Journal of Machine Learning Research 16 (2015) 2273-2328 Submitted 12/13; Revised 10/14; Published 8/15

Concave Penalized Estimation of Sparse Gaussian Bayesian Networks

Bryon Aragam BRYON@STAT.UCLA.EDU
Qing Zhou ZHOU@STAT.UCLA.EDU
Department of Statistics
University of California, Los Angeles
Los Angeles, CA 90024, USA

Editor: Max Chickering

Abstract

We develop a penalized likelihood estimation framework to learn the structure of Gaussian Bayesian networks from observational data. In contrast to recent methods which accelerate the learning problem by restricting the search space, our main contribution is a fast algorithm for score-based structure learning which does not restrict the search space in any way and works on high-dimensional data sets with thousands of variables. Our use of concave regularization, as opposed to the more popular ℓ_0 (e.g. BIC) penalty, is new. Moreover, we provide theoretical guarantees which generalize existing asymptotic results when the underlying distribution is Gaussian. Most notably, our framework does not require the existence of a so-called faithful DAG representation, and as a result, the theory must handle the inherent nonidentifiability of the estimation problem in a novel way. 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. Based on these experiments, we show that our algorithm obtains higher sensitivity with comparable false discovery rates for high-dimensional data and scales efficiently as the number of nodes increases. In particular, the total runtime for our method to generate a solution path of 20 estimates for DAGs with 8000 nodes is around one hour.

Keywords: Bayesian networks, concave penalization, directed acyclic graphs, coordinate descent, nonconvex optimization

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 work, we use a penalized likelihood estimation framework to learn the structure of Gaussian Bayesian networks from observational data. Our 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.

One of the key developments in our method is the application of modern regularization techniques, including both ℓ_1 and concave penalties. Although ℓ_1 regularization 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 ℓ_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, 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.

The organization of the rest of this paper is as follows: In the remainder of this section we review previous work and compare our contributions with the existing literature. In Section 2, we establish the necessary preliminaries for our approach via structural equations. In Section 3 we define and discuss the penalized estimator that is the focus of this paper. Section 4 then provides the necessary finite-dimensional theory to justify the use of our estimator. A complete description of our algorithm is outlined in Section 5, followed by an empirical evaluation of the algorithm in Section 6. Section 6 also offers a side-by-side comparison of our algorithm with four other structure learning algorithms, and Section 7 provides an evaluation of these algorithms using a real-world data set. We finally conclude with a discussion of some future directions for this research.

1.1 Related Work

The idea of using penalized likelihood estimation and sparse regression to learn Gaussian Bayesian networks in high dimensions is a recent development, and the theoretical basis for ℓ_0 penalization has been instigated by van de Geer and Bühlmann (2013). Their work relies on the interpretation of Gaussian Bayesian networks in terms of structural equation models (Drton and Richardson, 2008; Drton et al., 2011), which provides a natural interpretation of network edges in terms of coefficients of a regression model. To the best of our knowledge, the work of van de Geer and Bühlmann (2013) is the first high-dimensional analysis of a score-based approach in the literature, and has not yet been generalized to the case of continuous ℓ_1 or concave penalties. As the nontrivial and novel nature of this analysis would detract from our primary goal of addressing computational challenges, we will not pursue a corresponding high-dimensional theory here. Given this foundational work, our goal is to show that these ideas can be translated into a family of fast algorithms for score-based learning of Bayesian network structures.

While the traditional approach to estimating Bayesian networks uses ℓ_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 ℓ_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 ℓ_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. By contrast, the method proposed here adapts this algorithm for use with observational, high-dimensional data, and takes explicit advantage of convexity and sparsity. We also extend these ideas to a general class of penalties which includes both ℓ_0 and ℓ_1 regularization as special cases. The result is an algorithm which easily handles thousands of nodes in a matter of minutes. Moreover, in contrast to the theory proposed in Fu and Zhou (2013), our theory does not rely on faithfulness or identifiability.

1.2 Review of Structure Learning

Traditionally, there are three main approaches to learning Gaussian 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. In contrast, the method proposed in the present work is a genuine score-based

method and scales efficiently to graphs with thousands of variables. To the best of our knowledge, this is one of the first purely score-based methods that accomplishes this in the sense that we rely neither on significance tests (as in the constraint-based approach) nor pruning the search space (as in the hybrid approach).

2. 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. In what follows, the reader may recall that the structure of a Bayesian network is completely determined by a directed acyclic graph, and hence learning the structure of a Bayesian network reduces to learning directed acyclic graphs. In order to maintain consistency and ease of translation, much of our notation is adapted from van de Geer and Bühlmann (2013).

2.1 Background and Notation

We assume throughout that the data are generated from a p-variate Gaussian distribution,

$$(X_1, \ldots, X_p) \sim \mathcal{N}(0, \Sigma_0), \tag{1}$$

where the covariance matrix $\Sigma_0 \in \mathbb{R}^{p \times p}$ is positive definite. Such a model can always be written as a set of Gaussian structural equations as follows (see Dempster, 1969):

$$X_j = \sum_{i=1}^{p} \beta_{ij}^0 X_i + \varepsilon_j, \quad j = 1, \ldots, p, \tag{2}$$

where the ε_j are mutually independent with $\varepsilon_j \sim \mathcal{N}(0, (\omega_j^0)^2)$, ε_j is independent of $\Pi_j^0 = \{X_i : \beta_{ij}^0 \neq 0\}$, and $\beta_{jj}^0 = 0$. This decomposition is not unique, and we will let $B_0 = (\beta_{ij}^0)$ denote any matrix of coefficients that satisfies (2). The matrix $B_0 = (\beta_{ij}^0)$ can then be regarded as the weighted adjacency matrix of a directed acyclic graph and represents a Bayesian network for the distribution $\mathcal{N}(0, \Sigma_0)$. Recall that a *directed acyclic graph* B is a directed graph containing no directed cycles. 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 nodes of B are in one-to-one correspondence with the random variables $X_1, \ldots, X_p$ in our model. Following tradition, we make no distinction between random variables and nodes or vertices, and will use these terms interchangeably. We say that X_k is a *parent* of X_j if $X_k \to X_j$, and the set of parents of X_j will be denoted by $\Pi_j := \Pi_j(B)$. We will denote the number of edges in B by $s_B := |\{\beta_{ij} \neq 0\}|$. When the underlying graph is clear from context, we will suppress the dependence on B and simply denote the number of edges by s.

Unless otherwise noted, $\|\cdot\|$ shall always mean the standard Euclidean norm and $\|\cdot\|_F$ will denote the standard ℓ_2 Frobenius norm on matrices. 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 $a_1, \ldots, a_p$. In particular, we will write $B = [\beta_1 \,|\, \cdots \,|\, \beta_p]$ for an arbitrary DAG. The support of a matrix is defined by $\mathrm{supp}(B) := \{(i,j) : \beta_{ij} \neq 0\}$.

If $X = [x_1 \,|\, \cdots \,|\, x_p]$ is an $n \times p$ data matrix of i.i.d. observations from (1), then we can rewrite (2) as a matrix equation,

$$X = XB_0 + E, \tag{3}$$

where $E \in \mathbb{R}^{n \times p}$ is the matrix of noise vectors. This model has $p(p-1) + p = p^2$ free parameters, which we encode through two matrices given by (B_0, Ω_0). Here, $\Omega_0 = \mathrm{diag}((\omega_1^0)^2, \ldots, (\omega_p^0)^2)$ is the matrix of error variances. We denote the matrix of error variances by Ω in order to avoid confusion with the covariance matrix Σ.

There are thus two unknown parameters in (2):

$$B := (\beta_{ij}) \in \mathbb{R}^{p \times p},$$
$$\Omega := \mathrm{diag}(\omega_1^2, \ldots, \omega_p^2) \in \mathbb{R}^{p \times p}.$$

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

$$L(B, \Omega \,|\, X) = \sum_{j=1}^{p} \left[\frac{n}{2} \log(\omega_j^2) + \frac{1}{2\omega_j^2} \|x_j - X\beta_j\|^2 \right]. \tag{4}$$

Observe that the function in (4) is nonconvex; this fact will play an important role in the development of our method.

Remark 1. The vast majority of the literature on Bayesian networks focuses on discrete data, in contrast to our method 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 regression (Fu and Zhou, 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 7).

Thus far we have viewed the distribution $\mathcal{N}(0, \Sigma_0)$ as the data-generating mechanism, rewriting this in terms of (B_0, Ω_0) by using well-known properties of the Gaussian distribution. We could just as well have gone the other way around: Given a DAG B and variance matrix $\Omega = \mathrm{diag}(\omega_1^2, \ldots, \omega_p^2)$, the parameters (B, Ω) uniquely define a structural equation model as in (2), and this model defines a $\mathcal{N}(0, \Sigma)$ distribution. By (3), we have for any (B, Ω),

$$\Sigma = (I - B)^{-T} \Omega (I - B)^{-1}, \tag{5}$$

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

$$\Theta = \Theta(B, \Omega) = (I - B)\Omega^{-1}(I - B)^T. \tag{6}$$

By using (6) and defining $S_n := X^T X$, the negative log-likelihood in (4) can be rewritten in terms of $\Theta = \Theta(B, \Omega)$ directly as

$$L(\Theta \,|\, X) = -\frac{n}{2} \log\det \Theta + \frac{1}{2} \operatorname{tr}(\Theta S_n). \tag{7}$$

By combining (4) and (7), we have $L(B, \Omega \,|\, X) = L(\Theta(B, \Omega) \,|\, X)$. This expression shows how the weighted adjacency matrix of a DAG can be considered as a reparameterization of the usual normal distribution, and gives us an explicit connection between inverse covariance estimation and DAG estimation, which will be explored further in the next subsection.

Since the decomposition of a normal distribution as a linear structural equation model (SEM) as in (2) is not unique, we can define the following equivalence class of DAGs:

$$\mathcal{E}(\Theta) := \{(B, \Omega) : \Theta(B, \Omega) = \Theta\}. \tag{8}$$

When $(B, \Omega) \in \mathcal{E}(\Theta)$, we shall say that B *represents*, or *is consistent with*, Θ. Two DAGs $(B, \Omega), (B', \Omega')$ will be called *equivalent* if they belong to the same equivalence class $\mathcal{E}(\Theta)$.

This definition of equivalence in terms of equivalent parameterizations 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 B_0—which roughly speaking entails that B_0 contains exactly the same conditional independence constraints as the true distribution—we have deliberately sidestepped considerations of this hypothesis since our theory does not rely on faithfulness.

Remark 2. Strictly speaking, a Gaussian Bayesian network is specified by *both* a weighted adjacency matrix B and a variance matrix Ω, 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 Ω paired with B.

2.2 Comparison of Graphical Models

The previous section showed how the weighted adjacency matrix of a DAG can be considered as a reparameterization of the usual normal distribution, and gave an explicit connection between inverse covariance estimation and DAG estimation: Equation (6) shows how any DAG (B, Ω) uniquely defines an inverse covariance matrix $\Theta = \Theta(B, \Omega)$. It follows that any estimate $(\widehat{B}, \widehat{\Omega})$ of the true DAG yields an estimate of Θ_0 given by $\widehat{\Theta} := \Theta(\widehat{B}, \widehat{\Omega})$. In the context of the PC algorithm, this has been studied by Rütimann and Bühlmann (2009). As a result, one may also view our framework as defining an estimator for the inverse covariance matrix. Covariance selection and precision matrix estimation have 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; Friedman 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 here, however, a few comments are in order.

