Title
Penalized estimation of directed acyclic graphs from discrete data

Journal
Statistics and Computing, 29(1)

ISSN
0960-3174

Authors
Gu, Jiaying
Fu, Fei
Zhou, Qing

Publication Date
2019

DOI
10.1007/s11222-018-9801-y

Peer reviewed

Penalized Estimation of Directed Acyclic Graphs From Discrete Data

Jiaying Gu*, Fei Fu*, and Qing Zhou†
Department of Statistics, University of California, Los Angeles, CA 90095, USA

Abstract

Bayesian networks, with structure given by a directed acyclic graph (DAG), are a popular class of graphical models. However, learning Bayesian networks from discrete or categorical data is particularly challenging, due to the large parameter space and the difficulty in searching for a sparse structure. In this article, we develop a maximum penalized likelihood method to tackle this problem. Instead of the commonly used multinomial distribution, we model the conditional distribution of a node given its parents by multi-logit regression, in which an edge is parameterized by a set of coefficient vectors with dummy variables encoding the levels of a node. To obtain a sparse DAG, a group norm penalty is employed, and a blockwise coordinate descent algorithm is developed to maximize the penalized likelihood subject to the acyclicity constraint of a DAG. When interventional data are available, our method constructs a causal network, in which a directed edge represents a causal relation. We apply our method to various simulated and real data sets. The results show that our method is very competitive, compared to many existing methods, in DAG estimation from both interventional and high-dimensional observational data.

KEY WORDS: Coordinate descent; Discrete Bayesian network; Multi-logit regression; Group norm penalty.

1 Introduction

Bayesian networks are a class of graphical models whose structure implies conditional independence relationships among a set of random variables. It is graphically represented by a directed acyclic graph (DAG). Recent years have seen its popularity in the biological and medical sciences for inferring gene regulatory networks and cellular networks, partially attributed to the fact that it can be used for causal inference. Learning the structure of these biological networks from data is a key to understanding their functions. Most methods that have been proposed for structure learning of DAGs fall into two categories.

The first category encompasses the constraint-based methods that rely on a set of conditional independence tests. The tests are used to examine the existence of edges between nodes. In practice the assumptions behind these methods can be strong, which constitutes the main drawback of these methods. The PC algorithm proposed by Spirtes et al. (1993) and the MMPC algorithm by Tsamardinos et al. (2006) are two well-known examples. The second category includes score-based methods whose goal is to search for a DAG that maximizes certain scoring

*These two authors contributed equally to this work.
†To whom correspondence should be addressed (email: zhou@stat.ucla.edu).

function. The scoring functions that have been employed include several Bayesian Dirichlet metrics (Buntine 1991; Cooper and Herskovits 1992; Heckerman et al. 1995), Bayesian information criterion (Chickering and Heckerman 1997), minimum description length (Bouckaert 1993; Suzuki 1993; Bouckaert 1994; Lam and Bacchus 1994), entropy (Herskovits and Cooper 1990), et cetera. There are also Monte Carlo methods (Ellis and Wong 2008; Zhou 2011) which draw a sample of DAGs from a posterior distribution. Other recent developments on score-based methods include the work of Scutari (2016), which proposed a posterior score function with an uniform prior for discrete data. Additionally, there are hybrid methods which combine the two approaches. The idea is to narrow the search space first using a constraint-based method, and then use a score-based method to learn the DAG structure (Tsamardinos et al. 2006; Gámez et al. 2011).

With the rising interest in sparse statistical modeling, score-based methods seem particularly attractive since various sparse regularization techniques are potentially applicable. Assuming a given natural ordering among the nodes, Shojaie and Michailidis (2010) decomposed DAG estimation into a sequence of lasso regression problems. Schmidt and Murphy (2006) and Schmidt et al. (2007) exploited from a computational perspective the idea of using ℓ_1 regularization to learn the structure of DAGs. Fu and Zhou (2013) developed an ℓ_1-penalized likelihood approach to learn the structure of sparse DAGs from Gaussian data without assuming a given ordering. This method has been further generalized to the use of concave penalties by Aragam and Zhou (2015). Han et al. (2016) proposed a two-stage adaptive lasso approach for structure estimation of DAGs. There are also theoretical developments on ℓ_0-penalized estimation of high-dimensional DAGs under a multivariate Gaussian model (van de Geer and Bühlmann 2013).

Despite the recent fast developments on sparse regularization methods for learning Gaussian DAGs, a generalization to discrete data is highly nontrivial. First, each node now represents a factor coded by a group of dummy variables. In order to select a group of dummy variables together, we need to use a group norm penalty instead of penalizing individual coefficients. Second, the log-likelihood function for categorical data has more parameters, and development of an algorithm to maximize the penalized log-likelihood becomes much more challenging. In this paper, we propose a principled generalization of the penalized likelihood methodology in our previous work (Fu and Zhou 2013) to estimate sparse DAGs from categorical data without knowing the ordering among variables. To reduce the parameter space, we use a multi-logit regression to model the conditional distributions in a discrete Bayesian network. A blockwise coordinate descent (CD) algorithm is developed, which may take both observational and interventional data. Through extensive comparisons, we demonstrate that our method can outperform many competitors in learning discrete Bayesian networks from interventional data or from high-dimensional ($p > n$) observational data. Our algorithm has been implemented in the R package, `discretecdAlgorithm`, available on CRAN.

The remainder of this paper is organized as follows. Section 2 describes the proposed multi-logit regression model and formulates the structure learning problem. Section 3 develops in detail the blockwise CD algorithm for learning discrete Bayesian networks. Section 4 reports numerical results of our method on different types of simulated networks for both interventional and observational data, and Section 5 presents results on real networks. Sections 4 and 5 also include extensive comparisons with other competing methods. The paper is concluded with a discussion in Section 6. In the Appendix, we establish some asymptotic properties of our penalized DAG estimator.

2 Problem Formulation

2.1 Discrete Bayesian networks

The structure of a Bayesian network for p random variables $X_1, \ldots, X_p$ is given by a DAG $\mathcal{G} = (V, E)$. The set of nodes $V = \{1, \ldots, p\}$ represents the set of random variables $\{X_1, \ldots, X_p\}$, and the set of edges is given by $E = \{(j, i) \in V \times V : j \to i\}$, where $j \to i$ is a directed edge in $\mathcal{G}$. Given the structure of $\mathcal{G}$, the joint probability density (mass) function of $(X_1, \ldots, X_p)$ can be factorized as

$$p(x_1, \ldots, x_p) = \prod_{i=1}^{p} p(x_i | \Pi_i^{\mathcal{G}}), \tag{1}$$

where $\Pi_i^{\mathcal{G}} = \{j \in V : (j, i) \in E\}$ is called the set of parents of X_i and $p(x_i|\Pi_i^{\mathcal{G}})$ denotes the conditional density of X_i given $\Pi_i^{\mathcal{G}}$. Throughout the paper, we use j and X_j interchangeably.

Given a joint distribution, there may exist multiple factorizations of the form in (1), leading to different DAGs. The DAGs encoding the same set of conditional independence relations form an equivalence class. All DAGs in the same equivalence class have the same skeleton and v-structures. Here, a v-structure is a triplet $\{i, j, k\} \subset V$ of the form $i \to k \leftarrow j$, while i and j are not directly connected. However, when used for causal inference, equivalent DAGs do not have the same causal interpretation and can be differentiated based on experimental data. There are methods for learning causal DAGs from a mix of observational and experimental data (Cooper and Yoo 1999; Meganck et al. 2006; Ellis and Wong 2008; Hauser and Bühlmann 2015) and related work on inferring gene networks from perturbed expression data (Peér et al. 2001; Pournara and Wernisch 2004; Shojaie et al. 2014).

We describe briefly how the joint distribution of a Bayesian network can be modified to incorporate experimental data. For a detailed account of causal inference using Bayesian networks, please refer to Pearl (2003) and references therein. Assuming X_i, $i \in \mathcal{M} \subset \{1, \ldots, p\}$, is under experimental intervention, the joint density in (1) becomes

$$p(x_1, \ldots, x_p) = \prod_{i \notin \mathcal{M}} p(x_i | \Pi_i^{\mathcal{G}}) \prod_{i \in \mathcal{M}} p(x_i | \bullet), \tag{2}$$

where $p(x_i|\bullet)$ specifies the distribution of X_i under intervention. Experimental data generated from $\mathcal{G}$ can therefore be considered as being generated from the DAG $\mathcal{G}'$ obtained by removing all directed edges in $\mathcal{G}$ pointing to the variables under intervention. It should be noted that (2) also applies to observational data for which $\mathcal{M}$ is simply empty, and in this case, (2) reduces to (1). Hereafter, we develop our method under the assumption that part of the data are generated under experimental intervention, while regarding purely observational data as the special case of $\mathcal{M} = \varnothing$.

In a discrete Bayesian network, each variable X_i is considered a factor with r_i levels, indexed by $\{1, \ldots, r_i\}$. The set of its parents $\Pi_i^{\mathcal{G}}$ has a total of $q_i = \prod_{j \in \Pi_i^{\mathcal{G}}} r_j$ possible joint states $\{\boldsymbol{\pi}_k : k = 1, \ldots, q_i\}$. Let $\Theta_{ijk} = P\left(X_i = j \mid \Pi_i^{\mathcal{G}} = \boldsymbol{\pi}_k\right)$. A discrete Bayesian network $\mathcal{G}$ may be parameterized by $\boldsymbol{\Theta} = \{\Theta_{ijk} \geq 0 : \sum_j \Theta_{ijk} = 1\}$ via a product multinomial model given the graph structure. The number of parameters in this product multinomial model is

$$N(\boldsymbol{\Theta}) = \sum_{i=1}^{p} r_i q_i = \sum_{i=1}^{p} r_i \prod_{j \in \Pi_i^{\mathcal{G}}} r_j.$$

If we assume that each variable has $\mathcal{O}(r)$ levels, then

$$N(\boldsymbol{\Theta}) = \mathcal{O}\left(\sum_{i=1}^{p} r^{1+|\Pi_i^{\mathcal{G}}|}\right), \tag{3}$$

which grows exponentially as the size of the parent set $|\Pi_i^{\mathcal{G}}|$ increases. To reduce the number of free parameters, we propose a multi-logit model for discrete Bayesian networks under which development of a penalized likelihood method is straightforward. For the same DAG structure, the number of parameters can be much smaller compared to the product multinomial model.

