CONCAVE PENALIZED ESTIMATION OF SPARSE BAYESIAN NETWORKS

BRYON ARAGAM AND QING ZHOU

University of California, Los Angeles

Abstract. We develop a penalized likelihood approach to estimating the structure of a Gaussian Bayesian network, given by a directed acyclic graph, from observational data under a concave penalty. The framework introduced here does not rely on faithfulness or knowledge of the ordering of the variables and favours sparsity over complexity in estimating the underlying graph. Asymptotic theory for the estimator is provided in the finite-dimensional case, and a fast numerical scheme is offered that is capable of estimating the structure of graphs with thousands of nodes. By reparametrizing the usual normal log-likelihood, we obtain a convex loss function which accelerates computation of the proposed estimator. Our algorithm also takes advantage of sparsity and acyclicity by using coordinate descent, a computational approach which has recently become quite popular. Finally, we compare our method with the well-known PC algorithm, and show that our method is faster in general and does a significantly better job of handling small samples and very sparse networks. Our focus is on the Gaussian linear model, however, the framework introduced here can also be extended to non-Gaussian and non-linear designs, which is an attractive prospect for future applications.

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 very 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 parametric model, is it possible to construct a directed acyclic graph (DAG) which decomposes the estimated model 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 (e.g. Chickering (2003); Tsamardinos et al. (2006); Zhou (2011); Raskutti and Uhler (2013)). Unfortunately, as the number of variables increases, this problem becomes notoriously difficult. 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 propose using a penalized likelihood estimation framework for estimating the structure of a Bayesian network from observational data. Our use of concave regularization, as opposed to the more popular ℓ_1-regularization, is new. The main contribution is an efficient algorithm for estimating the structure of sparse Bayesian networks that does not require advance knowledge of the order of the variables or the restrictive faithfulness assumption. In addition, our algorithm easily handles data sets containing 1000 or more variables while searching through the entire space of DAGs for an optimal structure. We compare our algorithm with the PC algorithm (Spirtes and Glymour (1991)), and show that when $p \gg n$ our method can achieve greater accuracy in the same amount of time. We also provide finite-dimensional asymptotic performance guarantees; the high-dimensional case in which p is allowed to grow with n is left for future work.

The organization of the rest of this paper is as follows: In the remainder of Section 1, we review previous relevant work and establish the necessary background and notation. In Section 2 we describe the statistical framework and define the penalized estimator that is the focus of this paper. Section 3 then provides the necessary finite-dimensional theory to justify the use of our estimator. A complete description of our algorithm is outlined in Section 4, followed by an empirical evaluation of the algorithm in Section 5. Section 5 also offers a side-by-side comparison of our algorithm with the PC algorithm, and Section 6 provides an evaluation of these algorithms using a real-world dataset. We finally conclude with a discussion of some future directions for this research.

1.1. **Review of previous results.** Our primary focus in this work will be on the computational issues associated with estimating Bayesian networks from purely observational data. An exhaustive discussion of the existing algorithms for inference on Bayesian networks is beyond the scope of this paper, and we refer the reader to the excellent comparison of state-of-the-art algorithms contained in Tsamardinos et al. (2006).

While many of the existing algorithms in this area have good estimation performance, they do not scale well and have trouble estimating graphs with more than fifty or so nodes. This is largely due to the combinatorial nature of the algorithms. Since our main goal is the estimation of networks with well over one thousand nodes, we are only interested in algorithms that can handle hundreds of nodes with little difficulty. This excludes all of the algorithms compared at the time of the study conducted in Tsamardinos et al. (2006).

To our knowledge, the only existing algorithm that has been shown to gracefully handle networks with more than 1000 nodes is the high-dimensional implementation of the PC algorithm introduced in Kalisch and Bühlmann (2007).[1] The PC algorithm was originally introduced in Spirtes and Glymour (1991), and relies on repeated conditional independence tests to estimate the structure of a network. Because of its reliance on tests of conditional independence, the PC algorithm is also combinatorial in nature and suffers a worst-case exponential running time, although the recent high-dimensional modification reduces the expected running time to polynomial when the true graph is sufficiently sparse. For these reasons, the PC algorithm is the *de facto* gold standard for high-dimensional structural inference of Bayesian networks.

Our approach relies instead on solving a penalized maximum likelihood problem, which has been explored previously. While there is promising theory to support this approach (van de Geer and Bühlmann (2013)), solving these problems numerically has proven elusive when p is large because of the nonconvex nature of the DAG constraint. Many algorithms circumvent this issue by assuming prior knowledge of the ordering amongst the variables (Shojaie and Michailidis (2010)), which is a restrictive assumption. Without assuming knowledge of this ordering, the application of maximum likelihood techniques has proven difficult. By combining a clever method of enforcing acyclicity with a block coordinate descent algorithm, Fu and Zhou (2013) constructed an ℓ_1-penalized MLE-based algorithm for estimating Bayesian networks without prior knowledge of the ordering. The framework introduced in this paper is based on this work, and in fact, modulo a few mathematical simplifications, the work of Fu and Zhou (2013) is a special case of the new framework presented here.

1.2. **Background and notation.** 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. This has the upshot of avoiding unnecessary digressions into the calculus of Bayesian networks. 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.

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

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

[1]Note that this modification was introduced after the study by Tsamardinos et al. (2006). In the comparison study, the authors considered the older, slower variant of the PC algorithm.

where $\Sigma_0 \in \mathbb{R}^{p\times p}$ is an unknown covariance matrix. Such a model can always be written as a set of Gaussian structural equations as follows:

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

where the ε_j are mutually independent with $\varepsilon_j \sim N(0, (\omega_j^0)^2)$, ε_j is independent of $\Pi_j^0 = \{X_i : \beta_{ij}^0 \neq 0\}$, and $\beta_{jj}^0 = 0$. The matrix $B_0 = (\beta_{ij}^0)$ can be regarded as the weighted adjacency matrix of a directed acyclic graph and represents a Bayesian network for the distribution $N(0, \Sigma_0)$. 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. For a more thorough introduction to graphical modeling concepts, see Lauritzen (1996).

Unless otherwise noted, $\|\cdot\|$ shall always mean the standard Euclidean norm. 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.

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 in the two matrices (B_0, Ω_0). Here, $\Omega_0 = \operatorname{diag}((\omega_1^0)^2, \ldots, (\omega_p^0)^2)$ is the matrix of error variances. We denote the matrix of error variances by Ω in order to avoid confusion with Σ, the covariance matrix of X.

Given a DAG B and variance matrix $\Omega = \operatorname{diag}(\omega_1^2, \ldots, \omega_p^2)$, the parameters (B, Ω) uniquely define a structural equation model as in (2), and this model defines a $N(0, \Sigma)$ distribution. By (3), we have for any model (B, Ω),

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

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

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

This expression shows how the weighted adjacency matrix of a DAG can be considered as a reparametrization of the usual normal distribution, and gives us an explicit connection between inverse covariance estimation and DAG estimation. Strictly speaking, a DAG that fully encodes a Bayesian network is specified by *both* a weighted adjacency matrix B and a variance matrix Ω, however, we will frequently refer to a DAG simply by its adjacency matrix B. When there is any ambiguity one may assume that there is an assumed variance matrix Ω paired with B, although it may not be explicitly mentioned.

1.3. **Permutations and parametrization.** The decomposition of a normal distribution as a linear structural equation model (SEM) as in (2) is not unique; in general for a given Θ there are many paramterizations (B, Ω) such that $\Theta(B, \Omega) = \Theta$. This defines the following equivalence class of DAGs:

$$\mathcal{E}(\Theta) := \{(B, \Omega) \mid \Theta(B, \Omega) = \Theta\}. \tag{6}$$

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 $\mathcal{E}(\Theta)$.

Remark 1.1. Our definition of equivalence in terms of equivalent parametrizations is different from the usual definition of *distributional* or *Markov equivalence* that is common in the literature. Two graphs are called *Markov equivalent* if the sets of distributions which are compatible with each graph are the same (Chickering (2003)). For normal distributions it is true that Markov equivalent graphs encode the same conditional independence restrictions and have the same skeletons. This is obviously not the case for graphs which are equivalent in the parametric sense as we have defined here; see below for more details.

In this section we wish to exhibit the connection between equivalent DAGs and the choice of an ordering of the variables. We represent such an ordering as a permutation π on $\{1,\dots,p\}$, which can be interpreted as an ordering in the obvious way. For an arbitrary matrix A, 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 lower triangular. A DAG B will be called *consistent with the ordering* π if $P_\pi B$ is lower-triangular, which is equivalent to saying that $X_k \to X_j$ in B implies $\pi^{-1}(k) > \pi^{-1}(j)$.

Suppose Θ_0 is given and let π be an ordering of the variables. 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

$$P_\pi\Theta_0 = (I-L)D^{-1}(I-L)^T = \Theta(L,D), \tag{7}$$

where L is strictly lower triangular and D is diagonal. By using the identity $P_\pi\Theta(L,D) = \Theta(P_\pi L, P_\pi D)$ for any π, we can thus rewrite (7) as

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

This gives us the unique decomposition of Θ_0 into a DAG that is consistent with the ordering π. Following van de Geer and Bühlmann (2013), for each π we define

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

This defines a subset of the equivalence class, given by $\{(\tilde{B}_0(\pi),\tilde{\Omega}_0(\pi)) \mid \pi \text{ is a permutation}\}$. It is easy to check that in fact, this subset is the entire equivalence class.

Lemma 1.1. *Suppose Θ_0 is a given, fixed covariance matrix. Then*

$$\begin{aligned}\mathcal{E}(\Theta_0) &= \{(P_{\pi^{-1}}L, P_{\pi^{-1}}D) \mid P_\pi\Theta_0 = \Theta(L,D)\}\\ &= \{(\tilde{B}_0(\pi),\tilde{\Omega}_0(\pi)) \mid \pi \text{ is a permutation}\}.\end{aligned}$$

Example. 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

$$B_0 = \begin{pmatrix}0&1&0\\0&0&1\\0&0&0\end{pmatrix}, \quad \Omega_0 = \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}, \quad \Theta_0 = \Theta(B_0,\Omega_0) = \begin{pmatrix}2&-1&0\\-1&2&-1\\0&-1&1\end{pmatrix}.$$

The DAG B_0 is lower triangularized by the permutation $\pi_0 = (3,2,1)$ that swaps X_1 and X_3, and $B_0 = \tilde{B}_0(\pi_0)$. If we consider the alternative ordering of the variables given by $X_1 \prec X_3 \prec X_2$, we obtain the parametrization

$$B_1 = \begin{pmatrix}0&1/2&1\\0&0&0\\0&1/2&0\end{pmatrix}, \quad \Omega_1 = \begin{pmatrix}1&0&0\\0&1/2&0\\0&0&2\end{pmatrix}, \quad \Theta_0 = \Theta(B_1,\Omega_1) = \begin{pmatrix}2&-1&0\\-1&2&-1\\0&-1&1\end{pmatrix}.$$

Then $B_1 = \tilde{B}_0(\pi_1)$, where $\pi_1 = (2,3,1)$ is the permutation that induces this ordering. Moreover, $\Theta(B_0,\Omega_0) = \Theta(B_1,\Omega_1) = \Theta_0$, so that the two DAGs (B_0,Ω_0) and (B_1,Ω_1) are equivalent. They are, of course, not Markov equivalent.

Given this fact, the question arises: which DAG $(\tilde{B}_0(\pi),\tilde{\Omega}_0(\pi))$ do we want to estimate? What makes one DAG "better" than another? In the presence of experimental data, one may consider issues of causality, in which case each DAG represents a very different causal structure. In the absence of such data, however, we can make no such distinctions. In this case, a natural assumption is to choose the DAG that most parsimoniously represents the parameter Θ_0 in the

sense that it has the fewest number of edges. Let (B_0, Ω_0) be such a parametrization, and note that in general it may not be unique.

This gives us an intriguing prospect for estimating DAGs, which has been considered in Raskutti and Uhler (2013). If we have an estimate of Θ_0, then we can simply exploit efficient Cholesky solvers to compute $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$ for every choice of π, and choose the one with the fewest edges. Unfortunately there are $p!$ possible choices of π, and when p grows to even modest sizes, this approach is computationally infeasible.

