Optimizing Regularized Cholesky Score for Order-Based Learning of Bayesian Networks

Qiaoling Ye, Arash A. Amini, and Qing Zhou

Abstract—Bayesian networks are a class of popular graphical models that encode causal and conditional independence relations among variables by directed acyclic graphs (DAGs). We propose a novel structure learning method, annealing on regularized Cholesky score (ARCS), to search over topological sorts, or permutations of nodes, for a high-scoring Bayesian network. Our scoring function is derived from regularizing Gaussian DAG likelihood, and its optimization gives an alternative formulation of the sparse Cholesky factorization problem from a statistical viewpoint. We combine simulated annealing over permutation space with a fast proximal gradient algorithm, operating on triangular matrices of edge coefficients, to compute the score of any permutation. Combined, the two approaches allow us to quickly and effectively search over the space of DAGs without the need to verify the acyclicity constraint or to enumerate possible parent sets given a candidate topological sort. The annealing aspect of the optimization is able to consistently improve the accuracy of DAGs learned by greedy and deterministic search algorithms. In addition, we develop several techniques to facilitate the structure learning, including pre-annealing data-driven tuning parameter selection and post-annealing constraint-based structure refinement. Through extensive numerical comparisons, we show that ARCS outperformed existing methods by a substantial margin, demonstrating its great advantage in structure learning of Bayesian networks from both observational and experimental data. We also establish the consistency of our scoring function in estimating topological sorts and DAG structures in the large-sample limit. Source code of ARCS is available at https://github.com/yeqiaoling/arcs_bn.

Index Terms—Bayesian networks, proximal gradient, regularized likelihood, simulated annealing, sparse Cholesky factorization, structure learning, topological sorts.

1 Introduction

Bayesian networks (BNs) are a class of graphical models, whose structure is represented by a directed acyclic graph (DAG). They are commonly used to model causal networks and conditional independence relations among random variables. The past decades have seen many successful applications of Bayesian networks in computational biology, medical science, document classification, image processing, etc. As the relationships among variables in a BN are encoded in the underlying graph, various approaches have been put forward to estimate DAG structures from data. Constraint-based approaches, such as the PC algorithm [1], perform a set of conditional independence tests to detect the existence of edges. In score-based approaches, a network structure is identified by optimizing a score function [2, 3]. There are also hybrid approaches, such as the max-min hill-climbing algorithm [4], which use a constraint-based method to prune the search space, followed by a search for a high-scoring network structure.

The score-based search has been applied to three different search spaces: the DAG space [2, 5], the equivalence classes [2, 6] and the ordering space (or the space of topological sorts) [7, 8]. In this paper, we focus on the order-based search, which has two major advantages. First, the existence of an ordering among nodes guarantees a graph structure that satisfies the acyclicity constraint. Second, the space of orderings is significantly smaller than the space of DAGs or of the equivalence classes. Consequently, several lines of research have developed efficient order-based methods for DAG learning. Some methods adopt a greedy search in conjunction with various operators that propose moves in the ordering space [8, 9, 10, 11]. A greedy search, however, may easily be trapped in a local minimum, and thus different techniques were proposed to perform a more global search [7, 12, 13, 14]. In particular, stochastic optimization, such as the genetic algorithm [7, 11, 15] and Markov chain Monte Carlo [16, 17, 18], has been advocated as a promising way to perform global search over the ordering space. Under certain identifiability assumptions, sequential order search algorithms have been developed recently as well [19].

In spite of these methodological and algorithmic advances, there are a few difficulties in score-based learning of topological sorts for DAGs. First, the score of an ordering is usually defined by the score of the optimal DAG compatible with the ordering. The computational complexity of finding the optimal DAG given an ordering, typically by enumerating all possible parent sets for each node [20], can be as high as $O(p^{k+1})$ for p nodes and a pre-specified maximum indegree of k. Such computation is needed for every ordering evaluated by a search algorithm, which becomes prohibitive when k is large. Second, although the ordering space is smaller than the graph space, optimization over orderings is still a hard combinatorial problem due to the NP-hard nature of structure learning of BNs [21]. It is not surprising that the performance of the above stochastic optimization algorithms degrades severely for large graphs.

Motivated by these challenges, we develop a new order-based method for learning Gaussian DAGs by optimizing a regularized likelihood score. Representing an ordering by the corresponding permutation matrix P, the weighted adjacency matrix of a Gaussian DAG can be coded into a lower triangular matrix L. We add a continuous and

Department of Statistics, University of California, Los Angeles. Email: yeqiaoling@g.ucla.edu, aaamini@stat.ucla.edu, zhou@stat.ucla.edu.

concave penalty function to the likelihood to encourage sparsity in L, and thus achieve the goal of structure learning. Instead of a prespecified maximum indegree, which is *ad hoc* in nature, we provide a principled data-driven way to determine the tuning parameters for the penalty function. Finding the optimal DAG given P is then reduced to p decoupled penalized regression problems, which are solved by proximal gradient, an efficient first-order method, without enumerating possible parent sets for any node. Searching over P is done by simulated annealing (SA). We may also incorporate informative initial orderings, learned by an existing method, by setting a low starting temperature. Our numerical results demonstrate that this combined strategy substantially improves the accuracy of an estimated DAG. We note an interesting connection between our formulation and the sparse Cholesky factorization problem, and thus name our scoring function the *regularized Cholesky score* of orderings or permutations.

Regularizing likelihood with a continuous penalty function has been shown to be effective in learning Gaussian DAGs [22, 23, 24]. These methods optimize a regularized likelihood score over the DAG space by continuous optimization. They are likely to be trapped in a suboptimal structure due to the nonconvexity of the DAG parameter space. Using DAGs learned by such methods to generate initial orderings, our method can significantly improve the accuracy in structure learning. More recently, Champion et al. [15] developed a genetic algorithm optimizing over a triangular coefficient matrix and a permutation to learn Gaussian BNs. However, the authors did not provide a principled method to select the tuning parameter for the ℓ_1 penalty. Given a permutation, they optimize the network structure by an adaption of the least angle regression [25], which is closely related to the Lasso. In contrast, we use a more general and effective first-order method, the proximal gradient algorithm, which is applicable to many regularizers, including the ℓ_1 and concave penalties. As shown by our numerical experiments, our method substantially outperforms their genetic algorithm.

The paper is organized as follows. Section 2 covers some background on Gaussian BNs and the role of permutations in identifying the underlying DAGs. We introduce the regularized Cholesky loss and set up the optimization problem for BN learning in Section 3. Then we establish consistency results under our scoring function in the classical asymptotic setting in Section 4. In Section 5, we develop the annealing on regularized Cholesky score (ARCS) algorithm which combines simulated annealing to search over the permutation space and a proximal gradient algorithm to optimize the network structure given an ordering. We also propose a constraint-based approach to prune the estimated network structure after annealing process and a data-driven model selection technique to choose tuning parameters for the penalty function. Section 6 and 7 consist of exhaustive numerical experiments, where we compare our method to existing ones using both observational and experimental data in terms of structure learning accuracy and model fitting. Throughout the paper, experimental data refer to data generated under experimental interventions; see more detailed discussion in Section 3.3. We also study the empirical loss of our estimates and the effects of various components of our method. We conclude with a discussion in Section 8. All proofs and some additional numerical experiments are relegated to supplementary materials.

2 Background

We start with some background on Bayesian networks. A *Bayesian network* for a set of variables $\{X_1, \ldots, X_p\}$ consists of 1) a directed acyclic graph $\mathcal{G}$ that encodes a set of conditional independence assertions among the variables, and 2) a set of local probability distributions associated with each variable. It can be considered as a recipe for factorizing a joint distribution of $\{X_1, \ldots, X_p\}$ with probability density

$$p(x_1, \ldots, x_p) = \prod_{j=1}^{p} p(x_j \mid \Pi_j^{\mathcal{G}} = pa_j), \tag{1}$$

where $\Pi_j^{\mathcal{G}} \subset \{X_1, \ldots, X_p\} \setminus \{X_j\}$ is the parent set of variable X_j in $\mathcal{G}$ and pa_j its value. The DAG $\mathcal{G}$ is denoted by $\mathcal{G} = (V, E)$, where $V = \{1, \ldots, p\}$ is the vertex set corresponding to the set of random variables and $E = \{(i, j) : i \in \Pi_j^{\mathcal{G}}\} \subset V \times V$ is the edge set. We use variable X_j and node j exchangeably throughout the paper. DAGs contain no directed cycles, making the joint distribution in (1) well-defined.