2.2 A multi-logit model

We encode the r_i levels of X_i, $i = 1, \ldots, p$, by a group of $d_i = r_i - 1$ dummy variables. Let $\mathbf{x}_i \in \{0,1\}^{d_i}$ be the group of dummy variables for X_i and $\mathbf{x} = (1, \mathbf{x}_1, \ldots, \mathbf{x}_p)$ be a d-vector, where $d = 1 + \sum_{i=1}^{p} d_i$. For a discrete Bayesian network $\mathcal{G}$, we model the conditional distribution $[X_j | \Pi_j^{\mathcal{G}}]$, $j = 1, \ldots, p$, by the following multi-logit regression model

$$\begin{aligned} P(X_j = \ell \mid \Pi_j^{\mathcal{G}}) &= \frac{\exp(\beta_{j\ell 0} + \sum_{i=1}^{p} \mathbf{x}_i^T \boldsymbol{\beta}_{j\ell i})}{\sum_{m=1}^{r_j} \exp(\beta_{jm0} + \sum_{i=1}^{p} \mathbf{x}_i^T \boldsymbol{\beta}_{jmi})} \\ &= \frac{\exp(\mathbf{x}^T \boldsymbol{\beta}_{j\ell\cdot})}{\sum_{m=1}^{r_j} \exp(\mathbf{x}^T \boldsymbol{\beta}_{jm\cdot})} \triangleq p_{j\ell}(\mathbf{x}), \end{aligned} \tag{4}$$

for $\ell = 1, \ldots, r_j$, where $\beta_{j\ell 0}$ is the intercept, $\boldsymbol{\beta}_{j\ell i} \in \mathbb{R}^{d_i}$ is the coefficient vector for X_i to predict the ℓ^{th} level of X_j, and $\boldsymbol{\beta}_{j\ell\cdot} = \text{vec}(\beta_{j\ell 0}, \boldsymbol{\beta}_{j\ell 1}, \ldots, \boldsymbol{\beta}_{j\ell p}) \in \mathbb{R}^d$. Note that in (4), $\boldsymbol{\beta}_{j\ell i} = \mathbf{0}$ for all ℓ if $i \notin \Pi_j^{\mathcal{G}}$. Thus, our model indeed defines a joint distribution for $X_1, \ldots, X_p$ which factorizes according to the DAG $\mathcal{G}$. We choose to use a symmetric form of the multi-logit model here, as was done in Zhu and Hastie (2004) and Friedman et al. (2010). To make this model identifiable, we impose the following constraints on the intercepts

$$\beta_{j10} = 0, \quad j = 1, \ldots, p. \tag{5}$$

The nonidentifiability of other parameters can be resolved via regularization as demonstrated by Friedman et al. (2010). The particular form of regularization we use leads to the following constraints

$$\sum_{m=1}^{r_j} \boldsymbol{\beta}_{jmi} = \mathbf{0}, \quad \forall\, i, j = 1, \ldots, p. \tag{6}$$

Let $\boldsymbol{\beta} = (\boldsymbol{\beta}_{jmi})$, which is a four-way array, denote all the parameters. Given the structure of $\mathcal{G}$, the number of free parameters is

$$N(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left[(r_j - 1) + r_j \sum_{i \in \Pi_j^{\mathcal{G}}} d_i \right]. \tag{7}$$

If we further assume that $r_i = \mathcal{O}(r)$ for all i, then

$$N(\boldsymbol{\beta}) = \mathcal{O}(r^2)|E| + \mathcal{O}(rp), \tag{8}$$

which grows linearly in the total number of edges $|E|$. This rate of growth is much slower than that of the product multinomial model (3). Note that these two models are not equivalent. Numerical comparisons in Section 5 confirm that the proposed multi-logit model often serves as a good approximation to the product multinomial model.

Suppose that we have a data set $\mathcal{X} = (\mathcal{X}_{hi})_{n\times p}$ generated from a causal discrete Bayesian network $\mathcal{G}$, where $\mathcal{X}_{hi}$ is the level of X_i in the h^{th} data point, $h = 1, \ldots, n$, coded by dummy variables $\mathbf{x}_{h,i} \in \{0,1\}^{d_i}$. Let $\mathcal{I}_j$ be the index set of rows where X_j is under intervention and $\mathcal{O}_j = \{1, \ldots, n\} \setminus \mathcal{I}_j$ be the index set of rows in which X_j is observational. Note that $\mathcal{I}_j$ are not necessarily mutually exclusive, which means there can be more than one node under intervention for a data point. Under the multi-logit model (4), the log-likelihood function $\ell(\boldsymbol{\beta})$ can be written according to the factorization (2) as

$$\begin{aligned} \ell(\boldsymbol{\beta}) &= \sum_{j=1}^{p} \sum_{h\in\mathcal{O}_j} \log\left[p(\mathcal{X}_{hj}|\mathbf{x}_{h,i}, i \in \Pi_j^{\mathcal{G}})\right] \\ &= \sum_{j=1}^{p} \sum_{h\in\mathcal{O}_j} \left[\sum_{\ell=1}^{r_j} y_{hj\ell}\mathbf{x}_h^T\boldsymbol{\beta}_{j\ell\cdot} - \log\left\{\sum_{m=1}^{r_j}\exp(\mathbf{x}_h^T\boldsymbol{\beta}_{jm\cdot})\right\}\right], \end{aligned} \tag{9}$$

where $y_{hj\ell} = I(\mathcal{X}_{hj} = \ell)$ are indicator variables and $\boldsymbol{\beta}_{j\ell k} = \mathbf{0}$ for $k \notin \Pi_j^{\mathcal{G}}$.

Remark 1. Although we have assumed the availability of experimental data, it is easy to see that the log-likelihood (9) applies to observational data as well: If there are no experimental data for X_j, then $\mathcal{O}_j = \{1, \ldots, n\}$ in (9).

2.3 Group norm penalty

Define $\boldsymbol{\beta}_{j\cdot i} = \text{vec}(\boldsymbol{\beta}_{j1i}, \ldots, \boldsymbol{\beta}_{jr_ji}) \in \mathbb{R}^{d_i r_j}$ to be the vector of coefficients representing the influence of X_i on X_j and $\boldsymbol{\beta}_{j\cdot 0} = (\beta_{j10}, \ldots, \beta_{jr_j0}) \in \mathbb{R}^{r_j}$ to be the vector of intercepts for predicting X_j. The structure of $\mathcal{G}$ is coded by the sparsity of $\boldsymbol{\beta}_{j\cdot i}$ as

$$\boldsymbol{\beta}_{j\cdot i} = \mathbf{0} \qquad \Longleftrightarrow \qquad i \notin \Pi_j^{\mathcal{G}}. \tag{10}$$

In order to learn a sparse DAG from data, we estimate $\boldsymbol{\beta}$ via a penalized likelihood approach. It can be seen from (10) that, for discrete Bayesian networks, the set of parents of X_j is given by the set $\{i : \boldsymbol{\beta}_{j\cdot i} \neq \mathbf{0}\}$. The regular ℓ_1 penalty is inappropriate for this purpose since it penalizes each component of $\boldsymbol{\beta}$ separately. We instead penalize the vector $\boldsymbol{\beta}_{j\cdot i} \in \mathbb{R}^{d_i r_j}$ as a whole to obtain a sparse DAG via the use of a group norm penalty. Group norm penalties have been used in the group lasso and its generalizations (Yuan and Lin 2006; Meier et al. 2008). Let $\mathcal{G}_{\boldsymbol{\beta}}$ denote the graph induced by $\boldsymbol{\beta}$ so that $\Pi_j^{\mathcal{G}_{\boldsymbol{\beta}}} = \{i : \boldsymbol{\beta}_{j\cdot i} \neq \mathbf{0}\}$ for $j = 1, \ldots, p$. We define our group norm penalized estimator for a discrete Bayesian network by the following optimization program:

$$f_\lambda(\boldsymbol{\beta}) \triangleq -\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p}\sum_{i=1}^{p} \|\boldsymbol{\beta}_{j\cdot i}\|_2, \tag{11}$$

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}:\mathcal{G}_{\boldsymbol{\beta}} \text{ is a DAG}}{\arg\min} f_\lambda(\boldsymbol{\beta}), \tag{12}$$

where $\lambda > 0$ is a tuning parameter. See Section 3.3 for choosing the parameter λ. The feasible set of (12) is a DAG space, which imposes a highly nonconvex constraint. This is a major challenge for our optimization algorithm. Hereafter, we call $\boldsymbol{\beta}_{j\cdot i}$ a (component) group of $\boldsymbol{\beta}$.

3 Algorithm

Structure learning for discrete Bayesian networks is computationally demanding because of the nonlinear nature of the multi-logit model (4). We develop in this section a blockwise coordinate descent algorithm to solve (12). Coordinate descent algorithms have been proved successful in various settings (Fu 1998; Friedman et al. 2007; Wu and Lange 2008) and their implementations are relatively straightforward.

3.1 Single coordinate descent step

We first consider minimizing $f_\lambda(\boldsymbol{\beta})$ (11) with respect to $\boldsymbol{\beta}_{j\cdot i}$ while holding all the other parameters constant. We define

$$\begin{aligned} f_{\lambda,j}(\boldsymbol{\beta}_{j\cdot\cdot}) &= -\sum_{h\in\mathcal{O}_j}\left[\sum_{\ell=1}^{r_j} y_{hj\ell}\mathbf{x}_h^T\boldsymbol{\beta}_{j\ell\cdot} - \log\left\{\sum_{m=1}^{r_j}\exp\left(\mathbf{x}_h^T\boldsymbol{\beta}_{jm\cdot}\right)\right\}\right] + \lambda\sum_{k=1}^{p}\|\boldsymbol{\beta}_{j\cdot k}\|_2 \\ &\triangleq -\ell_j\left(\boldsymbol{\beta}_{j\cdot\cdot}\right) + \lambda\sum_{k=1}^{p}\|\boldsymbol{\beta}_{j\cdot k}\|_2, \end{aligned} \tag{13}$$

where $\boldsymbol{\beta}_{j\cdot\cdot} = (\boldsymbol{\beta}_{j\cdot 0}, \boldsymbol{\beta}_{j\cdot 1}, \ldots, \boldsymbol{\beta}_{j\cdot p})$. Considering the problem of minimizing $f_{\lambda,j}(\cdot)$ over $\boldsymbol{\beta}_{j\cdot i}$, we write $f_{\lambda,j}$ and ℓ_j as $f_{\lambda,j}(\boldsymbol{\beta}_{j\cdot i})$ and $\ell_j(\boldsymbol{\beta}_{j\cdot i})$, respectively.

Following the approach of Tseng and Yun (2009) and Meier et al. (2008), we form a quadratic approximation to $\ell_j(\boldsymbol{\beta}_{j\cdot i})$ using the second-order Taylor expansion at $\boldsymbol{\beta}_{j\cdot i}^{(t)}$, the current value of $\boldsymbol{\beta}_{j\cdot i}$. Adding the penalty, the quadratic approximation is

$$\begin{aligned} Q_{\lambda,j}^{(t)}(\boldsymbol{\beta}_{j\cdot i}) &= -\left\{\left(\boldsymbol{\beta}_{j\cdot i} - \boldsymbol{\beta}_{j\cdot i}^{(t)}\right)^T\nabla\ell_j\left(\boldsymbol{\beta}_{j\cdot i}^{(t)}\right)\right. \\ &\quad \left. + \frac{1}{2}\left(\boldsymbol{\beta}_{j\cdot i} - \boldsymbol{\beta}_{j\cdot i}^{(t)}\right)^T\mathbf{H}_{ji}^{(t)}\left(\boldsymbol{\beta}_{j\cdot i} - \boldsymbol{\beta}_{j\cdot i}^{(t)}\right)\right\} + \lambda\|\boldsymbol{\beta}_{j\cdot i}\|_2, \end{aligned} \tag{14}$$

up to an additive term that does not depend on $\boldsymbol{\beta}_{j\cdot i}$. The gradient of the log-likelihood function $\ell_j(\cdot)$ is

$$\nabla\ell_j(\boldsymbol{\beta}_{j\cdot i}^{(t)}) = \sum_{h\in\mathcal{O}_j}\begin{pmatrix}\left(y_{hj1} - p_{j1}^{(t)}(\mathbf{x}_h)\right)\mathbf{x}_{h,i}\\ \vdots \\ \left(y_{hjr_j} - p_{jr_j}^{(t)}(\mathbf{x}_h)\right)\mathbf{x}_{h,i}\end{pmatrix}, \tag{15}$$

where $p_{j\ell}^{(t)}(\mathbf{x})$, $\ell = 1, \ldots, r_j$, defined in (4) are evaluated at the current parameter values. To give a reasonable quadratic approximation, we use a negative definite matrix $\mathbf{H}_{ji}^{(t)} = h_{ji}^{(t)}\mathbf{I}_{d_i r_j}$ in (14) to approximate the Hessian of $\ell_j(\cdot)$, where the scalar $h_{ji}^{(t)} < 0$ and $\mathbf{I}_{d_i r_j}$ is the identity matrix of size $d_i r_j \times d_i r_j$. We choose

$$h_{ji}^{(t)} = h_{ji}(\boldsymbol{\beta}^{(t)}) \triangleq -\max\{\mathrm{diag}(-\mathbf{H}_{\ell_j}(\boldsymbol{\beta}_{j\cdot i}^{(t)})), b\}, \tag{16}$$

where $\mathbf{H}_{\ell_j}$ is the Hessian of the log-likelihood function $\ell_j(\cdot)$ and b is a small positive number used as a lower bound to help convergence. Note that it is not necessary to recompute $h_{ji}^{(t)}$ every iteration (Meier et al. 2008). See Section 3.3 for more details.

