control the probability of false selection for the neighbhourhood $M_j(S)$ alone. In turn, equation (3.47) says that in order to bound the probability that *any* neighbourhood estimator selects the wrong model, it suffices to control problems for $M_j(T)$. Since $|M_j(\Sigma)| = |m_j(\Sigma)| \leq \binom{p}{d}$, this is a substantial reduction from the original $p!$ possible neighbourhoods.

In analogy with Definition 2.7, let us define an abstract class exponent by

$$\nu_\lambda(\mathbf{X}_S) := -\log \mathbb{P}\left[\mathcal{F}_S \,|\, \mathbf{X}_S\right]. \tag{3.48}$$

*Remark* 2.4. Recall the definition of the model selection exponent in Definition 2.7. Since $Z$ is treated as fixed in this definition, we have

$$\Phi_\lambda(Z, \theta^*, \sigma^2) := -\log \mathbb{P}\left[\mathcal{A}(\mathbf{w}, Z, \theta^*) \,|\, Z\right], \quad \mathbf{w} \sim \mathcal{N}(0, \sigma^2 I_n).$$

In particular, if $\mathbf{X} \overset{\text{iid}}{\sim} \mathcal{N}(0, \Sigma)$ and $\boldsymbol{\varepsilon}_j(S) \perp\!\!\!\perp \mathbf{X}_S$, we have

$$\Phi_\lambda(\mathbf{X}_S, (\beta_j(S))_S, \omega_j^2(S)) := -\log \mathbb{P}\left[\mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S, (\beta_j(S))_S) \,|\, \mathbf{X}_S\right],$$

which corresponds to the abstract class exponent $\nu_\lambda(\mathbf{X}_S)$ in (3.48) when $\mathcal{F}_S = \mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S, (\beta_j(S))_S)$.

**Lemma 2.12.** *Let $\mathcal{F}$ be a monotonic class of sets indexed by $S \subset [p]_j$. Then*

$$\mathbb{P}\left[\bigcup_{S \subset [p]_j} \mathcal{F}_S\right] \leq \sum_{T \in M_j(\Sigma)} \mathbb{E}\big[\exp(-\nu_\lambda(\mathbf{X}_T))\big].$$

*Proof.* Using Lemma 2.11 and the union bound,

$$\mathbb{P}\left[\bigcup_{S \subset [p]_j} \mathcal{F}_S\right] = \mathbb{P}\left[\bigcup_{T \in M_j(S)} \mathcal{F}_T\right] \leq \sum_{T \in M_j(\Sigma)} \mathbb{E}\,\mathbb{P}\left[\mathcal{F}_T \,|\, \mathbf{X}_T\right].$$

The result then follows from (3.48). □

Lemma 2.12 is an abstract result that exploits the monotonicity property of $\mathcal{F}$, and will be used repeatedly in the sequel to provide uniform bounds over events of the form $\cup \mathcal{F}_S$.

### 2.4.2 Proof of Lemma 2.3

The first conclusion follows from elementary properties of conditional expectation and the identity

$$\mathbb{E}(X_j \,|\, X_{S_j(\pi)}) = \widetilde{\beta}_j(\pi)^T X.$$

Note that the third conclusion is a special case of the second, which we now prove. Fix $\pi \in \mathcal{P}$ and let $S_j = S_j(\pi)$. If $\widehat{\beta}_j(\pi) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S_j)$ for each $j$, then evidently $\widehat{B}(\pi) = [\,\widehat{\beta}_1(\pi) \,|\, \cdots \,|\, \widehat{\beta}_p(\pi)\,]$ minimizes $Q(B)$ over $\mathbb{D}[\pi]$.

For the reverse direction, note that by the definition of $\widehat{B}(\pi)$, we have

$$\frac{1}{2n}\|\mathbf{X} - \mathbf{X}\widehat{B}(\pi)\|_F^2 + \rho_\lambda(\widehat{B}(\pi)) \leq \frac{1}{2n}\|\mathbf{X} - \mathbf{X}B\|_F^2 + \rho_\lambda(B) \quad \forall\, B \in \mathbb{D}[\pi]. \quad (3.49)$$

Recall that $\mathbf{X}_{S_j}$ is the $n \times |S_j|$ matrix formed by extracting the columns in $S_j$, and similarly for $(\beta_j)_{S_j}$. For any $B \in \mathbb{D}[\pi]$ we have $(\beta_j)_{S_j^c} = 0$ for each $j$, so we can write

$$\begin{aligned}\frac{1}{2n}\|\mathbf{X} - \mathbf{X}B\|_F^2 + \rho_\lambda(B) &= \sum_{j=1}^p \Big\{ \frac{1}{2n}\|\mathbf{x}_j - \mathbf{X}\beta_j\|_2^2 + \rho_\lambda(\beta_j) \Big\} \\ &= \sum_{j=1}^p \Big\{ \frac{1}{2n}\|\mathbf{x}_j - \mathbf{X}_{S_j}(\beta_j)_{S_j}\|_2^2 + \rho_\lambda((\beta_j)_{S_j}) \Big\}.\end{aligned}$$

Using $\widehat{B}(\pi) = [\,\widehat{\beta}_1(\pi)\,|\,\cdots\,|\,\widehat{\beta}_p(\pi)\,] \in \mathbb{D}[\pi]$ and (3.49), we then have

$$\begin{aligned}&\sum_{j=1}^p \Big\{ \frac{1}{2n}\|\mathbf{x}_j - \mathbf{X}_{S_j}(\widehat{\beta}_j(\pi))_{S_j}\|_2^2 + \rho_\lambda((\widehat{\beta}_j(\pi))_{S_j}) \Big\} \\ &\qquad \le \sum_{j=1}^p \Big\{ \frac{1}{2n}\|\mathbf{x}_j - \mathbf{X}_{S_j}(\beta_j)_{S_j}\|_2^2 + \rho_\lambda((\beta_j)_{S_j}) \Big\} \quad \forall\, \beta_j \in \mathbb{R}^p, j = 1, \ldots, p,\end{aligned}$$

which implies $\widehat{\beta}_j(\pi) \in \widehat{\Theta}_\lambda(\mathbf{x}_j, \mathbf{X}; S_j)$ for each $j$. Since $\pi$ was arbitrary, the desired claim follows.