2.1 Gaussian Bayesian networks

In this paper, we focus on Gaussian BNs that can be equivalently represented by a set of linear structural equation models (SEMs):

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

where $\varepsilon_j \sim \mathcal{N}(0, (\omega_j^0)^2)$ are mutually independent and independent of $\{X_i : i \in \Pi_j^{\mathcal{G}}\}$. Defining $B_0 := (\beta_{ij}^0) \in \mathbb{R}^{p \times p}$, $\varepsilon := (\varepsilon_1, \ldots, \varepsilon_p)^\top \in \mathbb{R}^p$, and $X := (X_1, \ldots, X_p)^\top \in \mathbb{R}^p$, we rewrite (2) as

$$X = B_0^\top X + \varepsilon. \tag{3}$$

The model has two parameters: 1) B_0 as a coefficient matrix, sometimes called the weighted adjacency matrix, where β_{ij}^0 specifies a weight associated with the edge $i \to j$ and $\beta_{ij}^0 = 0$ for $i \notin \Pi_j^{\mathcal{G}}$, and 2) $\Omega_0 := \mathrm{diag}((\omega_j^0)^2)$ as a noise variance matrix. The SEMs in (2) define a joint Gaussian distribution, $X \sim \mathcal{N}(0, \Sigma_0)$, where Σ_0 is positive definite and given by

$$\Sigma_0^{-1} = (I - B_0)\Omega_0^{-1}(I - B_0)^\top. \tag{4}$$

2.2 Acyclicity and permutations

The support of B_0 in (3) defines the structure of $\mathcal{G}$, and thus it must satisfy the acyclicity constraint so that $\mathcal{G}$ is indeed a DAG. To facilitate the development of our likelihood score for orderings, we express the acyclicity constraint on B_0 via permutation matrices. Let $\{e_1, \ldots, e_p\}$ be the canonical basis of $\mathbb{R}^p$. To each permutation π on the set $[p] := \{1, \ldots, p\}$, we associate a permutation matrix P_π whose i^{th} row is $e_{\pi(i)}^\top$. For a vector $v = (v_1, \ldots, v_p)^\top$, we have

$$P_\pi v = v_\pi = (v_{\pi(1)}, \ldots, v_{\pi(p)})^\top, \tag{5}$$

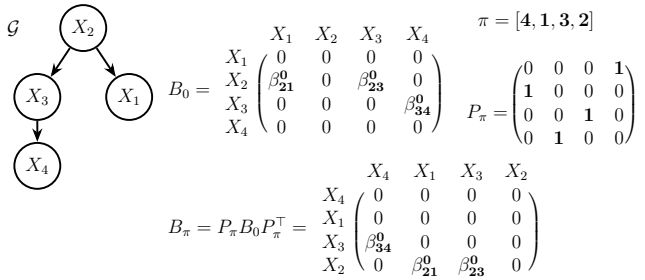


Fig. 1: An example DAG $\mathcal{G}$, its coefficient matrix B_0, and a permutation π. B_π permutes columns and rows of B_0 and is strictly lower triangular.

that is, P_π permutes the entries of v according to π. Since $P_\pi^\top P_\pi = I$, we can rewrite (3) as

$$P_\pi X = B_\pi^\top P_\pi X + P_\pi \varepsilon,$$

where $B_\pi := P_\pi B_0 P_\pi^\top$ is obtained by permuting the rows and columns of B_0 simultaneously according to π. Then, B_π will be a strictly lower triangular matrix if and only if π is the reversal of a topological sort of $\mathcal{G}$, i.e., $i \prec j$ in π for $j \in \Pi_i^{\mathcal{G}}$. See Figure 1 for an illustration. Under this reparametrization, the acyclicity constraint on B_0 translates to B_π being strictly lower triangular for some permutation π. Define $\Omega_\pi := P_\pi \Omega_0 P_\pi^\top$. Equivalently, one may think of node $\pi(i)$ as having been relabeled node i in B_π and Ω_π.

For simplicity, we drop the subscript π from P_π, B_π and Ω_π if no confusion arises. Therefore, throughout the paper, P defines a permutation π, B and Ω label nodes according to π and we write the permuted SEM as

$$PX = B^\top PX + P\varepsilon. \tag{6}$$

Denote by $\mathrm{cov}(X)$ the covariance matrix of X. Then we have $\Sigma := \mathrm{cov}(PX) = P\Sigma_0 P^\top$, obtained by permuting the rows and columns of Σ_0 (4) according to P.

3 Regularized likelihood score

In this section, we construct the objective function to estimate BN structure given data from the Gaussian SEM (2). We first focus on observational data in Sections 3.1 and 3.2: We re-parametrize the negative log-likelihood as a Cholesky loss and then impose sparse regularization to define our scoring function over permutations and DAG structures. In Section 3.3, we discuss how to modify the likelihood function for experimental data.

3.1 Cholesky loss

Let $\mathbf{X} := [\mathbf{X}_1, \ldots, \mathbf{X}_p] \in \mathbb{R}^{n \times p}$ be a data matrix where each row is an i.i.d. observation from (2). According to (6), we obtain a similar SEM on the data matrix:

$$\mathbf{X}P^\top = \mathbf{X}P^\top B + \mathbf{E}P^\top, \tag{7}$$

where each row of $\mathbf{E} \in \mathbb{R}^{n \times p}$ is an i.i.d. error vector from $\mathcal{N}(0, \Omega_0)$. In (7), $\mathbf{X}P^\top$ and $\mathbf{E}P^\top$ are $\mathbf{X}$ and $\mathbf{E}$ with columns permuted according to $P = P_\pi$. It then follows that each row of $\mathbf{X}P^\top$ is an i.i.d. observation from $\mathcal{N}(0, \Sigma)$ with $\Sigma^{-1} = (I - B)\Omega^{-1}(I - B)^\top$, and thus the negative log-likelihood of (7) is

$$\begin{aligned}&\ell(B, \Omega, P \mid \mathbf{X}) \\ &= \frac{1}{2} \operatorname{tr}\left[P\mathbf{X}^\top \mathbf{X} P^\top (I - B)\Omega^{-1}(I - B)^\top\right] + \frac{n}{2} \log|\Omega|. \end{aligned} \tag{8}$$

Recall that B and $\Omega = \mathrm{diag}((\omega_j)^2)$ are defined by permuting the rows and columns of B_0 and Ω_0 by the permutation matrix P. In particular, B is strictly lower triangular and we write its columns as $\beta_j \in \mathbb{R}^p$.

Denote by $L := (I - B)\Omega^{-\frac{1}{2}}$ a weighted coefficient matrix, where each column $L_j = (e_j - \beta_j)/\omega_j$ is a weighted coefficient vector for node $\pi(j)$. We define what we call the *Choleskly loss* function

$$\mathscr{L}_{\mathrm{chol}}(L; A) := \frac{1}{2} \operatorname{tr}\left(ALL^\top\right) - \log|L|, \tag{9}$$

where $|L|$ denotes the determinant of L. Noting that $|L| = |(I - B)\Omega^{-\frac{1}{2}}| = |\Omega|^{-\frac{1}{2}}$ and denoting by $\widehat{\Sigma} := \frac{1}{n}\mathbf{X}^\top \mathbf{X}$ the sample covariance matrix, one can re-parametrize the negative log-likelihood (8) with L and P and connect it to the Cholesky loss:

Lemma 1. *The negative log-likelihood (8) for observational data can be re-parametrized as*

$$\begin{aligned}\ell(L, P) &= n \cdot \mathscr{L}_{chol}(L; P\widehat{\Sigma}P^\top) \\ &= \frac{n}{2} \operatorname{tr}\left(P\widehat{\Sigma}P^\top LL^\top\right) - n \log|L|, \end{aligned} \tag{10}$$

where $L = (I - B)\Omega^{-\frac{1}{2}}$ is a lower triangular matrix and P is a permutation matrix.

The reason for naming (9) the Cholesky loss is that it provides an interesting variational characterization of the Cholesky factor of the inverse of a matrix as the following proposition shows. Let $\mathcal{L}_p$ be the set of $p \times p$ lower triangular matrices with positive diagonal entries, and for any positive definite matrix M, let $\mathcal{C}(M)$ be its unique Cholesky factor, i.e., the unique lower triangular matrix L with positive diagonal entries such that $M = LL^\top$. Let $\lambda_{\min}(A)$ and $\|A\|_F$ denote, respectively, the minimum eigenvalue and the Frobenius norm of a matrix A.