It is not difficult to show the following proposition, which is a direct consequence of the Karush-Kuhn-Tucker (KKT) conditions for minimizing (14).

Proposition 1. *Let* $\mathbf{H}_{ji}^{(t)} = h_{ji}^{(t)} \mathbf{I}_{d_i r_j}$ *for some scalar* $h_{ji}^{(t)} < 0$ *and* $\boldsymbol{d}_{ji}^{(t)} = \nabla \ell_j(\boldsymbol{\beta}_{j\cdot i}^{(t)}) - h_{ji}^{(t)} \boldsymbol{\beta}_{j\cdot i}^{(t)}$. *Then, the minimizer of* $Q_{\lambda,j}^{(t)}(\boldsymbol{\beta}_{j\cdot i})$ *in* (14) *is*

$$\bar{\boldsymbol{\beta}}_{j\cdot i}^{(t)} = \begin{cases} \mathbf{0} & \text{if } \|\boldsymbol{d}_{ji}^{(t)}\|_2 \le \lambda, \\ -\dfrac{1}{h_{ji}^{(t)}} \left[1 - \dfrac{\lambda}{\|\boldsymbol{d}_{ji}^{(t)}\|_2} \right] \boldsymbol{d}_{ji}^{(t)} & \text{otherwise.} \end{cases} \tag{17}$$

In order to achieve sufficient descent, an inexact line search by the Armijo rule is performed when $\bar{\boldsymbol{\beta}}_{j\cdot i}^{(t)} \neq \boldsymbol{\beta}_{j\cdot i}^{(t)}$, following the procedure in Meier et al. (2008). Put $\boldsymbol{s}_{ji}^{(t)} = \bar{\boldsymbol{\beta}}_{j\cdot i}^{(t)} - \boldsymbol{\beta}_{j\cdot i}^{(t)}$, and let $\Delta^{(t)}$ be the change in $f_{\lambda,j}$ when the log-likelihood is linearized at $\boldsymbol{\beta}_{j\cdot i}^{(t)}$, i.e.,

$$\Delta^{(t)} = -(\boldsymbol{s}_{ji}^{(t)})^T \nabla \ell_j(\boldsymbol{\beta}_{j\cdot i}^{(t)}) + \lambda \|\bar{\boldsymbol{\beta}}_{j\cdot i}^{(t)}\|_2 - \lambda \|\boldsymbol{\beta}_{j\cdot i}^{(t)}\|_2.$$

Pick $\eta, \delta \in (0,1)$ and $\alpha_0 > 0$, and let $\alpha^{(t)}$ be the largest value in the sequence $\{\alpha_0 \eta^k\}_{k \ge 0}$ such that

$$f_{\lambda,j}(\boldsymbol{\beta}_{j\cdot i}^{(t)} + \alpha^{(t)} \boldsymbol{s}_{ji}^{(t)}) \le f_{\lambda,j}(\boldsymbol{\beta}_{j\cdot i}^{(t)}) + \delta \alpha^{(t)} \Delta^{(t)}.$$

Then set

$$\boldsymbol{\beta}_{j\cdot i}^{(t+1)} = \boldsymbol{\beta}_{j\cdot i}^{(t)} + \alpha^{(t)} \boldsymbol{s}_{ji}^{(t)}, \tag{18}$$

which completes one iteration for updating $\boldsymbol{\beta}_{j\cdot i}$. In our implementation, we choose $\eta = 0.5$, $\delta = 0.1$ and $\alpha_0 = 1$ following the suggestion by Meier et al. (2008).

It follows from Proposition 1 with $\lambda = 0$ that for the unpenalized intercepts,

$$\boldsymbol{\beta}_{j\cdot 0}^{(t+1)} = \bar{\boldsymbol{\beta}}_{j\cdot 0}^{(t)} = -\boldsymbol{d}_{j0}^{(t)} / h_{j0}^{(t)}. \tag{19}$$

In addition, some of the parameters are always constrained to zero, e.g., $\boldsymbol{\beta}_{j\cdot j}$ and β_{j10} for all j.

3.2 Blockwise coordinate descent

Our CD algorithm consists of two layers of iterations. In the outer loop, we cycle through all pairs of nodes to update the active set of edges, including their directions. In the inner loop, we only cycle through the active edge set to update the parameter values.

We first describe the outer loop. Due to the acyclicity constraint in (12), we know *a priori* that $\boldsymbol{\beta}_{i\cdot j}$ and $\boldsymbol{\beta}_{j\cdot i}$ cannot simultaneously be nonzero for $i \neq j$. This suggests performing the minimization in blocks, minimizing over $\{\boldsymbol{\beta}_{i\cdot j}, \boldsymbol{\beta}_{j\cdot i}\}$ simultaneously. In order to enforce acyclicity, we use a simple heuristic (Fu and Zhou 2013): For each block $\{\boldsymbol{\beta}_{i\cdot j}, \boldsymbol{\beta}_{j\cdot i}\}$, we check if adding an edge from $i \to j$ induces a cycle in the estimated DAG. If so, we set $\boldsymbol{\beta}_{j\cdot i} = \mathbf{0}$ and minimize with respect to $\boldsymbol{\beta}_{i\cdot j}$. Alternatively, if the edge $j \to i$ induces a cycle, we set $\boldsymbol{\beta}_{i\cdot j} = \mathbf{0}$ and minimize with respect to $\boldsymbol{\beta}_{j\cdot i}$. If neither edge induces a cycle, we minimize over both parameters simultaneously. The cycle check is implemented by a breath-first search algorithm. We outline below (Algorithm 1) the complete blockwise CD algorithm for discrete Bayesian networks. In the algorithm, $\boldsymbol{\beta}_{j\cdot i} \Leftarrow \mathbf{0}$ is used to indicate that $\boldsymbol{\beta}_{j\cdot i}$ must be set to zero due to the acyclicity

Algorithm 1 CD algorithm for estimating discrete Bayesian networks

1: Initialize $\boldsymbol{\beta}$ such that $\mathcal{G}_{\boldsymbol{\beta}}$ is acyclic
2: **for** $i = 1, \ldots, p-1$ **do**
3: **for** $j = i+1, \ldots, p$ **do**
4: **if** $\boldsymbol{\beta}_{j\cdot i} \Leftarrow \mathbf{0}$ **then**
5: $\boldsymbol{\beta}_{i\cdot j} \leftarrow \arg\min_{\boldsymbol{\beta}_{i\cdot j}} f_{\lambda,i}(\cdot), \quad \boldsymbol{\beta}_{j\cdot i} \leftarrow \mathbf{0}$
6: **else if** $\boldsymbol{\beta}_{i\cdot j} \Leftarrow \mathbf{0}$ **then**
7: $\boldsymbol{\beta}_{i\cdot j} \leftarrow \mathbf{0}, \quad \boldsymbol{\beta}_{j\cdot i} \leftarrow \arg\min_{\boldsymbol{\beta}_{j\cdot i}} f_{\lambda,j}(\cdot)$
8: **else**
9: $S_1 \leftarrow \min_{\boldsymbol{\beta}_{i\cdot j}} f_{\lambda,i}(\cdot) + f_{\lambda,j}(\cdot)|_{\boldsymbol{\beta}_{j\cdot i}=\mathbf{0}}$
10: $S_2 \leftarrow f_{\lambda,i}(\cdot)|_{\boldsymbol{\beta}_{i\cdot j}=\mathbf{0}} + \min_{\boldsymbol{\beta}_{j\cdot i}} f_{\lambda,j}(\cdot)$
11: **if** $S_1 \leq S_2$ **then**
12: $\boldsymbol{\beta}_{i\cdot j} \leftarrow \arg\min_{\boldsymbol{\beta}_{i\cdot j}} f_{\lambda,i}(\cdot), \quad \boldsymbol{\beta}_{j\cdot i} \leftarrow \mathbf{0}$
13: **else**
14: $\boldsymbol{\beta}_{i\cdot j} \leftarrow \mathbf{0}, \quad \boldsymbol{\beta}_{j\cdot i} \leftarrow \arg\min_{\boldsymbol{\beta}_{j\cdot i}} f_{\lambda,j}(\cdot)$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: Update intercepts $\boldsymbol{\beta}_{j\cdot 0}$ for $j = 1, \ldots, p$
20: Inner loop given the active edge set (Algorithm 2)
21: Repeat step 2 to 20 until some stopping criterion is met

constraint given the current estimates of the other parameters. Minimization of $f_{\lambda,j}(\cdot)$ with respect to $\boldsymbol{\beta}_{j\cdot i}$ is done with the single CD step with line search (18).

Let $\boldsymbol{\beta}^{(t)}$ denote the parameter value after one cycle of the outer loop (after line 19 in Algorithm 1). Denote its active edge set by $E^{(t)} = \{(i,j) : \boldsymbol{\beta}^{(t)}_{j\cdot i} \neq \mathbf{0}\}$. The inner loop solves the following problem:

$$\min_{\boldsymbol{\beta}} f_\lambda(\boldsymbol{\beta}), \text{ subject to } \operatorname{supp}(\boldsymbol{\beta}) \subset E^{(t)}, \tag{20}$$

where f_λ is defined in (11). We use $\boldsymbol{\beta}^{(t)}$ as the initial value and cycle through $\boldsymbol{\beta}_{j\cdot i}$ for $(i,j) \in E^{(t)}$. In particular, the direction of an edge will not be reversed but edges may be deleted if their parameters $\boldsymbol{\beta}_{j\cdot i}$ are updated to zero. See Algorithm 2 for an outline of the inner loop.

Algorithm 2 Inner loop

1: Input $E^{(t)}$ and initialize $\boldsymbol{\beta} = \boldsymbol{\beta}^{(t)}$
2: **for** $(i,j) \in E^{(t)}$ **do**
3: $\boldsymbol{\beta}_{j\cdot i} \leftarrow \arg\min_{\boldsymbol{\beta}_{j\cdot i}} f_{\lambda,j}(\cdot)$
4: **end for**
5: Repeat step 2 to step 4 until convergence.

By construction, $E^{(t)}$ satisfies the acyclicity constraint and thus the feasible region in (20) is simply a Euclidean space. Since f_λ itself is convex, the CD algorithm for the inner loop has nice convergence properties. In analogy to Proposition 2 in Meier et al. (2008), we arrive at the following convergence result.