In Section 2 we introduce a new estimator, which seeks to estimate Θ_0 by decomposing this estimate into a sparse DAG (B_0, Ω_0), along with an algorithm for computing this estimator. This will be the focus of the remainder of the paper.

1.4. **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 *parameter* estimation. For this reason, we have chosen to view the data-generating 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. In this case, we are not only interested in which edges are present in the true structural model, but more importantly we wish to estimate the magnitude of the systematic effects and the noise, which are encoded in the parameters $(\beta_{ij}^0, (\omega_j^0)^2)$. It is not enough in this instance to simply estimate which β_{ij}^0 are nonzero.

This is the perspective commonly adopted in the social sciences and in public health, in which the structural equations model causal relationships between the measured 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. Instead, with the right assumptions one could view our method as eliminating causal DAGs that are unlikely to represent the data, without singling out a particular DAG as the "true" causal DAG.

Remark 1.2. When studying Bayesian networks, 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. This is also why in the previous section we somewhat arbitrarily chose to define B_0 as any DAG that represents Θ_0 with the fewest number of edges.

2. The Concave Penalization Framework

Our approach is to use a penalized maximum likelihood estimator to estimate a sparse DAG B_0 that represents Θ_0. Our model has two sets of unknown parameters:

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

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

$$L(B, \Omega \,|\, X) = \sum_{j=1}^{p} \left[\frac{n}{2} \log(\omega_j^2) + \frac{1}{2\omega_j^2} \|x_j - X\beta_j\|^2 \right]. \tag{8}$$

This will be our loss function. By using (5) and defining $S_n := X^T X$, equation (8) can be rewritten in terms of $\Theta = \Theta(B, \Omega)$ directly as

$$L(\Theta \,|\, X) = -\frac{n}{2} \log\det\Theta + \frac{1}{2} \operatorname{tr}(\Theta S_n). \tag{9}$$

By combining (8) and (9), we have $L(B, \Omega \,|\, X) = L(\Theta(B, \Omega) \,|\, X)$. This shows how we can simultaneously estimate Θ and decompose Θ into a DAG given by (B, Ω).

Following the usual practice, instead of considering the ordinary unpenalized MLE, we employ a penalty function in order to promote sparsity and avoid overfitting. In what follows, p_λ :

$[0,\infty) \to \mathbb{R}$ is an arbitrary nonnegative and nondecreasing penalty function that depends on a tuning parameter λ, and possibly depending on one or more additional shape parameters. In Section 2.1 we will discuss the details of choosing the penalty function.

The key in our approach is to penalize the DAG B, instead of the covariance Θ. Once p_λ is chosen, we may define a penalized estimator by

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

By only penalizing the edge weights in B, the minimization in (10) has the effect of finding the parameter Θ that simultaneously increases the likelihood while reducing the number of edges in B. In this way, the estimator (10) produces a sparse Bayesian network whose distribution is close to the true, underlying distribution.

Remark 2.1. If we further constrain the minimization problem in (10) to include only DAGs which are consistent with a given ordering, we can reduce the problem to a series of p individual penalized regression problems. For any ordering π, the parents of X_j must be a subset of the variables that come after X_j in the ordering. Formally, $\Pi_j^0 \subset \{X_k : \pi^{-1}(k) > \pi^{-1}(j)\}$. 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 terms of model selection and parameter estimation can then be established through standard penalized regression theory. Of course, in most practical applications the true ordering, or even an approximate ordering, is not known in advance. The key contribution of our framework is a method for adaptively selecting an ordering, which produces a sparse decomposition of the true distribution into a Bayesian network, based solely on observational data.

2.1. **Choice of penalty function.** 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 (see Antoniadis and Fan (2001) for proofs):

(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.

Note that 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.

We follow the above inspiration and choose to only consider penalty functions that satisfy (1-3) above, and hence in particular are concave functions. To simplify the theory, we also further assume that p_λ is continuous and $p_\lambda(0) = 0$. Examples of penalty functions of this type in the literature include the SCAD (Fan and Li (2001)) and MCP (Zhang (2010)). For our theory, we will not need to fix a specific p_λ, but in our computations we chose to use MCP. The MCP is defined for $t \geq 0$ by

$$\begin{aligned} p_\lambda(t;\gamma) &:= \lambda \left(t - \frac{t^2}{2\lambda\gamma} \right) 1(t < \lambda\gamma) + \frac{\lambda^2\gamma}{2} 1(t \geq \lambda\gamma) \\ &= \begin{cases} \lambda \left(t - \frac{t^2}{2\lambda\gamma} \right), & t < \lambda\gamma, \\ \frac{\lambda^2\gamma}{2}, & t \geq \lambda\gamma. \end{cases} \end{aligned} \tag{11}$$

The γ parameter controls the concavity of the penalty. We will thus refer to γ as the *concavity parameter* and λ as the *regularization parameter* in what follows. MCP is easily seen to be a quadratic spline between the origin and the ℓ_0 penalty with a knot at $t = \lambda\gamma$.

Another popular choice of penalty is the Lasso, or ℓ_1, penalty. This choice of penalty has the virtue of efficient computation, but unfortunately yields biased estimates in general (the Lasso does not satisfy Condition (1) above). The bias issues can be circumvented by employing the adaptive Lasso (Zou (2006)), an idea which has been explored in Fu and Zhou (2013). Despite these drawbacks, our algorithm can easily be adapted to employ the Lasso penalty if desired, although such a program would not have the same theoretical properties as the concave formulation (see Section 3).

In terms of computation, the only strict requirement on the penalty function is that there exists a corresponding threshold function $S(\cdot, \lambda)$ to perform the single parameter updates (see Section 4.2 for details). The MCP, SCAD, and Lasso penalties all satisfy this requirement.

2.2. **Reparametrization.** One of the drawbacks of the loss in (8) is that it is nonconvex, which complicates the minimization of the penalized loss. If we minimize (8) 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 reparametrization of the problem, introduced in Städler et al. (2010), which leads to a convex loss.

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

$$L(\Phi, R \,|\, X) = \sum_{j=1}^{p} \left[-n \log(\rho_j) + \frac{1}{2} \|\rho_j x_j - X\phi_j\|^2 \right], \tag{12}$$

where $\Phi = [\phi_1 \,|\, \cdots \,|\, \phi_p]$ and $R = \operatorname{diag}(\rho_1, \ldots, \rho_p)$. The loss function in (12) 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.

Once we add the penalty term, the resulting problem is again nonconvex. Moreover, the DAG constraint in (10) is also nonconvex. The idea behind this reparametrization is to allow our algorithm to exploit convexity wherever possible in order to reap at least *some* computational and analytical gains. As we shall see, the gains are indeed significant.

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 then 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 $(\widehat{\Phi}(\lambda), \widehat{R}(\lambda))$. This DAG estimator is the main focus of our work.

Remark 2.2. Note that the solution to (13) is *not* the same as the solution to (10) since we are penalizing different terms. In the original parametrization, we penalize the coefficients β_{ij}, whereas after reparametrizing we are penalizing the rescaled coefficients $\phi_{ij} = \beta_{ij}/\omega_j$. Thus we are also penalizing choices of coefficients which overfit the data, i.e., which have ω_j very small.

In the sequel, the pair $(\widehat{B}, \widehat{\Omega})$ shall always denote the estimator defined by (14), disregarding the estimator defined in (10).

Remark 2.3. By assuming the data is Gaussian, we obtain a tractable loss function in the form of the usual normal negative log-likelihood and by exploiting the structure of this loss function via the reparametrization above, we are able to compute approximate solutions to (13) very efficiently. It is clear, however, that even if the data is not Gaussian, as long as we can write down the negative log-likelihood (e.g. as in (8)), the estimator in (10) is well-defined. This continues to hold even if the relationships between the variables are nonlinear. In this sense, our theoretical framework is not dependent on the structural assumptions of our model.

Furthermore, the numerical scheme that we propose in Section 4 partially extends to such generalizations. In such cases there is no guarantee that a reparametrization such as the one that leads to (12) exists. Since we exploit the convexity of this reparametrization in designing and accelerating our algorithm, computing an estimate for such generalizations would likely be more difficult.

2.3. **The role of sparsity.** In principle, the estimator defined in (13, 14) does not require the true graph to be sparse. In what follows, however, we will assume that B_0 is sparse in the sense that $s_0 \ll p^2$. 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 few variables are truly involved, or (b) on average, each variable has only a few parents. In case (a), estimating B_0 is very similar to the variable screening problem. Both of these scenarios are realistic in practice. More importantly, for datasets 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. For this reason, we chose to tailor our algorithm to this case.

Given these practical concerns, our goal was to design an estimator that works best in the high-dimensional, sparse scenario. This is one of the main reasons for employing a concave penalty, whose superior performance in the $p \gg n$ regime has been highlighted in the context of regression (see Zhang and Zhang (2012); Fan and Lv (2010); Lv and Fan (2009); Zhang (2010); Fan et al. (2012)). We discuss this in more detail in Sections 2.4 and 3.5.

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. This is discussed in detail in Section 4.4. The result is an efficient, scalable, and accurate algorithm when we are confident that the true underlying graph is sparse.

2.4. **Comparison with ℓ_0-penalization.** In a recent paper, van de Geer and Bühlmann (2013) analyze an ℓ_0-penalized maximum likelihood estimator for sparse DAGs. Their estimator is in fact a special case of (10) with $p_\lambda(t) = \lambda^2 1(t \neq 0)$, that is, when the penalty is chosen to be the usual ℓ_0 penalty. It is no accident that we have adopted much of the same notation for ease of comparison. Note, however, that due to our reparametrization, even with the ℓ_0 penalty the estimator (13, 14) is not the same as the estimator defined in van de Geer and Bühlmann (2013).

They show that under certain conditions, if $\hat{\pi}$ represents the estimated ordering of the variables, their ℓ_0 estimator $\widehat{B}$ converges to $\tilde{B}_0(\hat{\pi})$, even when p grows with n. More importantly, they show that the number of edges in $\widehat{B}$ is of the same order as the true DAG, and under certain stronger conditions the ℓ_0 estimator has the variable screening property: with large probability, $\widehat{B}$ contains all the edges present in $\tilde{B}_0(\hat{\pi})$.

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. This is discussed in more detail in Section 3.5.

2.5. **Inverse covariance estimation.** Equation (5) 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 $\widehat{\Theta} := \Theta(\widehat{B}, \widehat{\Omega})$. Thus, in addition to estimating a DAG, the estimator we have defined gives us a free estimate of the true parameter. This is precisely what we mean when we say that our estimator "decomposes" Θ_0 into a Bayesian network.

As a result, we 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 (Stein (1956, 1975); Dempster (1972)), with recent approaches employing regularization in various incarnations (e.g. Huang et al. (2006); Banerjee et al. (2008); Friedman et al. (2008); Lam and Fan (2009); Rothman et al. (2010)). A detailed survey of recent progress in this area can be found in Pourahmadi (2011). We will not pursue this connection in detail here, however, a few comments are in order.

First, due to the reparametrization that leads to (12), our approach is distinct from existing methods that regularize based on Cholesky factors (Huang et al. (2006); Lam and Fan (2009)). In particular, the consistency theory in Lam and Fan (2009) for the sparse Cholesky decomposition does not apply directly to our method. Additionally, these methods make implicit use of an *a priori* ordering amongst the variables, which is unnecessary for our approach and is indeed one of its main advantages. By assuming a known ordering, the optimization problem becomes unconstrained, which simplifies the theory and computations substantially. In contrast, optimizing over all possible orderings simultaneously results in a constrained, nonconvex problem with more parameters.

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 is established through the theory presented in Section 3, as well as empirically via the simulations discussed in Section 5.3. 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.

3. 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.

3.1. 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 structure (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 sequence of local minimizers for the true parameter, which is unique (e.g. Fan and Li (2001)). It turns out that even if the model is nonidentifiable, we can still establish a consistent sequence of local minimizers for each equivalent parameter. As long as there are finitely many equivalent parameters, these sequences 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 1.1). 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* DAG in $\mathcal{E}_0$ in the sense that it will have strictly smaller penalized loss than its 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 DAG is the global minimizer. The simulations in Section 5, however, 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 consistent local maximizers before stating our main result in Section 3.3. We also briefly discuss the high-dimensional scenario in which p is allowed to depend on n.