Proposition 1 (Curvature). *For any positive definite matrix $A \in \mathbb{R}^{p \times p}$ and lower triangular matrix $L \in \mathcal{L}_p$,*

$$\mathscr{L}_{chol}(L; A) - \mathscr{L}_{chol}(L^*; A) \geq \frac{1}{2}\lambda_{\min}(A)\|L - L^*\|_F^2,$$

where $L^* = \mathcal{C}(A^{-1})$. Consequently, L^* is the unique minimizer of $\mathscr{L}_{chol}(\,\cdot\,; A)$ with optimal value

$$\mathscr{L}_{chol}^*(A) := \mathscr{L}_{chol}(\mathcal{C}(A^{-1}); A) = \frac{1}{2}\left(p + \log|A|\right).$$

In particular, $\mathscr{L}_{chol}^*(A) = \mathscr{L}_{chol}^*(PAP^\top)$ for any permutation matrix P.

Proposition 1 states that $\mathcal{C}(A^{-1})$ is the unique minimizer of $\mathscr{L}_{\mathrm{chol}}(\,\cdot\,; A)$ and bounds the curvature of the Cholesky loss near its minimum. The curvature bound will be used in the proof of consistency (cf. Theorem 1 in Section 4).

Now consider finding the maximum likelihood DAG for a fixed permutation P, which corresponds to minimizing $L \mapsto \ell(L, P)$ given by (10). Let $\ell^*(P)$ be the optimal value

of this problem, i.e.,

$$\ell^*(P) := \min_{L \in \mathcal{L}_p} \ell(L, P).$$

Then, Proposition 1 implies

$$\ell^*(P) = n \cdot \mathscr{L}^*_{\text{chol}}(P\widehat{\Sigma}P^T) = n \cdot \mathscr{L}^*_{\text{chol}}(\widehat{\Sigma}), \tag{11}$$

showing that ℓ^* is invariant to permutations, hence maximum likelihood estimation does not favor any particular ordering. In other words, all the maximum likelihood DAGs corresponding to different permutations give the same Gaussian likelihood. Moreover, they will always be complete DAGs, which has negative implications for both computational and interpretability concerns. These motivate our development of sparse regularization for this problem.

3.2 Sparse regularization

To break the permutation equivalence of the maximum likelihood (11), we add a regularizer to the Cholesky loss to favor sparse DAGs. Under faithfulness [26], the true DAG $\mathcal{G}$ in (2) and its equivalence class are the sparsest among all DAGs that can parameterize the joint distribution $\mathcal{N}(0, \Sigma_0)$. To start, let us point out some connections to the well-known "sparse Cholesky factorization" problem from linear algebra.

According to Proposition 1, the minimizer of $\ell(L, P)$ over L is the Cholesky factor of $(P\widehat{\Sigma}P^\top)^{-1} = P\widehat{\Sigma}^{-1}P^\top$. For a sparse $\widehat{\Sigma}^{-1}$, it is well-known that the choice of P greatly affects the sparsity of the resulting Cholesky factor. Heuristic approaches have been developed in numerical linear algebra to find a permutation that leads to a sparse factorization by trying to minimize the so-called "fill-in". An example is the maximum cardinality algorithm [27].

From a statistical perspective, however, $\widehat{\Sigma}^{-1}$ is, in general, not sparse (due to noise) even if the inverse of population covariance matrix $\Sigma = \mathbb{E}[\widehat{\Sigma}]$ is so. In such cases, one can first estimate a sparse precision matrix and then use the sparse estimate as the input to the sparse Cholesky factorization problem. We take a more direct alternative approach by adding a sparsity-measuring penalty to the Cholesky loss.

Let $\rho_\theta : \mathbb{R} \mapsto [0, \infty)$ be a nonnegative and nondecreasing regularizer with some tuning parameter(s) θ. We consider the following penalized loss function:

$$f_\theta(L; P) := \; n \cdot \mathscr{L}_{\text{chol}}(L; P\widehat{\Sigma}P^\top) + \sum_{i>j} \rho_\theta(L_{ij}), \tag{12}$$

where the penalty is only applied to the off-diagonal entries of a lower triangular matrix L. The loss depends on the regularization parameter θ, and for simplicity we write $f_\theta(L; P)$ as $f(L; P)$. In this paper, we focus on the class of regularizers called the minimax concave penalty (MCP) [28] which includes ℓ_1 and ℓ_0 as extreme cases; see (15) below. MCP is a sparsity-favoring penalty and adding it breaks the symmetry of the Cholesky loss w.r.t. permutations as in (11). As a result, the permutations leading to sparser lower-triangular factors L will have smaller loss values $f(L; P)$.

Let $\mathcal{P}_p$ be the set of $p \times p$ permutation matrices. Given $P \in \mathcal{P}_p$, the minimizer of $f(L; P)$ over L is a sparse DAG $\mathcal{G}(P)$ with a score $f(P)$ defined as

$$f(P) := \min_{L \in \mathcal{L}_p} f(L; P). \tag{13}$$

We minimize permutation score $f(P)$ over $\mathcal{P}_p$ to obtain an estimated ordering. The overall sparse BN learning problem is then

$$\min_{P \in \mathcal{P}_p} f(P) = \min_{P \in \mathcal{P}_p} \min_{L \in \mathcal{L}_p} \Big\{ \frac{n}{2} \operatorname{tr}\left(P\widehat{\Sigma}P^\top LL^\top\right) - n \log |L| + \sum_{i>j} \rho_\theta(L_{ij}) \Big\}. \tag{14}$$

In this formulation, the objective function only depends on the $p \times p$ sample covariance matrix $\widehat{\Sigma} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$. Thus, the computational complexity is determined by the dimension p once $\widehat{\Sigma}$ has been computed. In Section 5, we discuss our approach to solve this problem by optimizing over (L, P). It is worth noting that problem (14) can be considered both as 1) a penalized maximum likelihood BN estimator in the Gaussian case, and 2) a variational formulation of the sparse Cholesky factorization problem when the input matrix $\widehat{\Sigma}$ is noisy (hence its inverse usually not sparse). According to the second interpretation, we call $f(L; P)$ in (12) the *regularized Cholesky (RC) loss* function and $f(P)$ in (13) the *RC score* of a permutation P.

Throughout, let $\rho(\cdot) := \rho_\theta(\cdot)$ be the MCP with two parameters $\theta = (\gamma, \lambda)$ [28]:

$$\rho(x; \gamma, \lambda) = \begin{cases} \lambda|x| - \frac{x^2}{2\gamma}, & |x| < \gamma\lambda, \\ \frac{1}{2}\gamma\lambda^2, & |x| \geq \gamma\lambda, \end{cases} \tag{15}$$

where $\lambda \geq 0$ and $\gamma > 1$. Parameter λ measures the penalty level, while γ controls the concavity of the function. For a fixed value of λ, the MCP approaches the ℓ_1 penalty as $\gamma \to \infty$, and the ℓ_0 penalty as $\gamma \to 0^+$.

Figure 2 compares the MCP with $(\gamma, \lambda) = (2, 1)$ and the ℓ_1 penalty. The right derivative of MCP at zero is λ, which is the same as the derivative of the ℓ_1 penalty. The MCP function flats out when $|x| \geq \gamma\lambda$.

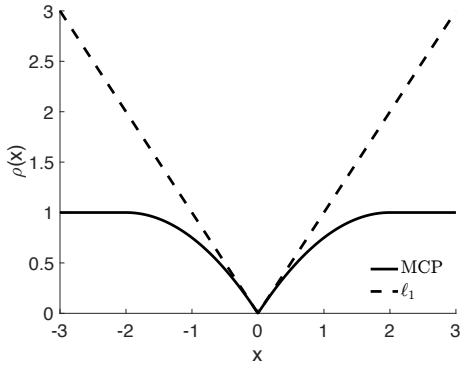


Fig. 2: A comparison between the MCP (solid line) and the ℓ_1 penalty (dashed line).

Remark 1. Aragam and Zhou [23] use an MCP regularized likelihood to estimate Gaussian DAGs as well. However, rather than searching over permutations, which automatically satisfies the acyclicity constraint, they perform a greedy coordinate descent to minimize the regularized loss

over DAGs. Thus, at each update in their algorithm the acyclicity constraint must be carefully checked.