Proposition 2. *Suppose that the sequence* $\{\boldsymbol{\beta}^{(k)}\}$ *is generated by the inner loop. If the matrix* $\mathbf{H}_{ji}^{(k)}$ *is chosen according to* (16)*, then every limit point of the sequence* $\{\boldsymbol{\beta}^{(k)}\}$ *is a minimizer of problem* (20)*.*

However, since the search space for the outer loop is the full DAG space, which is highly nonconvex, rigorous theory on its convergence is yet to be established. Therefore, a practical stopping criterion is employed. After the convergence of an inner loop, we obtain the current active set. If one more iteration of the outer loop does not change the active set, we then stop Algorithm 1. On the other hand, we also set a maximum number of iterations for the outer loop. For all the examples we have tested, our CD algorithm has shown no problem in convergence. This empirical observation is in line with recent theoretical work by Lee et al. (2016) who have established that gradient descent converges to a local minimizer of a nonconvex objective function for almost all initial values and have suggested similar behavior for coordinate descent.

3.3 Solution path

We use Algorithm 1 to compute $\hat{\boldsymbol{\beta}}_\lambda$ (12) over a grid of J values for the tuning parameter, $\lambda_1 > \ldots > \lambda_J > 0$, where at λ_1 every parameter other than the intercepts is estimated as zero. It follows from the KKT conditions for (11) that

$$\lambda_1 \quad = \quad \max_{1 \le i,j \le p} \|\nabla \ell_j(\boldsymbol{\beta}_{j\cdot i})|_{\boldsymbol{\beta}_{j\cdot i}=\mathbf{0}}\|_2, \tag{21}$$

in which $\boldsymbol{\beta}_{j\cdot 0}$ is set to the MLE of the intercept assuming all $\boldsymbol{\beta}_{j\cdot i}$, $i = 1, \ldots, p$, are zero.

The solution $\hat{\boldsymbol{\beta}}_{\lambda_m}$ is used as a warm start for estimating $\hat{\boldsymbol{\beta}}_{\lambda_{m+1}}$, $m = 1, \ldots, J-1$. To save computational time, we set $h_{ji}^{(t)} = h_{ji}(\hat{\boldsymbol{\beta}}_{\lambda_m})$ (16) in the CD algorithm for $\hat{\boldsymbol{\beta}}_{\lambda_{m+1}}$, instead of updating $h_{ji}^{(t)}$ every iteration.

Traditional model selection criteria such as BIC do not work well for the purpose of estimating DAGs from data. In our simulation results, the hill-climbing (HC) algorithm (Gámez et al. 2011), which uses BIC as the scoring function, always selects too many edges. There are also numerical studies (Scutari 2016) in which BIC tends to select too few edges on a different set of DAGs, showing that BIC could be sensitive and unstable. In order to select a suitable tuning parameter, we use an empirical model selection criterion proposed by Fu and Zhou (2013). Let $\hat{\mathcal{G}}_{\lambda_m}$ be the DAG induced by $\hat{\boldsymbol{\beta}}_{\lambda_m}$ and e_{λ_m} be the number of edges in $\hat{\mathcal{G}}_{\lambda_m}$. We reestimate $\boldsymbol{\beta}$ by the maximizer $\boldsymbol{\beta}_{\lambda_m}^\dagger$ of the log-likelihood $\ell(\boldsymbol{\beta})$ (9) given $\mathcal{G} = \hat{\mathcal{G}}_{\lambda_m}$ using the R package `nnet` (Venables and Ripley 2002). We define the difference ratio between two estimated DAGs $\hat{\mathcal{G}}_{\lambda_m}$ and $\hat{\mathcal{G}}_{\lambda_{m+1}}$ by $dr_{(m,m+1)} = \Delta\ell_{(m,m+1)}/\Delta e_{(m,m+1)}$, where $\Delta\ell_{(m,m+1)} = \ell(\boldsymbol{\beta}_{\lambda_{m+1}}^\dagger) - \ell(\boldsymbol{\beta}_{\lambda_m}^\dagger)$ and $\Delta e_{(m,m+1)} = e_{\lambda_{m+1}} - e_{\lambda_m}$, if $\Delta e_{(m,m+1)} \ge 1$. Otherwise, we set $dr_{(m,m+1)} = dr_{(m-1,m+1)}$. The selected tuning parameter is indexed by

$$m^* = \sup\left\{2 \le m \le J : dr_{(m-1,m)} \ge \alpha \cdot \max\{dr_{(1,2)}, \ldots, dr_{(J-1,J)}\}\right\}. \tag{22}$$

According to this criterion, an increase in model complexity, measured by the number of predicted edges, is accepted only if there is a substantial increase in the log-likelihood. We choose $\alpha = 0.3$ for all the results in this work.

4 Simulation Studies

We evaluate the CD algorithm on simulated data sets. As stated in Remark 1, the log-likelihood (9) applies to observational data as well. Therefore, we apply the CD algorithm on both interventional data and observational data. In order to assess the accuracy and efficiency of the CD algorithm, we compare it with a few competing methods. For interventional data, we compare our CD algorithm with the PC algorithm (Kalisch and Bühlmann 2007), the greedy interventional equivalent search (GIES) algorithm (Hauser and Bühlmann 2015) and the equi-energy sampler (EE sampler) (Kou et al. 2006). For observational data we compare the CD algorithm with the hill-climbing (HC) algorithm (Gámez et al. 2011), the max-min hill-climbing (MMHC) algorithm (Tsamardinos et al. 2006), and the PC algorithm. Among these competitors, the PC algorithm is a constraint-based method, the MMHC is a hybrid method and the others are all score-based.

Details about data generation and parameter choices will be discussed in Section 4.1. In Section 4.2, we compare DAGs estimated from interventional data. Section 4.3, on the other hand, presents results on high-dimensional observational data. The comparison of running times is provided in Section 4.4.

4.1 Experimental setup

Four types of networks are used to compare the methods: the bipartite graph, the scale-free network, the small-world network, and random DAGs. In each setting, we consider the combination of three main parameters: (n, p, s_0), where n is the sample size, p is the number of nodes, and s_0 is the number of true edges.

We generated bipartite graphs, scale-free networks, and small-world networks with the R package `igraph` (Csárdi and Nepusz 2006). The bipartite graphs were generated by the Erdős-Rényi model (Erdos and Rényi 1960). Each bipartite graph in our datasets had $0.2p$ top nodes, $0.8p$ bottom nodes, and $s_0 = p$ directed edges from the top to the bottom nodes. The structure of a scale-free network was generated using the Barabási-Albert model (Barabási and Albert 1999). These networks had $s_0 = p - 1$ directed edges. The small-world networks were generated by the Watts-Strogatz model (Watts and Strogatz 1998). A graph initially generated by the model was undirected. To convert it to a DAG, edge directions were chosen according to a randomly generated topological sort. In this way, a small-world network had $s_0 = 2p$ directed edges. Random DAGs were sampled using the R package `pcalg` (Kalisch et al. 2012), and each DAG had $s_0 \approx p$ edges.

In all the simulation studies, each variable was assumed to be binary, i.e., $r_j = 2$ for all j. In this case, each group of parameters $\boldsymbol{\beta}_{j\cdot i} = (\beta_{j1i}, \beta_{j2i}) \in \mathbb{R}^2$. If $\Pi_j = \varnothing$, X_j would be sampled from its two levels with equal probability. Otherwise, the parameters $\boldsymbol{\beta}_{j\cdot 0}$ and $\boldsymbol{\beta}_{j\cdot i}$, $i \in \Pi_j$, were chosen such that

$$p_{j\ell}(\mathbf{x}_h) = \frac{\exp(2\sum_{i\in\Pi_j} y_{hi\ell})}{\exp(2\sum_{i\in\Pi_j} y_{hi1}) + \exp(2\sum_{i\in\Pi_j} y_{hi2})}$$

for $\ell = 1, 2$, where $y_{hi\ell} = I(\mathcal{X}_{hi} = \ell)$. The value of a variable under intervention was randomly fixed to one of its levels regardless of its parents.

For each dataset, we input to our CD algorithm a sequence of 40 values of λ, starting from λ_1 (21) and ending at $0.01\lambda_1$. Since we assume the graphs are sparse, we stop a solution path when the number of predicted edges exceeds $3p$. Consequently, a sequence of DAGs is estimated

and one of them will be picked by our model selection criterion (22) with $\alpha = 0.3$. To avoid any potential bias in the estimation, we pick a random order to cycle through all the blocks in the outer loop of Algorithm 1.

The EE sampler is used for comparisons on interventional data. Its implementation for DAG estimation was done as in Zhou (2011). We will call it the EE-DAG sampler hereafter. In each run, we simulate 10 chains with a 0.1 chance of equi-energy jumps. To obtain an estimated DAG, we threshold the average graph from the target chain (the 10^{th} chain). More precisely, a directed edge will be predicted if its posterior inclusion probability is greater than 0.5.

The HC algorithm is a standard greedy method, while the MMHC algorithm is a hybrid method. For these two algorithms, we use the R package `bnlearn` (Scutari 2010, 2017). These methods are designed specifically for observational data and thus are compared with our method only on observational data. For the HC algorithm, since the number of predicted edges is too large, we set the maximum number of parents for each node to be 2. For the MMHC algorithm we also have the option to limit the number of parents per node. But since this algorithm can estimate a reasonable number of edges, we did not set an upper limit.

The PC algorithm is a popular constraint-based algorithm for learning Bayesian networks, with an efficient implementation in the R package `pcalg` (Kalisch et al. 2012; Hauser and Bühlmann 2012). However, it may not produce a DAG for every data set, and instead its output is a completed partially directed acyclic graph (CPDAG), which contains both directed and undirected edges. To make a fair comparison, we distinguish undirected edges from directed ones in our calculation of various performance metrics, with details provided later. The tuning parameter for the PC algorithm is the significance level for conditional independence tests, which is chosen as $\alpha = 0.01$ for all data.

The GIES algorithm can learn Bayesian networks from a mixture of observational and interventional data, by searching over the so-called interventional equivalence classes. However, the implementation of this algorithm in the `pcalg` package, the only implementation we found, can only take continuous data as input. So we generated continuous data from the simulated discrete data. The detailed procedure is described in the Supplemental Material.

4.2 Results for interventional data

For each type of network, we generated graphs with $p = 50$ and $p = 100$. For each node X_j, we generated n_j data points where the node is under intervention, so that the sample size $n = \sum_{j=1}^{p} n_j$ for interventional data. We chose $n_j \in \{1, 5\}$ for all $j = 1, \ldots, p$ to test the performance of the algorithms given different amount of intervention. In particular, when $n_j = 1$ we have $n = p$, which lies on the boundary between low- and high-dimensional settings. In combination, our choices of the data size were $(n, p) \in \{(50, 50), (250, 50), (100, 100), (500, 100)\}$. For each combination of (n, p), we generated 20 data sets.