3.2. Existence of local maximizers.

In analogy with the parametrization (B, Ω), let us define

$$\Theta(\Phi, R) = (R - \Phi)(R - \Phi)^T, \tag{15}$$

which gives us a formula for the inverse covariance matrix in the parametrization (Φ, R). Note that $\Theta(B, \Omega) = \Theta(\Phi, R)$ for any two equivalent parametrizations, and hence also $L(B, \Omega) = L(\Phi, R)$.

To simplify our notation, define $U := R - \Phi \in \mathbb{R}^{p \times p}$ and write $U = [u_1 \,|\, \cdots \,|\, u_p]$. Recall that after permutation, U is lower triangular with diagonal elements ρ_j. Now define $\boldsymbol{\nu} = (u_1, \ldots, u_p) \in \mathbb{R}^{p^2}$ to be the vectorized copy of U in $\mathbb{R}^{p^2}$; in a slight abuse of notation, we will also write $\boldsymbol{\nu} = (\Phi, R)$. As is customary, we denote the support set of a vector by

$\text{supp}(\boldsymbol{\nu}) := \{j \mid \nu_j \neq 0\}$, and likewise for matrices $\text{supp}(B) := \{(i,j) \mid \beta_{ij} \neq 0\}$. Let $\ell_n(\boldsymbol{\nu} \,|\, X)$ be the ordinary log-likelihood of the parameter vector $\boldsymbol{\nu}$ and define

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

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 the space of DAGs, which we denote by $\mathcal{D}$. For a more formal 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 (15), given $(\widehat{\Phi}, \widehat{R})$ we may consider the resulting estimate of the inverse covariance matrix $\widehat{\Theta} = \Theta(\widehat{\Phi}, \widehat{R})$. For any DAG $\boldsymbol{\nu} \in \mathbb{R}^{p^2}$, we may define in the obvious way the matrix $\Theta(\boldsymbol{\nu})$. Let $\mathcal{E}_0 = \mathcal{E}(\Theta_0) = \{\boldsymbol{\nu} \in \mathbb{R}^{p^2} : \Theta(\boldsymbol{\nu}) = \Theta_0\}$. For the remainder of this section we deviate from our original convention wherein a subscript zero denotes the sparsest DAG in $\mathcal{E}_0$. To distinguish between arbitrary DAGs and the equivalent DAGs in $\mathcal{E}_0$, we will denote an arbitrary element of $\mathcal{E}_0$ by $\boldsymbol{\nu}_0$, and denote a minimal-edge representation by $\boldsymbol{\nu}^*$.

For any $\boldsymbol{\nu}_0 = (\Phi_0, R_0) = ((\phi^0_{ij}), (\rho^0_j)) \in \mathcal{E}_0$, 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 is always finite by our assumptions on the penalty function (Section 2.1). Note that we also have $\tau(\lambda) = \lim_{t\to\infty} p_\lambda(t)$.

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

Theorem 3.1. *Fix $p \geq 1$. If there exists $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ with $b_n \to 0$, then there is a local maximizer $\widehat{\boldsymbol{\nu}}_n$ of $F(\boldsymbol{\nu})$ such that*

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

When $a_n = O(n^{-1/2})$, we obtain a $n^{1/2}$-consistent sequence of local maximizers. Note that 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 model selection when p remains fixed:

Theorem 3.2. *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 $a_n = O(n^{-1/2})$, $b_n \to 0$, and let $\widehat{\boldsymbol{\nu}}_n$ be a sequence of π-local maximizers from Theorem 3.1. If $\lambda_n \to 0$ and $\lambda_n n^{1/2} \to \infty$, then

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

In fact, this follows immediately from Theorem 3.1 and Theorem 2 in Fan and Li (2001). An obvious corollary is that $P(\hat{s}_n = s_0) \to 1$. Note also that assumption (20) is always satisfied when p_λ is the MCP.

Remark 3.1. If we assume that the conditions of Theorems 3.1 and 3.2 hold for *all* $\boldsymbol{\nu}_0 \in \mathcal{E}_0$, then we can conclude that every equivalent DAG has a sequence of π-local maximizers that is model selection consistent. 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.

It is important to interpret these theorems carefully: 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 sequence of local maximizers of F that is consistent. It remains to establish that the global maximizer of F is indeed one of these local maximizers.

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

Lemma 3.3. *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 parametrization (Φ, R).*

Recall that a *topological sort* of a DAG is an ordering $\prec$ of the variables such that $X_i \to X_j$ implies $X_i \prec X_j$. The assumption that two DAGs have a common topological sort is equivalent to each DAG being consistent 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 $B(\boldsymbol{\nu}_0, \varepsilon) := \{\boldsymbol{\nu} \in \mathcal{D} \mid \|\boldsymbol{\nu} - \boldsymbol{\nu}_0\| < \varepsilon\}$.

Lemma 3.4. *For any fixed 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.

3.3. **The main result.** We will now significantly strengthen Theorems 3.1 and 3.2 by showing that the 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 3.1, there is a sequence $\widehat{\boldsymbol{\nu}}_n^*$ of π-local maximizers 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. Of course, since the likelihood $\ell_n(\boldsymbol{\nu}_0)$ is constant for all $\boldsymbol{\nu}_0$, we would then have

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

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 penalty grows fast enough, this is precisely the case:

Theorem 3.5. *Suppose that $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$, $b_n(\boldsymbol{\nu}_0) \to 0$, and $\max_{i,j} p_{\lambda_n}(|\phi_{ij}^0|) = \tau(\lambda_n) + O(n^{-1/2})$ for every $\boldsymbol{\nu}_0 \in \mathcal{E}_0$. If $\sup_n \tau(\lambda_n) < \infty$ and $\tau(\lambda_n) n^{1/2} \to \infty$ 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 3.5 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 very 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 3.1, 3.2, and 3.5 uniformly, we can combine all of the results in order to characterize the behaviour of the estimates in terms of the parametrization $(\widehat{B}, \widehat{\Omega})$ given by (14). Before stating the main theorem, we will need some notation to distinguish π-local maximizers. When the conditions of Theorem 3.1 hold for all π, we will 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 3.6. *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 $a_n = O(n^{-1/2})$, $b_n \to 0$, *and* $\max_{i,j} p_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n) + O(n^{-1/2})$ *for each DAG in* $\mathcal{E}_0$. *If* $\lambda_n \to 0$ *and* $\tau(\lambda_n)n^{1/2} \to \infty$, *then for any permutation* π, *there is a local maximizer* $(\widehat{B}, \widehat{\Omega})$ *such that*

(1) $\|\widehat{B} - \tilde{B}_0(\pi)\|_F + \|\widehat{\Omega} - \tilde{\Omega}_0(\pi)\|_F = O_P(n^{-1/2})$,
(2) $P(\mathrm{supp}(\widehat{B}) = \mathrm{supp}(\tilde{B}_0(\pi))) \to 1$,
(3) $\|\widehat{\Theta} - \Theta_0\|_F = O_P(n^{-1/2})$.

Furthermore,

$$P\left((\widehat{B}^*, \widehat{\Omega}^*) \in \operatorname*{arg\,max}_{(\widehat{B},\widehat{\Omega})\in\mathcal{M}_n} F(B, \Omega)\right) \to 1.$$

The proof of Theorem 3.6 is immediate from the properties of the Frobenius norm and Theorems 3.1, 3.2, and 3.5.

3.4. **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 sequence of local maximizers 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 likelihood function, our intuition from classical maximum likelihood theory continues to hold true. In order to simultaneously guarantee consistency in estimation and in model selection, it is necessary that these parameters vanish asymptotically.

The conditions on $\tau(\lambda_n)$ 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—recall that 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 fast enough, 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.

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 $\max_{i,j} p_{\lambda_n}(|\phi^0_{ij}|)$. By rate of convergence, we refer to the fact that the assumptions on $a_n(\boldsymbol{\nu}_0)$ and $b_n(\boldsymbol{\nu}_0)$ imply that $p_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n) + o(1)$ for each $i \neq j$, and 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 3.1 and 3.2. Since $\tau(\lambda_n)$ must be bounded below asymptotically by $n^{-1/2}$, the convergence rate of $\max_{i,j} p_{\lambda_n}(|\phi^0_{ij}|)$ essentially must be exactly $n^{-1/2}$. When p_λ is a quadratic spline, such as the MCP or SCAD, the zeroth-order condition simply becomes $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$. In practice, it is sufficient to have $p_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n)$ 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 each of the conditions simultaneously, along with associated regularization parameters λ_n. When the penalty is chosen to be the MCP, we note that all of the conditions required for Theorem 3.6 are satisfied as long as

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

This shows how the concavity parameter must be correctly calibrated in order to guarantee consistency. Thus, given a sequence of λ_n that satisfy the required growth conditions, it merely suffices to choose γ_n so that (22) is satisfied, which can always be done.

Remark 3.2. 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, since we assume that $p'_\lambda(t) = \lambda\rho'(t) = 0$ for sufficiently large $t > 0$, we have $\lim_{t\to\infty} \rho(t) < \infty$ independent of the choice of λ. Thus $\tau(\lambda_n)/\lambda_n$ is constant and the conditions on λ_n in Theorem 3.6 become $\sup_n \lambda_n < \infty$ and $\lambda_n n^{1/2} \to \infty$. When $\lambda_n \to 0$, these conditions are simply the assumptions in Theorem 3.2.

The zeroth-order condition on $\max_{i,j} p_{\lambda_n}(|\phi^0_{ij}|)$ is trickier. If we assume that the function $\rho(t)$ does not depend on any other parameters, it is necessary that $\max_{i,j} p_{\lambda_n}(|\phi^0_{ij}|) = \tau(\lambda_n)$ for all n, which is restrictive. Penalties such as the MCP and SCAD, however, depend on an additional parameter which allows us to relax this restriction.

Remark 3.3. Although the usual formula for the MCP does not satisfy this factorization property, we may reparametrize it so that it does. To do this, define a new penalty by

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

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 (11). Furthermore, the condition in (22) becomes

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

which is independent of λ_n.

3.5. **High-dimensions and the oracle property.** The theory in this section so far has assumed that p is fixed with $n > p$, the so-called classical low-dimensional scenario. It would be interesting to obtain results for this method when $p = p_n$ is allowed to depend on n, and in particular the case when $p_n > n$. While the simulations in Section 5 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 2.1 how to modify our method in order to estimate the equivalent DAG that is consistent with π, which we have denoted by $(\tilde{B}_0(\pi), \tilde{\Omega}_0(\pi))$. When the order of the variables is fixed, our problem reduces to standard multiple regression with a concave penalty, in which case Theorems 3.1 and 3.2 can be generalized to high-dimensions, for instance using (Zhang and Zhang (2012); Fan and Lv (2010)). This is very much 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 ordering is optimal, so this does not tell the whole story. Theorem 3.5 shows how our estimator selects the right ordering automatically based on the data, and eliminates the need to assume prior knowledge of this ordering.

In Section 2.4, we discussed some positive results in the ℓ_0 case in which it is not assumed that π is known in advance. The key idea from 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 concave penalties, however, the details remain to be worked out and are expected to be technical.

One of the most attractive properties of concave regularization is the oracle property, which has been established in the context of regression under weak assumptions (Zhang and Zhang (2012); Fan and Lv (2010)). Additionally, Zhang and Zhang (2012) show how concave penalized least squares estimators are "close" to global ℓ_0 solutions under certain assumptions. As we wish to keep the focus on methodology, we do not go into the details here. Current work is underway to establish similar oracle properties for the estimator defined in (13, 14).

4. 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 reparametrization (12), however, we will be able to perform the single parameter updates and enforce the constraint with ruthless efficiency.

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) apply, 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.

4.1. **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

$$Q(\Phi, R) := L(\Phi, R) + \sum_{i,j} p_\lambda(|\phi_{ij}|) \tag{23}$$

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

$$Q_1(\phi_{kj}) = \frac{1}{2} \left\| \rho_j x_j - \sum_{i=1}^{p} \phi_{ij} x_i \right\|^2 + p_\lambda(|\phi_{kj}|), \tag{24}$$

$$Q_2(\rho_j) = -n \log \rho_j + \frac{1}{2} \left\| \rho_j x_j - \sum_{i=1}^{p} \phi_{ij} x_i \right\|^2. \tag{25}$$

Q_1 is the function Q 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}. Q_2 is the corresponding function for the parameter ρ_j. Note that Q_1 and Q_2 both depend on k and j, and we express this dependence through the 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 reparametrization (CCDr).