Remark 2. For a given permutation P, the minimizer of the RC loss $f(L;P)$ (13) estimates a sparse Cholesky factor of $P\Sigma_0^{-1}P^\top$, the precision matrix Σ_0^{-1} with rows and columns permuted. This is related to recent methods on covariance matrix estimation by Cholesky decomposition [29, 30, 31, 32, 33], which make two main assumptions: (i) a fixed variable ordering is provided, i.e. P is given, and (ii) the precision matrix has certain sparse structures, say it is banded. See [34] for a recent review on covariance matrix estimation. The key differences between these methods and our BN learning approach are: (i) We impose sparsity on the Cholesky factor L, which encodes a DAG structure with P, while those precision matrix estimation methods either impose sparsity on Σ_0^{-1}, such as in [29, 30], or assume L is banded [32]. (ii) More importantly, instead of assuming a known ordering, we minimize the RC score (14) jointly over (L,P) to estimate an ordering and a sparse DAG structure. Once an estimated $\widehat{P}$ and the associated $\widehat{L}$ have been found, we can estimate the precision matrix Σ_0^{-1} by $\widehat{P}^\top\widehat{L}\widehat{L}^\top\widehat{P}$. Under the Gaussian SEM (2), our estimate of Σ_0^{-1} will be more accurate than the above precision matrix estimation methods when the true ordering is unknown. See Section 6.5 for relevant numerical comparisons.

3.3 Likelihood for experimental data

It is well-known that DAGs in the same Markov equivalence class are observationally equivalent, and thus we cannot distinguish such DAGs from observational data alone. However, experimental interventions can help distinguish equivalent DAGs and construct causal networks. Following [35], intervention on a node X_j in a DAG is to impose a fixed external distribution on this node, denoted by $p(x_j \mid \bullet)$, independent of all X_{-j}, while keeping the structural equations (2) of the other nodes unchanged.

Suppose that our data $\mathbf{X} \in \mathbb{R}^{n\times p}$ consists of M blocks, where each block $\mathbf{X}^m \in \mathbb{R}^{n_m\times p}$ and $n = \sum_{m=1}^M n_m$. Denote by $X_{\mathcal{I}}^m \subset \{X_1,\dots,X_p\}$ the set of variables under experimental interventions in block m. Then, the data for $X_j \in X_{\mathcal{I}}^m$ in this block are generated independently from the distribution $p(x_j \mid \bullet)$, while for $X_i \notin X_{\mathcal{I}}^m$ from the conditional distribution $[X_i \mid \Pi_i^{\mathcal{G}}]$. Note that multiple nodes could be intervened for a block of data, in which case $|X_{\mathcal{I}}^m| \geq 2$.

Let $\mathcal{I}_j \subset \{1,2,\dots,n\}$ be the set of observations for which X_j is under experimental intervention, and let $\mathcal{O}_j = \{1,2,\dots,n\}\setminus\mathcal{I}_j$ be its complement. By the truncated factorization formula [26, 35, 36], the joint density of experimental data is

$$p(\mathbf{X}) = \prod_{j=1}^p \prod_{h\in\mathcal{O}_j} p(x_{hj} \mid pa_{hj}) \prod_{k\in\mathcal{I}_j} p(x_{kj} \mid \bullet), \qquad (16)$$

where x_{hj} is the value of the j^{th} variable in the h^{th} observation and pa_{hj} is the value for its parents. Let $\mathbf{X}_{\mathcal{O}_{\pi(j)}}$ be the submatrix of $\mathbf{X}$ with rows in $\mathcal{O}_{\pi(j)}$ and

$$\widehat{\Sigma}^j := \frac{1}{|\mathcal{O}_{\pi(j)}|}\mathbf{X}_{\mathcal{O}_{\pi(j)}}^\top \mathbf{X}_{\mathcal{O}_{\pi(j)}}$$

be the sample covariance matrix computed from data in these rows. Then the log-likelihood of experimental data can be re-parametrized into the Cholesky loss functions as well:

Lemma 2. *The negative log-likelihood for experimental data* (16) *can be written as*

$$\ell_{\mathcal{O}}(L,P) = \sum_{j=1}^p |\mathcal{O}_{\pi(j)}| \mathscr{L}_{chol}\left(L_j; P\widehat{\Sigma}^j P^\top\right), \qquad (17)$$

where $L_j = (e_j - \beta_j)/\omega_j \in \mathbb{R}^p$, $L = (L_j) \in \mathcal{L}_p$, *and* $|L_j| := L_{jj}$ *in* $\mathscr{L}_{chol}(\cdot)$ (9).

Though experimental data likelihood $\ell_{\mathcal{O}}(L,P)$ in (17) is not identical to the observational $\ell(L,P)$ in (10), searching strategies described in Section 5 can be applied to both observational and experimental data.

4 Consistency

We now show that the estimator that minimizes the regularized Cholesky score (14) is consistent as the sample size $n\to\infty$, while the dimension of the problem p remains fixed (the classical asymptotic setting). We establish the result for a general class of regularizers that contains MCP as a special case. We show two levels of consistency: (1) permutation consistency, in the sense of recovering a topological sort that is consistent with a DAG in the true Markov equivalence class, and (2) structure consistency or support recovery, showing that the estimated DAG has the same support as a DAG in the true equivalence class.

Let us give the high level ideas behind our proof of the consistency results. First, in the large-sample limit, the empirical loss $f(P_\pi)/n$ (13) converges to $F_{n,\pi}$ uniformly in π, where $F_{n,\pi}$ is its population version, obtained by replacing $\widehat{\Sigma}$ with Σ_0 in definition (12). Second, under certain identifiability assumptions, the population loss satisfies $F_{n,\pi^*} < F_{n,\pi}$, for any permutation π^* associated with the true Markov equivalence class and any π that is not. Together these two results allow us to guarantee both the permutation and structure consistency of the minimizers of the regularized Cholesky loss. We give a more detailed sketch in Section 4.3, after stating the main result.

To make the notion of permutation consistency more precise, we review some background on Markov equivalence classes and introduce the notion of *score-identifiability*.

4.1 Score-identifiability

Let P^* be the true data-generating distribution. We write $\mathcal{I}(P^*)$ for the set of conditional independence (CI) statements that hold in P^*. Similarly, we write $\mathcal{I}(B)$ for the set of CIs, implied by DAG B, in the sense of d-separation. Then, B is an I-map for P^* if $\mathcal{I}(B)\subset\mathcal{I}(P^*)$ and is *perfect* for (or *faithful* to) P^* if $\mathcal{I}(B) = \mathcal{I}(P^*)$. Two DAGs B and B' are (Markov) equivalent if $\mathcal{I}(B) = \mathcal{I}(B')$. We write $\mathcal{E}(B)$ for the equivalence class of B.

By a well-known result, all the DAGs in a Markov equivalence class have the same skeleton (hence the same number of edges) and the same v-structures. We write $\mathcal{E}$ for a generic equivalence class and $\|\mathcal{E}\|_0$ for the number of edges of any DAG in the equivalence class. Consider the following definition:

Definition 1. *We say that a Markov equivalence class $\mathcal{E}$ is score-identifiable (for P^*) if for any DAG B which is an I-map for P^*, either $B \in \mathcal{E}$ or $\|B\|_0 > \|\mathcal{E}\|_0$.*

The case that this definition is ruling out is when there exists a DAG outside equivalence class $\mathcal{E}$ that has the same number of edges but a different set of v-structures. By definition, if a score-identifiable class exists, it is "the unique" sparsest equivalence class that is an I-map for P^*. We have the following which is essentially the same as [6, Proposition 8]:

Lemma 3. *If $\mathcal{E}$ is perfect for P^*, then $\mathcal{E}$ is score-identifiable.*

Therefore, score-identifiability is implied by, and thus no stronger than, faithfulness.

4.2 Consistent structure learning

With some abuse of notation, throughout this section, let $B = P_\pi^\top (I - A)\Omega^{-1/2} P_\pi$ denote a normalized DAG, where A is a strictly lower triangular matrix and Ω is a diagonal matrix with positive diagonal elements. Note that $L := (I - A)\Omega^{-1/2} \in \mathcal{L}_p$ is a lower triangular matrix with positive diagonals, and π is a reversed topological sort (RTS) of the DAG defined by the support of B (cf. Figure 1). Let $\mathfrak{D}_\pi$ be the set of DAGs whose RTS is consistent with permutation π, that is,

$$\mathfrak{D}_\pi = \{P_\pi^\top L P_\pi : L \in \mathcal{L}_p\}$$

and let K_π be the (unique) complete DAG reversely sorted by π. For every $B \in \mathfrak{D}_\pi$, we have $\mathrm{supp}(B) \subset K_\pi$, where $\mathrm{supp}(B)$ denotes the "off-diagonal" support of B, i.e., the set of indices of nonzero off-diagonal elements of B. Unless otherwise stated, the support of B refers to this off-diagonal support. We let B_π be the unique minimizer of the (unregularized) population Cholesky loss over $\mathfrak{D}_\pi$ in this section:

$$B_\pi = \operatorname*{arg\,min}_{B \in \mathfrak{D}_\pi} \mathscr{L}_{\mathrm{chol}}(B; \Sigma_0) \tag{18}$$

and let $S_\pi = \mathrm{supp}(B_\pi)$ be its (off-diagonal) support. It is not hard to see that the support of B_π is the (minimal) I-map corresponding to permutation π. We simply refer to B_π as the I-map associated with π.