We compare the DAGs estimated by four algorithms, the CD algorithm, the EE-DAG sampler, the PC algorithm, and the GIES algorithm. For an estimated DAG, we distinguish between expected edges, which are estimated edges in the true skeleton with the correct direction, and reversed edges, which are in the true skeleton but with a reversed direction. Let P, E, R, and FP denote, respectively, the numbers of predicted edges, expected edges, reversed edges, and false positive edges (excluding the reversed ones) in an estimated DAG, and recall that s_0 is the number of edges in the true graph. Then the number of missing edges is $\mathrm{M} = s_0 - \mathrm{E} - \mathrm{R}$. The accuracy of DAG estimation is measured by the true positive rate (TPR), the false discovery

Table 1: Comparison between our CD algorithm and the PC algorithm on simulated interventional data

Graph	(n, p, s_0)	Method	P	E	R	FP	TPR	FDR	SHD	JI
Bipartite	$(100, 100, 100)$	CD	98.2	63.0	18.6	16.5	0.630	0.355	**53.5**	**0.466**
		CD*	60.5	41.2	14.5	4.7	0.412	0.316	63.5	0.345
		PC	50.0	5.6(18.3)	21.9	4.2	0.239	0.085	(80.3, 98.7)	(0.039, 0.191)
	$(500, 100, 100)$	CD	104.2	81.7	17.1	5.5	0.816	0.217	**23.8**	**0.666**
		CD*	86.3	68.8	15.8	1.7	0.688	0.203	32.9	0.586
		PC	80.5	29.2(27.1)	20.5	3.8	0.562	0.047	(47.5, 74.6)	(0.193, 0.454)
Scale-free	$(100, 100, 99)$	CD	99.2	74.8	17.9	6.5	0.756	0.245	**30.8**	0.610
		CD*	103.3	76.3	18.1	8.8	0.771	0.260	31.5	**0.611**
		PC	59.1	4.9(49.5)	2.8	1.9	0.549	0.032	(46.5, 96.0)	(0.032, 0.525)
	$(500, 100, 99)$	CD	98.8	85.0	13.4	0.4	0.859	0.140	**14.5**	**0.758**
		CD*	105.2	85.5	13.6	6.2	0.863	0.186	19.8	0.726
		PC	74.0	1.2(71.0)	0.0	1.8	0.729	0.023	(28.6, 99.5)	(0.007, 0.717)
Small-world	$(100, 100, 200)$	CD	145.2	69.1	49.1	26.9	0.346	0.523	157.9	0.250
		CD*	121.1	59.1	42.7	19.2	0.296	0.511	160.1	0.226
		PC	67.7	11.7(45.4)	7.7	3.0	0.285	0.045	(**146.0**, 191.3)	(0.046, **0.271**)
	$(500, 100, 200)$	CD	168.8	98.8	53.0	17.0	0.494	0.412	**118.2**	**0.367**
		CD*	135.1	82.0	46.6	6.5	0.410	0.393	124.5	0.324
		PC	117.0	43.0(26.2)	46.2	1.5	0.346	0.013	(132.2, 158.4)	(0.158, 0.283)
Random DAG	$(100, 100, 101.5)$	CD	91.8	56.6	24.7	10.4	0.556	0.384	55.4	0.414
		CD*	61.0	38.5	18.9	3.5	0.381	0.365	66.5	0.311
		PC	66.2	22.4(28.7)	12.5	2.6	0.506	0.040	(**53.0**, 81.8)	(0.156, **0.441**)
	$(500, 100, 102.0)$	CD	103.4	76.2	23.4	3.8	0.746	0.263	29.8	0.591
		CD*	85.8	64.2	19.8	1.8	0.636	0.253	39.6	0.524
		PC	96.7	60.5(24.2)	10.6	1.3	0.837	0.014	(**18.6**, 42.9)	(0.438, **0.755**)

CD is the result of our CD algorithm with the smallest SHD along the solution path; CD* is the result of our CD algorithm using our model selection criterion; The number in parentheses in column E for the PC algorithm reports the number of predicted undirected edges in the true skeleton

rate (FDR), the structural Hamming distance (SHD) (Tsamardinos et al. 2006), and the Jaccard index (JI), defined as TPR = E/s_0, FDR = (R + FP)/P, SHD = (R + M + FP), and JI = E/(P + s_0 − E). Note that SHD was originally defined for CPDAGs, and our definition here measures the Hamming distance between two DAGs. Both SHD and JI are single performance metrics for DAG estimation. We mark in boldface results with the optimum SHD and JI scores in the subsequent tables.

Results reported in Table 1 are the comparisons between our CD algorithm and the PC algorithm, while in Table 2 are the comparisons between our CD algorithm and the EE-DAG sampler. Results are averages over 20 data sets for each setting (n, p, s_0). For our CD algorithm, we report two results for each setting, (i) result with the smallest SHD along the solution path; (ii) result using our model selection criterion (22) with $\alpha = 0.3$. In Table 1, in order to make a clear comparison, we report lower and upper bounds of the SHD and the Jaccard index for CPDAGs estimated by the PC algorithm. Counting all undirected edges in a CPDAG that are in the true skeleton as expected edges, we will have a lower bound for the SHD and an upper bound for the Jaccard index. Counting these undirected edges as reversed edges will give us an upper bound for the SHD and a lower bound for the Jaccard index.

It is obvious from Table 1 that in majority of the cases our CD algorithm outperformed the

Table 2: Comparison between our CD algorithm and the EE-DAG sampler on simulated interventional data

Graph	(n, p, s_0)	Method	P	E	R	FP	TPR	FDR	SHD	JI
Bipartite	$(50, 50, 50)$	CD	30.8	19.4	6.5	4.8	0.389	0.362	**35.4**	0.316
		CD*	29.6	17.8	6.5	5.3	0.355	0.398	37.6	0.286
		EE	53.9	27.4	4.8	21.6	0.548	0.486	44.2	**0.362**
Scale-free	$(50, 50, 49)$	CD	48.5	30.2	6.8	11.4	0.617	0.376	**30.1**	0.452
		CD*	51.0	31.0	6.8	13.2	0.633	0.387	31.2	**0.454**
		EE	60.6	32.9	3.1	24.6	0.670	0.453	40.8	0.434
Small-world	$(50, 50, 100)$	CD	70.2	35.0	25.9	9.2	0.350	0.498	**74.2**	**0.260**
		CD*	39.5	21.2	15.8	2.5	0.212	0.458	81.2	0.181
		EE	62.4	30.6	16.1	15.7	0.306	0.507	85.0	0.234
Random DAG	$(50, 50, 46.8)$	CD	34.1	20.4	8.3	5.5	0.441	0.400	**31.9**	0.340
		CD*	26.1	16.4	6.8	2.9	0.361	0.363	33.2	0.296
		EE	50.0	25.2	7.0	17.8	0.547	0.492	39.3	**0.358**

CD is the result of our CD algorithm with the smallest SHD along the solution path; CD* is the result of our CD algorithm using our model selection criterion

PC algorithm. The SHD score is smaller than the lower bound of the PC algorithm, and the Jaccard index is higher than the upper bound of the PC algorithm. Only for random DAGs and small-world networks with $n = 100$, the SHD of our CD algorithm is slightly higher than the lower bound of the SHD while the Jaccard index is slightly lower than the upper bound of the PC algorithm, showing that our algorithm was close to the best performance one could hope for the PC algorithm. Note that when calculating the TPRs and FDRs in the table, the undirected edges are counted as expected ones, which clearly favors the PC algorithm.

We were only able to test the EE-DAG sampler on small graphs with $p = 50$ because its computing time was too long for graphs with $p = 100$. For the cases of $n = p = 50$ (Table 2), we see that the smallest SHD our CD algorithm can achieve along a solution path is 20% lower than the EE-DAG sampler for bipartite graphs, random DAGs and scale-free networks, and 10% lower for small-world networks. The EE-DAG sampler predicted much more false positive edges (FP) but slightly fewer reversed edges (R) than our CD algorithm. For the cases of $n = 5p$, we report our comparisons in the Supplemental Material. The EE-DAG sampler had an outstanding performance, with a lower SHD and a higher Jaccard index. These observations are largely in agreement with our expectation. The EE-DAG sampler searches for DAGs by sampling from a posterior distribution under the product multinomial model with a conjugate Dirichlet prior, which is close to ℓ_0 regularization when n is large. Thus, it is expected to have good performance when p is small and n is large. However, the search is combinatorial in nature, which makes it impractical for even moderately large networks (such as the graphs with $p = 100$ here). On the contrary, our CD algorithm showed no problem in estimating graphs with hundreds of nodes and can obtain comparable or better results in the cases of $n = p = 50$.

We also did a comparison between the CD algorithm and the GIES algorithm, reported in the Supplemental Material. The results seem to suggest that the GIES algorithm often selects

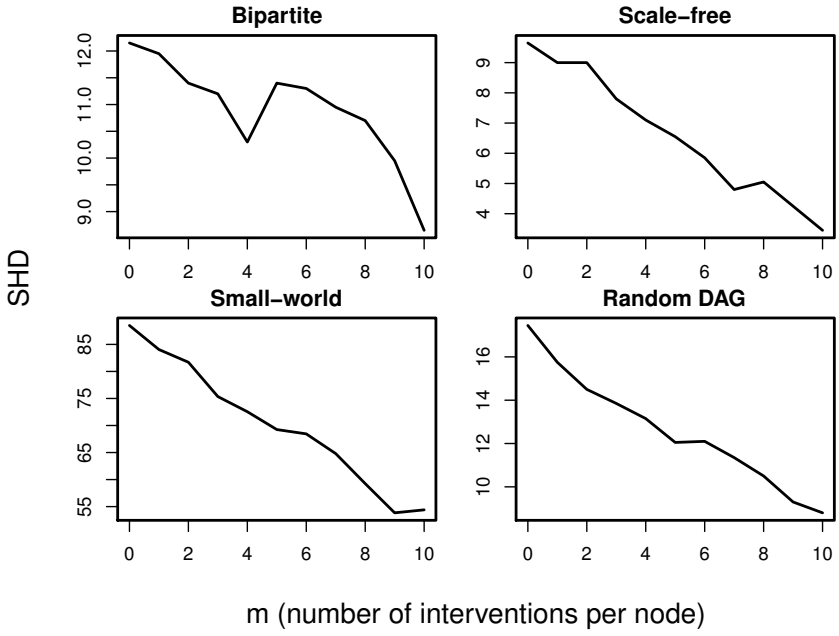


Figure 1: The effect of interventions in terms of the SHD, where each node has m interventional data points while the total sample size n is fixed

too many edges. When $n = p = 100$, the number of edges the GIES algorithm predicted was around $3s_0$ in most of the cases. Consequently, it showed a much higher FDR as well as a larger SHD. Recall that the available package for the GIES algorithm can only take continuous data, which were generated by taking a transformation of the simulated discrete data. As a result, this comparison could be confounded by the use of different although related data sets, and thus is only intended to illustrate how the two algorithms would work.

In order to show how interventional data improve the accuracy, we did more experiments. Figure 1 shows how the SHD decreases when adding intervention to an observational data set, for all four types of graphs with $n = 500$ and $p = 50$. We started with a purely observational data set, and replaced m observational data points by m interventional data points for each node, for $m = 1, 2, ..., 10$. The sample size was fixed as $n = 500$. Therefore, we would finally have a data set with 10 interventions for each node. Figure 1 shows the average of 20 experiments, with a very clear downward trend in all plots. The curve for the bipartite graph is not as smooth as the curves for the other types of networks. This is because the improvement for bipartite graphs is not as significant as the other networks.

4.3 Results for high-dimensional observational data

In this section, we apply our CD algorithm to high-dimensional observational data and compare its performance with the PC algorithm, the MMHC algorithm, and the HC algorithm.