### 2.4.3 Proof of Lemma 2.4

We will show the complement of (3.30), i.e. $A(Z, \theta^*; T)^c \subset A(Z, \theta^*; S)^c$. Suppose $w \in A(Z, \theta^*; T)^c$, i.e. $\mathrm{supp}(\widetilde{\theta}) = \mathrm{supp}(\theta^*) := S^*$ for any $\widetilde{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z; T)$. We wish to show that for any $\widehat{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z; S)$, it must also be true that $\mathrm{supp}(\widehat{\theta}) = S^*$. Note that $w$ is nonrandom in this argument.

Since $\mathrm{supp}(\widehat{\theta}) \subset S \subset T$, we have

$$F(\widetilde{\theta}) \le F(\widehat{\theta})$$

for any $\widetilde{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z; T)$. But $\widetilde{\theta}$ is also feasible for the $S$-restricted problem since $\mathrm{supp}(\widetilde{\theta}) = S^* \subset S$, so that

$$F(\widetilde{\theta}) \ge F(\widehat{\theta}) \implies F(\widetilde{\theta}) = F(\widehat{\theta}).$$

Since the value $F(\widetilde{\theta})$ is by definition the global minimum of $F$ for the $T$-restricted problem, $F(\widehat{\theta})$ must be the global minimum of $F$ for the $S$-restricted problem. Thus $\widehat{\theta} \in \widehat{\Theta}_\lambda(Z\theta^* + w, Z; T)$, whence $\mathrm{supp}(\widehat{\theta}) = S^*$ as desired.

### 2.4.4 Proof of Lemma 2.1

The proof relies on the following property of $L^2$ projections:

**Lemma 2.13.** *For any two sets $S, R \subset [p]_j$, we have*

$$\beta_j(S \cup R) = \beta_j(S) \iff \varepsilon_j(S) \perp X_i,\ \forall i \in R$$

*Proof of Lemma 2.1.* To lighten the notation, let $\beta = \beta_j(S)$ and $S^* = m_j(S)$.

Since $\mathrm{supp}(\beta_j(S)) = S^*$, we have $\beta_j(S) = \beta_j(S^*)$. It follows from Lemma 2.13 that $\varepsilon_j(S^*) \perp\!\!\!\perp X_i$ for $i \in S \setminus S^*$. Similarly, since $\mathrm{supp}(\beta_j(T_k)) = S^*$, we have $\varepsilon_j(S^*) \perp\!\!\!\perp X_i$ for $i \in T_k \setminus S^*$ and $k = 1, 2$. It follows that

$$\varepsilon_j(S^*) \perp\!\!\!\perp X_i,\ \forall i \in (T_1 \setminus S^*) \cup (T_2 \setminus S^*)$$

hence the application of Lemma 2.13 in the reverse direction yields

$$\beta_j(T_1 \cup T_2) = \beta_j\big(S^* \cup (T_1 \setminus S^*) \cup (T_2 \setminus S^*)\big) = \beta_j(S^*) = \beta_j(S). \qquad \square$$

### 2.4.5 Proof of Proposition 2.6

Fix $S \subset [p]_j$ and let $\theta^* = \beta_j(S)$, $s^* = |m_j(S)| = \|\theta^*\|_0$ and $\varepsilon^* = \boldsymbol{\varepsilon}_j(S)$. Consider the event $\mathcal{A}(\varepsilon^*, \mathbf{X}, \theta^*; S) = \mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}, \beta_j(S); S)$. The complement $\mathcal{A}(\varepsilon^*, \mathbf{X}, \theta^*; S)^c$ represents the following model selection failure:

$$\mathrm{supp}(\widehat{\theta}) \neq \mathrm{supp}(\theta^*) \quad \exists\, \widehat{\theta} \in \widehat{\Theta}_\lambda(\mathbf{X}\theta^* + \varepsilon^*, \mathbf{X}; S)$$

Since $\mathrm{supp}(\theta^*) \subset S$, we can restrict $\mathbf{X}$ and $\theta^*$ to the columns in $S$, so that the above is equivalent to

$$\mathrm{supp}(\widehat{\theta}) \neq \mathrm{supp}(\theta^*_S) \quad \exists\, \widehat{\theta} \in \widehat{\Theta}_\lambda(\mathbf{X}_S\theta^*_S + \varepsilon^*, \mathbf{X}_S).$$

The key here is that $\varepsilon^*$ is independent of $\mathbf{X}_S$ by Lemma 2.3(a). Hence, conditioning on $\mathbf{X}_S$, we are dealing with a fixed design regression problem, with Gaussian noise $\varepsilon^* = \boldsymbol{\varepsilon}_j(S) \sim N(0, \omega_j^2(S) I_n)$. We obtain

$$\mathbb{P}\left(\mathcal{A}(\boldsymbol{\varepsilon}_j(S), \mathbf{X}_S, \theta^*)^c \;\middle|\; \mathbf{X}_S\right) \leq \exp[-\Phi_\lambda(\mathbf{X}_S, \theta_S^*, \omega_j^2(S))] = \exp(-\Xi_j(S)).$$

The desired result then follows from Corollary 2.5 and Lemma 2.12.