4.2. **Coordinate descent.** In what follows, we assume that the data has been appropriately normalized so that each column x_j has unit norm, $\|x_j\|^2 = \sum_h x_{hj}^2 = 1$. Furthermore, although the details of the algorithm do not depend on the choice of penalty, we will only work out the details for MCP.

4.2.1. *Update for ϕ_{kj}.* Mazumder et al. (2011) show that the minimum of (24) can be found by solving

$$\arg\min_{\beta} Q^1(\beta), \quad \text{where } Q^1(\beta) := \frac{1}{2}(\beta - \tilde{\beta})^2 + p_\lambda(|\beta|). \tag{26}$$

For the MCP with $\gamma > 1$, the solution is given by the so-called threshold function

$$S_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \le \lambda, \\ \operatorname{sgn}(\tilde{\beta}) \left(\frac{|\tilde{\beta}| - \lambda}{1 - 1/\gamma} \right), & \lambda < |\tilde{\beta}| \le \lambda\gamma, \\ \tilde{\beta}, & |\tilde{\beta}| > \lambda\gamma. \end{cases} \tag{27}$$

To see how to convert (24) into (26), note that

$$\begin{aligned} Q_1(\phi_{kj}) &= \frac{1}{2} \sum_{h=1}^n \left(\rho_j x_{hj} - \sum_{i \neq k} \phi_{ij} x_{hi} - \phi_{kj} x_{hk} \right)^2 + p_\lambda(|\phi_{kj}|) \\ &= \frac{1}{2} \sum_{h=1}^n x_{hk}^2 \left(\frac{1}{x_{hk}} r_{kj}^{(h)} - \phi_{kj} \right)^2 + p_\lambda(|\phi_{kj}|), \end{aligned} \tag{28}$$

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

$$Q_1(\phi_{kj}) = \frac{1}{2} \left\{ \sum_{h=1}^n (r_{kj}^{(h)})^2 - 2\phi_{kj} \sum_{h=1}^n x_{hk} r_{kj}^{(h)} + \phi_{kj}^2 \right\} + p_\lambda(|\phi_{kj}|) \tag{29}$$

$$= \frac{1}{2} \left(\phi_{kj} - \sum_{h=1}^n x_{hk} r_{kj}^{(h)} \right)^2 + p_\lambda(|\phi_{kj}|) + \text{const}. \tag{30}$$

The constant term in (30) 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 (26) with $\tilde{\beta} = \sum_h x_{hk} r_{kj}^{(h)}$, and hence for MCP with $\gamma > 1$,

$$\arg\min Q_1(\phi_{kj}) = S_\gamma \left(\sum_h x_{hk} r_{kj}^{(h)}, \lambda \right). \tag{31}$$

The existence of a closed-form solution to the single parameter update for ϕ_{kj} is one of the main reasons we employ MCP for our penalty. Many other penalty functions, however, allow for closed-form solutions to (26), and our algorithm applies for any such penalty function.

4.2.2. *Update for ρ_k.* The single parameter update for ρ_j is straightforward to compute and is given by

$$\arg\min Q_2(\rho_j) = \frac{c + \sqrt{c^2 + 4n}}{2}, \quad \text{with } c = \sum_{i \neq j} \phi_{ij} \sum_h x_{hi} x_{hj}. \tag{32}$$

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

4.3. **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 chose to fix $\gamma = 2$ and postpone further study of the algorithm'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 log-scale. One can easily check that if we choose 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 somehow 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. See Wang et al. (2007) for some positive results concerning the SCAD penalty, and Fu and Zhou (2013) for a 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.

4.4. **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 (31) can be written as

$$\sum_{h=1}^{n} x_{hk} r_{kj}^{(h)} = \rho_j \langle x_j, x_k \rangle - \sum_{i \neq k} \phi_{ij} \langle x_i, x_k \rangle. \tag{33}$$

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 (32), which highlights why the reparametrization (12) is useful: the single parameter update for each ρ_j only requires $O(p)$ operations, which is an order of magnitude smaller than the computation of the standard residual estimate for ω_j^2 in the original parametrization. Since we perform p of these updates in each cycle, we reduce the total number of operations per cycle from $O(p^3)$ down to $O(p^2)$, which is a substantial savings. Moreover, by leveraging sparsity, both (31) and (32) 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 very 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. Moreover, as shown in Section 5.4, the performance of the CCDr algorithm is less competitive as s_0 becomes much larger than p.

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

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.

our algorithm is suboptimal. 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*.

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

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

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

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

5. Numerical Examples and Results

In order to assess the accuracy and efficiency of the CCDr algorithm, we analyze the consistency of our estimates and then compare our algorithm with the PC algorithm. We emphasize once more that due to the scale of problems we are interested in, the PC algorithm is the only known algorithm that is feasible for comparison. We are mainly interested in the accuracy and timing performance of each algorithm as a function of the model parameters (p, s_0, n).

In our first test, we assess the accuracy of the estimated inverse covariance matrices, given by $\widehat{\Theta} = \Theta(\widehat{B}, \widehat{\Omega})$, as a function of the sparsity of the estimates. This will be done by computing an R^2 goodness-of-fit statistic. We then move on to a side-by-side comparison of the PC algorithm with the CCDr algorithm, in which we will observe the following general behaviour:

(1) When $s_0 < p$, CCDr outperforms PC in both accuracy and timing,
(2) When $s_0 \geq p$, PC outperforms CCDr in accuracy, but not timing.

The scenario in (1) will be referred to as the *ultra-sparse regime*, while case (2) will be called the *standard sparsity regime* or simply the *sparse regime*. We also run additional simulations to give a more complete picture of the sparse, high-dimensional scenario in which CCDr performs best. In particular, even when $n \ll p$, CCDr shows very good performance.

In what follows, we used the R package `pcalg` (v1.1-5) and a significance level of $\alpha = 0.01$ for all tests involving the PC algorithm. All of the tests cited here 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.

5.1. **The setting.** A total of four simulated experiments were run, each described in Sections 5.3-5.5. In each experiment, an array of values were chosen for each of the three main DAG 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). 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 model parameters were fixed at

$$\beta_{ij}^0 = \begin{cases} 1, & \text{if } X_i \to X_j \\ 0, & \text{otherwise} \end{cases},$$
$$\omega_j^0 = 1 \quad \text{for all } j.$$

The CCDr algorithm has three 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: $\varepsilon = 10^{-2}$, $M = p^{1/2} \vee 10$, and $\alpha = 3$. Recall that in the full algorithm (see Section 4.5), for each λ_i there are at most $M^2 = p \vee 100$ sweeps. When p is very small a maximum of 100 iterations is more than enough.

5.2. **Performance metrics.** In relying on different assumptions, the outputs of the PC algorithm and the CCDr algorithm are subtly different. Under the assumption that the underlying distribution is faithful to a DAG, one can represent the entire Markov equivalence class of DAGs by a *completed partially directed acyclic graph* (CPDAG for short, see Chickering (2003) for the relevant definitions). The output of the PC algorithm is accordingly a CPDAG, as opposed to a DAG. Given a DAG estimated by the CCDr algorithm, we may in turn compute the CPDAG to which this estimate belongs, and compare this CPDAG to the CPDAG produced by the PC algorithm. In our evaluations, we compared both skeletons and CPDAGs for completeness.

Let P = number of estimated edges, T = number of edges in B_0, $F = \binom{p}{2} - T$, TP = true positives, and FP = false positives. We gauged the performance of the algorithms on the following metrics:

(1) True positive rate (TPR $= TP/T$),
(2) True discovery rate (TDR $= TP/P$),
(3) False positive rate (FPR $= FP/F$),
(4) Structural Hamming distance (SHD),
(5) SHD for the estimated CPDAG,
(6) Computational time in seconds,
(7) R^2 goodness-of-fit.

A few comments on these metrics are in order. In some literature, TPR is also variously called *recall* or *sensitivity*, and TDR is called *precision* or *specificity*. Metrics (1-4) were computed by comparing the estimated skeleton against the skeleton of the true graph. It is worth emphasizing that as a consequence, the TDR values cited in the subsequent sections *ignore directionality of the estimated graphs*. Metric (5) was computed by comparing the estimated CPDAG against the true CPDAG. The `pcalg` function `dag2cpdag` was used to convert the CCDr estimates to CPDAGs for comparison. Due to the heavy computational burden associated with computing $L + 1$ CPDAGs for each CCDr model, comparisons of CPDAGs were excluded for the case $p = 1000$ in Section 5.4.1 and all of the high-dimensional tests discussed in Section 5.4.2.

For the precise definition of the structural Hamming distance, see Tsamardinos et al. (2006). Intuitively, SHD measures the number of edge reversals, additions, and/or removals necessary to convert an estimated graph into the true graph. This is a useful metric since it gives an absolute sense of "how far" away the estimates are from the true graph.

The R^2 goodness-of-fit in metric (7) is defined by the following normalized residual factor,

$$R^2(\Theta) = 1 - \frac{\|\Theta - \Theta_0\|_F^2}{\|nI_p - \Theta_0\|_F^2}, \tag{34}$$

where $\|\cdot\|_F$ is the usual Frobenius norm for matrices. The term in the numerator is simply the residual sum of squares in estimating Θ_0, and the denominator is the residual sum of squares assuming a null model with $B_0 \equiv 0$. This statistic gives us a sense of how much improvement our estimates provide over the null model. For any reasonable estimator—one that does not

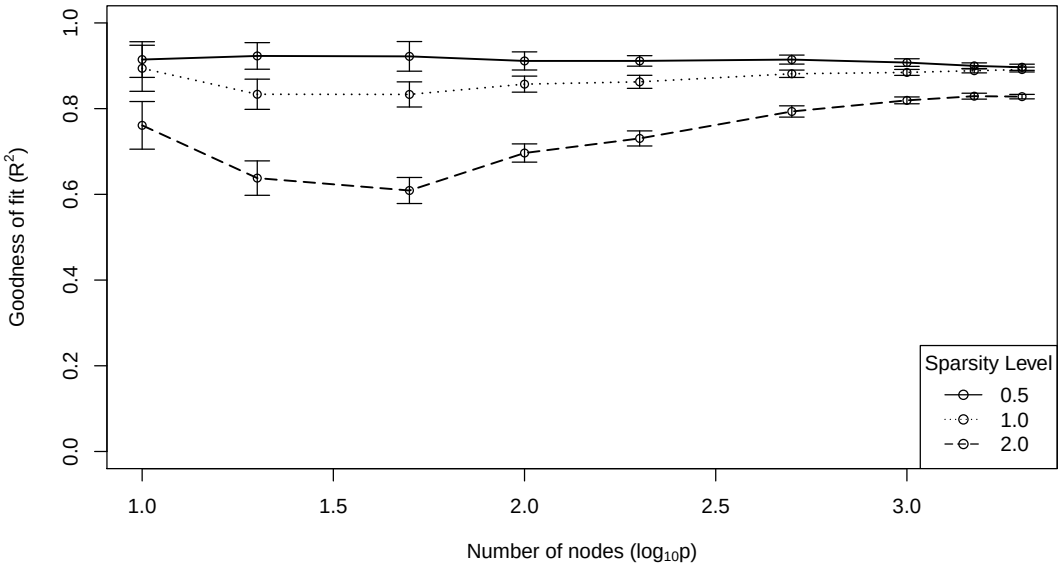


FIGURE 1. $R^2(\widehat{\Theta})$ for different sparsity levels (s_0/p), with 95% confidence intervals.

perform worse than the null estimate—we have $0 \leq R^2(\widehat{\Theta}) \leq 1$. Since this statistic is meant to evaluate the performance of an estimator that approximates Θ_0, we did not compute this value for the PC algorithm.