Metrics for estimation accuracy in this section are modified from those for interventional data. Since equivalent DAGs cannot be distinguished with observational data, we define reversed edges with regard to CPDAGs. A CPDAG is a partially directed graph that has all compelled

Table 3: Comparison among our CD algorithm and other algorithms on simulated observational data

Network	(n, p, s_0)	Method	P	E	R	FP	TPR	FDR	SHD	JI
Bipartite	$(50, 200, 200.0)$	CD	108.7	69.6	20.6	18.6	0.348	0.357	**148.9**	**0.290**
		CD*	90.2	59.6	17.8	12.8	0.298	0.333	153.2	0.258
		PC	75.7	26.9	34.2	14.6	0.134	0.643	187.7	0.108
		MMHC	175.4	72.2	20.4	82.8	0.361	0.588	210.6	0.239
		HC	378.1	111.5	32.9	233.8	0.557	0.705	322.4	0.239
Scale-free	$(50, 200, 199.0)$	CD	139.6	83.8	15.8	39.9	0.421	0.402	**155.1**	0.326
		CD*	201.8	106.8	21.6	73.4	0.537	0.470	165.6	**0.365**
		PC	99.5	46.5	23.1	30.0	0.234	0.532	182.5	0.185
		MMHC	176.8	93.2	16.1	67.5	0.468	0.472	173.2	0.330
		HC	377.8	121.0	28.1	228.8	0.608	0.680	306.9	0.266
Small-world	$(50, 200, 400.0)$	CD	88.2	28.9	36.2	23.0	0.072	0.533	**394.1**	0.058
		CD*	296.5	84.6	110.8	101.1	0.212	0.714	416.5	0.138
		PC	70.2	7.3	54.4	8.6	0.018	0.898	401.2	0.016
		MMHC	249.2	84.3	85.7	79.3	0.211	0.661	395.0	**0.150**
		HC	357.3	87.9	97.5	171.9	0.220	0.754	484.1	0.131
Random DAG	$(50, 200, 203.6)$	CD	113.0	68.8	28.1	16.1	0.339	0.386	**150.9**	0.278
		CD*	101.2	63.7	25.1	12.4	0.315	0.364	152.3	0.265
		PC	97.3	47.1	37.0	13.2	0.233	0.515	169.7	0.187
		MMHC	179.8	86.4	31.1	62.4	0.427	0.520	179.6	**0.292**
		HC	376.3	96.3	52.1	227.8	0.475	0.744	335.1	0.200

CD* is the result of our CD algorithm using our model selection criterion

(directed) edges in the equivalent class of a DAG. We calculate CPDAGs for both an estimated DAG and the true DAG. A reversed edge (R) refers to a predicted edge that satisfies the following two conditions: i) Its direction in the estimated DAG is wrong compared to the true DAG. ii) The direction of this edge is inconsistent between the CPDAGs of the true and estimated DAGs, including the case where the edge is directed in one CPDAG but undirected in the other. Likewise, we define reversed edges in a CPDAG predicted by the PC algorithm as edges in the true skeleton that have an inconsistent direction with the true CPDAG. The number of expected edges (E) is the number of estimated edges in the true skeleton excluding those reversed ones.

For high-dimensional data, we generated graphs with $p = 200$ for each type of the networks. We chose $n = 50$ and the number of true edges s_0 ranged from 190 to 400 for these graphs. Again, 20 data sets were generated for each combination of (n, p).

Table 3 summarizes the comparison results. One sees that our model selection criterion works quite well: The SHDs of CD and CD* in Table 3 are very close. Our CD algorithm has the lowest SHD for all networks, and the Jaccard index is also quite high compared to other algorithms for most of the networks. When running the HC algorithm with the default setting, it tends to predict too many edges, which makes the comparison meaningless. For example, for a scale-free network with $p = 200$, the HC algorithm predicted more than 2,000 edges. Therefore, we set the maximum number of parents for each node to be 2 for the HC algorithm. However, it still predicted too many edges that it had a much higher TPR as well as a higher FDR and larger SHD than all the other algorithms. The PC algorithm predicted too few edges in scale-

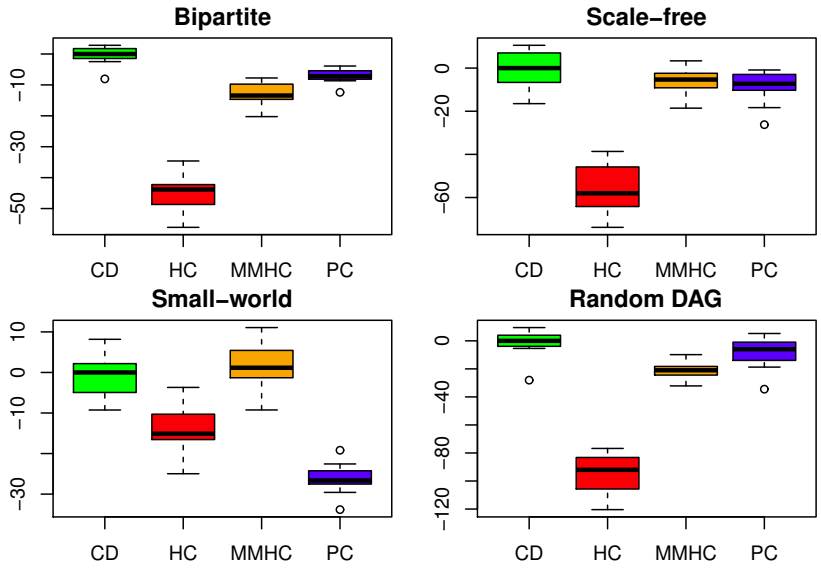


Figure 2: Box-plot of test data log-likelihood for four algorithms with log-likelihood scaled by the sample size $n = 50$

free networks and small-world networks, which led to a substantially lower TPRs and JIs, and the FDRs were quite high for bipartite networks. The MMHC algorithm, on the other hand, predicted a comparable number of edges as the true DAG in most cases. Its performance was in between of our CD algorithm and the PC algorithm. Our CD algorithm presents a clear advantage over all other algorithms in these high-dimensional cases.

To further evaluate the quality of estimated networks, we computed test data log-likelihood to compare the predictive power. We generated 500 test data sets of the same size ($n = 50$) for each DAG with $p = 200$. We used each estimated graph to calculate the total log-likelihood of a test data set. Note that the output graph of the PC algorithm is a CPDAG, for which we cannot directly calculate the test data log-likelihood. However, since the likelihood for any data set under every DAG in an equivalence class is the same, we converted a CPDAG output by the PC algorithm to an arbitrary DAG in the equivalence class and then calculated its test data log-likelihood. Figure 2 is the box-plot of test data log-likelihood for the four types of graphs in terms of the difference from the median of the test data log-likelihood of the CD algorithm. DAGs estimated by the CD algorithm were chosen by our model selection criterion.

It is seen from Figure 2 that our CD algorithm has the highest test data log-likelihood for three out of the four types of networks. Only for small-world networks, our CD algorithm has a slightly lower log-likelihood than the MMHC algorithm. This shows that our method also has a very good predictive power in high-dimensional cases. We see that the HC algorithm has a much lower test data log-likelihood for most cases, which suggests overfitting given the observation that this algorithm often predicts too many edges. These results demonstrate the critical role of sparsity not only in structure estimation but also in predictive modeling.

4.4 Timing comparison

We comment briefly on the comparison of running time among the algorithms. The running time for generating a whole solution path for the interventional data (results in Tables 1 and 2) was within 40 seconds for all graphs. The PC algorithm was about two times faster than our CD algorithm for bipartite graphs and random DAGs, but it did not scale well for scale-free networks and small-world networks. For these two types of networks, running time of the PC algorithm was much longer for $n = 500$ and $p = 100$. For the high-dimensional data in Table 3, it took between 2 and 20 seconds for our method to compute the entire solution path. The speed of the PC algorithm was quite comparable to our CD algorithm on these data sets. The MMHC algorithm was faster which took at most 5 seconds for all data sets. The fastest algorithm was the HC algorithm, however, its accuracy in learning Bayesian networks was too bad so we will not go into details for this algorithm. Our method gives a principled way to incorporate interventional data and is often more accurate than the other competitors. These merits in performance justify its utility. In addition, there is room for a more efficient implementation of our algorithm which may improve its speed substantially.

5 Applications to Real Networks

In this section, we apply our CD algorithm to real networks. In Section 5.1, we examine how the proposed multi-logit model compares to the product multinomial model by comparing our CD algorithm to the K2 algorithm (Cooper and Herskovits 1992). We will then apply our method to a real data set in Section 5.2.

5.1 Comparison with the K2 Algorithm

The K2 algorithm is a well-known method for learning discrete Bayesian networks based on a product multinomial model. However, it requires an input ordering of the nodes. A wrong ordering can severely damage the quality of the estimated graph. Therefore, we provide the K2 algorithm with an ordering that is compatible with the true DAG to obtain the best estimation. In order to conduct a fair comparison, we also input the same ordering to our CD algorithm by only running the inner loop of Algorithm 2, which is equivalent to a sequence of $p - 1$ penalized multi-logit regression problems. With a known ordering, a main difference between the two algorithms is the underlying model, the multi-logit model for our CD algorithm and the multinomial model for the K2 algorithm. We used a Matlab package `K2` (Bielza et al. 2011) to run the algorithm. The K2 algorithm also requires an upper bound for the maximum number of parents for each node. In our experiments, we set the upper bound to be 4. We chose 8 real networks provided by the `bnlearn` package, where p ranges from 8 to 441. Observational data were simulated from these networks, and for each DAG, 20 data sets were generated independently according to a product multinomial model. This comparison will demonstrate how well our proposed multi-logit model approximates the multinomial model.

Summary of the comparison for the 8 networks is provided in Table 4. Since a correct ordering is given, there will not be any reversed edges, and thus, in this table, we only report P, TPR, FDR, SHD, and JI. Here we matched the number of predicted edges of our CD algorithm with the K2 algorithm. It can be seen that for most graphs the SHD for our CD algorithm is lower than that of the K2 algorithm, while the JI is higher, except the networks asia and hailfinder.

Table 4: Comparison between our CD algorithm and the K2 Algorithm

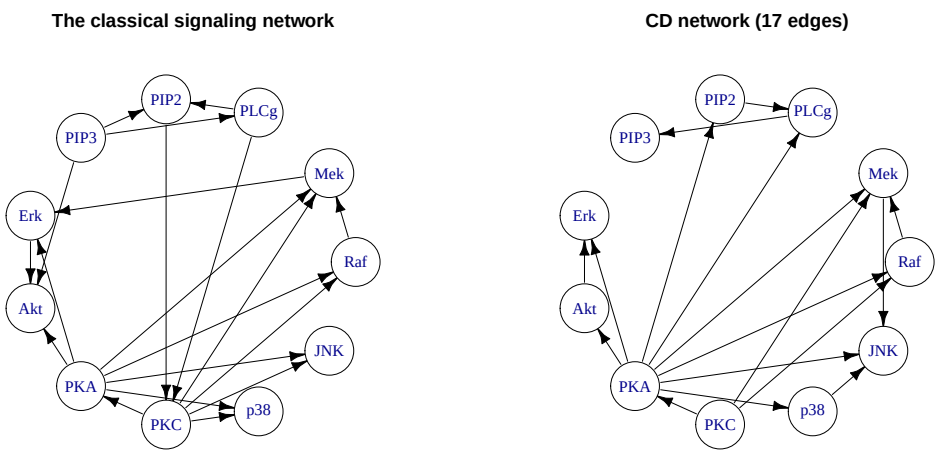
Network	(n, p, s_0)	CD Algorithm P	CD Algorithm TPR	CD Algorithm FDR	CD Algorithm SHD	CD Algorithm JI	K2 Algorithm P	K2 Algorithm TPR	K2 Algorithm FDR	K2 Algorithm SHD	K2 Algorithm JI
asia	$(250, 8, 8)$	10.2	0.719	0.430	6.7	0.469	10.1	0.838	0.331	**4.7**	**0.597**
sachs	$(250, 11, 17)$	14.5	0.732	0.133	**6.6**	**0.659**	14.7	0.538	0.374	13.3	0.408
child	$(250, 20, 25)$	31.1	0.656	0.469	**23.3**	**0.416**	30.1	0.602	0.500	25.0	0.376
insurance	$(250, 27, 52)$	50.4	0.473	0.511	**53.2**	**0.316**	51.1	0.414	0.578	60.0	0.265
alarm	$(250, 37, 46)$	60.8	0.664	0.497	**45.6**	**0.401**	60.8	0.618	0.531	49.9	0.364
hailfinder	$(250, 56, 66)$	80.2	0.525	0.558	76.9	0.313	79.1	0.546	0.542	**73.0**	**0.331**
hepar2	$(250, 70, 123)$	137.6	0.269	0.756	**194.5**	**0.146**	139.9	0.236	0.792	204.8	0.124
pigs	$(250, 441, 592)$	773.65	0.863	0.334	**343.5**	**0.600**	788.8	0.704	0.472	547.4	0.432

Since the data sets were simulated by product multinomial models, this result confirms that our proposed multi-logit model serves as a good approximation to the full multinomial model. This comparison also suggests that the group norm regularization in our method may be more efficient than using an upper bound on the parent size as in the K2 algorithm.