### 2.4.6 Proof of Theorem 2.3

To lighten the notation, for any vector $u$ let $u_1 := u_{S^*}$, $u_2 = u_{(S^*)^c}$, and also $\Delta = \widehat{\theta} - \theta^*$. Then, invoking the subadditivity of $\rho_\lambda$ (this follows from Condition 2.1),

$$\begin{aligned}
\rho_\lambda(\widehat{\theta}) - \rho_\lambda(\theta^*) &= \rho_\lambda(\Delta + \theta^*) - \rho_\lambda(\theta^*) \\
&= \rho_\lambda(\Delta_1 + \theta_1^* + \Delta_2) - \rho_\lambda(\theta_1^*) \\
&= \rho_\lambda(\Delta_1 + \theta_1^*) + \rho_\lambda(\Delta_2) - \rho_\lambda(\theta_1^*) \\
&\geq -\rho_\lambda(\Delta_1) + \rho_\lambda(\Delta_2).
\end{aligned}$$

It is straightforward to derive

$$\frac{1}{2n}\|y - Z\widehat{\theta}\|_2^2 - \frac{1}{2n}\|y - Z\theta^*\|_2^2 = \frac{1}{2n}\|Z\Delta\|^2 + \frac{1}{n}\langle w, Z\Delta\rangle \tag{3.50}$$

Since $(w, Z) \in \mathrm{GW}_{\rho_\lambda}(\delta)$, we can invoke the Gaussian width condition with $u = \Delta$,

$$\frac{1}{n}\langle w, Z\Delta\rangle \geq -\frac{1}{n}|\langle w, Z\Delta\rangle| \geq -\delta\frac{1}{2n}\|Z\Delta\|^2 - \delta\rho_\lambda(\Delta). \tag{3.51}$$

It follows that

$$\begin{aligned}
0 &\geq \frac{1}{2n}\|y - Z\widehat{\theta}\|_2^2 - \frac{1}{2n}\|y - Z\theta^*\|_2^2 + \rho_\lambda(\widehat{\theta}) - \rho_\lambda(\theta^*) \\
&\geq \frac{1}{2n}\|Z\Delta\|^2 + \frac{1}{n}\langle w, Z\Delta\rangle - \rho_\lambda(\Delta_1) + \rho_\lambda(\Delta_2) \\
&\geq \frac{1-\delta}{2n}\|Z\Delta\|^2 - \delta\rho_\lambda(\Delta) - \rho_\lambda(\Delta_1) + \rho_\lambda(\Delta_2),
\end{aligned}$$

where we have used (3.50) in the second line and (3.51) in the third. Thus

$$\begin{aligned} 0 &\geq \frac{1-\delta}{2n}\|Z\Delta\|^2 - \delta\rho_\lambda(\Delta) - \rho_\lambda(\Delta_1) + \rho_\lambda(\Delta_2) \\ &= \frac{1-\delta}{2n}\|Z\Delta\|^2 - (1+\delta)\rho_\lambda(\Delta_1) + (1-\delta)\rho_\lambda(\Delta_2) \\ &= (1-\delta)\Big[\frac{1}{2n}\|Z\Delta\|^2 + \rho_\lambda(\Delta_2) - \xi(\delta)\rho_\lambda(\Delta_1)\Big]. \end{aligned} \tag{3.52}$$

Thus

$$\rho_\lambda(\Delta_2) \leq \xi(\delta)\rho_\lambda(\Delta_1) \implies \Delta \in C_\rho(S^*, \xi(\delta)).$$

Recalling the definition of $\phi^2(Z, C_\rho(S^*, \xi))$ in (3.36), we conclude that

$$\frac{1}{2n}\|Z\Delta\|^2 \geq \frac{\phi^2}{2}\|\Delta\|_2^2$$

Using (3.52),

$$0 \geq \frac{1}{2n}\|Z\Delta\|^2 + \rho_\lambda(\Delta_2) - \xi(\delta)\rho_\lambda(\Delta_1) \tag{3.53}$$

$$\geq \frac{\phi^2}{2}\|\Delta\|_2^2 - \xi(\delta)\rho_\lambda(\Delta_1) \tag{3.54}$$

$$\geq \frac{\phi^2}{2}\|\Delta\|_2^2 - \xi(\delta)\rho'_\lambda(0+)\|\theta^*\|_0^{1/2}\|\Delta\|_2 := g(\|\Delta\|_2). \tag{3.55}$$

where we have used $\rho_\lambda(\Delta_1) \leq \rho'_\lambda(0+)\|\Delta_1\|_1 \leq \rho'_\lambda(0+)\|\theta^*\|_0^{1/2}\|\Delta\|_2$ in the last line. Finally, since $g(\|\Delta\|_2) \leq 0$, it follows after re-arranging that

$$\|\Delta\|_2 \leq \frac{2\,\xi(\delta)}{\phi^2}\,\rho'_\lambda(0+)\|\theta^*\|_0^{1/2}.$$

This proves (3.37).

For (3.38), since $\Delta \in C_\rho(S^*, \xi(\delta))$, we can use Lemma 2.14 to construct a set $M \subset [p]$ with $|M| = s^*$ such that $\Delta \in C_1(M, \xi(\delta))$. Then

$$\begin{aligned} \|\Delta\|_1 &= \|\Delta_M\|_1 + \|\Delta_{M^c}\|_1 \\ &\leq (1+\xi(\delta))\|\Delta_M\|_1 \\ &\leq (1+\xi(\delta))\|\theta^*\|_0^{1/2}\|\Delta_M\|_2 \\ &\leq \frac{2\,\xi(\delta)(1+\xi(\delta))}{\phi^2} \cdot \rho'_\lambda(0+)\|\theta^*\|_0. \end{aligned}$$

### 2.4.7 Proof of Proposition 2.7

We have the following lemma:

**Lemma 2.14.** *Suppose that $\rho_\lambda$ satisfies Condition 2.1. Then for any $A \subset [m]$ and $\xi > 0$,*

$$C_\rho^m(A, \xi) \subset \bigcup_{\substack{A' \subset [m] \\ |A'| = |A|}} C_1^m(A', \xi)$$

*In particular, for any fixed $Z$,*

$$\phi_\rho^2(Z, A) \geq \inf_{\substack{A' \subset [m] \\ |A'| = |A|}} \phi_1^2(Z, A').$$

*Proof of Lemma 2.14.* We follow the proof of (Zhang and Zhang, 2012, Proposition 2). Suppose $u \in C_\rho(A, \xi)$ and let $M$ be the index set of the $|A|$ largest $|u_i|$. Then $\rho_\lambda(u_{A^c}) \leq \xi \rho_\lambda(u_A)$ implies $\rho_\lambda(u_{M^c}) \leq \xi \rho_\lambda(u_M)$ since $\rho_\lambda$ is nondecreasing. Now,

$$\|u_{M^c}\|_1 = \sum_{j \in M^c} \rho_\lambda(|u_j|) \frac{|u_j|}{\rho_\lambda(|u_j|)} \leq \sum_{j \in M^c} \rho_\lambda(|u_j|) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)} = \rho_\lambda(u_{M^c}) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)},$$

where we have used the fact that $t/\rho_\lambda(t)$ is nondecreasing, which is an elementary consequence of Condition 2.1. Thus

$$\|u_{M^c}\|_1 \leq \rho_\lambda(u_{M^c}) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)} \leq \xi \rho_\lambda(u_M) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)}. \tag{3.56}$$

Now, on the right side we have

$$\begin{aligned} \xi \rho_\lambda(u_M) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)} &= \xi \sum_{i \in M} \rho_\lambda(|u_i|) \frac{\|u_{M^c}\|_\infty}{\rho_\lambda(\|u_{M^c}\|_\infty)} \\ &\leq \xi \sum_{i \in M} \rho_\lambda(u_i) \frac{|u_i|}{\rho_\lambda(|u_i|)} \\ &= \xi \|u_M\|_1, \end{aligned} \tag{3.57}$$

where we have invoked the monotonicity of $t/\rho_\lambda(t)$ once again. Combining (3.56) and (3.57) implies the desired result. □

The next two results are simple consequences of the definitions which will prove useful later on:

**Lemma 2.15.** *If $S \subset T$, then $C_1(S, \xi) \subset C_1(T, \xi)$ for any $\xi > 0$.*

*Proof.* Suppose $u \in C_1(S, \xi)$. Since $S \subset T$, we have $\|u_S\|_1 \leq \|u_T\|_1$ and $\|u_{T^c}\|_1 \leq \|u_{S^c}\|_1$. Thus,

$$\|u_{T^c}\|_1 \leq \|u_{S^c}\|_1 \leq \xi \|u_S\|_1 \leq \xi \|u_T\|_1,$$

whence $u \in C_1(T, \xi)$. □

*Definition* 2.13. For any fixed matrix $Z \in \mathbb{R}^{n \times m}$, we say that $Z$ satisfies a *restricted eigenvalue condition of order $k$* with parameters $\gamma > 0$ and $\xi > 0$ and write $Z \in \mathrm{RE}_\rho(k, \gamma; \xi)$ if

$$\frac{1}{n}\|Zu\|_2^2 \geq \gamma^2 \|u\|_2^2 \quad \forall u \in C_\rho(A, \xi),$$

uniformly for all $A \subset [m]$ with $|A| = k$. The dependence on $\xi$ will typically be ignored, in which case we write simply $Z \in \mathrm{RE}_\rho(k, \gamma)$.

*Definition* 2.14. We say that $\Sigma \in \mathbb{R}^{m \times m}$ satisfies a *restricted eigenvalue condition of order $k$* if

$$\|\Sigma^{1/2} u\|_2 \geq \gamma \|u\|_2 \quad \forall u \in C_\rho(A, \xi),$$

uniformly for all $A \subset [m]$ with $|A| = k$. When this holds, we shall write $\Sigma \in \mathrm{RE}_\rho(k, \gamma) = \mathrm{RE}_\rho(k, \gamma; \xi)$.

When $Z \in \mathrm{RE}_\rho(k, \gamma)$ it is necessarily true that

$$\inf_{\substack{A \subset [m] \\ |A| = k}} \phi_\rho^2(Z, A) \geq \gamma^2 > 0.$$

*Remark* 2.5. Another way of stating Lemma 2.14 is that if $Z \in \mathrm{RE}_1(k, \gamma)$, then $\phi_\rho^2(Z, A) \geq \gamma^2 > 0$ for any $A \subset [m]$ with $|A| = k$.

**Lemma 2.16.** *Suppose $Z \in \mathbb{R}^{n \times m}$ is a fixed matrix. If $Z \in \mathrm{RE}_\rho(k, \gamma)$ for some $k > 0$ then $Z_S \in \mathrm{RE}_\rho(k, \gamma)$ for all $S \subset [m]$ such that $|S| \geq k$.*

*Proof.* We proceed by induction: We will show that $Z \in \mathrm{RE}_\rho(k, \gamma)$ implies $Z_S \in \mathrm{RE}_\rho(k, \gamma)$ for any $|S| = m - 1$. Induction on $m$ then completes the proof.