Since both algorithms require the empirical correlation matrix of X as an input, the time to compute this matrix is included in the timing for each algorithm. All of the timing was done in R, and represents the raw processor time it took for each algorithm to compute the correlation matrix, run the main steps of the algorithm, and manipulate the output into R-friendly data structures. In some cases for the CCDr algorithm, the last step was dominant, but we included it since we did not perform similar fine-grained benchmarking for the `pcalg` function `pc`. Finally, when comparing the timing data it is important to note that the computational time for the CCDr algorithm is the *total* time to compute all the models $(\widehat{\Phi}(\lambda_i), \widehat{R}(\lambda_i))$ for $i = 0, 1, \ldots, L$. The average time per model is actually much lower (see Section 5.5).

5.3. **Goodness-of-fit.** In order to measure the estimation performance of CCDr, we first ran a series of tests intended to assess the goodness of fit of the estimated models using the R^2 statistic defined in (34). Because it does not seek to fit a parametric model, this assessment does not apply to the PC algorithm.

We ran $N = 20$ tests using all combinations of the following parameter settings:

- $p \in \{10, 20, 50, 100, 200, 500, 1000, 1500, 2000\}$;
- $s_0/p \in \{0.5, 1.0, 2.0\}$;
- $n = 1000$.

For each test, given a randomly generated DAG B_0, we computed the oracle DAG by regressing each node on its parents in B_0, using the simulated data X. This yielded an oracle (B_0^o, Ω_0^o), with corresponding $\Theta_0^o = \Theta(B_0^o, \Omega_0^o)$. The R^2 value $R^2(\Theta_0^o)$ could then be used as a standard against which to judge the CCDr estimates. We then ran the CCDr algorithm and similarly used an OLS regression to refit the model parameters for each estimated edge. This produced a final estimate $\widehat{\Theta}_i = \Theta(\widehat{B}_i^{OLS}, \widehat{\Omega}_i^{OLS})$ for each λ_i, for which we computed $R^2(\widehat{\Theta}_i)$. We then selected the model with the ratio $\hat{s}_i/s_0$ closest to one, in order to give us a sense of how good the fit of our estimate was when the sparsity level was fixed to be approximately at the true level. While it may seem like this choice artificially inflates the R^2 values, in fact, as with other goodness-of-fit measures, $R^2(\Theta)$ increases monotonically as a function of the model complexity, and so sparser graphs have smaller R^2 in general. This behaviour was indeed observed in our simulations.

The results are given in Figure 1. The oracle R^2 values were, as expected, all very close to 1. In fact, they were all larger than 0.98 with an average of 0.997. This gives us good empirical

validation for the statistic defined in (34). By comparison, 75% of the CCDr estimated models satisfy $R^2 \geq 0.8$ with a third of the models achieving $R^2 \geq 0.9$. The most striking feature of the results is that R^2 does not depreciate significantly as p increases all the way up to $p = 2000$, where the dimension is much higher than the sample size $n = 1000$. This shows that the CCDr has very good estimation performance in terms of both sparsity and goodness of fit.

5.4. **Comparison with the PC algorithm.** In this section we provide detailed results comparing the two algorithms across a wide range of choices of (p, s_0, n) using first six metrics described in Section 5.2.

In order to properly compare the two algorithms, a single model needed to be selected from each sequence of estimates $(\widehat{\Phi}(\lambda_i), \widehat{R}(\lambda_i))$ generated by CCDr. To make fair comparisons, we used the following heuristic: Given the estimated PC graph, we chose the CCDr model whose TPR was closest to the TPR of the PC graph. This allows us to compare the relative cost of using CCDr over PC. That is, for a given TPR, how do the algorithms compare on the various metrics? If we employed a more sophisticated model selection criterion, the data show that CCDr could obtain much better TDR at only a modest sacrifice to TPR (see Section 5.6). Since we have chosen not to address the model selection problem here, however, we do not reward our algorithm for models that cannot be reliably discerned based on current work.

5.4.1. *Various parameter settings.* We first tested the two algorithms with the following parameter choices:

- $p \in \{10, 20, 50, 100, 200, 500, 1000\}$;
- $s_0/p \in \{0.2, 0.5, 1.0, 2.0\}$;
- $n/p \in \{0.2, 1, 10\}$.

These choices of model parameters allow us to compare the two algorithms in the most common scenarios encountered in practice: (1) Small p ($p < 100$), (2) Large p ($p \geq 100$), (3) High-dimensions ($n < p$), (4) Low-dimensions ($n \geq p$), (5) Ultra-sparse ($s_0 < p$), (6) Sparse ($s_0 \geq p$). In scenario (6), each node has at least one parent on average. In scenario (5), there are many nodes with no edges which represent independent variables, and thus is reminiscent of variable screening.

For $p \leq 500$, we ran $N = 50$ tests each. Due to time constraints, we could only run $N = 20$ tests each for $p = 1000$. This represents a total of 3,840 tests. The results are shown in Figure 2. This figure shows a clear separation into three separate parameter regimes. In the upper left corner, when n and s_0 are small relative to p, CCDr outperforms PC. In the lower right corner, when n and s_0 are larger than p, the PC algorithm outperforms CCDr. In the middle of the figure, when $n \approx s_0 \approx p$, the algorithms are comparable. To offer a sense of the overall picture of the algorithm performance, Table 1 gives the averages over all 3,840 tests of the various statistics for each algorithm. We have chosen to omit the CPDAG statistics from this table since we did not compute CPDAGs for $p = 1000$. When restricted to $p \leq 500$, the SHDs for the estimated CPDAGs were 92.61 and 116.02 (p-value $< 10^{-3}$), respectively for CCDr and PC.

Since Figure 2 only gives relative timing information, we provide some absolute timing data in Table 2. This table gives the average total running time in R for each algorithm, averaged across all (s_0, n) for each value of p tested. We see that the CCDr algorithm is generally faster when compared with the PC algorithm. There are, however, certain cases in which PC is more efficient, which are not evident from this table: the CCDr algorithm slows down when the true graph is dense or when there is limited data. The former is of course expected since the active set rapidly expands as λ decreases, and the latter bottleneck is likely due to slowed convergence when n is small.

Figures S1 and S2 in the Supplementary Materials compare the average estimation performance of the two algorithms between the two sparsity regimes across all six metrics. The metrics are plotted as functions of the number of nodes in the true graph, p, on a log-scale. Each point corresponds to the average performance of an algorithm for all models that fit the designated sparsity level, averaged over all choices of n, along with 95% confidence intervals for each p. The general pattern is clear: in the ultra-sparse regime, CCDr outperforms PC on all six metrics. Note in particular the difference in SHD as p increases. When $s_0 \geq p$, however, PC outperforms CCDr, while the gap in SHD is smaller than in the ultra-sparse regime. This gives us good quantitative justification for the pattern observed in the previous paragraphs.

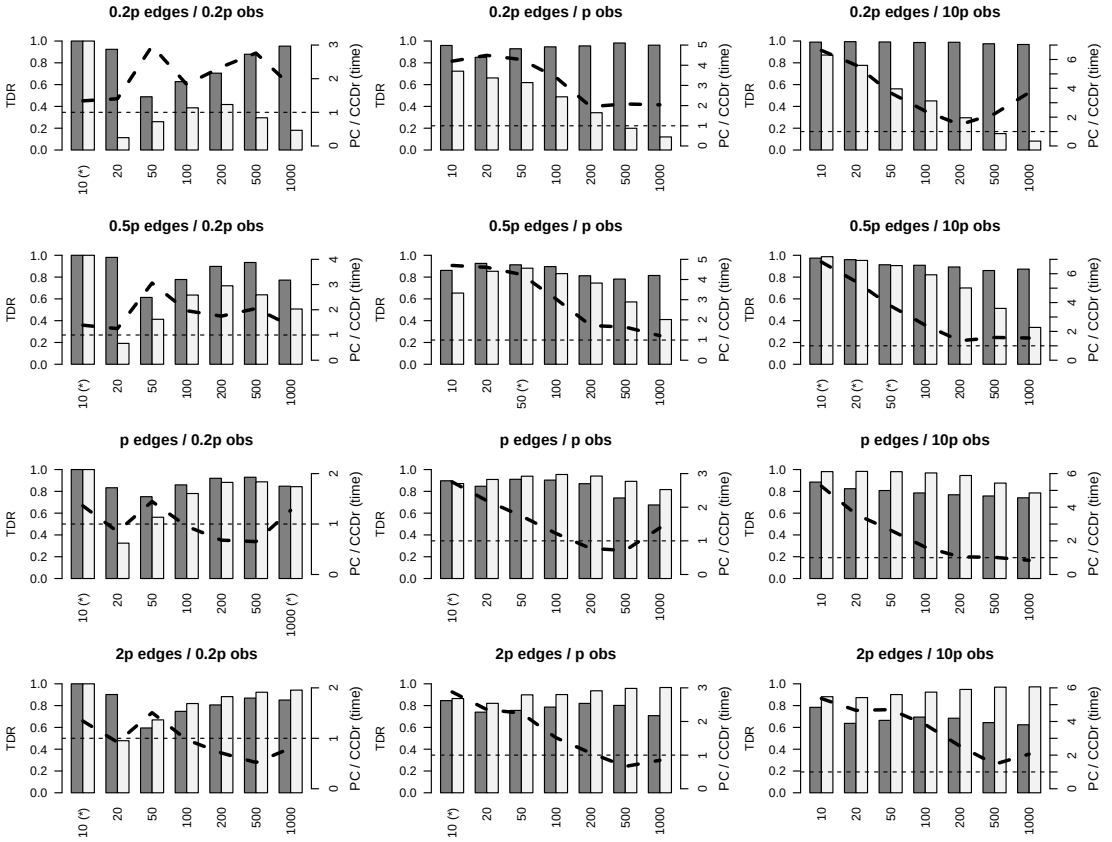


FIGURE 2. Comparison of the performance of CCDr algorithm with that of the PC algorithm. The bars represent TDR values for the estimated skeletons, where the x-axis corresponds to the number of nodes in each graph (p). The dark bars represent CCDr, and a (*) below a pair of bars indicates that the difference in the TDR values between the two algorithms is not significant at the 5% level. The overlaid dashed line is the ratio of the PC processor time to the CCDr processor time.

TABLE 1. Average estimation performance of CCDr and PC.

	CCDr	PC	p-value
TPR	0.6401	0.6419	0.8222
TDR	0.8471	0.7258	$< 10^{-3}$
FPR	0.0071	0.0041	$< 10^{-3}$
SHD (skeleton)	89.00	116.46	$< 10^{-3}$
Timing (s)	18.90	24.20	0.0062

5.4.2. *The high-dimensional case.* In the previous section we observed that the CCDr algorithm exhibits good performance when n is very small relative to p, which was anticipated since the CCDr algorithm was designed to better handle very small sample sizes through its use of concave regularization. To further test this hypothesis, we performed more tests using the following parameter choices:

- $p \in \{100, 200, 500\}$;
- $s_0/p \in \{0.5, 1.0, 2.0\}$;
- $n/p \in \{0.1, 0.15, 0.2, 0.25\}$;
- $N = 50$ for all models.

These tests give us a better sense of the performance of the algorithms when the sample size is very small. The results are shown in Figure 3, with a summary provided in Table 3. As

TABLE 2. Average timing of CCDr and PC (in seconds).

p	CCDr	PC	p-value
10	0.0079 (0.0027)	0.0269 (0.0156)	$< 10^{-3}$
20	0.0281 (0.0155)	0.0656 (0.0367)	$< 10^{-3}$
50	0.1014 (0.0612)	0.2925 (0.2660)	$< 10^{-3}$
100	0.5602 (0.3758)	1.158 (1.344)	$< 10^{-3}$
200	3.568 (2.766)	5.117 (6.480)	$< 10^{-3}$
500	40.24 (45.33)	42.31 (53.69)	0.4708
1000	191.1 (183.1)	264.7 (291.4)	0.00999

Note: The numbers in parentheses represent standard deviations across all choices of (s_0, n) tested for each value of p, a total of $N = 600$ tests when $p < 1000$ and $N = 240$ when $p = 1000$.

anticipated, the CCDr algorithm shows significantly better performance without a significant computational cost. This again provides strong empirical evidence that our method should have good theoretical properties in the sparse, high-dimensional regime (see Section 3.5).