Let $\widehat{\Sigma}_n$ be the sample covariance matrix. Consider the following permutation score

$$\widehat{F}_{n,\pi} := \min_{L \in \mathcal{L}_p} \Big[\mathscr{L}_{\mathrm{chol}}(L; P_\pi \widehat{\Sigma}_n P_\pi^\top) + \frac{\rho_n(L)}{n}\Big], \tag{19}$$

for some permutation-invariant penalty $\rho_n = \rho_{\theta_n}$. Note that $\widehat{F}_{n,\pi} = f(P_\pi)/n$, where $f(\cdot)$ is the regularized Cholesky score in (13). We assume that ρ_n is defined over all $n \times n$ matrices (not just lower triangular ones). Then, we can alternatively write

$$\widehat{F}_{n,\pi} = \min_{B \in \mathfrak{D}_\pi} \Big[\mathscr{L}_{\mathrm{chol}}(B; \widehat{\Sigma}_n) + \frac{\rho_n(B)}{n} := \mathcal{L}_n(B; \widehat{\Sigma}_n)\Big]. \tag{20}$$

Let $\widehat{B}_{n,\pi}$ be a (global) minimizer of $B \mapsto \mathcal{L}_n(B; \widehat{\Sigma}_n)$ over $\mathfrak{D}_\pi$ so that $\widehat{F}_{n,\pi} = \mathcal{L}_n(\widehat{B}_{n,\pi}; \widehat{\Sigma}_n)$. We estimate the permutation by minimizing $\pi \mapsto \widehat{F}_{n,\pi}$, with a minimizer denoted as $\widehat{\pi}_n$. Then, our estimated weighted adjacency matrix will be $\widehat{B}_{n,\widehat{\pi}_n}$, and its support defines the structure of an estimated DAG $\widehat{G}_n$.

We need some regularity conditions on the regularizer $\rho_n(\cdot)$ and the collection $\{B_\pi\}$ of I-maps of Σ_0. Let $\lambda_{\min} = \lambda_{\min}(\Sigma_0)$ be the minimum eigenvalue of Σ_0, and assume that $\rho_n(B) = \sum_{i \neq j} r_n(|B_{ij}|)$ for some $r_n : \mathbb{R}_+ \to \mathbb{R}_+$ that satisfies the following:

(R1) We say that r_n is (a_n, b_n)-flat if it is bounded by a_n and

$$r_n(t) = a_n \text{ for } t \geq b_n.$$

for some $b_n = O(1)$.

(R2) Assume that r_n is twice differentiable on $[0, b_n)$, with the derivatives at 0 interpreted as one-sided, and assume that r_n' has a left limit at b_n. Moreover $r_n''(\cdot)/n$ is C_0-Lipschitz on $[0, b_n)$, for some constant $C_0 \geq 0$, and

$$|r_n''(0+)| \leq C_1 n, \quad |r_n''(b_n-)| = O(n),$$

with $C_1 \leq \lambda_{\min}/2$. Let $\lambda_n := r_n'(0+)$ be the right derivative of r_n at 0.

Consider the so-called $\beta_{\min}$ condition

$$2b_n \leq \min_\pi \tau(B_\pi), \tag{21}$$

where $\tau(B) = \min\{|B_{ij}| : (i,j) \in \mathrm{supp}(B)\}$. Assume further that

$$\liminf \frac{n b_n^2}{a_n} > \frac{2}{\lambda_{\min}} \max_\pi |S_\pi|. \tag{22}$$

Let us write $\Pi(\mathcal{E})$ for the collection of RTSs for some DAGs in equivalence class $\mathcal{E}$. If $\mathcal{E}^*$ is the true equivalence class, then $\Pi(\mathcal{E}^*)$ is the collection of true RTSs. We have the following consistency result:

Theorem 1. *Assume that P^*, or equivalently Σ_0 in (4), has a (unique) score-identifiable Markov equivalence class $\mathcal{E}^*$, and let*

$$\widehat{\pi}_n \in \operatorname*{arg\,min}_\pi \widehat{F}_{n,\pi}.$$

(a) Assume that $r_n(\cdot)$ is (a_n, b_n)-flat for sequences that satisfy (21), (22), $n^{-1/2} a_n \to \infty$ and $n^{1/4} b_n \to \infty$, and (R2) holds with $n^{-1}\lambda_n = O(1)$. Then, $\mathbb{P}(\widehat{\pi}_n \in \Pi(\mathcal{E}^*)) \to 1$ as $n \to \infty$. Moreover, $\widehat{B}_{n,\widehat{\pi}_n}$ is a $\sqrt{n}$-consistent estimate of $B_{\widehat{\pi}_n}$.

(b) If in addition $n^{-1}\lambda_n \to 0$ and $n^{-1/2}\lambda_n \to \infty$, then we also have $\mathbb{P}(\widehat{G}_n \in \mathcal{E}^*) \to 1$ as $n \to \infty$, where $\widehat{G}_n$ is the DAG defined by $\mathrm{supp}(\widehat{B}_{n,\widehat{\pi}_n})$.

Part (a) of Theorem 1 establishes permutation consistency by guaranteeing that $\widehat{\pi}_n$ eventually does not leave $\Pi(\mathcal{E}^*)$, although it can move around in this set indefinitely. Part (b) establishes the structure consistency.

In the case of the MCP, we can take $r_n(t)/n = \rho(t; \gamma_n, \xi_n)$ giving $\lambda_n = r_n'(0+) = n\xi_n$. Then, (R1) holds with $a_n = \frac{1}{2} n \gamma_n \xi_n^2$ and $b_n = \gamma_n \xi_n$, and we have $r_n''(t)/n = 1/\gamma_n$ for $t \in [0, b_n)$. Assuming $1/\gamma_n \leq \lambda_{\min}/2$ after proper re-scaling of the data or the regularizer, (R2) holds with $C_0 = 0$. Since the right-hand sides of (21) and (22) are constants, these conditions hold if γ_n is sufficiently large and $\xi_n \to 0$. Conditions for part (a) of the theorem hold if $\xi_n = O(1)$, $\sqrt{n}\gamma_n \xi_n^2 \to \infty$ and $n^{1/4}\gamma_n \xi_n \to \infty$, so it suffices to have $n^{-1/4} \ll \xi_n \lesssim 1$ and $\gamma_n \gtrsim 1$. For part (b), we need $\xi_n = o(1)$

and $\sqrt{n}\xi_n \to \infty$, that is, $n^{-1/2} \ll \xi_n \ll 1$. All the conditions are satisfied if $\gamma_n \gtrsim 1$ and $n^{-1/4} \ll \xi_n \ll 1$. In particular, if we let $\gamma_n \to \infty$ and $\gamma_n \xi_n \to 0$ at a rate slower than $n^{-1/4}$, then our consistency results apply to the capped ℓ_1 penalty, i.e. $r_n(t)/n = \xi_n t \wedge \gamma_n \xi_n^2$.

Remark 3. Aragam and Zhou [23] also provide asymptotic results for the estimator we consider here. Under appropriate conditions, they show that (i) in small neighborhoods (of radius $\sim n^{-1/2}$) of every B_π (18), there are "good" local minimizers of (20) (i.e., with correct support) and (ii) if one B_π has more edges than the other, the corresponding nearby local minimizer gives a higher value of the objective function. Their results, however, provide no guarantee for all local minimizers of the problem. In particular, their results are silent about the global minimizer(s) of (20). In contrast, we provide conditions, under which, any "global minimizer" of (20) is both permutation and structure consistent. Proving such global results requires significantly more technical effort. For example, even showing that a global minimizer is within a neighborhood of radius $\sim n^{-1/2}$ of some B_π is nontrivial, as the proof demonstrates.

4.3 Proof sketch

We give a brief sketch of the proof of Theorem 1 here. A detailed proof can be found in the supplement. For two symmetric matrices A and B of the same dimension, we write $A \succeq B$ if $A - B$ is positive semidefinite.