5.2 Application to flow cytometry data

We consider in this section applying the CD algorithm to a real data set that has been extensively studied. The data set was generated from a flow cytometry experiment conducted by Sachs et al. (2005), who studied a well-known signaling network in human primary CD4+ T-cells of the immune system. This chosen network was perturbed by various stimulatory and inhibitory interventions. Each interventional condition was applied to an individual component of the network. Simultaneous measurements were taken on $p = 11$ proteins and phospholipids of this network from individual cells under each condition. Since three interventions were targeted at proteins that were not measured, samples collected under these conditions were observational. Among the 11 measured components, five proteins and phospholipids were perturbed. The data set contains measurements for $n = 5,400$ cells. Each variable has three levels (high, medium and low), and consequently, the size of a component group of $\boldsymbol{\beta}$ is 6 for this data set.

Figure 3A is a plot for the known causal interactions among the 11 components of this signaling network. These causal relationships are well-established, and no consensus has been reached on interactions beyond those present in the network. This network structure is often used as the benchmark to assess the accuracy of an estimated network. Therefore, we call it the consensus model. Our estimated network by the CD algorithm with the smallest SHD along the solution path is shown in Figure 3B. The DAG is qualitatively close to the consensus model. More detailed performance measures are reported in Table 5, including both results for the DAG with the smallest SHD (CD algorithm) and the one selected by our model selection criterion (CD algorithm*). As a comparison, we include the DAGs estimated by three competing methods, the order-graph sampler (Ellis and Wong 2008), the EE-DAG sampler, and the PC algorithm. Our CD algorithm showed a very competitive performance, predicting more or comparable number of expected edges and fewer reversed edges than the other three methods. Our method also had the least number of false positive edges among all the methods. All these led to the lowest SHD and highest Jaccard index for our CD algorithm. Note that for the PC algorithm, we counted



(A) (B)

Figure 3: (A) The consensus signaling network in human immune system cells, (B) DAG estimated by the CD algorithm

all 3 undirected edges in the true skeleton as expected edges in the calculation of the SHD and JI. Yet our CD algorithm still outperformed it. Markov chain Monte Carlo (MCMC) methods for DAG estimation often have good performance when the number of nodes p is small, but they do not scale well. Thus, it is comforting to see that our method, which can handle larger networks, outperforms MCMC methods on this relatively small network.

Table 5: Comparison on the flow cytometry data set

Method	P	E	R	M	FP	SHD	JI
CD algorithm	17	10	3	7	4	**14**	**0.370**
CD algorithm*	14	8	3	9	3	15	0.308
PC algorithm	17	5(3)	4	8	5	17	0.276
Order-graph sampler	20	8	4	8	8	20	0.250
EE-DAG sampler	26	9	6	5	11	22	0.243

The order-graph sampler result comes from the mean graph (Figure 11 in Ellis and Wong 2008)

6 Discussions

We have developed a maximum penalized likelihood method for estimating sparse discrete Bayesian networks under a multi-logit model. In order to avoid penalizing separately individual dummy variables for a factor, a group norm penalty is utilized to encourage sparsity at the factor level. A blockwise coordinate descent algorithm is developed where each coordinate descent step is solved by iteratively applying a quadratic approximation. The acyclicity constraint imposed on the structure of Bayesian networks can be enforced in a natural way by the coordinate descent algorithm. Our method has been evaluated on simulated graphs and

real-world networks, with both interventional and observational data. We have demonstrated that the CD algorithm outperforms many existing methods, particularly when $n \leq p$. We have also performed an analysis of a flow cytometry data set generated from a signaling network in human immune system cells. The DAG estimated by the CD algorithm is close to the consensus model. Since the true network is not available, the estimated edges provide candidate causal interactions that could be tested in future experiments.

Computation for estimating discrete Bayesian networks is demanding due to the size of the parameter space and the nonlinear nature of the multi-logit model. There is room for improving the efficiency of the CD algorithm. For example, one may incorporate the idea of stochastic gradient descent in the quadratic approximation step, which will reduce significantly the computation. Moreover, since our search space is nonconvex, introducing such components of stochastic optimization may also increase the chance of finding a global minimizer of the penalized loss function. Other future directions include studying the consistency of our penalized estimator when the number of nodes $p = p_n$ grows with the sample size n and investigating the use of group concave penalties.

Appendix: Asymptotic Theory

In this Appendix, we establish asymptotic theory for the DAG estimator $\hat{\boldsymbol{\beta}}_\lambda$ (12) assuming that p is fixed and $n \to \infty$. By rearranging and relabeling individual components, we rewrite $\boldsymbol{\beta}$ as $\boldsymbol{\phi} = (\boldsymbol{\phi}_{(1)}, \boldsymbol{\phi}_{(2)})$, where $\boldsymbol{\phi}_{(1)} = \text{vec}(\boldsymbol{\beta}_{1\cdot 1}, \ldots, \boldsymbol{\beta}_{1\cdot p}, \ldots\ldots, \boldsymbol{\beta}_{p\cdot 1}, \ldots, \boldsymbol{\beta}_{p\cdot p})$ is the parameter vector of interest and $\boldsymbol{\phi}_{(2)} = \text{vec}(\boldsymbol{\beta}_{1\cdot 0}, \ldots, \boldsymbol{\beta}_{p\cdot 0})$ denotes the vector of intercepts. Hereafter, we denote by ϕ_j the j^{th} group of $\boldsymbol{\phi}$, such that $\phi_1 = \boldsymbol{\beta}_{1\cdot 1}$, $\phi_2 = \boldsymbol{\beta}_{1\cdot 2}$, $\ldots$, $\phi_{p^2} = \boldsymbol{\beta}_{p\cdot p}$, and so on. We say $\boldsymbol{\phi}$ is acyclic if the graph $\mathcal{G}_{\boldsymbol{\phi}}$ induced by $\boldsymbol{\phi}$ (or the corresponding $\boldsymbol{\beta}$) is acyclic.

Define $\boldsymbol{\phi}_{[k]}$ ($k \in \{1, \ldots, p\}$) to be the parameter vector obtained from $\boldsymbol{\phi}$ by setting $\boldsymbol{\beta}_{k\cdot i} = \mathbf{0}$ for $i = 1, \ldots, p$. In other words, the DAG $\mathcal{G}_{\boldsymbol{\phi}_{[k]}}$ is obtained by deleting all edges pointing to the k^{th} node in $\mathcal{G}_{\boldsymbol{\phi}}$; see (10). We assume the data set $\mathcal{X}$ consists of $(p+1)$ blocks, denoted by $\mathcal{X}^j$ of size $n_j \times p$, $j = 1, \ldots, p+1$. The node X_j is experimentally fixed in $\mathcal{X}^j$ for the first p blocks, while the last block contains purely observational data. Let $\mathcal{I}_j$ be the set of row indices of $\mathcal{X}^j$. As demonstrated by (2), we can model interventional data in the k^{th} block of the data matrix $\mathcal{X}^k$ as *i.i.d.* observations from a joint distribution factorized according to $\mathcal{G}_{\boldsymbol{\phi}_{[k]}}$. Denote the corresponding probability mass function by $p(\mathbf{x}|\boldsymbol{\phi}_{[k]})$, where $\mathbf{x} = (x_1, \ldots, x_p)$ and $x_j \in \{1, \ldots, r_j\}$ for $j = 1, \ldots, p$. To simplify our notation, denote the parameter for the $(p+1)$th block by $\boldsymbol{\phi}_{[p+1]} = \boldsymbol{\phi}$. Then the log-likelihood of $\mathcal{X}$ is

$$L(\boldsymbol{\phi}) = \sum_{k=1}^{p+1} L_k(\boldsymbol{\phi}_{[k]}) = \sum_{k=1}^{p+1} \log p(\mathcal{X}^k \mid \boldsymbol{\phi}_{[k]}), \tag{23}$$

where $\log p(\mathcal{X}^k|\boldsymbol{\phi}_{[k]}) = \sum_{h \in \mathcal{I}_k} \log(p(\mathcal{X}_{h\cdot}|\boldsymbol{\phi}_{[k]}))$ and $\mathcal{X}_{h\cdot} = (\mathcal{X}_{h1}, \ldots, \mathcal{X}_{hp})$. The penalized log-likelihood function with a tuning parameter $\lambda_n > 0$ is

$$\begin{aligned} R(\boldsymbol{\phi}) &= L(\boldsymbol{\phi}) - \lambda_n \sum_{j=1}^{p^2} \|\phi_j\|_2 \\ &= \sum_{k=1}^{p+1} L_k(\boldsymbol{\phi}_{[k]}) - \lambda_n \sum_{j=1}^{p^2} \|\phi_j\|_2, \end{aligned} \tag{24}$$

where the component group ϕ_j ($j = 1, \ldots, p^2$) represents the influence of one variable on another. Let $\Omega = \{\boldsymbol{\phi} : \mathcal{G}_{\boldsymbol{\phi}} \text{ is a DAG}\}$ be the parameter space. A penalized estimator $\hat{\boldsymbol{\phi}}$ is obtained by maximizing $R(\boldsymbol{\phi})$ in Ω.

Though interventional data help distinguish equivalent DAGs, the following notion of natural parameters is needed to completely establish identifiability of DAGs for the case where each variable has interventional data. We say that i is an ancestor of j in a DAG $\mathcal{G}$ if there exists at least one path from i to j. Denote the set of ancestors of j by $\text{an}(j)$.

Definition 1 (Natural parameters)**.** We say that $\boldsymbol{\phi} \in \Omega$ is natural if $i \in \text{an}(j)$ in $\mathcal{G}_{\boldsymbol{\phi}}$ implies that j is not independent of i under the joint distribution given by $\boldsymbol{\phi}_{[i]}$ for all $i, j = 1, \ldots, p$.

For a causal DAG, a natural parameter implies that the effects along multiple causal paths connecting the same pair of nodes do not cancel. This is a reasonable assumption for many real-world problems, and is much weaker than the faithfulness assumption. Under the faithfulness assumption, all conditional independence restrictions can be read off from d-separations in the DAG. If nodes i and j are independent in $\boldsymbol{\phi}_{[i]}$, then by faithfulness the nodes i and j must be separated by empty set and thus $i \notin \text{an}(j)$ in $\mathcal{G}_{\boldsymbol{\phi}_{[i]}}$. This implies that $i \notin \text{an}(j)$ in $\mathcal{G}_{\boldsymbol{\phi}}$ as well, by the construction of $\mathcal{G}_{\boldsymbol{\phi}_{[i]}}$. Indeed, we see that the faithfulness assumption implies the natural parameter assumption.

To establish asymptotic properties of our penalized likelihood estimator, we make the following assumptions:

(A1) The true parameter $\boldsymbol{\phi}^*$ is natural and an interior point of Ω.