First, while these two problems are deeply connected, estimating an inverse covariance matrix is significantly easier: The estimation problem is statistically identifiable and the

parameter space is convex. This stands in stark contrast to the more difficult problem of estimating an underlying DAG, which we have shown to be simultaneously *nonidentifiable* and *nonconvex*. As a result, while the high-dimensional properties of regularized covariance estimation are well-understood, the high-dimensional properties of DAG estimation have proven much more difficult to ascertain. The only significant results we are aware of are in van de Geer and Bühlmann (2013) and Kalisch and Bühlmann (2007).

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. For applications of Bayesian networks to inferring so-called Markov blankets, see Aliferis et al. (2010a,b).

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 Θ_0. This will be established through the theory presented in Section 4, as well as empirically via the simulations discussed in Section 6. Because of the significance and popularity of covariance selection methods, it would of course be interesting to compare our estimate of Θ_0 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.

2.3 Permutations and Equivalence

In this section we wish to exhibit the connection between equivalent DAGs as defined in (8) 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 $\{1, \dots, p\}$. For an arbitrary matrix A and any $\pi \in \mathcal{P}$, let us denote by $P_\pi A$ the matrix obtained by permuting the rows and columns of A according to π, so that $(P_\pi A)_{ij} = a_{\pi(i)\pi(j)}$. Then a DAG can be equivalently defined as any graph whose adjacency matrix B admits a permutation π 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* π 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 $\pi^{-1}(k) > \pi^{-1}(j)$. Similarly, π will also be called *compatible* with B. Such a permutation π 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

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

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 equation (6) above with (9) below).

Suppose Θ_0 is a fixed positive definite matrix and $\pi \in \mathcal{P}$. Then the matrix $P_\pi \Theta_0$ represents the same covariance structure as Θ_0 up to a reordering of the variables. We may use the Cholesky decomposition to write $P_\pi \Theta_0$ uniquely as

$$P_\pi \Theta_0 = (I - L)D^{-1}(I - L)^T = \Theta(L, D), \tag{9}$$

where L is strictly lower triangular and D is diagonal. It follows from Lemma 8 in the Appendix that $P_\pi \Theta(L, D) = \Theta(P_\pi L, P_\pi D)$ for any π, so we can rewrite (9) as

$$\Theta_0 = \Theta(P_{\pi^{-1}} L, P_{\pi^{-1}} D).$$

For each π, define

$$\tilde{B}_0(\pi) := P_{\pi^{-1}} L,$$
$$\tilde{\Omega}_0(\pi) := P_{\pi^{-1}} D.$$

By (6), this gives us the unique decomposition of Θ_0 into a DAG $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ that is compatible with the permutation π. The DAGs $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ that are compatible with some permutation π define a subset of the equivalence class $\mathcal{E}(\Theta_0)$; it is easy to check that in fact, this subset is the entire equivalence class.

Lemma 1. *Suppose Σ_0 is a positive definite covariance matrix and let $\Theta_0 := \Sigma_0^{-1}$. Then*

$$\begin{aligned}\mathcal{E}(\Theta_0) &= \{(P_{\pi^{-1}} L, P_{\pi^{-1}} D) : P_\pi \Theta_0 = \Theta(L, D),\ \pi \in \mathcal{P}\} \\ &= \{(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi)) : \pi \in \mathcal{P}\}.\end{aligned}$$

Note that the relationship between DAGs and permutations is not bijective: multiple permutations can lead to the same DAG. For example, the trivial DAG with no edges is compatible with all possible permutations.

The question now arises: which DAG $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ 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 $\mathcal{E}(\Theta_0)$ are statistically indistinguishable based on observational data alone, so a natural objective is to estimate the DAG that most parsimoniously represents the parameter Θ_0 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 π 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 Θ_0, this is exactly the same problem as finding a permutation π such that $\tilde{B}_0(\pi)$ has the fewest number of edges. This connection has been studied in more detail by Raskutti and Uhler (2014). They show that in this oracle setting, there is an equivalence between ℓ_0-penalized estimation and

sparse Cholesky factorization. In contrast, here we seek to estimate Θ_0 *as well as* find a sparse permutation π, and in this sense we provide a non-oracular, computationally feasible alternative to searching across all $p!$ permutations when p is large.

Example 1. 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

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

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

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

Since $\Theta(B_1, \Omega_1) = \Theta(B_0, \Omega_0)$, the DAG (B_1, Ω_1) is equivalent to (B_0, Ω_0). Thus, according to Lemma 1, there must be a permutation π_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 (9)) that these identities hold. Furthermore, if we reverse the order of the variables in π_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 Θ_0, we wish to find one which has the fewest number of edges.

2.4 Structural Equation Modeling

We have chosen to focus on the problem of structure estimation of Bayesian networks, which is not to be confused with the problem of causal inference. We view the data-generation mechanism as a multivariate Gaussian distribution as in (1). From this perspective, there are many linear structural equations (2) that may generate (1). Our focus is on finding the most parsimonious representation of the true distribution as a set of structural equations.

Alternatively, one could view the structural equation model (2) 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.

3. The Concave Penalization Framework

Now that the necessary preliminaries have been discussed, in the remainder of the paper we will develop the estimation framework thus far described at a high-level. Our approach is to use a penalized maximum likelihood estimator to estimate a sparse DAG B_0 that represents Θ_0. Recall that the negative log-likelihood is given by $L(B, \Omega \,|\, X)$ in (4). This will be our loss function, however in order to promote sparsity and avoid overfitting, we will minimize a penalized loss instead. In what follows, let $p_\lambda : [0, \infty) \to \mathbb{R}$ be a nonnegative and nondecreasing penalty function that depends on the tuning parameter λ and possibly one or more additional shape parameters. Our framework is valid for a general class of penalties, so in what follows we will allow $p_\lambda(\cdot)$ to be arbitrary. The details of choosing the penalty function will be discussed in Section 3.3.

Once p_λ is chosen, one may seek to find a solution to

$$\arg\min_{B,\Omega} \big\{ L(B, \Omega \,|\, X) + n \sum_{i,j} p_\lambda(|\beta_{ij}|) : B \text{ is a DAG} \big\}. \tag{10}$$

When L is taken to be a more general scoring function such as a posterior probability, (10) resembles most familiar score-based methods. When $p_\lambda(\cdot)$ is taken to be the ℓ_0 penalty, we recover the estimator discussed in van de Geer and Bühlmann (2013). Our approach differs from the aforementioned in two ways:

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

2. Due to the nonconvexity of the loss function, we reparameterize the problem in order to obtain a convex loss function.

Thus, in general our estimator will not be the same as (10).

Remark 3. If we further constrain the minimization problem in (10) to include only DAGs which are compatible with a fixed topological sort, we can reduce the problem to a series of p individual regression problems. Given a topological sort $\prec$, the parents of X_j must be a subset of the variables that precede X_j in $\prec$. In terms of the permutation π described in Section 2.3, we require $\Pi_j^0 \subset \{X_k : \pi^{-1}(k) > \pi^{-1}(j)\}$. The true neighbourhood of X_j can then be determined by projecting X_j onto this subset of nodes, which can be done via penalized least squares. Consistency in structure learning and parameter estimation can then be established through standard penalized regression theory.

3.1 Reparameterization

One of the drawbacks of the loss in (4) is that it is nonconvex, which complicates the minimization of the penalized loss. If we minimize (4) with respect to Ω and use the adaptive Lasso penalty, we obtain the estimator described in Fu and Zhou (2013). By keeping the p variance terms, however, we can exploit a clever reparameterization 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 reparameterized negative log-likelihood

$$L(\Phi, R \,|\, X) = \sum_{j=1}^{p} \left[-n \log(\rho_j) + \frac{1}{2} \|\rho_j x_j - X\phi_j\|^2 \right], \tag{11}$$

where $\Phi = [\phi_1 \,|\, \cdots \,|\, \phi_p]$ and $R = \operatorname{diag}(\rho_1, \ldots, \rho_p)$. The loss function in (11) is easily seen to be convex. Furthermore, if we interpret Φ as the adjacency matrix of a directed graph, then Φ has exactly the same edges and nonzero entries as B, and thus in particular Φ is acyclic if and only if B is acyclic.

In analogy with the parameterization (B, Ω), define

$$\Theta(\Phi, R) = (R - \Phi)(R - \Phi)^T, \tag{12}$$

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

This reparameterization is *not* the same as the likelihood in (7), which is well-known to lead to a convex program (see, for instance, Boyd and Vandenberghe, 2009, §7.1). Indeed, plugging (6) into (7) leads back to (4), which is nonconvex in the parameters β_{ij} and ω_j. To wit, the problem is convex in Θ but not in (B, Ω). The key insight from Städler et al. (2010) is to observe that one may recover convexity by switching to the alternate parameterization in terms of ϕ_{ij} and ρ_j. Unfortunately, the DAG constraint in (10) is still nonconvex. The idea behind this reparameterization 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.

3.2 The Estimator

We are now prepared to introduce the formal definition of the DAG estimator which is the focus of this work.

Fix a penalty function $p_\lambda(\cdot)$. Then given

$$\begin{aligned} (\widehat{\Phi}, \widehat{R}) &:= \operatorname*{arg\,min}_{\Phi, R} \big\{ L(\Phi, R \,|\, X) + n \sum_{i,j} p_\lambda(|\phi_{ij}|) : \Phi \text{ is a DAG} \big\} \\ &= \operatorname*{arg\,min}_{\Phi, R} \big\{ \sum_{j=1}^{p} \left[-n \log(\rho_j) + \frac{1}{2} \|\rho_j x_j - X\phi_j\|^2 \right] \\ &\qquad\qquad + n \sum_{i,j} p_\lambda(|\phi_{ij}|) : \Phi \text{ is a DAG} \big\}, \end{aligned} \tag{13}$$

we define our estimator to be

$$(\widehat{B}, \widehat{\Omega}) = \begin{cases} \hat{\beta}_{ij} = \hat{\phi}_{ij}/\hat{\rho}_j, & i \neq j \\ \hat{\beta}_{jj} = 0, & \\ \hat{\omega}_j^2 = 1/\hat{\rho}_j^2, & j = 1, \ldots, p \end{cases} \tag{14}$$

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 λ, we shall denote it by $(\widehat{\Phi}(\lambda), \widehat{R}(\lambda))$.

There is an intuitive interpretation of the problem in (13): By the identity $L(\Phi, R \,|\, X) = L(\Theta(\Phi, R) \,|\, X)$, it is evident that the loss function for (Φ, R) is simply the negative log-likelihood of the resulting estimate of $\Theta = \Theta(\Phi, R)$. In this sense, we are implicitly approximating the true parameter Θ_0. The key ingredient, however, is the penalty term: We only penalize the edge weights ϕ_{ij}, which has the effect of self-selecting for DAGs which are sparse. In this way, the solution to (13) produces a sparse Bayesian network whose distribution is close to the true, underlying distribution.

Remark 4. For most choices of the penalty, the solution to (13) is *not* the same as the solution to (10) since we are penalizing different terms. In the original parameterization, we penalize the coefficients β_{ij}, whereas after reparameterizing 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 ω_j. A notable exception, however, occurs when $p_\lambda(\cdot)$ is taken to be the ℓ_0 penalty. In this special case, the problems in (10) and (13) are the same, and thus in particular the analysis in van de Geer and Bühlmann (2013) applies.