First, we show that under (21), for any π and $B \in \mathfrak{D}_\pi$,

$$\mathcal{L}_n(B;\Sigma_0) - \mathcal{L}_n(B_\pi;\Sigma_0) \geq \frac{\lambda_{\min}}{2}\big(\|B - B_\pi\|_F^2 \wedge c_{n,\pi}\big), \tag{23}$$

where $c_{n,\pi} := b_n^2 - 2a_n|S_\pi|/(\lambda_{\min} n)$. Combined with (22), this implies that the population version of (20), namely,

$$F_{n,\pi} := \min_{B \in \mathfrak{D}_\pi} \mathcal{L}_n(B;\Sigma_0), \tag{24}$$

has B_π as its unique (isolated) minimizer. A so-called "basic inequality argument" further implies that $\mathcal{L}_n(\widehat{B}_{n,\pi};\Sigma_0) - \mathcal{L}_n(B_\pi;\Sigma_0) = O_p(n^{-1/2})$. Together with (23), we get $\|\widehat{B}_{n,\pi} - B_\pi\|_F = O_p(n^{-1/4})$, that is, $\widehat{B}_{n,\pi}$ is $n^{1/4}$-consistent for B_π.

Using $n^{1/4} b_n \to \infty$, we conclude that $\|\widehat{B}_{n,\pi} - B_\pi\|_F < b_n$, eventually. Recalling assumption (21), this implies that for any DAG B which is between $\widehat{B}_{n,\pi}$ and B_π elementwise, the absolute coordinates $|B_{ij}|$ are either in $[0, b_n)$ or (b_n, ∞). Since $r_n(\cdot)$ is smooth over each of these intervals, we can apply a Taylor expansion of $\mathcal{L}_n(\,\cdot\,, \widehat{\Sigma}_n)$ around B_π. Taking into account the one-sided differentiability of the regularized loss at zero, we obtain a quadratic inequality for $\Delta_{n,\pi} = \widehat{B}_{n,\pi} - B_\pi$, where the quadratic term is controlled by the Hessian $\nabla^2 \mathcal{L}_n(\widetilde{B}_{n,\pi};\widehat{\Sigma}_n)$ for some $\widetilde{B}_{n,\pi}$ that is between $\widehat{B}_{n,\pi}$ and B_π elementwise. A further argument shows that $\nabla^2 \mathcal{L}_n(\widetilde{B}_{n,\pi};\widehat{\Sigma}_n) \succeq \frac{1}{2}\lambda_{\min} I_{p^2}$ when n is large, which together with the quadratic equality implies

$$\|\Delta_{n,\pi}\|_F \leq \frac{2}{\lambda_{\min}} \|\nabla \mathscr{L}_{\text{chol}}(B_\pi;\widehat{\Sigma}_n) - \nabla \mathscr{L}_{\text{chol}}(B_\pi;\Sigma_0)\|_F.$$

Algorithm 1 Annealing on regularized Cholesky score (ARCS).

Input: Dataset $\mathbf{X}$, initial permutation matrix P_0, a temperature schedule $\{T^{(i)}, i = 0, \dots, N\}$, constant m.
Output: Adjacency matrix $\widehat{B}$.

1: Select tuning parameters (γ, λ) for $f(L;P)$ according to Algorithm 4 (Section 5.4).
2: $\widehat{P} \leftarrow P_0$, $\widehat{L} \leftarrow \arg\min_{L \in \mathcal{L}_p} f(L;\widehat{P})$ by Algorithm 2, $f(\widehat{P}) \leftarrow f(\widehat{L};\widehat{P})$.
3: **for** $i = 0, \dots, N$ **do**
4: $\quad T \leftarrow T^{(i)}$.
5: $\quad$ Propose P^* by flipping a random length-m interval in the permutation defined by $\widehat{P}$.
6: $\quad L^* \leftarrow \arg\min_{L \in \mathcal{L}_p} f(L;P^*)$ using Algorithm 2, $f(P^*) \leftarrow f(L^*;P^*)$.
7: $\quad \alpha \leftarrow \min\Big\{1, \exp\big(-\frac{1}{T}[f(P^*) - f(\widehat{P})]\big)\Big\}$.
8: $\quad$ Set $(\widehat{P}, \widehat{L}, f(\widehat{P})) \leftarrow (P^*, L^*, f(P^*))$ with prob. α.
9: **end for**
10: Refine adjacency matrix $\widehat{B}$ given $(\widehat{L}, \widehat{P})$ by Algorithm 3 (Section 5.3).

The right-hand side can be shown to be $O_p(n^{-1/2})$ from which we conclude the $\sqrt{n}$-consistency of $\widehat{B}_{n,\pi}$ for B_π. All the consistency arguments hold uniformly over π.

Equipped with $\sqrt{n}$-consistency, we then show that $\widehat{F}_{n,\pi} - F_{n,\pi} = O_p(n^{-1/2})$, uniformly in π. For any $\pi^* \in \Pi(\mathcal{E}^*)$ and any $\pi \notin \Pi(\mathcal{E}^*)$,

$$\begin{aligned}\widehat{F}_{n,\pi} - \widehat{F}_{n,\pi^*} &\geq F_{n,\pi} - F_{n,\pi^*} - O_p(n^{-1/2}) \\ &= \frac{a_n}{n}(|S_\pi| - |S_\pi^*|) - O_p(n^{-1/2}).\end{aligned}$$

See the supplement for the details of getting to the second line. Since by score-identifibility $|S_\pi| \geq |S_{\pi^*}| + 1$, and by assumption $n^{-1/2} a_n \to \infty$, with high probability (w.h.p.) $\widehat{F}_{n,\pi} - \widehat{F}_{n,\pi^*} > 0$ for all $\pi \notin \Pi(\mathcal{E}^*)$, when n is sufficiently large. This proves permutation consistency.

For the structure consistency, since $\widehat{B}_{n,\pi}$ is within a neighborhood of radius $O_p(n^{-1/2})$ around B_π, an argument similar to that of Lemma 1 in [37] shows that $\operatorname{supp}(\widehat{B}_{n,\pi}) = \operatorname{supp}(B_\pi)$ for all π w.h.p. In particular, $\operatorname{supp}(\widehat{B}_{n,\hat{\pi}_n}) = \operatorname{supp}(B_{\hat{\pi}_n})$, w.h.p. and the proof is complete.

5 Optimization

We now describe how we solve the optimization problem (14). The main steps are outlined in Algorithm 1, where we use simulated annealing to search over the permutation space to minimize the RC score defined in (13). To obtain the RC score for a given permutation, we need to solve a continuous optimization problem (line 2 and 6) for which we propose a proximal gradient algorithm (Algorithm 2).

5.1 Searching over permutations

The ARCS algorithm is detailed in Algorithm 1. At each iteration, we propose a permutation P^* and decide whether to stay at the current permutation or move to the proposed one

with probability α given in line 7. The probability is determined by the difference between the proposed and current scores $f(P^*)-f(\widehat{P})$ normalized by a temperature parameter T. For $T \to \infty$, the jumps are completely random and for $T \to 0^+$ completely determined by the RC score $f(\cdot)$. The algorithm follows a temperature schedule which is often taken to be a decreasing sequence $T^{(0)} \geq T^{(1)} \geq \ldots \geq T^{(N)}$ allowing the algorithm to explore more early on and zoom in on a solution as time progresses.

The proposed permutation matrix P^* is constructed as follows. Let $\widehat{\pi}$ and π^* be the permutations associated with $\widehat{P}$ and P^* as in (5). We propose π^* by flipping (i.e., reversing the order of) a random interval of length m in the current permutation $\widehat{\pi}$. For example, with $m = 3$ we may flip $\widehat{\pi} = (1, 2, 3, 4, \ldots, p)$ to $\pi^* = (1, 4, 3, 2, \ldots, p)$ in the proposal. Equivalently, we flip a contiguous block of m rows of $\widehat{P}$ to generate P^*.

As a byproduct of evaluating the RC score for the proposed permutation P^*, we also obtain the corresponding lower triangular matrix L^*, representing the associated DAG. We keep track of these DAGs as well as the permutations throughout the algorithm (line 6).

Remark 4. There is no theoretical guarantee for ARCS to find a global minimizer using simulated annealing. In this sense, ARCS performs a heuristic search over the permutation space to learn a topological sort. For smaller DAGs, however, our numerical results in Section 6.8 show that ARCS often finds solutions that are close to the global minimizers.

5.2 Computing RC score

We propose a proximal gradient algorithm to evaluate the RC score $f(P)$ at each permutation matrix P (line 2 and 6, Algorithm 1). This algorithm belongs to a class of first-order methods that are quite effective at optimizing functions composed of a smooth loss and a nonsmooth penalty [38].