Suppose $Z \in \mathrm{RE}_\rho(k, \gamma)$ and $S \subset [m]$ with $|S| = m - 1$. Let $A \subset S$ be any subset of columns with $|A| = k$. Choose $u \in C_\rho^{m-1}(A)$. We may augment $u$ with an extra zero in the missing column from $\mathbf{X}$ (there is only one since $|S| = m - 1$ by the inductive assumption); call this new vector $\tilde{u}$. Then $\tilde{u} \in C_\rho^m(A)$, whence

$$\frac{1}{n}\|Z_S u\|_2^2 = \frac{1}{n}\|Z\tilde{u}\|_2^2 \geq \gamma^2 \|\tilde{u}\|_2^2 = \gamma^2 \|u\|_2^2.$$

Since $A$ was arbitrary, it follows that $Z_S \in \mathrm{RE}_\rho(k, \gamma)$. □

**Lemma 2.17.** *Suppose $Z \in \mathbb{R}^{n \times m}$ is a fixed matrix. If $Z \in \mathrm{RE}_\rho(d, \gamma)$ for some $\gamma > 0$, then*

$$\inf_{1 \leq j \leq p} \inf_{S \subset [p]_j} \inf_{\substack{A \subset S \\ |A| \leq d}} \phi_\rho^2(Z_S, A) \geq \gamma^2 > 0. \tag{3.58}$$

*Proof.* Choose $A \subset [p]_j$ with $|A| = d$. By Lemma 2.15, if $A' \subset A$ then $\phi_\rho^2(Z, A') \geq \phi_\rho^2(Z, A)$, i.e. if $Z$ satisfies REC on $A$, then it satisfies REC on any subset of $A$. Thus, it suffices to show (3.58) for $|A| = d$.

Since $Z \in \mathrm{RE}_\rho(d, \gamma)$, we have

$$\phi_\rho^2(Z, A) \geq \gamma^2 > 0,$$

which, combined with the observation in the previous paragraph implies

$$\inf_{\substack{A \subset [p]_j \\ |A| = d}} \phi_\rho^2(Z, A) \geq \gamma^2 > 0.$$

This implies that for any $S \subset [p]_j$,

$$\inf_{\substack{A \subset S \\ |A|=d}} \phi^2(Z, A) \geq \gamma^2 > 0.$$

Invoking Lemma 2.16,[1] we have for any $S \subset [p]_j$,

$$\inf_{\substack{A \subset S \\ |A|=d}} \phi^2(Z_S, A) \geq \gamma^2 > 0.$$

Since $\gamma^2$ is a constant that is independent of $j$ and $S$, we may take the infimum over $j$ and $S$ on the left to obtain the desired result. □

**Proposition 2.18.** *Suppose $Z \in \mathbb{R}^{n \times m}$ is a fixed matrix and $\rho = \rho_\lambda$ satisfies Condition 2.1. If $Z \in \mathrm{RE}_1(d, \gamma)$ for some $\gamma > 0$, then $Z \in \mathrm{RE}_\rho(d, \gamma)$ and thus*

$$\inf_{1 \leq j \leq p} \inf_{S \subset [p]_j} \inf_{\substack{A \subset S \\ |A| \leq d}} \phi_\rho^2(Z_S, A) \geq \gamma^2 > 0.$$

*Proof.* Lemma 2.14 implies

$$\inf_{\substack{A \subset S \\ |A| \leq d}} \phi_\rho^2(Z_S, A) \geq \inf_{\substack{A \subset S \\ |A| \leq d}} \phi_1^2(Z_S, A),$$

so the result follows from Lemma 2.17 with $\rho_\lambda = \lambda \| \cdot \|_1$. □

According to Proposition 2.18, in order to obtain uniform control over each (generalized) restricted eigenvalue for all possible neighbourhood regression problems, it suffices to show that $\mathbf{X} \in \mathrm{RE}_1(d, \gamma)$ for some constant $\gamma > 0$.

The following theorem is a quick consequence of Corollary 1 in Raskutti et al. (2010):

**Lemma 2.19.** *Assume $\Sigma \in \mathbb{R}^{p \times p}$ is positive definite and satisfies Condition 2.2. Let $\xi > 0$ and assume $\mathbf{X} \overset{iid}{\sim} \mathcal{N}(0, \Sigma)$. Then there exist universal constants $c, c', c''$, such that if*

$$n > c'' \frac{\sigma_{\max}^2 (1 + \xi)^2}{r_{\min}(\Sigma)} d(\Sigma) \log p$$

[1] We may dispense with the condition that $|S| \geq d$ since we have just shown that $Z \in \mathrm{RE}(k, \gamma)$ for all $k \leq d$.

*then with probability at least* $1 - c' \exp(-cn)$, $\mathbf{X} \in \mathrm{RE}_1(k, r_{\min}(\Sigma)^{1/2})$ *for any* $k \leq d(\Sigma)$.

*Proof.* Since $\Sigma$ is positive definite, then so is $\Sigma^{1/2}$, that is

$$\|\Sigma^{1/2} u\|_2^2 \geq r_{\min}(\Sigma) \|u\|_2^2 \quad \forall u \in \mathbb{R}^p.$$

Thus, in particular $\Sigma \in \mathrm{RE}(k, r_{\min}(\Sigma)^{1/2})$ for all $k \leq p$ and any $\xi > 0$. The desired result then follows from Corollary 1 from Raskutti et al. (2010) applies and

$$n > c'' \frac{\sigma_{\max}^2 (1+\xi)^2}{r_{\min}(\Sigma)} d(\Sigma) \log p > c'' \frac{\sigma_{\max}^2 (1+\xi)^2}{r_{\min}(\Sigma)} k \log p$$

for any $k \leq d(\Sigma)$. □

*Proof of Proposition 2.7.* Lemma 2.19 implies $\mathbf{X} \in \mathrm{RE}_1(d, r_{\min}(\Sigma)^{1/2})$ with the desired probability, and Proposition 2.18 ensures the desired lower bound holds. □