Figure 3 must be interpreted carefully. For instance, the case $n = 0.1p$ corresponds to sample sizes of $n = 10, 20, 50$ respectively for $p = 100, 200, 500$. As expected, the PC algorithm shows improved performance as n gets larger, but when n is at its smallest, CCDr convincingly outperforms PC. This is highlighted, for example, when $p = 100$, $s_0 = 50$, and $n = 10$. In this case, CCDr had an average TDR of 59.1% compared to 32.8% for PC, an almost two-fold improvement when the sample was extremely limited. In the presence of more samples when $p = 500$ and $n = 50$, the PC algorithm performed much better (64.8%), but still significantly worse than CCDr (91.3%). What's more, this improvement in accuracy comes at no cost to efficiency—in this case the CCDr algorithm is still significantly faster than the PC algorithm.

5.5. **Timing considerations.** The previous section offered a detailed comparison of the real processor time required for running the CCDr and PC algorithms when $p \leq 1000$. In order to test the limits of the CCDr algorithm, we ran further tests with $p > 1000$. Since the timing of the algorithm depends crucially on the total number of models estimated, and also on the threshold parameter α, instead of comparing the full running time of the algorithm as in the previous section, here we consider the average running time per λ. This gives us a better sense of how the algorithm scales for more complicated models, and how future improvements in selecting λ would affect the total running time of the algorithm.

Since the timing is acutely dependent on the relationship between the dimension, the sparsity of the true graph, and the number of independent observations, 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 $s_0/p = 1.097$. The results are displayed in Figure 4. Here, the dashed line counts only the time to run the optimization step, omitting the timing for precomputing the correlation matrix for X, which takes 15s on average when $p = 2000$, as well as the extra computational time required to perform object management in R (see Section 5.2). The thick solid line represents the average processor time including these extra steps. When p is small, the difference is negligible, however, when p becomes large the postprocessing step takes up more and more time. When $p = 2000$, the total computational cost is roughly two minutes per λ, but 25% of this time is spent in postprocessing.

We also briefly mention here some tests when $p > 2000$. Because models of this size are burdensome to test, we did not have the luxury of running a variety of repeated simulations. We can report, however, that for graphs with $(p, s_0, n) = (2500, 2500, 5000)$, the largest graphs

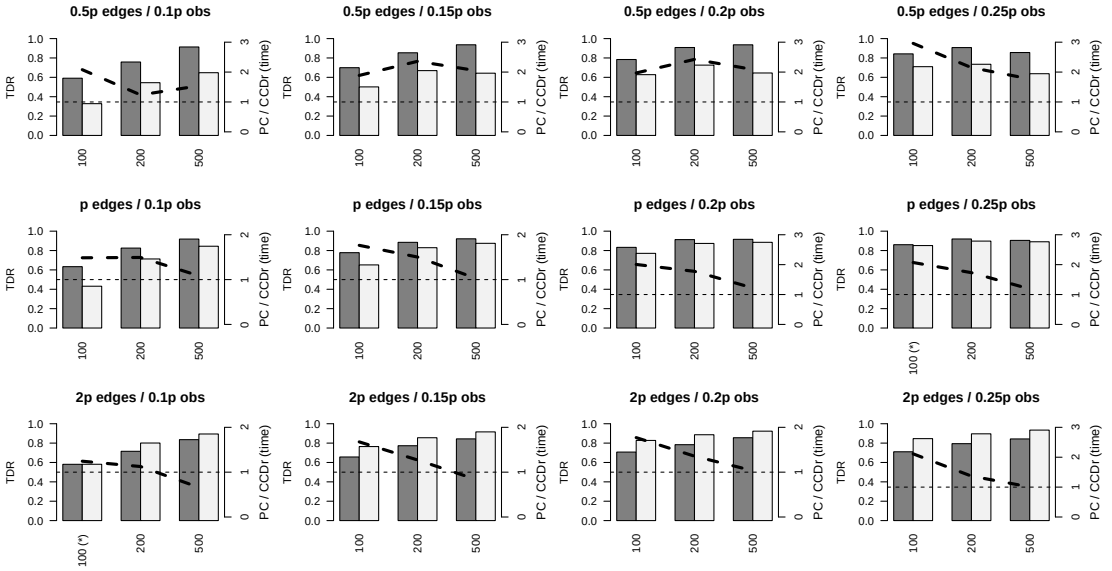


FIGURE 3. Comparison of the performance of CCDr algorithm with the PC algorithm when $p \gg n$, plotted using the same format as Figure 2.

TABLE 3. Average estimation performance of CCDr and PC when $p \gg n$.

	CCDr	PC	p-value
TPR	0.4692	0.4676	0.8497
TDR	0.8164	0.7515	$< 10^{-3}$
FPR	0.0011	0.0013	$< 10^{-3}$
SHD (skeleton)	185.07	194.94	0.0951
Timing (s)	7.724	8.866	0.0026

tested in the literature that we are aware of, the main optimization step takes 30 minutes, with just under one minute of this time to compute the correlations.

5.6. **Further discussion.** The simulations presented here were deliberately designed to handicap the CCDr algorithm by creating a fair comparison with the PC algorithm. If we instead consider the most accurate model produced by the CCDr, then we get a glimpse at its potential. To quantify this, we calculated how the overall TDR is affected by selecting a different model. Specifically, for all the models computed in the first comparison test (Section 5.4.1), instead of selecting a CCDr model by matching TPR with PC as closely as possible, we chose the model with the highest TDR among those CCDr models whose TPR was within $\pm\delta$ of the PC algorithm. For each $\delta \in \{0.01, 0.02, \ldots, 1\}$, we calculated the average improvement in TDR across all simulations as well as the proportion of models that showed any improvement. The results are plotted in Figure 5. To summarize, by sacrificing the TPR by at most 10%, 53% of models were improved with an average increase in TDR of 10.3%. If we sacrifice at most 20% on average, 61% of models were improved with an average increase of 14.6%. In actuality, the average cost in TPR when capped at 10% was only 6.4%, and 14.0% when capped at 20%. These numbers highlight both the potential advantages of CCDr as well as the need for a consistent model selection criterion.

In both algorithms, we kept key parameters fixed in order to keep the number of free parameters down to a reasonable size. For the CCDr algorithm, we fixed $\gamma = 2$, although this parameter was observed to have a non-negligible effect on the results. A more in-depth study in the future would account for the effect of this parameter. For the PC algorithm we fixed $\alpha = 0.01$, which was determined in Kalisch and Bühlmann (2007) to be in the optimal range based on tests designed to minimize the SHD of estimates. Moreover, these tests showed that the average SHD does not fluctuate by a substantial amount as α changes. That being said, allowing $\alpha = \alpha_n$ to change with n might lead to better results with the PC algorithm.

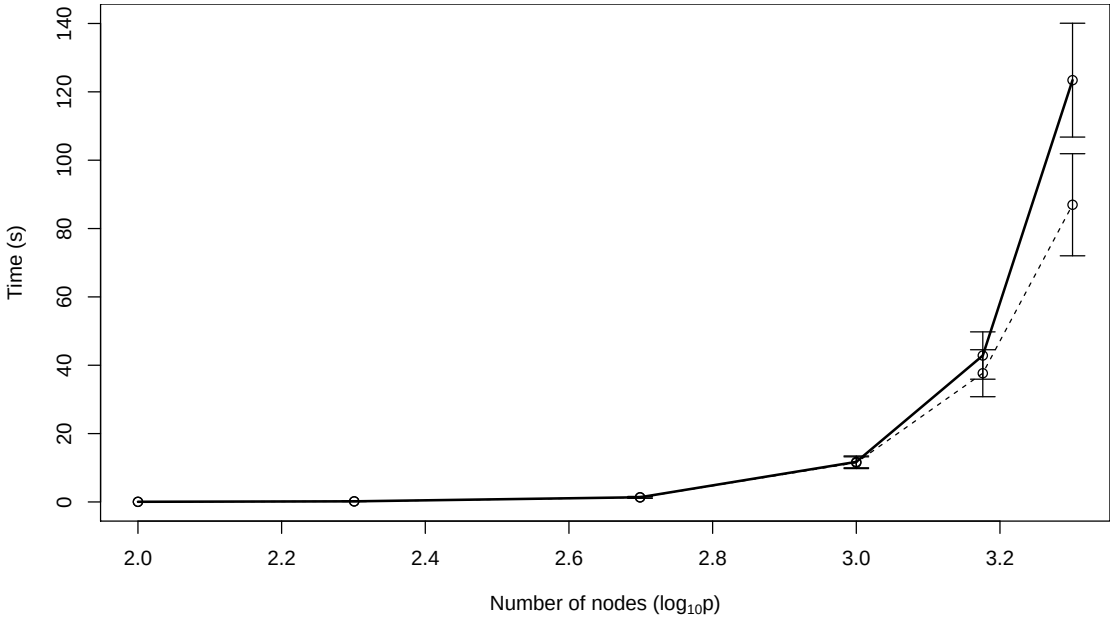


FIGURE 4. Average processor time to compute a single model $(\widehat{\Phi}(\lambda_i), \widehat{R}(\lambda_i))$ for the CCDr algorithm including 95% confidence intervals over 20 random simulations. The solid line includes both pre- and post-processing steps; the dashed line counts only the optimization step.

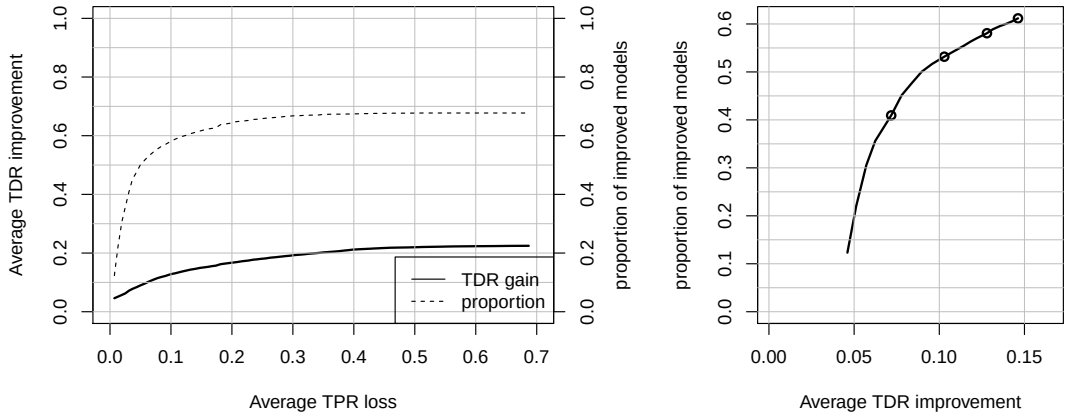


FIGURE 5. (Left) Improvement for CCDr (measured by change in TDR) as a function of the average loss in TPR. (Right) Alternative plot showing the proportion of models which are improved as a function of the average TDR change with $\delta \leq 0.2$. The four dots correspond to the points at which the maximum losses in TPR (δ) are given by 0.05, 0.1, 0.15, and 0.2. See Section 5.6 for details.

One model parameter which we have not discussed is the neighbourhood size, 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, and is the main reason we bounded this quantity in our simulations. 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. For brevity we have chosen to omit the details.

6. Application to Real Data

We provide in this section some tests comparing the two algorithms on real data. We analyzed the well-known flow cytometry dataset, 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 contains $n = 7466$ measurements of $p = 11$ continuous variables corresponding to proteins and phospholipids in human immune system cells. The consensus network, constructed by experimental annotations, has $s_0 = 20$ edges, and so falls into the standard sparsity regime. Hereafter, we regard this consensus network as the true network in order to assess the algorithms. The data contains a mixture of both observational and experimental data. We also analyzed a discretized version of the dataset which contains $n = 5400$ measurements.

Since both the PC and CCDr algorithms assume the data is 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. The results of estimating the true graph are shown at the top of Figure 6. For model selection, we chose the best CCDr model with $\leq 5\%$ difference in TPR compared to the PC model. As expected, the CCDr model is sparser with a much smaller false positive rate (17% vs. 37%), higher true discovery rate (60% vs. 43%), and lower SHD (24 vs. 31). For comparison purposes, the estimates obtained by using the untransformed data are given in Supplementary Figure S3.