The RC score is obtained by minimizing the RC loss $f(L; P)$ over L as shown in (13). Recall that $\mathcal{L}_p$ is the set of $p \times p$ lower triangular matrices, and let

$$\rho(u) := \sum_{i>j} \rho(u_{ij}), \quad \text{for} \quad u = (u_{ij}) \in \mathcal{L}_p. \tag{25}$$

Note that we are leaving out the diagonal elements of u in defining $\rho(u)$. Then, the RC loss is $f(u; P) = \ell(u, P) + \rho(u)$, where $\ell(u, P)$ (10) is differentiable and $\rho(u)$ is nonsmooth. The idea of the proximal gradient algorithm is to replace $\ell(u, P)$ with a local quadratic function at the current estimate L and optimize the resulting approximation to $f(u; P)$ to get a new estimate L^+:

$$\begin{aligned} L^+ &= \operatorname*{arg\,min}_{u \in \mathcal{L}_p} \ell(L) + \nabla\ell(L)^\top (u - L) + \frac{1}{2t}\|u - L\|^2 + \rho(u) \\ &= \operatorname*{arg\,min}_{u \in \mathcal{L}_p} \frac{1}{2t}\|L - t\nabla\ell(L) - u\|^2 + \rho(u), \end{aligned} \tag{26}$$

where $\ell(L) = \ell(L, P), \nabla\ell(L) := \nabla_L \ell(L, P)$ is the gradient of $\ell(L, P)$ w.r.t. L, and $t > 0$ is a step size. Consider the proximal operator $\mathbf{prox}_\rho : \mathcal{L}_p \to \mathcal{L}_p$ associated with ρ defined by

$$\mathbf{prox}_\rho(x) := \operatorname*{arg\,min}_{u \in \mathcal{L}_p} \left(\rho(u) + \frac{1}{2}\|x - u\|^2 \right),$$

where $x \in \mathcal{L}_p$ and $\|\cdot\|$ is the usual Euclidean norm. Then, (26) is equivalent to

$$L^+ = \mathbf{prox}_{t\rho}\left(L - t\nabla\ell(L)\right), \tag{27}$$

where $\mathbf{prox}_{t\rho}(\cdot)$ is the proximal operator applied to the scaled function $t\rho(\cdot)$. Since $\rho(u)$ is separable across the coordinates $\{u_{ij}, i \geq j\}$, we have for $x \in \mathcal{L}_p$,

$$\left(\mathbf{prox}_\rho(x)\right)_{ij} = \begin{cases} \mathbf{prox}_\rho(x_{ij}), & i > j, \\ \mathbf{prox}_0(x_{ii}) = x_{ii}, & i = j. \end{cases}$$

The proximal operators on the RHS are univariate, and the distinction between the two cases is because we do not penalize the diagonal entries, i.e., $\rho(u_{ii}) = 0$.

The overall procedure is summarized in Algorithm 2. To choose the step size t normalized by $\|\nabla\ell(L)\|_F$ (line 4), we have used a line search strategy [38], where we repeatedly reduce the step size by a factor $\kappa \in (0, 1)$ until a quadratic upper bound is satisfied by the new update (line 9). To implement Algorithm 2, we need two more ingredients, $\nabla\ell(L)$ and the univariate $\mathbf{prox}_\rho(\cdot)$, both of which have nice closed-form expressions:

Algorithm 2 Compute the RC score by proximal gradient.

Input: P, $L^{(0)} \in \mathcal{L}_p$, $t^{(0)} > 0$, $\kappa \in (0, 1)$, `max-iter`, `tol`.
Output: L.

1: $k \leftarrow 0$, err $\leftarrow \infty$, $L \leftarrow L^{(0)}$.
2: **while** $k <$ `max-iter` and err $>$ `tol` **do**
3: Compute $\nabla\ell(L)$ using either Lemma 4 or 5.
4: $t \leftarrow t^{(0)}/\|\nabla\ell(L)\|_F$.
5: **repeat**
6: $\widetilde{L} \leftarrow L - t\nabla\ell(L)$.
7: $L^+_{ij} \leftarrow \mathbf{prox}_{t\rho}(\widetilde{L}_{ij})$ for $i > j$ (using Lemma 6).
8: $L^+_{ii} \leftarrow \widetilde{L}_{ii}$.
9: **break if** $\ell(L^+, P) \leq \ell(L, P) + \langle \nabla\ell(L), L^+ - L \rangle + \frac{1}{2t}\|L^+ - L\|_F^2$.
10: $t \leftarrow \kappa t$.
11: err $\leftarrow \max_j \delta(L^+_j, L_j)$ where $\delta(x, y) := \frac{\|x-y\|}{\max\{1, \|y\|\}}$.
12: $L \leftarrow L^+$ and $k \leftarrow k + 1$.
13: **end while**

Lemma 4. *The gradient of $\ell(L, P)$ in (10) w.r.t. L is*

$$\nabla\ell(L) = n\left(\Pi_{\mathcal{L}}(P\widehat{\Sigma}P^\top L) - \operatorname{diag}\left(\{1/L_{ii}\}_{i=1}^p\right)\right),$$

where $\Pi_{\mathcal{L}} : A \mapsto (A_{ij}\mathbb{1}\{i \geq j\})_{p\times p}$ maps a matrix to its lower triangular projection.

Lemma 5. *The gradient of $\ell_{\mathcal{O}}(L, P)$ in (17) w.r.t. L_j is*

$$\nabla_{L_j}\ell_{\mathcal{O}}(L, P) = \left|\mathcal{O}_{\pi(j)}\right|\left(\Pi_j\left(P\widehat{\Sigma}^j P^\top L_j\right) - \frac{e_j}{L_{jj}}\right),$$

where $\Pi_j : v \mapsto (v_i\mathbb{1}\{i \geq j\})_{p\times 1}$ and $\{e_j\}$ is the canonical basis of $\mathbb{R}^p$.

Lemma 6. *Let ρ be the scalar MCP with parameter (γ, λ) defined*

in (15)*, and let* ρ_1 *be the same penalty for* $\lambda = \gamma = 1$*. Then, for any* $t > 0$*,*

$$\mathbf{prox}_{t\rho}(x) = \lambda\gamma\, \mathbf{prox}_{(t/\gamma)\rho_1}\left(\frac{x}{\lambda\gamma}\right), \tag{28}$$

and for any $\alpha > 0$*,*

$$\mathbf{prox}_{\alpha\rho_1}(x) = \begin{cases} 0, & 0 \le x < \min\{\alpha, 1\} \text{ or} \\ & 1 < x < \sqrt{\alpha}; \\ \dfrac{x-\alpha}{1-\alpha}, & \alpha < x \le 1; \\ x, & x > \max\{\alpha, 1\} \text{ or} \\ & 1 < \sqrt{\alpha} < x \le \alpha. \end{cases} \tag{29}$$

Moreover, $\mathbf{prox}_{\alpha\rho_1}(-x) = -\mathbf{prox}_{\alpha\rho_1}(x)$ *for all* $x \in \mathbb{R}$*.*

We have excluded two special cases in (29) in which the minimizer is not unique: 1) If $x = \alpha = 1$, $\mathbf{prox}_{\alpha\rho_1}(x) = [0, 1]$; 2) If $x = \sqrt{\alpha} > 1$, $\mathbf{prox}_{\alpha\rho_1}(x) = \{0, \sqrt{\alpha}\}$. We set $\mathbf{prox}_{\alpha\rho_1}(x) = 0$ in our implementation if these special cases occur. The MCP has parameter $\gamma > 1$, and usually the step size $t < 1$. Thus, the cases with $\alpha < 1$ are the most common scenario in our numerical study.

5.3 Structure refinement after annealing

At the end of the annealing loop (line 9, Algorithm 1), a pair $(\widehat{L}, \widehat{P})$ is found. Accordingly, an estimated reversal of a topological sort is $\widehat{\pi} = \widehat{P}(1, \ldots, p)^\top$. Define $\widetilde{L} = \widehat{P}^\top \widehat{L} \widehat{P}$, and $\widehat{B}$ by $\widehat{B}_{ij} = -\widetilde{L}_{ij}/\widetilde{L}_{jj}$ for $i \neq j$ and $\widehat{B}_{ii} = 0$. Then, $\widehat{B}$ is the estimated weighted adjacency matrix for a DAG, i.e., an estimate for B_0. The support of $\widehat{B}$ gives the estimated parent sets $\widehat{pa}_j = \{i : \widehat{B}_{ij} \neq 0\}$ for $j = 1, \ldots, p$. The use of a continuous regularizer, i.e. MCP, eases our optimization problem; however, this may lead to more false positive edges compared to ℓ_0 regularization. To improve structure learning accuracy, we add a refinement step to remove some predicted edges by conditional independence tests, which borrows the strength from a constraint-based approach.