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 ℓ_1-based methods have been slowly gaining in popularity. Traditionally, ℓ_1 regularization is viewed as a convex relaxation of optimal ℓ_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 ℓ_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) $p'_\lambda(t) = 0$ for large t;
2. (Sparsity) The minimum of $t + p'_\lambda(t)$ is positive;
3. (Continuity) The minimum of $t + p'_\lambda(t)$ 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 p_λ 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 Section 4 will illuminate how the properties of the penalty function influence the theoretical properties of the estimator (13, 14), 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

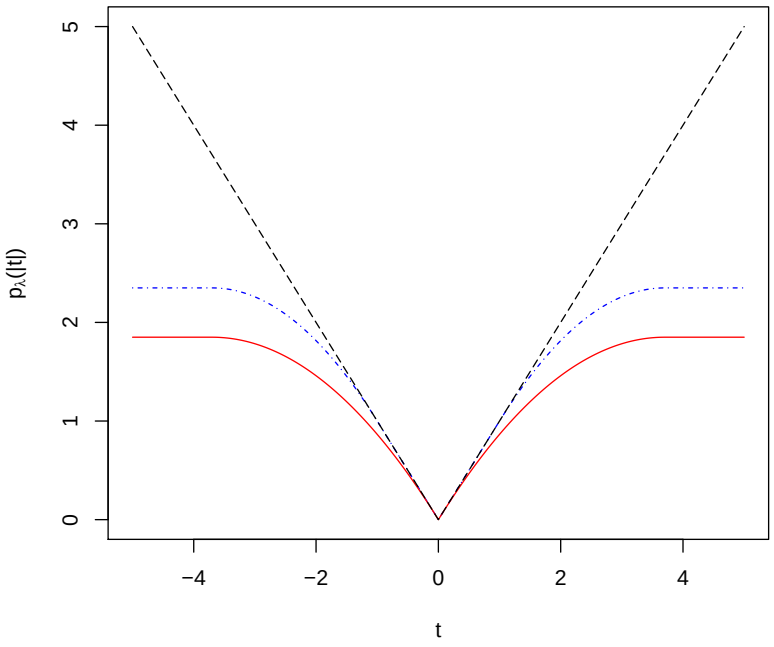

Figure 1: Comparison of penalty functions. The solid red line is the minimax concave penalty (MCP), the dot-dashed blue line is the smoothly clipped absolute deviation penalty (SCAD), and the dashed black line is the ℓ_1 or Lasso penalty. Both the MCP and SCAD represent smooth interpolations of the ℓ_1 and ℓ_0 penalties and hence have better statistical properties, whereas the ℓ_1 penalty exhibits bias due to its divergence as $t \to \infty$.

the single parameter updates (see Section 5.2 for details). Examples of common penalty functions in the literature include ℓ_1 (or Lasso, Tibshirani, 1996), SCAD (Fan and Li, 2001) and MCP (Zhang, 2010). The SCAD penalty represents a smooth quadratic interpolation between the ℓ_1 and ℓ_0 penalties, and the MCP translates the ℓ_1 portion of the SCAD to the origin. See Figure 1 for a visual comparison of these three penalties. The key difference between the ℓ_1 penalty and SCAD or MCP is the flat part of the penalty, which helps to reduce bias.

In our computations we chose to use the MCP, defined for $t \geq 0$ by

$$p_\lambda(t;\gamma) := \lambda\left(t - \frac{t^2}{2\lambda\gamma}\right)1(t < \lambda\gamma) + \frac{\lambda^2\gamma}{2}1(t \geq \lambda\gamma) \tag{15}$$
$$= \begin{cases} \lambda\left(t - \frac{t^2}{2\lambda\gamma}\right), & t < \lambda\gamma, \\ \frac{\lambda^2\gamma}{2}, & t \geq \lambda\gamma. \end{cases}$$

The γ parameter in the MCP controls the concavity of the penalty: As $\gamma \to 0$, MCP approaches the ℓ_0 penalty and as $\gamma \to \infty$, it approaches the ℓ_1 penalty. In the sequel we will thus refer to γ as the *concavity parameter* and λ as the *regularization parameter*. From the above formula, MCP is easily seen to be a quadratic spline between the origin and the

ℓ_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 ℓ_1 penalty, $p_\lambda(|t|) = \lambda|t|$.

As the ℓ_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 *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 ℓ_1 penalties, the assumptions on the data needed for consistency can be relaxed substantially. Generalizing these ideas to Bayesian network models, we will show in Section 4 how our estimator is consistent in both parameter estimation and structure learning when concave regularization is used; with ℓ_1 regularization we only obtain parameter estimation consistency. These theoretical results are supported by the comparisons in Section 6.

3.4 The Role of Sparsity

For a given Θ_0, the equivalence class $\mathcal{E}(\Theta_0)$ will typically consist of graphs with different numbers of edges, and in general there need not be a sparse representation $(\tilde{B}_0(\pi), \tilde{\Omega}(\pi))$ with $s_{\tilde{B}_0(\pi)} := \tilde{s}_0(\pi) \ll p^2$. Moreover, the asymptotic theory to be developed in Section 4 will not require such an assumption. When we evaluate our method in Sections 5-7, however, we will focus our attention on the case where there exists a DAG in $\mathcal{E}(\Theta_0)$ which is sparse, that is, satisfying the condition $\tilde{s}_0(\pi) = O(p)$.

Our justification for this assumption is both practical and theoretical. In terms of the true graph, sparsity implies that we expect either (a) only a subset of the variables are truly involved, or (b) on average, each variable has only a few parents. In case (a), estimating a Bayesian network is similar to the variable screening problem. Both of these scenarios are commonly encountered in practice, as many realistic DAG models tend to be sparse in one of these two senses. Moreover, for data sets with p very large, we typically have fewer observations than variables. In fact, we expect $p \gg n$, with p on the order of thousands or tens of thousands. When this happens, we can only expect to obtain reasonable results when each node has at most n parents, although in practice far fewer than n parents is typical. For these reasons, we chose to tailor our algorithm to the sparse, high-dimensional regime. Along with the nonconvexity of the constraint space, this is the main reason for emphasizing the use of concave penalties, whose superior performance in the $p \gg n$ regime has been already established for regression models. Furthermore, by assuming that the true graph is sparse, we can take advantage of several computational enhancements that allow our algorithm to leverage sparsity for speed. The result is an efficient algorithm when we are confident that the underlying model admits a sparse representation.

4. Asymptotic Theory

In this section we provide theoretical justification for the use of the estimator (13, 14) in the finite-dimensional regime. That is, we will assume p is fixed and let $n \to \infty$. The purpose of this section is not to provide novel theoretical insights, but rather simply to

show that under the right conditions we can always guarantee that the estimator defined in the previous section has good estimation properties. Most importantly, we establish that these conditions can always be satisfied when the MCP is used for regularization.

In the statistics literature, a procedure which attains consistency in structure learning with high probability is sometimes referred to as *model selection consistent*. This can be confusing as model selection is also used to refer to the problem of selecting the tuning parameter λ. In the sequel, we use the following conventions: (i) A procedure is *structure estimation consistent* if $P(\text{supp}(\widehat{B}) = \text{supp}(B_0)) \to 1$, (ii) A procedure is *parameter estimation consistent* if $\|\widehat{B} - B_0\|_F \xrightarrow{P} 0$, and (iii) *Model selection* will refer only to the problem of choosing λ.

4.1 Nonidentifiability and Sparsity

Since our optimization problem is nonconvex, we must be careful when discussing "solutions" to (13). The estimator is defined to be the global minimum of the penalized loss, but theoretical guarantees are generally only available for local minimizers. Our theory is no exception, and it is furthermore complicated by identifiability issues: Based on observational data alone, the inverse covariance matrix Θ_0 is identifiable, but the DAG (B_0, Ω_0) 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 $\mathcal{E}_0$ (Lemma 1). 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 $\mathcal{E}_0$ 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 in Section 6 give us good empirical evidence that our estimator indeed approximates the sparsest DAG representation of Θ_0, 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 (13) 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 consis-

tent local maximizers before stating our main result in Section 4.3. We also briefly discuss the high-dimensional scenario in which p is allowed to depend on n.

Remark 5. 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; Anandkumar et al., 2013). In contrast to these developments, the main technical difficulty in our analysis is the nonidentifiability of the general Gaussian model.

4.2 Existence of Local Maximizers

In the ensuing theoretical analysis, it will be easier to work with a single parameter vector (vs. the two matrices Φ and R), so we first transform our parameter space in this way without any loss of generality. To the end, define $U := R + \Phi$ and let $\boldsymbol{\nu} = \mathrm{vec}(U) = \mathrm{vec}(R + \Phi) \in \mathbb{R}^{p^2}$ be the vectorized copy of U in $\mathbb{R}^{p^2}$. Our parameter space is then the subset $\mathcal{D}$ of $\mathbb{R}^{p^2}$ such that $\boldsymbol{\nu} \in \mathcal{D}$ implies (Φ, R) is a DAG, where $\boldsymbol{\nu} = \mathrm{vec}(R + \Phi)$. In the sequel, we will refer to such a $\boldsymbol{\nu}$ as a DAG. For a more in-depth treatment of the abstract framework, see Section A.1 in the Appendix.

The true distribution is uniquely defined by its inverse covariance matrix, Θ_0. By equation (12), given $(\widehat{\Phi}, \widehat{R})$ we may consider the resulting estimate of the inverse covariance matrix $\widehat{\Theta} = \Theta(\widehat{\Phi}, \widehat{R})$. By analogy, for any DAG $\boldsymbol{\nu} \in \mathbb{R}^{p^2}$, we may define in the obvious way the matrix $\Theta(\boldsymbol{\nu})$. Thus the parameter $\boldsymbol{\nu}$ is simply another parameterization of the normal distribution: For any Θ_0, there exists $\boldsymbol{\nu} \in \mathcal{D}$ such that $\Theta_0 = \Theta(\boldsymbol{\nu})$. Let $\mathcal{E}_0 = \mathcal{E}(\Theta_0) = \{\boldsymbol{\nu} \in \mathbb{R}^{p^2} : \Theta(\boldsymbol{\nu}) = \Theta_0\}$. We will denote an arbitrary element of $\mathcal{E}_0$ by $\boldsymbol{\nu}_0$ and a minimal-edge DAG in $\mathcal{E}_0$ by $\boldsymbol{\nu}^*$.

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

$$p_\lambda(\boldsymbol{\nu}) = \sum_{i \neq j} p_\lambda(|u_{ij}|), \tag{16}$$

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 Φ. Now let

$$F(\boldsymbol{\nu}) := \ell_n(\boldsymbol{\nu} \,|\, X) - n\, p_{\lambda_n}(\boldsymbol{\nu}). \tag{17}$$

We are interested in maximizing F over $\mathcal{D}$.