(A2) The parameter $\boldsymbol{\theta}_j$ of the conditional distribution $[X_j | \Pi_j^{\mathcal{G}}; \boldsymbol{\theta}_j]$ is identifiable for each $j = 1, \ldots, p$. The log-likelihood function $\ell_j(\boldsymbol{\theta}_j) = \log p(x_j | \Pi_j^{\mathcal{G}}; \boldsymbol{\theta}_j)$ is strictly concave and continuously three times differentiable for any interior point.

Recall that the k^{th} block of our data, $\mathcal{X}^k$, can be regarded as an *i.i.d.* sample of size n_k from the distribution $p(\mathbf{x}|\boldsymbol{\phi}^*_{[k]})$ for all k, while we define $\boldsymbol{\phi}^*_{[p+1]} = \boldsymbol{\phi}^*$ for the last block of observational data.

Theorem 1. *Assume (A1) and (A2). If $p(\mathbf{x}|\boldsymbol{\phi}_{[k]}) = p(\mathbf{x}|\boldsymbol{\phi}^*_{[k]})$ for all possible $\mathbf{x}$ and all $k = 1, \ldots, p$, then $\boldsymbol{\phi} = \boldsymbol{\phi}^*$. Furthermore, if $n_k \gg \sqrt{n}$ for all $k = 1, \ldots, p$, then for any $\boldsymbol{\phi} \neq \boldsymbol{\phi}^*$,*

$$P(L(\boldsymbol{\phi}^*) > L(\boldsymbol{\phi})) \to 1 \quad \text{as } n \to \infty. \tag{25}$$

Theorem 2. *Assume (A1) and (A2). If $\lambda_n / \sqrt{n} \to 0$ and $n_k \gg \sqrt{n}$ for all $k = 1, \ldots, p$, then there exists a global maximizer $\hat{\boldsymbol{\phi}}$ of $R(\boldsymbol{\phi})$ such that $\|\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}^*\|_2 = O_p(n^{-1/2})$.*

Proofs of the two theorems are relegated to the Supplemental Material. Theorem 1 confirms that the causal DAG model is identifiable with interventional data assuming a natural parameter. Theorem 2 implies that there is a $\sqrt{n}$-consistent global maximizer of $R(\boldsymbol{\phi})$ with the group norm penalty. Note that Assumption (A2) does not specify a particular choice of model for the conditional distribution $[X_j | \Pi_j^{\mathcal{G}}]$ and thus these theoretical results apply to a large class of DAG models for discrete data. In particular, the multi-logit regression model (4) satisfies (A2).

Remark 2. The assumption on the sample size of interventional data, $n_k \gg \sqrt{n}$, imposes a lower bound on how fast the fraction $\alpha_k = n_k / n \gg n^{-1/2}$ can approach zero for $k = 1, \ldots, p$. Although

this allows the observational data to dominate when $\alpha_k \to 0$, the fractions of interventional data must be larger than the typical order $O_p(n^{-1/2})$ of statistical errors so that (25) can hold to establish identifiability of the true causal DAG parameter $\boldsymbol{\phi}^*$. This guarantees that the global maximizer $\hat{\boldsymbol{\phi}}$ will locate in a neighborhood of $\boldsymbol{\phi}^*$ with high probability. Once in this neighborhood, the convergence rate of $\hat{\boldsymbol{\phi}}$ then depends on the size n of all data, both interventional and observational. Therefore, increasing the size of observation data will lead to more accurate estimate $\hat{\boldsymbol{\phi}}$ as long as we keep $\alpha_k \gg n^{-1/2}$ for $k = 1, \ldots, p$.

Remark 3. It is interesting to generalize the above asymptotic results to the case where $p = p_n$ grows with the sample size n, say, by developing nonasymptotic bounds on the ℓ_2 estimation error $\|\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}^*\|_2$. However, in order to estimate the causal network consistently, sufficient interventional data are needed for each node, i.e., n_k must approach infinity, and thus $p/n \to 0$ as $n \to \infty$. This limits us to the low-dimensional setting with $p < n$. Suppose we have a large network with $p \gg n$. One may first apply some regularization method on observational data to screen out independent nodes and to partition the network into small subgraphs that are disconnected to one another. Then for each small subgraph, we can afford to generate enough interventional data for every node and apply the method in this paper to infer the causal structure. Our asymptotic theory provides useful guidance for the analysis in the second step.

For purely observational data, the theory becomes more complicated due to the existence of equivalent DAGs and parameterizations. It is left as future work to establish the consistency of a global maximizer for high-dimensional observational data.

References

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

Barabási, A.-L. and Albert, R. (1999), "Emergence of Scaling in Random Networks," *Science*, 286, 509–512.

Bielza, C., Li, G., and Larranaga, P. (2011), "Multi-dimensional classification with Bayesian networks," *International Journal of Approximate Reasoning*, 52, 705–727.

Bouckaert, R. R. (1993), "Probabilistic Network Construction Using the Minimum Description Length Principle," in *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: European Conference ECSQARU '93*, Springer, vol. 747 of *Lecture Notes in Computer Science*, pp. 41–48.

— (1994), "Probabilistic Network Construction Using the Minimum Description Length Principle," Tech. Rep. RUU-CS-94-27, Department of Computer Science, Utrecht University.

Buntine, W. (1991), "Theory Refinement on Bayesian Networks," in *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., pp. 52–60.

Chickering, D. M. and Heckerman, D. (1997), "Efficient Approximations for the Marginal Likelihood of Bayesian Networks with Hidden Variables," *Machine Learning*, 29, 181–212.

Cooper, G. F. and Herskovits, E. (1992), "A Bayesian Method for the Induction of Probabilistic Networks from Data," *Machine Learning*, 9, 309–347.

Cooper, G. F. and Yoo, C. (1999), "Causal discovery from a mixture of experimental and observational data," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., pp. 116–125.

Csárdi, G. and Nepusz, T. (2006), "The igraph Software Package for Complex Network Research," *InterJournal: Complex Systems*, 1695.

Ellis, B. and Wong, W. H. (2008), "Learning Causal Bayesian Network Structures from Experimental Data," *Journal of the American Statistical Association*, 103, 778–789.

Erdos, P. and Rényi, A. (1960), "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, 5, 17–60.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007), "Pathwise Coordinate Optimization," *The Annals of Applied Statistics*, 1, 302–332.

Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, 33, 1–22.

Fu, F. and Zhou, Q. (2013), "Learning Sparse Causal Gaussian Networks With Experimental Intervention: Regularization and Coordinate Descent," *Journal of the American Statistical Association*, 108, 288–300.

Fu, W. (1998), "Penalized Regressions: The Bridge versus the Lasso," *Journal of Computational and Graphical Statistics*, 7, 397–416.

Gámez, J. A., Mateo, J. L., and Puerta, J. M. (2011), "Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood," *Data Mining and Knowledge Discovery*, 22, 106–148.

Han, S. W., Chen, G., Cheon, M.-S., and Zhong, H. (2016), "Estimation of Directed Acyclic Graphs Through Two-Stage Adaptive Lasso for Gene Network Inference," *Journal of the American Statistical Association*, 111, 1004–1019.

Hauser, A. and Bühlmann, P. (2012), "Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs," *Journal of Machine Learning Research*, 13, 2409–2464.

Hauser, A. and Bühlmann, P. (2015), "Jointly interventional and observational data: estimation of interventional Markov equivalence classes of directed acyclic graphs," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77, 291–318.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995), "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Machine Learning*, 20, 197–243.

Herskovits, E. and Cooper, G. (1990), "Kutató: An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases," in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, Elsevier Science Inc., pp. 117–128.

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

Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. (2012), "Causal Inference Using Graphical Models with the R Package pcalg," *Journal of Statistical Software*, 47, 1–26.

Kou, S., Zhou, Q., and Wong, W. H. (2006), "Equi-energy sampler with applications in statistical inference and statistical mechanics (with discussion)," *The Annals of Statistics*, 34, 1581–1652.

Lam, W. and Bacchus, F. (1994), "Learning Bayesian Belief Networks: An Approach Based on the MDL Principle," *Computational Intelligence*, 10, 269–293.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016), "Gradient descent only converges to minimizers," vol. 49, pp. 1–12.

Meganck, S., Leray, P., and Manderick, B. (2006), "Learning causal Bayesian networks from observations and experiments: A decision theoretic approach," in *International Conference on Modeling Decisions for Artificial Intelligence*, Springer, pp. 58–69.

Meier, L., van de Geer, S., and Bühlmann, P. (2008), "The Group Lasso for Logistic Regression," *Journal of the Royal Statistical Society. Series B (Statistical Methodology).*, 70, 53–71.

Pearl, J. (2003), "Causality: Models, Reasoning, and Inference," *Econometric Theory*, 19, 675–685.

Peér, D., Regev, A., Elidan, G., and Friedman, N. (2001), "Inferring subnetworks from perturbed expression profiles," *Bioinformatics*, 17, S215–S224.

Pournara, I. and Wernisch, L. (2004), "Reconstruction of gene networks using Bayesian learning and manipulation experiments," *Bioinformatics*, 20, 2934–2942.

Sachs, K., Perez, O., Peér, D., Lauffenburger, D. A., and Nolan, G. P. (2005), "Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data," *Science*, 308, 523–529.

Schmidt, M. and Murphy, K. (2006), "LassoOrderSearch: Learning Directed Graphical Model Structure using ℓ_1-Penalized Regression and Order Search," *Learning*, 8, 2.

Schmidt, M., Niculescu-Mizil, A., Murphy, K., et al. (2007), "Learning graphical model structure using ℓ_1-regularization paths," in *AAAI*, vol. 7, pp. 1278–1283.

Scutari, M. (2010), "Learning Bayesian Networks with the bnlearn R Package," *Journal of Statistical Software*, 35, 1–22.

— (2016), "An Empirical-Bayes Score for Discrete Bayesian Networks," in *Conference on Probabilistic Graphical Models*, pp. 438–448.

— (2017), "Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package," *Journal of Statistical Software*, 77, 1–20.

Shojaie, A., Jauhiainen, A., Kallitsis, M., and Michailidis, G. (2014), "Inferring regulatory networks by combining perturbation screens and steady state gene expression profiles," *PloS one*, 9, e82393.

Shojaie, A. and Michailidis, G. (2010), "Penalized Likelihood Methods for Estimation of Sparse High-Dimensional Directed Acyclic Graphs," *Biometrika*, 97, 519–538.

Spirtes, P., Glymour, C., and Scheines, R. (1993), *Causation, Prediction, and Search*, Springer-Verlag.

Suzuki, J. (1993), "A Construction of Bayesian Networks from Databases Based on an MDL Principle," in *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 266–273.

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

Tseng, P. and Yun, S. (2009), "A Coordinate Gradient Descent Method for Nonsmooth Separable Minimization," *Mathematical Programming*, 117, 387–423.

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

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S*, New York: Springer, 4th ed., iSBN 0-387-95457-0.

Watts, D. J. and Strogatz, S. H. (1998), "Collective Dynamics of 'Small-World' Networks," *Nature*, 393, 440–442.

Wu, T. and Lange, K. (2008), "Coordinate Descent Algorithms for Lasso Penalized Regression," *The Annals of Applied Statistics*, 2, 224–244.

Yuan, M. and Lin, Y. (2006), "Model Selection and Estimation in Regression with Grouped Variables," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68, 49–67.

Zhou, Q. (2011), "Multi-Domain Sampling with Applications to Structural Inference of Bayesian Networks," *Journal of the American Statistical Association*, 106, 1317–1330.

Zhu, J. and Hastie, T. (2004), "Classification of Gene Microarrays by Penalized Logistic Regression," *Biostatistics*, 5, 427–443.