The discrete version of the dataset was 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)). Since this data contained only three levels, we did not transform the data, and simply ran both algorithms on the supplied data. The results are shown at the bottom of Figure 6. The same pattern emerges here: CCDr estimates a sparser graph with a smaller false positive rate, higher true discovery rate, and lower SHD.

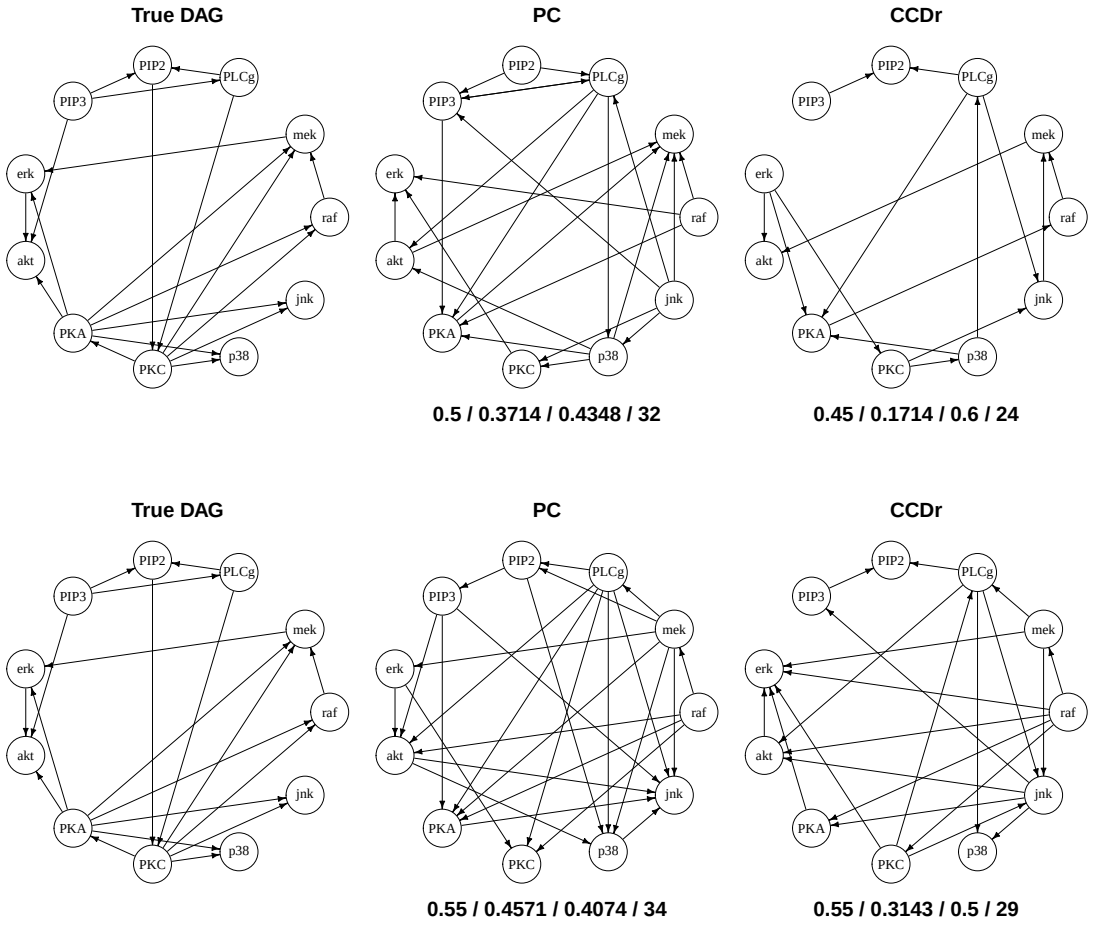


FIGURE 6. Estimated DAGs for the cytometry dataset. The numbers below each graph are TPR / FPR / TDR / SHD (CPDAG). (top) Log-transformed continuous data ($n = 7466$). (bottom) Discretized data ($n = 5400$).

7. Conclusion

We have introduced a new penalized likelihood framework for estimating sparse Bayesian networks, along with a fast, scalable algorithm that is easily implemented on a personal computer. In the finite dimensional scenario, our 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 can handle networks with up to 2500 nodes as well as the high-dimensional case where $p \gg n$, and even outperforms the well-known PC algorithm in both speed and accuracy when the true graph is very sparse.

We would like to emphasize that our results do *not* indicate that the CCDr algorithm represents a substitute for the PC algorithm. In fact, while both algorithms ultimately estimate the structure of a DAG, they are solving fundamentally different problems. The PC algorithm seeks to estimate the structure of a DAG that is faithful to an observational distribution, while the CCDr algorithm solves a parametric problem while simultaneously estimating a sparse DAG that is consistent with the observed data. The objective of this comparison was merely to benchmark our new framework against one of the current state of the art algorithms for Bayesian network inference.

The central theme of exploiting convexity to solve nonconvex problems is an intriguing prospect for the development of new algorithms in computational statistics. Indeed, historically the main difficulty with nonconvex regularization has been computation. Although recent progress has broken this barrier in the case of least squares regression, to our knowledge the algorithm presented here is the first to solve such a highly nonconvex statistical optimization problem—whose dimension scales quadratically—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. Such approaches tend to scale very poorly as the number of nodes increases due to the combinatorial nature of the algorithms. Unlike these methods, future advances in nonconvex optimization will directly affect how we solve the maximum likelihood problem presented here.

Perhaps even more exciting is the potential for extending this algorithm in new ways. Our focus on the Gaussian linear model was merely for convenience, and we have discussed how both our penalized likelihood framework and the coordinate descent algorithm apply equally well to any model for which the likelihood can be written down. One could imagine extending this framework to more general loss functions as well. It would be of great interest to extend these ideas to structural inference of Bayesian networks which are not necessarily Gaussian, or for which the functional relationships are allowed to be nonlinear.

Appendix

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 reparametrization (Φ, R) given by

$$\Phi = B\Omega^{-1/2} \tag{35}$$

$$R = \Omega^{-1/2}. \tag{36}$$

This is of course just the matrix version of the reparametrization that leads to (12). Now define the following function which maps $(\Phi, R) \in \mathbb{R}^{p\times p} \times \mathbb{R}^{p\times p}$ into $\mathbb{R}^{p^2}$:

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

In the sequel, 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 parametrizations. 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 recast in terms of the original model (B, Ω).

We formally define the space of DAGs as follows:

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

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 $F(\boldsymbol{\nu}) = \ell_n(\boldsymbol{\nu}) - np_{\lambda_n}(\boldsymbol{\nu})$. We make the following assumptions on the penalty function p_λ:

(1) p_λ is concave and piecewise twice differentiable,
(2) p_λ is nonnegative, nondecreasing and $p'_\lambda(t) = 0$ for all large t,
(3) $p_\lambda(0) = 0$ for all λ,
(4) $0 < \lim_{t\to 0+} p'_\lambda(t) < \infty$ for all λ.

Condition (4) ensures that the penalty is singular with a well-defined right-hand derivative at the origin, and combined with Conditions (1) and (2) this implies that p_λ is Lipschitz continuous for each λ.

A.2. **Proofs.** We first prove Lemma 3.3, which will be used in the proof of the Theorem 3.1.

Proof of Lemma 3.3. We only prove this for the original parametrization (B, Ω); the reparametrized 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

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

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 3.1. 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 3.1). Since the densities $f(\cdot \mid \boldsymbol{\nu})$ 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 3.1 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 in a neighbourhood of $\boldsymbol{\nu}_0$. Since f is always a Gaussian density, it will suffice to show that $f(\cdot \mid \boldsymbol{\nu}) \neq f(\cdot \mid \boldsymbol{\nu}_0)$ 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 $\phi_{ij}\phi^0_{ij} > 0$ whenever $\phi^0_{ij} \neq 0$. Indeed, otherwise

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

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 3.3 that $\Theta(\boldsymbol{\nu}) \neq \Theta(\boldsymbol{\nu}_0)$, whence $f(\cdot \mid \boldsymbol{\nu}) \neq f(\cdot \mid \boldsymbol{\nu}_0)$. □

Proof of Lemma 3.4. Note that Lemma 1.1 implies that the equivalence class $\mathcal{E}_0$ is finite. Set $\varepsilon = \min_{\boldsymbol{\nu}_0 \in \mathcal{E}_0} \min_{i,j} \{|\phi^0_{ij}|^2 : \phi^0_{ij} \neq 0\} > 0$. Then if $\|\Phi - \Phi_0\| \leq \|\boldsymbol{\nu} - \boldsymbol{\nu}_0\| < \varepsilon$, the arguments in the proof of Theorem 3.1 guarantee the existence of an ordering that is compatible with Φ and Φ_0, and the result follows from Lemma 3.3. □

Instead of directly proving Theorem 3.5, we will prove a slightly more general statement which requires weaker assumptions. Theorem 3.5 will then follow as a special case. First we will need the following lemma, which is a standard application of the uniform law of large numbers (see, for example, §16 in Ferguson (1996)).

Lemma A.1. *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*

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

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 3, 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 3.1.

Theorem A.2. *Suppose $a_n(\boldsymbol{\nu}^*) = O(n^{-1/2})$ and $b_n(\boldsymbol{\nu}^*) \to 0$ and $\boldsymbol{\nu}_0 \in \mathcal{E}_0$ has strictly more edges than $\boldsymbol{\nu}^*$. Let $\alpha_n := n^{-1/2} + a_n(\boldsymbol{\nu}_0) + a_n(\boldsymbol{\nu}^*)$. If*

(1) $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$ *and* $b_n(\boldsymbol{\nu}_0) \to 0$,
(2) $\max_{i,j} p_{\lambda_n}(|\phi_{ij}^0|) = \tau(\lambda_n) + O(\alpha_n)$,
(3) $\tau(\lambda_n) = \omega(\alpha_n)$,
(4) $\sup_n \tau(\lambda_n) < \infty$,

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. Let $\alpha_n := n^{-1/2} + a_n(\boldsymbol{\nu}_0) + a_n(\boldsymbol{\nu}^*)$. Since ℓ_n is continuous for each n, $\|\widehat{\boldsymbol{\nu}}_n - \boldsymbol{\nu}_0\| = O_P(n^{-1/2} + a_n(\boldsymbol{\nu}_0))$, and $\|\widehat{\boldsymbol{\nu}}_n^* - \boldsymbol{\nu}^*\| = O_P(n^{-1/2} + a_n(\boldsymbol{\nu}^*))$, the above lemma 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 $n^{-1}(\ell(\widehat{\boldsymbol{\nu}}_n) - \ell(\widehat{\boldsymbol{\nu}}_n^*)) = O_P(\alpha_n)$.

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

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

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

Since p_{λ_n} is Lipschitz continuous and bounded (by Condition (4)) for each n and $\widehat{\boldsymbol{\nu}}_n \to \boldsymbol{\nu}_0$, we can write $p_{\lambda_n}(\widehat{\boldsymbol{\nu}}_n) = p_{\lambda_n}(\boldsymbol{\nu}_0) + O(\alpha_n)$ and similarly for $\widehat{\boldsymbol{\nu}}^*$. We can thus rewrite this inequality as

$$p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) > M\alpha_n.$$

Now,

$$\begin{aligned} p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) &= \sum_{\phi_{ij}^0 \neq 0} p_{\lambda_n}(|\phi_{ij}^0|) - \sum_{\phi_{ij}^* \neq 0} p_{\lambda_n}(|\phi_{ij}^*|) \\ &= (s_0 - s^*)\tau(\lambda_n) - \sum_{\phi_{ij}^0 \neq 0} \varepsilon_n(\phi_{ij}^0) + \sum_{\phi_{ij}^* \neq 0} \varepsilon_n(\phi_{ij}^*), \end{aligned}$$

where $\varepsilon_n(t) := \tau(\lambda_n) - p_{\lambda_n}(|t|)$. Since $\varepsilon_n(\phi_{ij}^0) = O(\alpha_n)$, we thus have

$$p_{\lambda_n}(\boldsymbol{\nu}_0) - p_{\lambda_n}(\boldsymbol{\nu}^*) \geq (s_0 - s^*)\tau(\lambda_n) - O(\alpha_n) = \omega(\alpha_n),$$

from which the claim follows. □

Proof of Theorem 3.5. When $a_n(\boldsymbol{\nu}_0) = O(n^{-1/2})$ for all $\boldsymbol{\nu}_0 \in \mathcal{E}_0$, Condition (3) in Theorem A.2 reduces to $\tau(\lambda_n)/n^{-1/2} \to \infty$, and Theorem 3.5 follows as a special case. □

References

A. Antoniadis and J. Fan. Regularization of wavelet approximations. *Journal of the American Statistical Association*, 96(455), 2001.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

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

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

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

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

J. Fan, L. Xue, and H. Zou. Strong oracle optimality of folded concave penalized estimation. *arXiv preprint arXiv:1210.5992*, 2012.