For any $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ 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 p_λ:

$$a_n(\boldsymbol{\nu}_0) := \max\{|p'_{\lambda_n}(|\phi^0_{ij}|)| : \phi^0_{ij} \neq 0\}, \tag{18}$$

$$b_n(\boldsymbol{\nu}_0) := \max\{|p''_{\lambda_n}(|\phi^0_{ij}|)| : \phi^0_{ij} \neq 0\}. \tag{19}$$

When it is clear from context, the dependence of a_n and b_n on $\boldsymbol{\nu}_0$ will be suppressed. Finally, let $\tau(\lambda) := \sup_t p_\lambda(t)$, which may be infinite. For the MCP we have $\tau(\lambda) = \lambda^2\gamma/2$ and for the ℓ_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 2. *Fix $p \geq 1$. If there exists $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ with $b_n(\boldsymbol{\nu}_0) \to 0$, then there is a local maximizer $\widehat{\boldsymbol{\nu}}_n$ of $F(\boldsymbol{\nu})$ such that*

$$\|\widehat{\boldsymbol{\nu}}_n - \boldsymbol{\nu}_0\| = O_P(n^{-1/2} + a_n(\boldsymbol{\nu}_0)).$$

When $a_n = O(n^{-1/2})$, we obtain a $n^{1/2}$-consistent estimator of $\boldsymbol{\nu}_0$. Note that by Lemma 1, if $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ then $\boldsymbol{\nu}_0 = (\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ for some permutation π. For this reason, in the sequel we shall refer to the local maximizer $\widehat{\boldsymbol{\nu}}_n$ as the *π-local maximizer* of F for the permutation π. This theorem says that as long as the curvature of the penalty at $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ tends to zero, the penalized likelihood has a π-local maximizer that converges to $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ 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 3. *Assume that the penalty function satisfies*

$$\liminf_{n\to\infty} \liminf_{t\to 0^+} p'_{\lambda_n}(t)/\lambda_n > 0. \tag{20}$$

Assume further that $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ satisfies $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$, $b_n(\boldsymbol{\nu}_0) \to 0$, and let $\widehat{\boldsymbol{\nu}}_n$ be a π-local maximizer from Theorem 2. If $\lambda_n \to 0$ and $\lambda_n n^{1/2} \to \infty$, then

$$P(\operatorname{supp}(\widehat{\boldsymbol{\nu}}_n) = \operatorname{supp}(\boldsymbol{\nu}_0)) \to 1. \tag{21}$$

In fact, this follows immediately from Theorem 2 above and Theorem 2 in Fan and Li (2001). An obvious corollary is that $P(\hat{s}_n = s_0) \to 1$.

We must be careful in interpreting these theorems correctly: They do not imply necessarily that the estimator defined in (13, 14) 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 6. If we assume that the conditions of Theorems 2 and 3 hold for *all* $\boldsymbol{\nu}_0 \in \mathcal{E}_0$, then we can conclude that every equivalent DAG has a π-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 $\boldsymbol{\nu}_0$ 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 4. *If $B_1 \neq B_2$ are DAGs that have a common topological sort, then for any choices of Ω_1 and Ω_2, we have $\Theta(B_1, \Omega_1) \neq \Theta(B_2, \Omega_2)$. A similar result holds in the parameterization (Φ, R).*

The assumption that two DAGs have a common topological sort is equivalent to each DAG being compatible with the same permutation π. The following lemma shows that the $\boldsymbol{\nu}_0$ are isolated, which guarantees that π-local maximizers do not cluster around multiple $\boldsymbol{\nu}_0$. For any $\varepsilon > 0$, we denote the ε-neighbourhood of $\boldsymbol{\nu}_0$ in $\mathcal{D}$ by $B(\boldsymbol{\nu}_0, \varepsilon) := \{\boldsymbol{\nu} \in \mathcal{D} : \|\boldsymbol{\nu} - \boldsymbol{\nu}_0\| < \varepsilon\}$.

Lemma 5. *For any positive definite Θ_0 there exists $\varepsilon > 0$ such that $\mathcal{E}_0 \cap B(\boldsymbol{\nu}_0, \varepsilon) = \{\boldsymbol{\nu}_0\}$ for any $\boldsymbol{\nu}_0 \in \mathcal{E}_0$.*

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

4.3 The Main Result

We will now significantly strengthen Theorems 2 and 3 by showing that, under a concave penalty, a sparsest DAG $\boldsymbol{\nu}^* \in \mathcal{E}_0$ maximizes the penalized likelihood amongst all the possible equivalent representations of the covariance matrix Θ_0. Under the assumptions of Theorem 2, there is a π-local maximizer $\widehat{\boldsymbol{\nu}}_n^*$ of $F(\boldsymbol{\nu})$ such that $\|\widehat{\boldsymbol{\nu}}_n^* - \boldsymbol{\nu}^*\| = O_P(n^{-1/2} + a_n(\boldsymbol{\nu}^*))$. Ideally, when $\boldsymbol{\nu}_0$ has more edges than $\boldsymbol{\nu}^*$, we would like these π-local maximizers to satisfy $F(\widehat{\boldsymbol{\nu}}_n^*) > F(\widehat{\boldsymbol{\nu}}_n)$ with high probability.

Intuitively, when $a_n(\boldsymbol{\nu}_0) = b_n(\boldsymbol{\nu}_0) = 0$, all of the nonzero coefficients lie in the flat part of the penalty where $p'_{\lambda_n}(|\phi_{ij}^0|) = p''_{\lambda_n}(|\phi_{ij}^0|) = 0$. When this happens, the penalty "acts" like the ℓ_0 penalty by penalizing all of the coefficients equally by the amount $\tau(\lambda_n)$, and any DAG with more edges than $\boldsymbol{\nu}^*$ will see a heavier penalty. In order to quantify "how close" $\boldsymbol{\nu}_0$ is to lying in the flat part of the penalty, we define

$$c_n(\boldsymbol{\nu}_0) := \min\{p_{\lambda_n}(|\phi_{ij}^0|) : \phi_{ij}^0 \neq 0\}.$$

When $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n)$, the penalty mimics the ℓ_0 penalty, and since the likelihood $\ell_n(\boldsymbol{\nu}_0 \,|\, X)$ is constant for all $\boldsymbol{\nu}_0$, we would then have

$$p_{\lambda_n}(\boldsymbol{\nu}^*) < p_{\lambda_n}(\boldsymbol{\nu}_0) \iff \ell_n(\boldsymbol{\nu}^* \,|\, X) - n\, p_{\lambda_n}(\boldsymbol{\nu}^*) > \ell_n(\boldsymbol{\nu}_0 \,|\, X) - n\, p_{\lambda_n}(\boldsymbol{\nu}_0).$$

One would hope that for local maximizers $\widehat{\boldsymbol{\nu}}_n$ that are sufficiently close to the $\boldsymbol{\nu}_0$, 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 6. *Suppose that $p_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $p_\lambda(0) = 0$. Assume further that the conditions for Theorem 3 hold for all $\boldsymbol{\nu}_0 \in \mathcal{E}_0$. Recall that $\tau(\lambda_n) := \sup_t p_{\lambda_n}(t)$. If*

1. *$c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n) + O(n^{-1/2})$ for all $\boldsymbol{\nu}_0 \in \mathcal{E}_0$,*
2. *$\limsup_n \tau(\lambda_n) < \infty$,*
3. *$\tau(\lambda_n) n^{1/2} \to \infty$,*

then for any DAG $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ with strictly more edges than $\boldsymbol{\nu}^*$, $P(F(\widehat{\boldsymbol{\nu}}_n^*) > F(\widehat{\boldsymbol{\nu}}_n)) \to 1$ as $n \to \infty$.

The restriction to $\boldsymbol{\nu}_0$ with strictly more edges than $\boldsymbol{\nu}^*$ is necessary since $\boldsymbol{\nu}^*$ may not be unique in general. Theorem 6 essentially answers the question of which DAG in the true equivalence class $\mathcal{E}_0$ our estimator approximates. As we have discussed, there is a subtle technicality in which it is possible that there are *other* maximizers of $F(\boldsymbol{\nu})$ besides the π-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 2, 3, and 6 uniformly, we can combine all of the results in order to characterize the behaviour of the estimates in terms of the parameterization $(\widehat{B}, \widehat{\Omega})$ given by (14). Before stating the main theorem, we will need some notation to distinguish π-local maximizers. Assuming the conditions of Theorem 2 hold for all π, denote the collection of π-local maximizers by $\mathcal{M}_n$. Continuing our notation from the previous section, we also let (B^*, Ω^*) denote any graph in $\mathcal{E}_0$ with the fewest number of edges, and let $(\widehat{B}^*, \widehat{\Omega}^*)$ be the corresponding π-local maximizer. Recall that given a DAG estimate $(\widehat{B}, \widehat{\Omega})$, we define $\widehat{\Theta} = \Theta(\widehat{B}, \widehat{\Omega})$.

Theorem 7. *Suppose that $p_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $p_\lambda(0) = 0$. Fix $p \geq 1$ and assume that the penalty function satisfies*

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

Assume further that $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$, $b_n(\boldsymbol{\nu}_0) \to 0$, and $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n) + O(n^{-1/2})$ for each DAG in $\mathcal{E}_0$. 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 π, 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. $P(\operatorname{supp}(\widehat{B}) = \operatorname{supp}(\tilde{B}_0(\pi))) \to 1$,
3. $\|\widehat{\Theta} - \Theta_0\|_F = O_P(n^{-1/2})$.

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 7 is immediate from the properties of the Frobenius norm and Theorems 2, 3, and 6.

Remark 7. Using an adaptive ℓ_1 penalty, Fu and Zhou (2013) first obtained results similar to Theorems 2 and 3. 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 (8) (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 6.

4.4 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 (13). We pause now to discuss these conditions more carefully, and show that they can always be satisfied.

The parameters $a_n(\boldsymbol{\nu}_0)$ and $b_n(\boldsymbol{\nu}_0)$ 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 2 and 3 highlight the advantages of concave regularization over ℓ_1 regularization. In particular, the ℓ_1 penalty trivially satisfies $b_n \to 0$, but cannot simultaneously satisfy $a_n(\boldsymbol{\nu}_0) = \lambda_n = O(n^{-1/2})$ and $\lambda_n n^{1/2} \to \infty$. Thus, for the ℓ_1 penalty, we may apply Theorem 2 to obtain a local maximizer which is consistent in *parameter estimation*, but we cannot guarantee structure estimation consistency through Theorem 3. In contrast, these conditions are easily satisfied by a concave penalty; in particular they are satisfied when p_λ is the MCP. These observations were first made in Fan and Li (2001).

The conditions on $\tau(\lambda_n)$ in Theorem 6 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 $\boldsymbol{\nu}_0 \in \mathcal{E}_0$. 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^*, Ω^*) emerges as the best estimate of the true graph. Since $\tau(\lambda_n) = +\infty$ for any convex penalty, Theorem 6 along with the remainder of this discussion do not apply to ℓ_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 $c_n(\boldsymbol{\nu}_0)$. By rate of convergence, we refer to the fact that the assumptions on $a_n(\boldsymbol{\nu}_0)$ and $b_n(\boldsymbol{\nu}_0)$ alone require that $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n) + o(1)$, or equivalently $p_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n) + o(1)$ whenever $\phi^0_{ij} \neq 0$. The stronger assumption that $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n) + O(n^{-1/2})$ in Theorem 6 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 p_{λ_n}, in contrast to the first- and second-order convergence required by Theorems 2 and 3. In practice, it is sufficient to have $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n)$ 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 λ_n. When the penalty is chosen to be the MCP, all of the conditions required for Theorem 7 are satisfied as long as

$$\limsup_n \lambda_n \gamma_n < \min_{\boldsymbol{\nu}_0 \in \mathcal{E}_0} \min\{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\} \quad \text{and} \quad \lambda_n = O(n^{-\alpha}),\ 0 < \alpha < 1/2. \tag{22}$$

Remark 8. To better understand the conditions on $\tau(\lambda_n)$ in Theorems 6 and 7, it is instructive to consider the simplified case in which the penalty factors as $p_{\lambda_n}(t) = \lambda_n \rho(t)$ for some function $\rho(t)$ (not to be confused with the parameters ρ_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 6 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 3. Thus, the extra conditions on $\tau(\lambda_n)$ in Theorems 6 and 7 are redundant when the penalty factors in this way.

Example 2. Although the usual formula for the MCP does not satisfy the factorization property in Remark 8, we may reparameterize 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 (15). Furthermore, the condition in (22) becomes

$$\limsup_n \delta_n < \min_{\boldsymbol{\nu}_0 \in \mathcal{E}_0} \min\{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\},$$

which is independent of λ_n.