### 2.4.8 Proof of Lemma 2.8

Let $\mathbf{w} = \boldsymbol{\varepsilon}_j(S)$. Then on $\mathcal{E}_S(\delta, \lambda)^c$ we have

$$\begin{aligned} & \frac{\delta}{2n} \|\mathbf{X}u\|_2^2 - \frac{1}{n} \mathbf{w}^T \mathbf{X} u + \delta \rho_\lambda(u) > 0 \\ \iff \quad & \frac{1}{2n} \|\mathbf{w}/\delta\|_2^2 + \frac{1}{2n} \|\mathbf{X}u\|_2^2 - \frac{1}{\delta n} \mathbf{w}^T \mathbf{X} u + \rho_\lambda(u) = \frac{1}{2n} \|\mathbf{w}/\delta - \mathbf{X}u\|_2^2 + \rho_\lambda(u) \\ & > \frac{1}{2n} \|\mathbf{w}/\delta\|_2^2. \end{aligned}$$

The latter inequality implies that

$$0 \in \arg\min_u \|\mathbf{w}/\delta - \mathbf{X}u\|_2^2/(2n) + \rho_\lambda(u)$$

is the unique global minimizer of the right hand side. Recalling the definition of $\mathcal{A}\Big(\boldsymbol{\varepsilon}_j(S)/\delta, \mathbf{X}, 0; S\Big)$ in (3.20), we obtain the desired result.

### 2.4.9 Proof of Proposition 2.9

By analogy with (3.33), define for any neighbourhood $S \subset [p]_j$,

$$\xi_j(S) = \xi_j(S;\delta) := \Phi_\lambda(\mathbf{X}_S,\, 0,\, \omega_j^2(S)/\delta^2). \tag{3.59}$$

Using Definition 2.7,

$$\begin{aligned}\xi_j(M_j(T);\delta) &= \Phi_\lambda(\mathbf{X}_{M_j(T)},\, 0,\, \omega_j^2(T)/\delta^2) \\ &= -\log \mathbb{P}\left[\mathcal{A}(\mathbf{w}, \mathbf{X}_{M_j(T)}, 0)\right], \quad \mathbf{w} \sim \mathcal{N}\Big(0, \frac{\omega_j^2(T)}{\delta^2} I_n\Big).\end{aligned}$$

We have

$$\begin{aligned}\Pr(\mathcal{E}_S(\delta,\lambda)) &= \Pr\left(\mathcal{A}\Big(\boldsymbol{\varepsilon}_j(S)/\delta, \mathbf{X}, \mathbf{0}; S\Big)\right) \\ &\le \Pr\left(\mathcal{A}\Big(\boldsymbol{\varepsilon}_j(M_j(S))/\delta, \mathbf{X}, \mathbf{0}; M_j(S)\Big)\right) \\ &= e^{-\xi_j(M_j(S))}.\end{aligned}$$

The first line follows from Lemma 2.8, the second from monotonicity (cf. Corollary 2.5), and the last line is the definition of the model selection exponent. Thus, Lemma 2.12 implies

$$\mathbb{P}\left(\bigcup_{S\subset[p]_j} \mathcal{E}_S(\delta,\lambda)\right) \le \sum_{T\in m_j(\mathcal{S})} \mathbb{E}e^{-\xi_j(M_j(T))}.$$

Now take the union bound over $j = 1, \ldots, p$ to get

$$\Pr(\mathcal{E}(\delta,\lambda)^c) \le \sum_{j=1}^{p} \sum_{T\in m_j(\mathcal{S})} \mathbb{E}e^{-\Phi_\lambda(\mathbf{X}_{M_j(T)}, 0, \sigma_{\max}^2/\delta^2)} \le p\binom{p}{d} \mathbb{E}e^{-\psi_\lambda(\mathbf{X}, \sigma_{\max}^2; \delta)},$$

as desired.

# CHAPTER 4

# Contributions and future directions

The focus of this dissertation, at a high-level, has been the study of high-dimensional Bayesian network under a Gaussian model. We will conclude by summarizing the main contributions at a high-level in Section 1, and then discuss some current directions for this research, some open problems, and suggestions for future work in Section 2.

## 1 High-level overview

Traditionally, the statistics literature has focused on convex, identifiable problems such as the undirected Gaussian graphical model. Departing somewhat from this tradition, a unifying theme of this dissertation has been the development of analytical and computational techniques to address nonidentifiability and nonconvexity in statistical problems. With this in mind, the high-level contributions of this dissertation can be summarized as follows:

1. Exploiting algebraic assumptions such as linearity in order to simplify the learning problem for continuous Bayesian networks,

2. Drawing attention to some issues with existing methods based on greedy heuristics and large sample theory,

3. Extending previous intuition regarding neighbourhood estimation in undirected graphical models to directed graphical models.

The general structure learning problem for Bayesian networks is known to be NP-hard, and hence it is generally accepted that one must resort to heuristics when developing algorithms for solving (or, more accurately, approximating solutions to) this problem. The existing literature has focused on assumptions such as identifiability, faithfulness, and the availability of large samples, which can be restrictive in practice and lead to undesirable output. Moreover, for score-based learning in particular, the literature has almost exclusively considered greedy approaches to optimization. In contrast, the present work trades off these assumptions at the cost of assuming a linear model, which is of course can be a strong assumption in and of itself. The motivation has been to address what *can* be accomplished in such a setting, with the hopes that future work will address more general nonlinear and non-Gaussian models.

The algorithm introduced in Chapter 2 offers some new insight into how to accelerate general score-based structure learning. In particular, the use of cyclic coordinate descent should be contrasted with greedy hill climbing, which is the approach most commonly found in the literature. By avoiding a complete greedy search with each update, the cyclic approach allows for fast updating of the parameters with no degradation in solution quality. Even though the parameter space is nonconvex, by restricting the search to one direction at a time, the single parameter updates are very efficient due to the convex reparamterization of the Gaussian likelihood. This shows that even though the overall program is indeed nonconvex, there are still gains to be reaped by convexifying the program as much as possible. One open problem that remains is to study the convergence of this scheme, which is one of the future directions discussed in the next section.