T. S. Ferguson. *A course in large sample theory*, volume 38. CRC Press, 1996.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

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

J. Z. Huang, N. Liu, M. Pourahmadi, and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98, 2006.

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

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *Annals of statistics*, 37(6B):4254, 2009.

S. L. Lauritzen. *Graphical models*. Oxford University Press, 1996.

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

R. Mazumder, J. H. Friedman, and T. Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

M. Pourahmadi. Covariance estimation: The glm and regularization perspectives. *Statistical Science*, 26(3):369–387, 2011.

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

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

A. J. Rothman, E. Levina, and J. Zhu. A new approach to cholesky-based covariance regularization in high dimensions. *Biometrika*, 97(3):539–550, 2010.

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

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

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

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

C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley symposium on mathematical statistics and probability*, volume 1, pages 197–206, 1956.

C. Stein. Estimation of a covariance matrix. *Rietz Lecture*, 1975.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

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

H. Wang, R. Li, and C.-L. Tsai. Tuning parameter selectors for the smoothly clipped absolute deviation method. *Biometrika*, 94(3):553–568, 2007.

T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

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

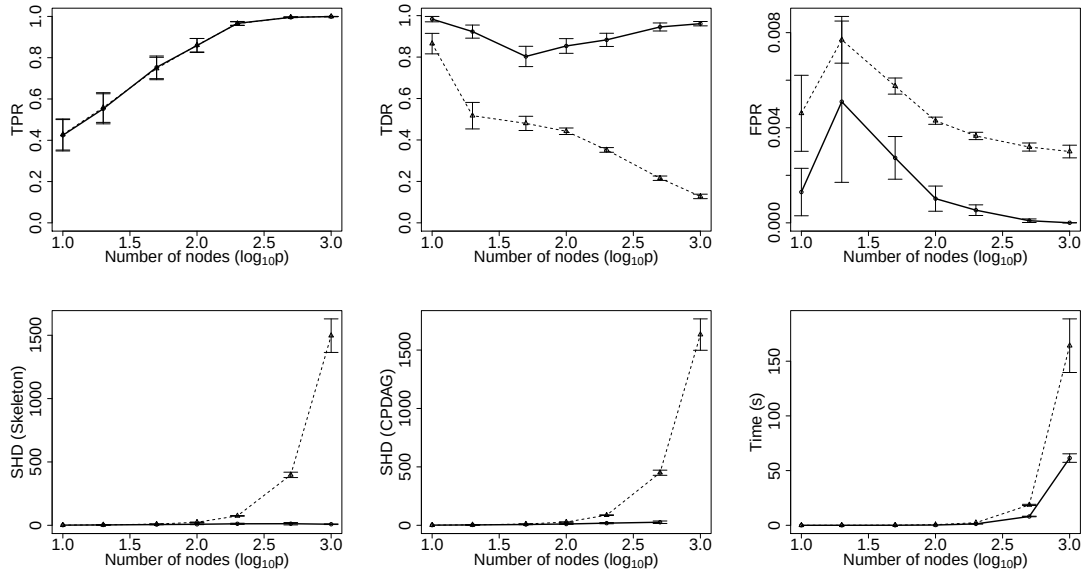
C.-H. Zhang and T. Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

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

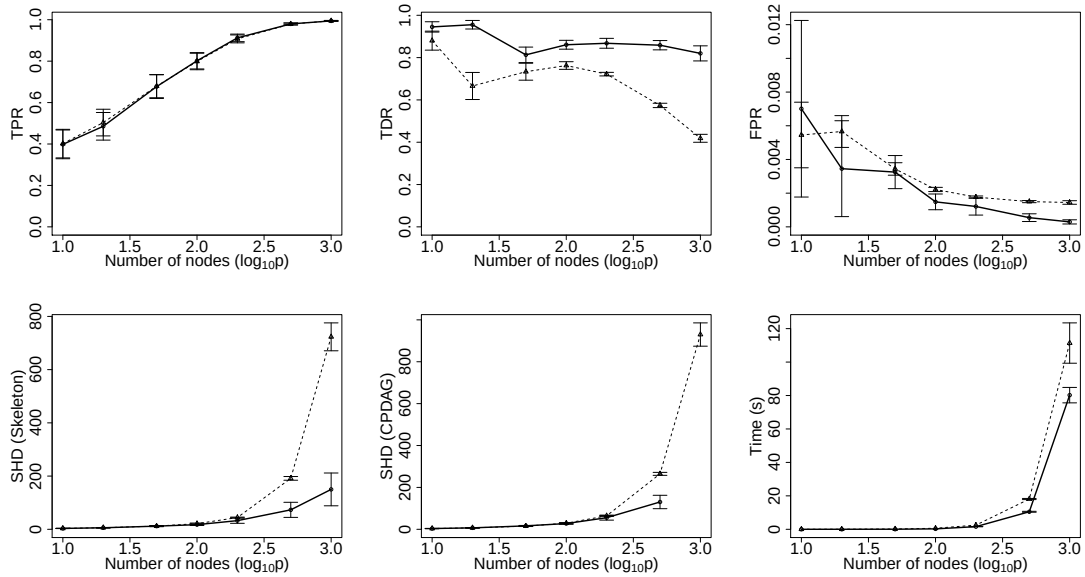
H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Supplementary Material

Detailed plots of the six metrics for each sparsity level ($s_0/p = 0.2, 0.5, 1, 2$). Note that CPDAGs were not computed for CCDr when $p = 1000$, so this data is omitted from the figures.

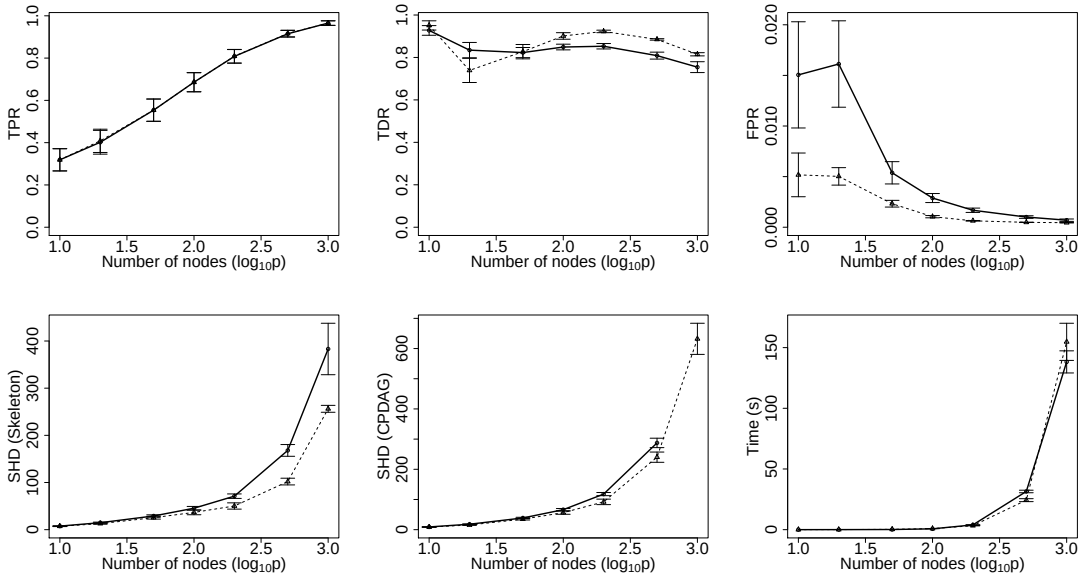


(A) $s_0/p = 0.2$

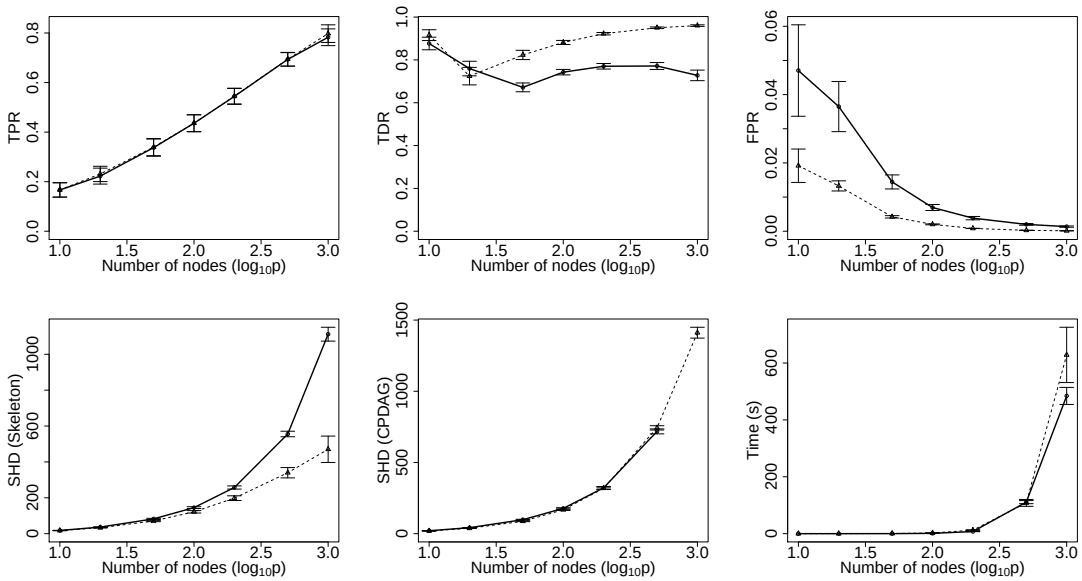


(B) $s_0/p = 0.5$

FIGURE S1. Six metrics for ultra-sparse graphs ($s_0/p < 1$). The dashed lines correspond to the PC algorithm, solid lines to the CCDr algorithm.



(A) $s_0/p = 1.0$



(B) $s_0/p = 2.0$

FIGURE S2. Six metrics for sparse graphs ($s_0/p \geq 1$). The dashed lines correspond to the PC algorithm, solid lines to the CCDr algorithm.

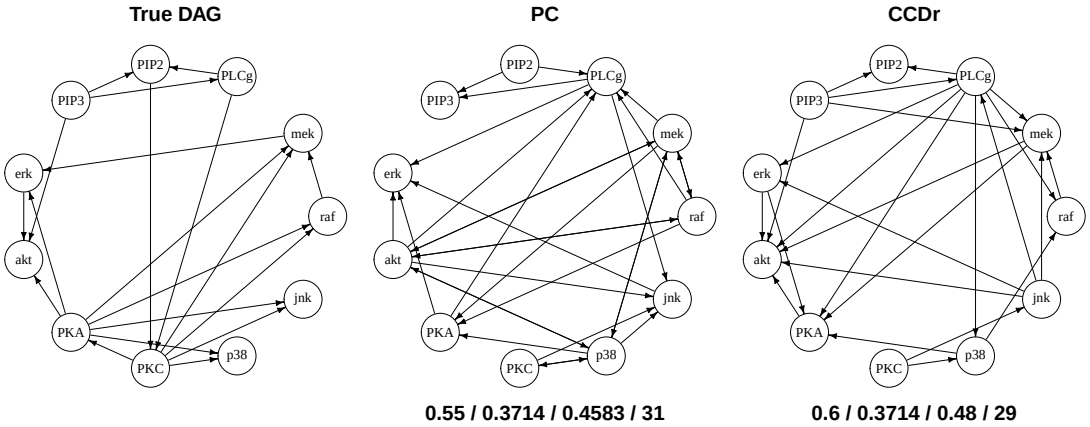


FIGURE S3. Estimated DAGs for the cytometry dataset, using the untransformed continuous data ($n = 7466$). The numbers below each graph are TPR / FPR / TDR / SHD (CPDAG).