4.5 Score-Based Theory in High-Dimensions

The theory in this section so far has assumed that p is fixed with $n > p$, the classical low-dimensional scenario. It would be interesting to obtain results for this method when p is allowed to depend on n, and in particular the case when $p > n$. While the simulations in Section 6 give good empirical evidence that our method is applicable to this scenario, formal theoretical results are not available yet. Here we take a moment to discuss some current work in this direction.

If we fix a permutation π, we have already described in Remark 3 how to modify our method in order to estimate the equivalent DAG that is compatible with π, which we have denoted by $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$. When the order of the variables is fixed, the problem reduces to standard multiple regression with a concave penalty, in which case Theorems 2 and 3 can be generalized to high-dimensions, for instance using the results in Fan and Lv (2010). This is in the spirit of similar results in the ℓ_1 case obtained by Shojaie and Michailidis (2010). Of course, in our set-up, we do not know in advance which permutation is optimal, so this does not tell the whole story. Theorem 6 shows how our estimator selects the right permutation automatically based on the data, and eliminates the need to assume this prior knowledge.

Recently, van de Geer and Bühlmann (2013) obtained some positive results using ℓ_0 regularization in which it is not assumed that π is known in advance. Under the same Gaussian framework we have adopted in this work, they show the following: When $p_\lambda(t) = \lambda^2 1(t \neq 0)$ and under certain strong regularity conditions, any *global minimizer* of (10) satisfies

$$\|\widehat{B} - \tilde{B}_0(\hat{\pi})\|_F^2 + \|\widehat{\Omega} - \tilde{\Omega}_0(\hat{\pi})\|_F^2 = O_P(\lambda^2 s_0), \tag{23}$$

where $\hat{\pi}$ is the permutation compatible with $(\widehat{B}, \widehat{\Omega})$. Furthermore, they establish that the estimated number of edges are all of the same order: $\hat{s} = O_P(\tilde{s}_0(\hat{\pi})) = O_P(s_0)$. These results represent the first significant analysis of score-based structure learning in high-dimensions that we know of, however, they have some drawbacks. First, they do not guarantee structure estimation consistency, and instead only give an upper bound on the number of estimated edges, which is to be of the same order as a minimal-edge DAG. With respect to computations, these results only hold for the intractable ℓ_0 penalty, and no suggestions are made to allow computation of this estimator in practice. Furthermore, since the optimization problem is nonconvex, theoretical guarantees for global minimizers are less practical than guarantees for local minimizers. We have already observed (Remark 4) that the estimator defined in van de Geer and Bühlmann (2013) is a special case of (13), and so this theory applies to our framework under ℓ_0 regularization.

A common interpretation of concave penalization is as a continuous relaxation of the discrete ℓ_0 penalty. Our framework can thus be seen in this light. Previous work has shown that penalized likelihood estimators can have near optimal performance when compared with the ℓ_0 estimator (Zhang and Zhang, 2012), and thus we have good reason to believe the same holds true for our estimator. The key idea from the analysis in van de Geer and Bühlmann (2013) is to control the behaviour of the estimates over all $p!$ possible permutations, which requires careful analysis using exponential-type concentration inequalities. Based on our preliminary work, we believe that such an analysis can be carried out for more general penalties, however, the details remain to be worked out and are expected to be technical.

Recently there has been some reported progress in high-dimensions for hybrid methods that consist of multiple learning stages. The general outline of these 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. In this light, Loh and Bühlmann (2013) consider using inverse covariance estimation to restrict the search space, and Bühlmann et al. (2014) convert the problem into three separate steps: preliminary neighborhood selection, order search, and maximum likelihood estimation. Since these ideas use multiple stages, they do not apply directly to the framework developed here.

5. Algorithm Details

Both the objective function and the constraint set in (13) are nonconvex, which makes traditional gradient descent algorithms for performing the necessary minimization inapplicable. One could employ naive gradient descent to find a local minimizer of (13), 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 reparameterization (11), however, we will be able to perform the single parameter updates and enforce the constraint with ruthless efficiency.

5.1 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 ϕ_{kj} and ϕ_{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 ϕ_{jk}. Alternatively, if the edge $X_j \to X_k$ induces a cycle, we set $\phi_{jk} = 0$ and minimize with respect to ϕ_{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

$$Q(\Phi, R) := L(\Phi, R) + \sum_{i,j} p_\lambda(|\phi_{ij}|) \tag{24}$$

to be our objective function for coordinate descent. Note that we have suppressed the dependence of the log-likelihood on the data 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 p_λ. This simply amounts to rescaling the regularization parameter λ, which causes no problems in computing $(\widehat{\Phi}, \widehat{R})$. Thus solving (13) is equivalent to minimizing Q.

Now define the single-variable functions

$$Q_1(\phi_{kj}) = \frac{1}{2}\left\|\rho_j x_j - \sum_{i=1}^{p} \phi_{ij} x_i\right\|^2 + p_\lambda(|\phi_{kj}|), \tag{25}$$

$$Q_2(\rho_j) = -n \log \rho_j + \frac{1}{2}\left\|\rho_j x_j - \sum_{i=1}^{p} \phi_{ij} x_i\right\|^2. \tag{26}$$

The function Q_1 is $Q(\Phi, R)$ in (24) considered as a function of the single parameter ϕ_{kj}, while holding the other $p^2 - 1$ variables fixed and ignoring terms that do not depend on ϕ_{kj},

Algorithm 1 CCDr Algorithm

Input: Initial estimates (Φ^0, R^0); penalty parameters (λ, γ); error tolerance $\varepsilon > 0$; maximum number of iterations M.

1. Cycle through ρ_j for $j = 1, \ldots, p$, minimizing Q_2 with respect to ρ_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 ϕ_{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 ϕ_{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 (14)) and output these values.

and Q_2 is the corresponding function for the parameter ρ_j. We express the dependence of Q_1 and Q_2 on k and/or j implicitly through their respective argument, ϕ_{kj} or ρ_j.

An overview of the algorithm is given in Algorithm 1. We use the notation $\phi_{kj} \Leftarrow 0$ to mean that ϕ_{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 reparameterization (CCDr).

5.2 Coordinate Descent

In what follows, we assume that the data have been appropriately normalized so that each column x_j has unit norm, $\|x_j\|^2 = \sum_h x^2_{hj} = 1$. Furthermore, although the details of the algorithm do not depend on the choice of penalty, we will focus on the MCP and ℓ_1 penalties, as these are the methods implemented and discussed in Sections 6 and 7.

5.2.1 Update for ϕ_{kj}

Mazumder et al. (2011) show that the minimum of (25) can be found by solving

$$\arg\min_{\beta} Q^1(\beta), \quad \text{where } Q^1(\beta) := \frac{1}{2}(\beta - \tilde{\beta})^2 + p_\lambda(|\beta|). \tag{27}$$

The solution to (27) 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

$$S_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \leq \lambda, \\ \operatorname{sgn}(\tilde{\beta}) \left(\frac{|\tilde{\beta}| - \lambda}{1 - 1/\gamma} \right), & \lambda < |\tilde{\beta}| \leq \lambda\gamma, \\ \tilde{\beta}, & |\tilde{\beta}| > \lambda\gamma. \end{cases} \tag{28}$$

For the ℓ_1 penalty, we have

$$S(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \leq \lambda, \\ \operatorname{sgn}(\tilde{\beta})(|\tilde{\beta}| - \lambda), & |\tilde{\beta}| > \lambda. \end{cases} \tag{29}$$

To see how to convert (25) into (27), note that

$$\begin{aligned} Q_1(\phi_{kj}) &= \frac{1}{2} \sum_{h=1}^n \left(\rho_j x_{hj} - \sum_{i \neq k} \phi_{ij} x_{hi} - \phi_{kj} x_{hk} \right)^2 + p_\lambda(|\phi_{kj}|) \qquad (30) \\ &= \frac{1}{2} \sum_{h=1}^n x_{hk}^2 \left(\frac{1}{x_{hk}} r_{kj}^{(h)} - \phi_{kj} \right)^2 + p_\lambda(|\phi_{kj}|), \end{aligned}$$

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 $\sum_h x_{hk}^2 = 1$,

$$\begin{aligned} Q_1(\phi_{kj}) &= \frac{1}{2} \left\{ \sum_{h=1}^n (r_{kj}^{(h)})^2 - 2\phi_{kj} \sum_{h=1}^n x_{hk} r_{kj}^{(h)} + \phi_{kj}^2 \right\} + p_\lambda(|\phi_{kj}|) \qquad (31) \\ &= \frac{1}{2} \left(\phi_{kj} - \sum_{h=1}^n x_{hk} r_{kj}^{(h)} \right)^2 + p_\lambda(|\phi_{kj}|) + \text{const.} \qquad (32) \end{aligned}$$

The constant term in (32) does not depend on ϕ_{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 (27) with $\tilde{\beta} = \sum_h x_{hk} r_{kj}^{(h)}$. Hence for MCP with $\gamma > 1$,

$$\arg\min Q_1(\phi_{kj}) = S_\gamma \left(\sum_h x_{hk} r_{kj}^{(h)}, \lambda \right), \tag{33}$$

and similarly for the ℓ_1 penalty. The existence of a closed-form solution to the single parameter update for ϕ_{kj} is a key ingredient to our method, and is one of the reasons we chose the MCP and ℓ_1 penalties in our comparisons. Many other penalty functions, however, allow for closed-form solutions to (27), and our algorithm applies for any such penalty function.

5.2.2 Update for ρ_k

The single parameter update for ρ_j is straightforward to compute and is given by

$$\arg\min Q_2(\rho_j) = \frac{c + \sqrt{c^2 + 4n}}{2}, \quad \text{with } c = \sum_{i \neq j} \phi_{ij} \sum_h x_{hi} x_{hj}. \tag{34}$$

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

5.3 Regularization Paths

In practice, it is difficult to select optimal choices of the penalty parameters (λ, γ) in advance. Thus it is necessary to compute several models at many discrete choices of (λ_i, γ_j), and then perform model selection. In testing, we observed a dependence on the concavity parameter γ, however, for simplicity we will consider γ fixed in the sequel, and postpone further study of the method's dependence on γ to future work.

The regularization parameter λ, 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 λ_0 so that $\widehat{\Phi}(\lambda_0) = \mathbf{0}$, with successive values of λ_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 model selection problem, and is beyond the scope of this paper. The present work should be considered a "proof of concept," showing that under the right conditions, there exists a λ 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 6.5. 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.

5.4 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 Φ 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 (33) can be written as

$$\sum_{h=1}^{n} x_{hk} r_{kj}^{(h)} = \rho_j \langle x_j, x_k \rangle - \sum_{i \neq k} \phi_{ij} \langle x_i, x_k \rangle. \tag{35}$$

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 (34), which highlights why the reparameterization (11) is useful: the single parameter update for each ρ_j only requires $O(p)$ operations, compared with $O(p^2)$ required operations for the standard residual estimate for ω_j^2 in the original parameterization. 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 (33) and (34) 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 λ-values and compute an estimate $(\widehat{\Phi}(\lambda_i), \widehat{R}(\lambda_i))$ for all $L+1$ choices of λ_i. In general, we do not know in advance what the smallest value of λ appropriate for the data is, and we typically choose λ_L as some small value. Since the model complexity (in terms of the number of edges) increases as λ decreases, more and more time is spent computing complex models for small λ. We can exploit these facts in order to avoid wasting time on computing unnecessarily complex models. As the algorithm proceeds calculating estimates for each λ_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 λ. 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 α should be application driven, we will use $\alpha = 3$ unless reported otherwise. In the sequel, α shall be referred to as the *threshold parameter*.

5.5 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

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

if one wishes to adjust the γ parameter in addition to λ.