Chapter 3 puts the focus on statistical aspects of score-based learning. Since the estimators we are interested in are the result of nonconvex programs, it is of interest to study both local and global solutions. The results for local minimizers give a concrete example of how we can use the same arguments and intuition from

the theory classical of maximum likelihood estimation in order to study a nonconvex, nonidentifiable problem. The global theory, which is of a quite different nature, yields an elegant characterization of the penalized least squares estimator in terms of neighbourhood regression. This analysis allows us to reduce the study of linear structural equations to the more familiar linear regression setting. This reduction is made quite explicit by the introduction of model selection exponents, which are used to provide useful probability bounds that can be computed based on existing results. Finally, our analysis confirms long-held intuition regarding the performance differences between convex regularization via $\ell_1$ penalties and nonconvex penalties such as the MCP, SCAD, and $\ell_0$ penalty, and extends this intuition to a new setting.

## 2 Future directions

The work in this dissertation should be considered as a mere springboard for future advances in structure learning in Bayesian networks. We outline some of these directions here.

### 2.1 Optimization

A significant hanging thread from this work remains understanding the convergence and other optimization-theoretic properties of the algorithm outlined in Chapter 2. Along this axis, many questions linger:

- Does this algorithm always converge?
- Which kind of stationary points does it converge to?
- How can the optimization step be improved?

A growing body of literature in optimization suggests that coordinate descent works well for certain kinds of nonconvex programs. Due to the intrinsic nonconvex constraint imposed by enforcing acyclicity, it not clear whether or not these results apply to the method developed here, and it would be of great interest to address this. Our empirical results indicate that this method performs quite well—particularly in comparison to existing approaches—but this does not tell with confidence that the stationary points attracted by the algorithm are the ones we seek. Finally, it would be interesting to study how the method can be improved through the use of more sophisiticated optimization routines such as stochastic and adaptive coordinate descent.

## 2.2 Experimental comparisons

Coordinate descent is a general procedure that can be applied to a wide variety of optimization problems. A natural question is to ask how the kind of procedure outlined in Chapter 2 would perform in different scenarios. Thus far, we have only consider the simple case of linear, Gaussian models, for which the log-likelihood is explicit and tractable. One could inquire into a variety of generalizations, including:

*Different error distributions.* It would be of interest to study how the performance of the estimator varies as the properties of the error distribution change, for example, the variance or the skewness. This should be straightforward and remains for future work. Along these lines, adapting the method presented here to discrete data as in Fu et al. (2014) would also provide further justification for the general framework contained herein.

*Different loss functions.* One could imagine replacing the reparameterized negative log-likelihood with other loss functions in the CCDr algorithm. As long as we can still derive closed-form solutions to the single parameter updates, the

basic algorithm makes sense as an approximation. For example, issues of speed and scalability aside, it would be interesting to directly compare the estimation performance of the nonconvex negative log-likelihood (1.9) to the convexified version, (2.1). Motivated by our high-dimensional results, one could also study the empirical performance of the least squares loss function.

*Different structural models.* Finally, one might consider generalizing the underlying model itself to allow for dependent errors or even nonlinear models. In the latter case, one could consider GLMs, additive models, or even the general case of nonlinear functional models. Additionally, theoretical guarantees in these types of settings would be of interest in their own right.

## 2.3 High-dimensional theory

In this section we discuss some possible extensions to the theory developed in Section 2 of Chapter 3.

### 2.3.1 Applications and examples

The probability bounds in both of the main theorems involve so-called model selection exponents, which are general quantities that depend on regularizer $\rho_\lambda$ and the strength of the assumptions placed on the underlying linear models. We have not considered any explicit examples of these exponents in this dissertation, and a natural first step is to derive explicit expressions for these exponents in some special cases. This should be relatively straightforward for the well-understood $\ell_1$-regularized or "Lasso" estimator. Unfortunately, in the literature on concave regularization, it is more popular to derive expressions for *local minimizers*, and generally these results apply to certain special local minimizers, and not necessarily *all* local minimizers. There are, however, results for global minimizers (Zhang and Zhang, 2012; Huang et al., 2012; Fan and Lv, 2013), and we expect these

types of results to apply to the framework developed here.

### 2.3.2 Local minimizers and stationary points

Considering global minimizers (as we have done) instead of local minimizers greatly simplifies the analysis, but it would be interesting to derive similar results for local minimizers. Such an extension would require establishing useful results for all possible local minimizers of the restricted program over $\mathbb{D}[\pi]$. It seems like the most direct approach would be to study stationary points of the global program (i.e. on $\mathbb{D}$) and relate them somehow to stationary points of the restricted programs (i.e. on $\mathbb{D}[\pi]$). This seems feasible, but will require different arguments and some new ideas. Such an extension would provide a significant practical justification for this type of estimator.

### 2.3.3 Extensions

Finally, it would of course be interesting and relevant to extend this analysis to non-Gaussian models. As long as the errors remain independent of the neighbourhood (i.e. $\varepsilon_j(S) \perp\!\!\!\perp \mathbf{X}_S$), much of this analysis carries over in a straightforward manner. Eliciting when this is a reasonable assumption remains for future work, as is comparing the empirical performance of this estimator under such assumptions.

# Bibliography

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2009.

P. Bühlmann and S. van de Geer. *Statistics for high-dimensional data: Methods, theory and applications*. Springer, 2011.

P. Bühlmann, J. Peters, J. Ernest, et al. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42 (6):2526–2556, 2014.

S. Chaudhuri, M. Drton, and T. S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94(1):199–216, 2007.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from data*, pages 121–130. Springer, 1996.

D. M. Chickering. Optimal structure identification with greedy search. *The Journal of Machine Learning Research*, 3:507–554, 2003.