Algorithm 2 Full CCDr Algorithm

Input: Initial estimates (Φ_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 $\|x_j\|^2 = 1$ and compute the inner products $\langle x_i, x_j \rangle$ for all $i, j = 1, \ldots, p$.
2. For each λ_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 (Φ_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 (14)) and output these values.

6. Numerical Simulations and Results

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 6.1.

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 ℓ_1 penalty (CCDr-ℓ_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 model selection and timing. We conclude this section with some detailed discussions about our experiments.

6.1 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 (2). 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 β^0_{ij} were chosen randomly and uniformly from the interval $[0.5, 2]$ and the error variances were fixed at $\omega^0_j = 1$ 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 λ, which controls the amount of regularization, and for PC and MMHC it is α, 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 α 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 γ in addition to λ. 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: ε, M, and α. Based on our simulations, ε 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 λ_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 9. 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.

6.2 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 minimum 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.

6.3 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 6.2.

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 model 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 model 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 6.5 we will discuss some interesting issues related to model selection.

6.3.1 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-ℓ_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 Table 1 and Figure 2. 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 (Table 1). 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 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-ℓ_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 6.3.3).

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 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,

$p = 50$, $T = 46.48$	CCDr-MCP	CCDr-ℓ_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**
$p = 100$, $T = 91.48$	CCDr-MCP	CCDr-ℓ_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**
$p = 200$, $T = 185.06$	CCDr-MCP	CCDr-ℓ_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 1: Average estimation performance of algorithms in low-dimensions.

which is further evidence that these methods are not necessarily overfitting. Rather, going back to (7), we see that the log-likelihood is a function of Θ 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 $\|\widehat{\Theta} - \Theta_0\|_F$ is small, i.e. $\widehat{\Theta}$ 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.

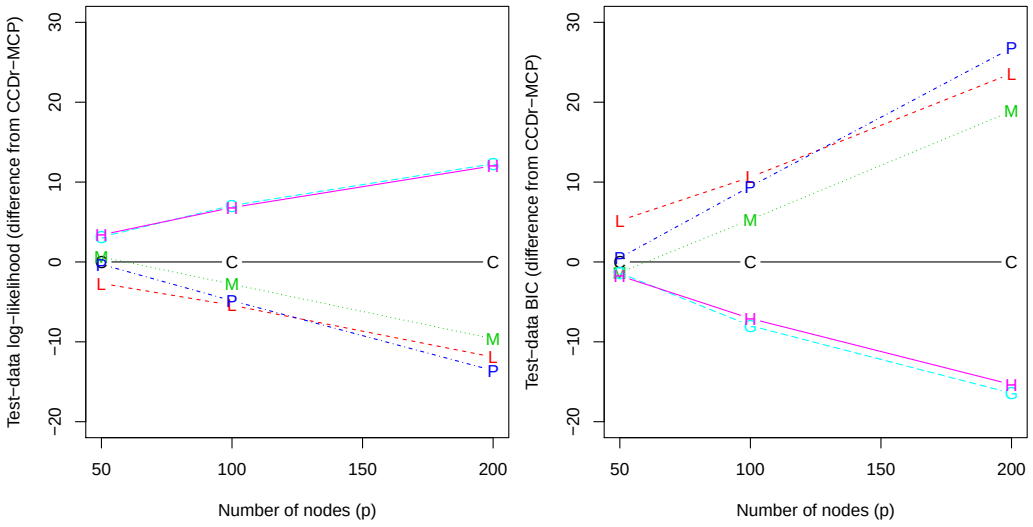

Figure 2: Comparison of test-data log-likelihood and BIC scores (low dimensions). The data are presented relative to the scores for CCDr-MCP. For log-likelihood, larger scores (positive values in the plot) are better; for BIC smaller scores (negative values) are better. (C = CCDr-MCP, L = CCDr-ℓ_1, G = GES, H = HC, M = MMHC, P = PC)

This explains why GES and HC perform the best on this metric: They do a good job of estimating Θ_0, 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). 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 6.5.

6.3.2 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-ℓ_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 Table 2. 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-ℓ_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 ℓ_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 Tables 1 and 2 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-ℓ_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-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 ℓ_1 regularization.

6.3.3 Timing Comparison

A comparison of the total and average runtimes for all the algorithms is provided by Figures 3 and 4. The results are displayed graphically here; detailed tables can be found in the Supplementary Materials (Tables S1 and S2).

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-ℓ_1, in which time both methods compute approximately 20 estimates. Amongst all the compared methods, the fastest alternative is the PC algorithm, however, the difference in timing is still roughly an order of magnitude: When $p = 200$, PC

$p = 100$, $T = 92.31$	CCDr-MCP	CCDr-ℓ_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**
$p = 200$, $T = 181.89$	CCDr-MCP	CCDr-ℓ_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**
$p = 500$, $T = 460.21$	CCDr-MCP	CCDr-ℓ_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: Average estimation performance of algorithms in high-dimensions.

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-ℓ_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 (see Tables S1 and S2 in the Supplementary Materials), which we suspect is due to how these algorithms scale with n: data sets with more samples require more time to process (see Section 6.6 for more details).

Combined with the improved performance in high-dimensions (Section 6.3.2), these results support our claim that CCDr is an improvement in both timing and accuracy over

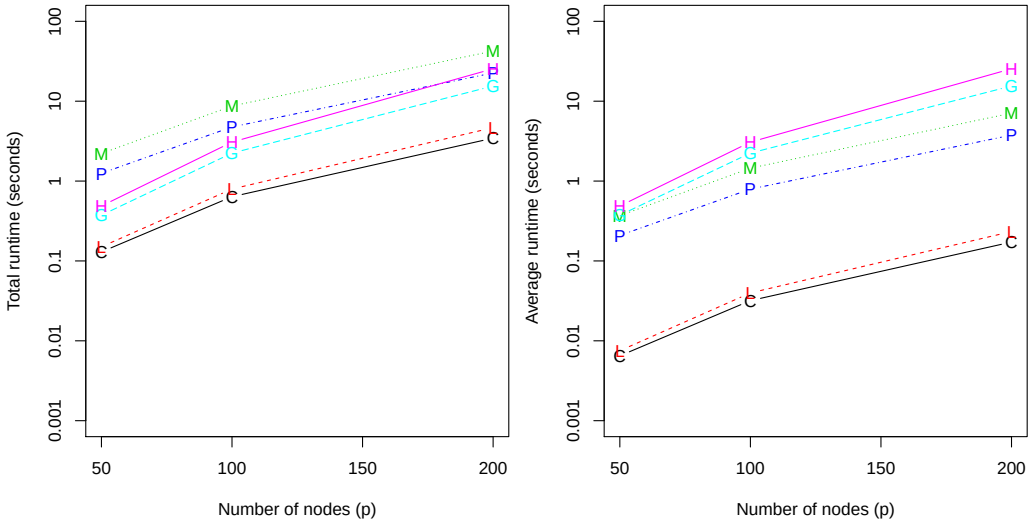

Figure 3: Timing comparison in low dimensions for all six algorithms (C = CCDr-MCP, L = CCDr-ℓ_1, G = GES, H = HC, M = MMHC, P = PC).

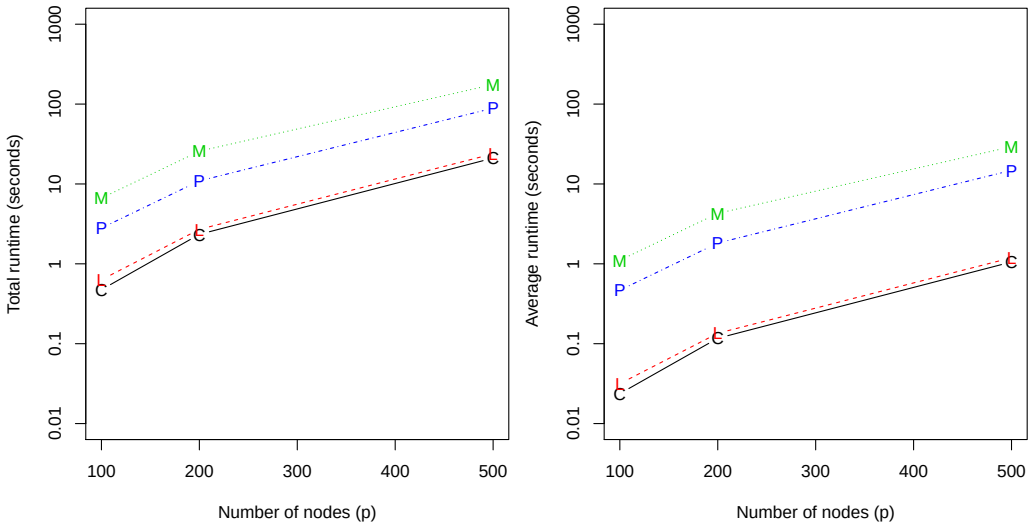

Figure 4: Timing comparison in high dimensions, excluding GES and HC (C = CCDr-MCP, L = CCDr-ℓ_1, M = MMHC, P = PC).

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

6.4 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:

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

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 3 and Figure 5. Since the timing of the algorithm depends crucially on the total number of models estimated, and also on the threshold parameter α, 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 3 shows that the results are comparable to those in Section 6.3. Furthermore, as p increases we notice that TPR increases while 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 5, 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.[1] 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 3. The total runtime for 20 estimates was 75 minutes, which may be compared with the 13 days reported for

1. A comprehensive comparison of the updated implementation vs. the numbers reported here can be found in Figure S1 in the Supplementary Materials.

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 3: Average estimation performance of CCDr-MCP from Section 6.4, averaged over $N = 20$ random choices of s_0 and n for each p.

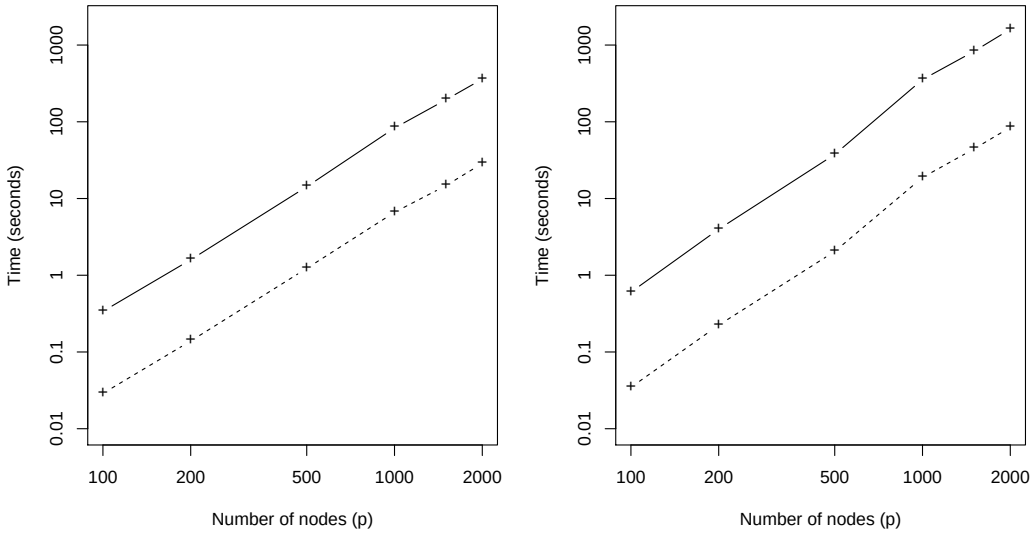

Figure 5: Timing data for CCDr-MCP up to $p = 2000$. The solid line is the total runtime and the dashed line is the average runtime. (left) Time to estimate graphs with at most p edges; (right) Full runtime with edge threshold $\alpha = 3$.

MMHC on a graph with $p = 5000$ in Tsamardinos et al. (2006). Regarding the internal implementation of our method, we did not make 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.

6.5 Model 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 model selection techniques such as BIC or cross-validation. It has already been noted that these empirical model 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-ℓ_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. 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 model 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 6.3.1, the results resemble those previously reported (Table S3 in the Supplementary Materials). Across the board, graphs with more edges were selected, but the qualitative observations between CCDr and PC / MMHC remain the same.

6.6 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 6.3, 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 6.3.1 and 6.3.2 persists (see Figures S3 and S4 in the Supplementary Materials). 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. Moreover, restricting the maximum size of the conditioning sets used in the conditional independence tests in the PC algorithm, as suggested by the work of Anandkumar et al. (2012), also had a negligible effect. 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.

7. 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,[2] 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).

7.1 Bayesian Network Repository

All of the networks examined in this experiment were loaded using the `bnlearn` package.[3] 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

2. The original repository can be found at: `http://www.cs.huji.ac.il/site/labs/compbio/Repository/`.
3. A mirror of the repository used by the `bnlearn` package can be found at: `http://www.bnlearn.com/bnrepository/`.

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 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 6.5 how traditional model 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 6.

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 instance, `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-ℓ_1. Compared with the simulation results in Section 6, there is a more pronounced difference between the performance of concave vs ℓ_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 ℓ_1 regularization is preferable is `pathfinder`, where CCDr-ℓ_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 S2 in the Supplementary Materials 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-ℓ_1, 71s for PC, and 147s for MMHC. In both cases CCDr-MCP easily did the best job reconstructing the true networks.

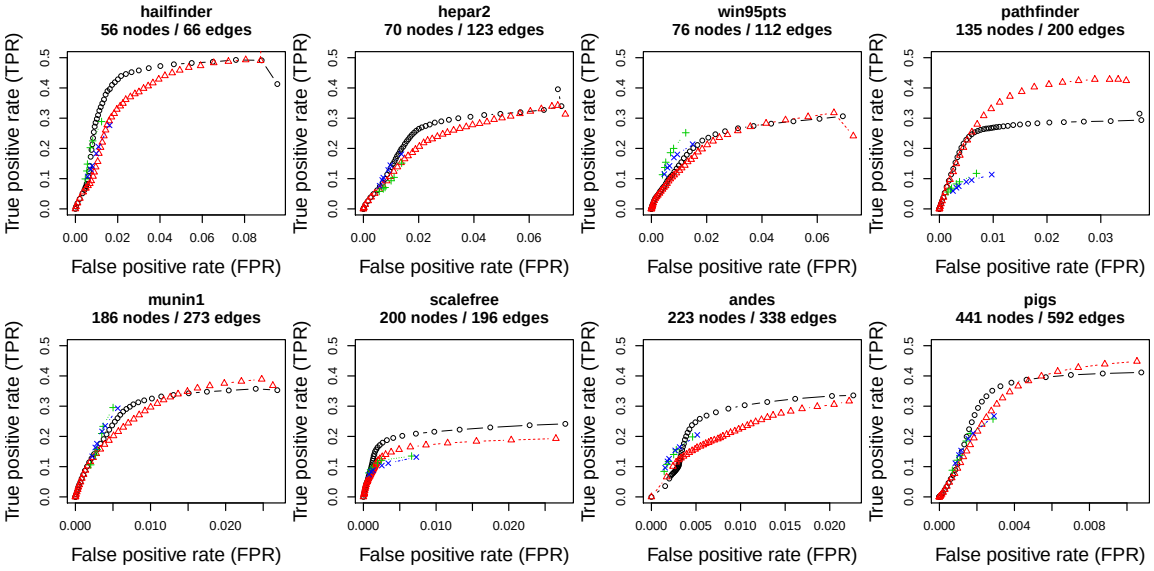

Figure 6: ROC curves for real networks (black ◯ = CCDr-MCP, red △ = CCDr-ℓ_1, blue × = MMHC, green + = PC).

7.2 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 9). 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 4 and 5 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 6, 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 7. Both GES and HC consistently estimated too many edges, which matches the behaviour observed in Section 6.3.1. 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.

8. Conclusion

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. In the finite dimensional scenario, the resulting estimator has good theoretical properties. Through a series of tests designed to test the limits of this new algorithm, we 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-

$p = 11$, $T = 20$	CCDr-MCP	CCDr-ℓ_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 4: Structure estimation performance for all algorithms using the log-transformed continuous cytometry data.

$p = 11$, $T = 20$	CCDr-MCP	CCDr-ℓ_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 5: Structure estimation performance for all algorithms using discretized cytometry data.

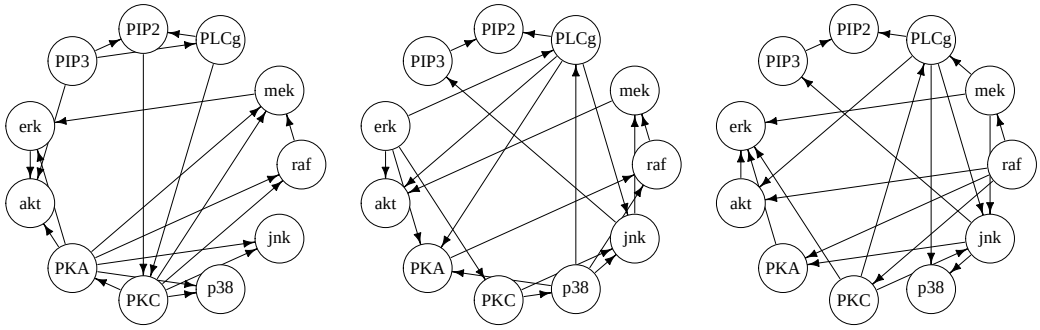

Figure 7: Comparison of the consensus network (left) against the DAGs estimated by the CCDr-MCP algorithm for both data sets: (middle) Log-transformed continuous data set; (right) Discretized data set.

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 in this work 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 knowledge 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 6.3.1 why this may be, and it remains for future work to study this phenomenon in more depth.

While we have focused on the use of cyclic coordinate descent to minimize the penalized log-likelihood, it would be interesting to compare more sophisticated optimization techniques such as adaptive and stochastic coordinate descent. It also remains to incorporate prior knowledge either via whitelists and blacklists, or through a more sophisticated hybrid Bayesian approach. As nonconvex optimization is a rapidly developing field of study, the methods presented here merely scratch the surface of how such techniques can be applied to the structure learning problem for Bayesian networks. An `R` package which implements the proposed algorithm along with some of these improvements is currently under development.

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.

Acknowledgements

We would like to thank the referees for their helpful comments. We would also like to thank Marco Scutari and Sara van de Geer for their thoughtful discussions, as well as Damon Alexander for his assistance and suggestions in implementing the algorithm. This work was supported by NSF grants DMS-1055286 and DMS-1308376 (to Q.Z.) and NSF graduate research fellowships DGE-1144087 and DGE-0707424. B.A. was also supported by a UCLA Dissertation Year Fellowship.

Appendix A. Proofs of Main Results

We collect here the proofs of our main results.

A.1 Formal Preliminaries

Conceptually our theory is quite simple: we have a function F on $\mathbb{R}^{p^2}$ which we would like to maximize over a subset defined by the space of DAGs, $\mathcal{D}$. In order to properly specify a topology for this space, and to ensure that the translation between our statistical model for

(B, Ω) and the mathematical model for $\boldsymbol{\nu}$ is coherent, we carefully outline the mathematical set-up here.

Given a DAG (B, Ω), consider the reparameterization (Φ, R) given by

$$\Phi = B\Omega^{-1/2} \tag{36}$$

$$R = \Omega^{-1/2}. \tag{37}$$

This is of course just the matrix version of the reparameterization that leads to (11). Now define the following function which maps $(\Phi, R) \in \mathbb{R}^{p\times p} \times \mathbb{R}^{p\times p}$ into $\mathbb{R}^{p^2}$:

$$\boldsymbol{\nu}(\Phi, R) = \text{vec}(U) = (u_1, \dots, u_p), \quad \text{where } U = [u_1 \,|\, \dots \,|\, u_p] = R + \Phi.$$

Recall that Φ 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 Φ separately. Furthermore, the sparsity pattern of the off-diagonal elements of U exactly matches that of Φ.

In the proofs, when there is no confusion we will simply write $\boldsymbol{\nu} = U = (\Phi, R) = (B, \Omega)$ to mean that these are all equivalent representations of the same DAG in various parameterizations. In particular, for any $\boldsymbol{\nu}_0 \in \mathcal{E}_0$, we have $\boldsymbol{\nu}_0 = U_0 = (\Phi_0, R_0) = (B_0, \Omega_0)$. Mathematically, we will work with $\boldsymbol{\nu}$, however, our results should always be interpreted in terms of the original model (B, Ω).

The space of DAGs is formally defined as follows:

$$\mathcal{D} := \left\{ \boldsymbol{\nu} = \boldsymbol{\nu}(\Phi, R) \in \mathbb{R}^{p^2} : \Phi \in \mathbb{R}^{p\times p} \text{ is a DAG, } \rho_j > 0 \text{ for all } j \right\}.$$

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 $F(\boldsymbol{\nu}) = \ell_n(\boldsymbol{\nu}) - np_{\lambda_n}(\boldsymbol{\nu})$.

A.2 Proof of Theorem 2

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 4. First recall the following standard result:

Lemma 8. *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 Θ is given and use the Cholesky decomposition to write $\Theta = \Theta(L, D)$ as in (9). Then, taking $A = \Theta(L, D)$ in Lemma 8, we obtain $P_\pi \Theta(L, D) = \Theta(P_\pi L, P_\pi D)$. Alternatively, suppose $(B, \Omega) \in \mathcal{E}(\Theta)$ and suppose $\pi \in \mathcal{P}$ is compatible with (B, Ω). Since $P_\pi B$ is lower-triangular, by taking $A = \Theta(P_\pi B, P_\pi \Omega)$, we may similarly deduce

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

This proves the following lemma, which will be useful:

Lemma 9. *Let (B, Ω) be a DAG. For any permutation $\pi \in \mathcal{P}$ that is compatible with (B, Ω), we have*

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

We now prove Lemma 4, which will be used in the proof of Theorem 2.

Proof of Lemma 4 We only prove this for the original parameterization (B, Ω); the reparameterized case is similar.

Since B_1 and B_2 have a common topological sort, there is a permutation π 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 $\Theta(B_1, \Omega_1) = \Theta(B_2, \Omega_2) := \tilde{\Theta}$, so that (using Lemma 9 above)

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

The last expression is equal to $P_\pi \tilde{\Theta}$, 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 2 Suppose $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ with $b_n(\boldsymbol{\nu}_0) \to 0$. 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 4.1). Since the densities $f(\cdot \,|\, \boldsymbol{\nu})$ are Gaussian, the only condition that needs to be checked is that the Fisher information is positive definite at $\boldsymbol{\nu}_0$ restricted to the DAG space $\mathcal{D}$. Theorem 2 will then follow immediately from Theorem 1 in Fan and Li (2001).

Let $I(\boldsymbol{\nu}_0)$ denote the usual Fisher information matrix at this point; we will show that $I(\boldsymbol{\nu}_0)$ is positive definite. Since f is always a Gaussian density, it will suffice to show that $f(\cdot \,|\, \boldsymbol{\nu}) \neq f(\cdot \,|\, \boldsymbol{\nu}_0)$ for $\boldsymbol{\nu}$ in a sufficiently small neighbourhood of $\boldsymbol{\nu}_0$.