D. M. Chickering and C. Meek. Finding optimal Bayesian networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 94–102. Morgan Kaufmann Publishers Inc., 2002.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian

networks is NP-hard. *The Journal of Machine Learning Research*, 5:1287–1330, 2004.

A. P. Dempster. *Elements of continuous multivariate analysis*, volume 388. Addison-Wesley Reading, Mass., 1969.

A. P. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

M. Drton, B. Sturmfels, and S. Sullivant. *Lectures on algebraic statistics*. Springer, 2009.

M. Drton, R. Foygel, and S. Sullivant. Global identifiability of linear structural equation models. *The Annals of Statistics*, 39(2):865–886, 2011.

B. Ellis and W. H. Wong. Learning causal Bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482), 2008.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456): 1348–1360, 2001.

J. Fan and J. Lv. A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101, 2010.

J. Fan and J. Lv. Nonconcave penalized likelihood with NP-dimensionality. *Information Theory, IEEE Transactions on*, 57(8):5467–5484, 2011.

Y. Fan and J. Lv. Asymptotic equivalence of regularization methods in thresholded parameter space. *Journal of the American Statistical Association*, 108 (503):1044–1061, 2013.

T. S. Ferguson. *A course in large sample theory*, volume 38. CRC Press, 1996.

R. Foygel and M. Drton. Extended Bayesian information criteria for Gaussian graphical models. In *Advances in Neural Information Processing Systems*, pages 604–612, 2010.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the Graphical Lasso. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

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

F. Fu, J. Gu, and Q. Zhou. Adaptive penalized estimation of directed acyclic graphs from categorical data. *arXiv preprint arXiv:1403.2310*, 2014.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, 2011.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. One iteration CHC algorithm for learning Bayesian networks: An effective and efficient algorithm for high dimensional problems. *Progress in Artificial Intelligence*, 1(4):329–346, 2012.

D. Geiger and D. Heckerman. Learning Gaussian networks. *arXiv preprint arXiv:1302.6808*, 2013.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3): 197–243, 1995.

J. Huang, P. Breheny, and S. Ma. A selective review of group selection in high-dimensional models. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 27(4), 2012.

J. Z. Huang, N. Liu, M. Pourahmadi, and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98, 2006.

A. Hyvärinen, K. Zhang, S. Shimizu, and P. O. Hoyer. Estimation of a structural vector autoregression model using non-Gaussianity. *The Journal of Machine Learning Research*, 11:1709–1731, 2010.

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

M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *The Annals of Statistics*, 37(6B):4254, 2009.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational intelligence*, 10(3):269–293, 1994.

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

J. Lv and Y. Fan. A unified approach to model selection and sparse recovery using regularized least squares. *The Annals of Statistics*, 37(6A):3498–3528, 2009.

R. Mazumder, J. H. Friedman, and T. Hastie. SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106 (495):1125–1138, 2011.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

J. Peters, J. Mooij, D. Janzing, and B. Schölkopf. Identifiability of causal graphs using functional models. *arXiv preprint arXiv:1202.3757*, 2012.

M. Pourahmadi. *High-dimensional covariance estimation*. John Wiley & Sons, 2013.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL `http://www.R-project.org`.

G. Raskutti and C. Uhler. Learning directed acyclic graphs based on sparsest permutations. *arXiv preprint arXiv:1307.0366*, 2014.

G. Raskutti, M. J. Wainwright, and B. Yu. Restricted eigenvalue properties for correlated Gaussian designs. *The Journal of Machine Learning Research*, 11: 2241–2259, 2010.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing $\ell_1$-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

R. Redner. Note on the consistency of the maximum likelihood estimate for nonidentifiable distributions. *The Annals of Statistics*, 9(1):225–228, 1981.

R. W. Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial mathematics V*, pages 28–43. Springer, 1977.

P. Rütimann and P. Bühlmann. High dimensional sparse covariance estimation via directed acyclic graphs. *Electronic Journal of Statistics*, 3:1133–1160, 2009.

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

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using L1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(i03), 2010.

M. Scutari. Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn R package. *arXiv preprint arXiv:1406.7648*, 2014.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *The Journal of Machine Learning Research*, 7:2003–2030, 2006.

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

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

N. Städler, P. Bühlmann, and S. Van De Geer. $\ell_1$-penalization for mixture regression models. *Test*, 19(2):209–256, 2010.

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

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

C. Uhler, G. Raskutti, P. Bühlmann, and B. Yu. Geometry of the faithfulness assumption in causal inference. *The Annals of Statistics*, 41(2):436–463, 2013.

S. van de Geer and P. Bühlmann. $\ell_0$-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics*, 41(2):536–567, 2013.

H. Wang, R. Li, and C.-L. Tsai. Tuning parameter selectors for the Smoothly Clipped Absolute Deviation method. *Biometrika*, 94(3):553–568, 2007.

S. Wright. On the nature of size factors. *Genetics*, 3(4):367, 1918.

S. Wright. Correlation and causation. *Journal of agricultural research*, 20(7): 557–585, 1921.

S. Wright. The method of path coefficients. *The Annals of Mathematical Statistics*, 5(3):161–215, 1934.

T. T. Wu and K. Lange. Coordinate descent algorithms for Lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

J. Xiang and S. Kim. A* Lasso for learning a sparse Bayesian network structure for continuous variables. In *Advances in Neural Information Processing Systems*, pages 2418–2426, 2013.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

C.-H. Zhang and T. Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

J. Zhang and P. Spirtes. Strong faithfulness and uniform consistency in causal inference. In *Proceedings of the nineteenth conference on uncertainty in artificial intelligence*, pages 632–639. Morgan Kaufmann Publishers Inc., 2002.

P. Zhao and B. Yu. On model selection consistency of Lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

Q. Zhou. Multi-domain sampling with applications to structural inference of Bayesian networks. *Journal of the American Statistical Association*, 106(496): 1317–1330, 2011.

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