Now suppose $\boldsymbol{\nu} = (\Phi, R)$ is in an arbitrarily small neighbourhood of $\boldsymbol{\nu}_0 = (\Phi_0, R_0)$. 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 Φ_0 implies $i \to j$ in any DAG close to Φ_0. In particular, Φ contains all the edges (including orientation) in Φ_0, with the possible addition of extra edges. That is, Φ_0 is a subgraph of Φ. It follows that there is an ordering of the vertices that is compatible with Φ and Φ_0 simultaneously. Since $\Phi \neq \Phi_0$, it follows from Lemma 4 that $\Theta(\boldsymbol{\nu}) \neq \Theta(\boldsymbol{\nu}_0)$, whence $f(\cdot \,|\, \boldsymbol{\nu}) \neq f(\cdot \,|\, \boldsymbol{\nu}_0)$. ■

Proof of Lemma 5 Note that Lemma 1 implies that the equivalence class $\mathcal{E}_0$ is finite. Set $\varepsilon = \min_{\boldsymbol{\nu}_0 \in \mathcal{E}_0} \min_{i,j}\{|\phi^0_{ij}| : \phi^0_{ij} \neq 0\} > 0$. Then if $\|\Phi - \Phi_0\| \leq \|\boldsymbol{\nu} - \boldsymbol{\nu}_0\| < \varepsilon$, the arguments in the proof of Theorem 2 guarantee the existence of an ordering that is compatible with Φ and Φ_0, and the result follows from Lemma 4. ■

A.3 Proof of Theorem 6

Instead of directly proving Theorem 6, we will prove a slightly more general statement under weaker assumptions. Theorem 6 will then follow as a special case.

The following technical lemmas ensure that the objective function $F(\boldsymbol{\nu})$ 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 10. *Fix $\boldsymbol{\nu}_0$ and suppose $\boldsymbol{\nu}_n$ is a sequence with $\|\boldsymbol{\nu}_n - \boldsymbol{\nu}_0\| = o(1)$. If the empirical log-likelihood $\ell_n(\boldsymbol{\nu})$ is continuous for all n, then*

$$P\left(\lim_{n\to\infty} \frac{1}{n}\ell_n(\boldsymbol{\nu}_n) = \lim_{n\to\infty} \frac{1}{n}\ell_n(\boldsymbol{\nu}_0)\right) = 1.$$

Lemma 11. *Suppose that $p_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $p_\lambda(0) = 0$. 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*

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

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 4, we use $\widehat{\boldsymbol{\nu}}_n$ and $\widehat{\boldsymbol{\nu}}^*_n$ to denote the local maximizers close to $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}^*$, respectively, whose existence is guaranteed by Theorem 2.

Theorem 12. *Suppose that $p_\lambda(t)$ is nondecreasing and concave for $t \geq 0$ with $p_\lambda(0) = 0$. Let $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ be a DAG with strictly more edges than $\boldsymbol{\nu}^*$. Assume further that the conditions for Theorem 3 hold for both $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}^*$. If*

1. *$c_n(\boldsymbol{\nu}^*) = \tau(\lambda_n) + O(n^{-1/2})$ and $c_n(\boldsymbol{\nu}_0) = \tau(\lambda_n) + O(n^{-1/2})$,*
2. *$\limsup_n \tau(\lambda_n) < \infty$,*

3. $\tau(\lambda_n) = \omega(n^{-1/2})$,

then for every $\varepsilon > 0$,

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

Proof Since we assume Theorem 3 holds for both $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}^*$, we may assume without loss of generality that $\mathrm{supp}(\widehat{\boldsymbol{\nu}}_n^*) = \mathrm{supp}(\boldsymbol{\nu}^*)$ and $\mathrm{supp}(\widehat{\boldsymbol{\nu}}_n) = \mathrm{supp}(\boldsymbol{\nu}_0)$.

Since ℓ_n is continuous for each n, $\|\widehat{\boldsymbol{\nu}}_n - \boldsymbol{\nu}_0\| = O_P(n^{-1/2})$, and $\|\widehat{\boldsymbol{\nu}}_n^* - \boldsymbol{\nu}^*\| = O_P(n^{-1/2})$, Lemma 10 implies that

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

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

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

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

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

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

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

Lemma 11 implies that for each $\phi_{ij}^0 \neq 0$,

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

and similarly for all $\phi_{ij}^* \neq 0$. Thus we can write $p_{\lambda_n}(\widehat{\boldsymbol{\nu}}_n) = p_{\lambda_n}(\boldsymbol{\nu}_0) + O_P(n^{-1/2})$ and similarly for $\widehat{\boldsymbol{\nu}}^*$. It thus suffices to show that

$$p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) = \omega(n^{-1/2}).$$

Now, using Condition 1,

$$\begin{aligned}
p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) &= \sum_{\phi_{ij}^0 \neq 0} p_{\lambda_n}(|\phi_{ij}^0|) - \sum_{\phi_{ij}^* \neq 0} p_{\lambda_n}(|\phi_{ij}^*|) \\
&\geq s_0 c_n(\boldsymbol{\nu}_0) - s^* \tau(\lambda_n) + s^* \tau(\lambda_n) - \sum_{\phi_{ij}^* \neq 0} p_{\lambda_n}(|\phi_{ij}^*|) \\
&= (s_0 - s^*)\tau(\lambda_n) + O(n^{-1/2}) + \sum_{\phi_{ij}^* \neq 0} (\tau(\lambda_n) - p_{\lambda_n}(\phi_{ij}^*)) \\
&\geq (s_0 - s^*)\tau(\lambda_n) + O(n^{-1/2}).
\end{aligned}$$

Since $\tau(\lambda_n) = \omega(n^{-1/2})$ (Condition 3), it follows that $p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) \geq \omega(n^{-1/2})$, from which the claim follows. ■

Proof of Theorem 6 Condition 3 in Theorem 12 is equivalent to $\tau(\lambda_n)/n^{-1/2} \to \infty$, and Theorem 6 follows as a special case since the equivalence class $\mathcal{E}_0$ is finite. ■

References

Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification Part I: Algorithms and empirical evaluation. *The Journal of Machine Learning Research*, 11:171–234, 2010a.

Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification Part II: Analysis and extensions. *The Journal of Machine Learning Research*, 11:235–284, 2010b.

Animashree Anandkumar, Vincent YF Tan, Furong Huang, and Alan S Willsky. High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion. *The Journal of Machine Learning Research*, 13(1):2293–2337, 2012.

Animashree Anandkumar, Daniel Hsu, Adel Javanmard, and Sham Kakade. Learning linear Bayesian networks with latent variables. In *Proceedings of The 30th International Conference on Machine Learning*, pages 249–257, 2013.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre 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.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer, 2011.

Peter Bühlmann, Jonas Peters, Jan Ernest, et al. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

Sanjay Chaudhuri, Mathias Drton, and Thomas S Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94(1):199–216, 2007.

David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning From Data*, pages 121–130. Springer, 1996.

David Maxwell Chickering. Optimal structure identification with greedy search. *The Journal of Machine Learning Research*, 3:507–554, 2003.

David Maxwell Chickering and Christopher Meek. Finding optimal Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 94–102. Morgan Kaufmann Publishers Inc., 2002.

Arthur P Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

Arthur Pentland Dempster. *Elements of Continuous Multivariate Analysis*, volume 388. Addison-Wesley Reading, Mass., 1969.

Mathias Drton and Thomas S Richardson. Graphical methods for efficient likelihood inference in Gaussian covariance models. *The Journal of Machine Learning Research*, 9: 893–914, 2008.

Mathias Drton, Rina Foygel, and Seth Sullivant. Global identifiability of linear structural equation models. *The Annals of Statistics*, 39(2):865–886, 2011.

Byron Ellis and Wing Hung Wong. Learning causal Bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482), 2008.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

Jianqing Fan and Jinchi Lv. A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101, 2010.

Jianqing Fan and Jinchi Lv. Nonconcave penalized likelihood with NP-dimensionality. *IEEE Transactions on Information Theory*, 57(8):5467–5484, 2011.

Yingying Fan and Jinchi Lv. Asymptotic equivalence of regularization methods in thresholded parameter space. *Journal of the American Statistical Association*, 108(503):1044–1061, 2013.

Thomas Shelburne Ferguson. *A Course in Large Sample Theory*, volume 38. CRC Press, 1996.

Rina Foygel and Mathias Drton. Extended Bayesian information criteria for Gaussian graphical models. In *Advances in Neural Information Processing Systems*, pages 604–612, 2010.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the Graphical Lasso. *Biostatistics*, 9(3):432–441, 2008.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

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

Fei Fu and Qing Zhou. Penalized estimation of sparse directed acyclic graphs from categorical data under intervention. *arXiv Preprint arXiv:1403.2310*, 2014.

José A Gámez, Juan L Mateo, and José 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.

José A Gámez, Juan L Mateo, and José 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.

Dan Geiger and David Heckerman. Learning Gaussian networks. *arXiv Preprint arXiv:1302.6808*, 2013.

David Heckerman, Dan Geiger, and David M Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

Jian Huang, Patrick Breheny, and Shuangge Ma. A selective review of group selection in high-dimensional models. *Statistical Science*, 27(4), 2012.

Jianhua Z Huang, Naiping Liu, Mohsen Pourahmadi, and Linxu Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98, 2006.

Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. Estimation of a structural vector autoregression model using non-Gaussianity. *The Journal of Machine Learning Research*, 11:1709–1731, 2010.

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

Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H Maathuis, and Peter Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

Clifford Lam and Jianqing Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *The Annals of Statistics*, 37(6B):4254, 2009.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

Po-Ling Loh and Peter Bühlmann. High-dimensional learning of linear causal networks via inverse covariance estimation. *arXiv Preprint arXiv:1311.3492*, 2013.

Jinchi Lv and Yingying Fan. A unified approach to model selection and sparse recovery using regularized least squares. *The Annals of Statistics*, 37(6A):3498–3528, 2009.

Rahul Mazumder, Jerome H Friedman, and Trevor Hastie. SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495): 1125–1138, 2011.

Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

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

Mohsen 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`.

Garvesh Raskutti and Caroline Uhler. Learning directed acyclic graphs based on sparsest permutations. *arXiv Preprint arXiv:1307.0366*, 2014.

Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

Richard Redner. Note on the consistency of the maximum likelihood estimate for nonidentifiable distributions. *The Annals of Statistics*, 9(1):225–228, 1981.

Robert W Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V*, pages 28–43. Springer, 1977.

Philipp Rütimann and Peter Bühlmann. High dimensional sparse covariance estimation via directed acyclic graphs. *Electronic Journal of Statistics*, 3:1133–1160, 2009.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

Mark Schmidt, Alexandru Niculescu-Mizil, and Kevin Murphy. Learning graphical model structure using L1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(i03), 2010.

Marco Scutari. Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn R package. *arXiv Preprint arXiv:1406.7648*, 2014.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-Gaussian acyclic model for causal discovery. *The Journal of Machine Learning Research*, 7:2003–2030, 2006.

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

Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

Nicolas Städler, Peter Bühlmann, and Sara Van De Geer. ℓ_1-penalization for mixture regression models. *Test*, 19(2):209–256, 2010.

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

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

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

Hansheng Wang, Runze Li, and Chih-Ling Tsai. Tuning parameter selectors for the Smoothly Clipped Absolute Deviation method. *Biometrika*, 94(3):553–568, 2007.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for Lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

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

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

Cun-Hui Zhang and Tong Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

Peng Zhao and Bin Yu. On model selection consistency of Lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

Qing Zhou. Multi-domain sampling with applications to structural inference of Bayesian networks. *Journal of the American Statistical Association*, 106(496):1317–1330, 2011.

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