The refinement step outlined in Algorithm 3 is based on the following fact. If $k \prec j$ in a topological sort and there is no edge $k \to j$, then $X_k \perp X_j \mid \Pi_j$, where Π_j is the parent set of X_j. For each $k \in \widehat{pa}_j$, we test the null hypothesis that X_k and X_j are conditionally independent given $\widehat{pa}_j \setminus \{k\}$ using the Fisher Z-score. We remove the edge $k \to j$ if the null hypothesis is not rejected at a given significance level. The conditional independence tests are performed in a sequential manner for the nodes in $\widehat{pa}_j$ according to the estimated topological sort: For $k_1, k_2 \in \widehat{pa}_j$, if $k_1 \prec k_2$ in the sort, we carry out the test for k_2 prior to that for k_1.

5.4 Selection of the tuning parameters

Before starting the iterations in Algorithm 1, we select and fix the tuning parameters $\theta = (\gamma, \lambda)$ of MCP (line 1), hence fixing a particular scoring function $f(L, P) = f_\theta(L, P)$ throughout the algorithm.

We use the Bayesian information criterion (BIC) [39] to select the tuning parameters, given an initial permutation P_0. The details are summarized in Algorithm 4. For every pair $(\gamma^{(i)}, \lambda^{(i)})$ over a grid of values, we evaluate the BIC score given in line 2, where $\|\widehat{L}(\cdot)\|_0$ is the number of nonzero entries in $\widehat{L}(\cdot)$, and then we output the one with the lowest

Algorithm 3 Constraint-based structure refinement.

Input: $\mathbf{X}$, adjacency matrix $\widehat{B}$, significance level α.
Output: Adjacency matrix $\widehat{B}$.

$Z_\alpha \leftarrow \Phi^{-1}(1 - \frac{\alpha}{2})$, where $\Phi(x)$ is the CDF of $\mathcal{N}(0, 1)$.
for $j = 1, \ldots, p$ **do**
  $\widehat{pa}_j \leftarrow \{i : \widehat{B}_{ij} \neq 0\}$.
  for $k \in \widehat{pa}_j$ **do**
    $\mathbf{s} \leftarrow \widehat{pa}_j \setminus \{k\}$.
    $\mathbf{X}^j \leftarrow$ observations for which j is not intervened
    $n \leftarrow$ number of rows in $\mathbf{X}^j$.
    $r_{j,k|\mathbf{s}} \leftarrow$ sample partial correlation between X_j and X_k given $X_\mathbf{s}$ based on $\mathbf{X}^j$.
    $z \leftarrow \frac{1}{2}\sqrt{n - |\mathbf{s}| - 3}\log\left(\frac{1+r_{j,k|\mathbf{s}}}{1-r_{j,k|\mathbf{s}}}\right)$.
    Remove k from $\widehat{pa}_j$, if $|z| < Z_\alpha$.
  end for
  $\widehat{B}_{ij} \leftarrow 1$ if $i \in \widehat{pa}_j$ and $\widehat{B}_{ij} \leftarrow 0$ otherwise.
end for

Algorithm 4 Tuning parameter selection by BIC.

Input: Initial permutation P_0 and a grid of values $\{\theta^{(i)}\} = \{(\gamma^{(i)}, \lambda^{(i)})\}$.
Output: Optimal index i^* in the grid.

Define $\widehat{L}(\theta) := \arg\min_{L \in \mathcal{L}_p} f_\theta(L; P_0)$ computed by Algorithm 2.
Let $\mathrm{BIC}(\theta) := 2\ell(\widehat{L}(\theta), P_0) + \|\widehat{L}(\theta)\|_0 \log(\max\{n, p\})$.
Output $i^* = \arg\min_i \mathrm{BIC}(\theta^{(i)})$.

BIC score. The regularization parameter in BIC(θ) is adapted to $\log(\max\{n, p\})$, which works well for both low and high-dimensional data. To construct the grid, possible choices for the concavity parameter γ are $\{2, 10, 50, 100\}$ based on our tests. Note that $\gamma > 1$ is required in the definition of MCP (15), while the behavior of MCP for $\gamma \ge 100$ is essentially the same as the ℓ_1 penalty. For the regularization parameter λ, we select 20 equi-spaced points from the interval $[0.1\sqrt{n}, \sqrt{n}]$. The choice of $\sqrt{n}$ often leads to an empty graph when the data are standardized, hence a natural end point.

6 Results on Observational Data

6.1 Methods and data

Recall that p is the number of variables and n is the number of observations. For a thorough evaluation of the algorithm, we simulated data for both $n > p$ and $n < p$ cases.

We used real and synthetic networks to simulate data. Real networks were downloaded from the Bayesian networks online repository [40]. We duplicated some of them to further increase the network size. Synthetic DAG structures were constructed using the `sparsebn` package [41]. Given a DAG structure, we sampled the edge coefficients β_{ij} uniformly from $[-0.8, -0.5] \cup [0.5, 0.8]$ and set the noise variance to one. We then calculated the covariance matrix according to (4) and normalized its diagonal elements to one. Consequently, the variances of $\{X_1, \ldots, X_p\}$ were identical. We used the following networks to generate observational data, denoted by the network name and (p, s_0),

where s_0 is the number of edges after duplication: 4 copies of `Hailfinder` (224, 264), 1 copy of `Andes` (223, 338), 2 copies of `Hepar2` (280, 492), 4 copies of `Win95pts` (304, 448), 1 copy of `Pigs` (441, 592), and random DAGs, `rDAG1` (300, 300) and `rDAG2` (300, 600). The sample size $n = 200$ for real networks and $n = 240$ for synthetic graphs in the $n < p$ case. In the low-dimensional setting $n > p$, the sample size $n = 450, 500, 600, 600$ for `rDAG1`, `Win95pts`, `Pigs` and `rDAG2`, respectively, and $n = 400$ for the other networks.

In the observational data setting, we compared our algorithm with the following BN learning algorithms: the coordinate descent (CD) algorithm [23], the standard greedy hill climbing (HC) algorithm [5], the greedy equivalence search (GES) [6], the Peter-Clark (PC) algorithm [1], the max-min hill-climbing (MMHC) algorithm [4], and the genetic algorithm (GA) [15].

The CD algorithm optimizes a regularized log-likelihood function by a blockwise update on (β_{ij}, β_{ji}) while checking the acyclicity constraint before each update. The HC algorithm performs a greedy search over the DAG space by starting from a certain initial state, performing a finite number of local changes and selecting the DAG with the best improvement in each local change. The GES algorithm searches over the equivalence classes and utilizes greedy search operators on the current state to find the next one, of which the output is an equivalence class of DAGs. The PC algorithm performs conditional independence tests to identify edges and orients edge directions afterwards. The MMHC algorithm constructs the skeleton of a Bayesian network via conditional independence tests and then performs a greedy hill climbing search to orient the edges via optimizing a Bayesian score. The GA decomposes graph estimation into two optimization sub-problems: node ordering search with mutation and crossover operators, and structure optimization by an adaption of the least angle regression [25].

Among these methods, PC is a constraint-based method, and MMHC is a hybrid method. Other methods, CD, HC, GES and GA, are all score-based, where CD and HC search over the DAG space, GES searches over the equivalence classes, and GA searches over the permutation space. Our method is a score-based search over the permutation space, similar to GA.

Our ARCS algorithm (Algorithm 1) may take an initial permutation P_0 provided by a local search method. In this study, we use the CD and GES algorithms to provide an initial permutation, and call the corresponding implementation ARCS(CD) and ARCS(GES). To partially preserve properties of the input initial permutation, we start with a low temperature $T^{(0)} = 1$. The output of the CD algorithm is a DAG for which we find a topological sort to define P_0. The GES algorithm outputs a completed partially directed acyclic graph (CPDAG). We then generate a DAG in the equivalence class of the estimated CPDAG, and initialize ARCS with a topological sort of this DAG.

We implemented the ARCS algorithm in MATLAB, with source code available at https://github.com/yeqiaoling/arcs_bn. We used the following R packages for other methods: `sparsebn` [41] for the CD algorithm, `rcausal` [42] for the GES, GIES (for experimental data) and PC algorithms, `bnlearn` [43] for the MMHC and HC algorithms, and `GADAG` [15] for the GA. Among score-based methods, HC and GES used the BIC scoring function for Gaussian data; CD used an MCP-regularized likelihood scoring function, with an internal tuning parameter selection method; GA used an ℓ_1-regularized likelihood, for which we applied grid search for tuning parameter selection.

6.2 Accuracy metrics

Among all methods applied to observational data, ARCS, CD, HC and MMHC output DAGs, while the GES and PC algorithms output CPDAGs. Given these estimates, we need to evaluate the performance of each method. To standardize the performance metrics in observational data setting, we transfer an estimated DAG into its CPDAG before calculating the following metrics.

Define P, TP, FP, M, R as the numbers of estimated edges, true positive edges, false positive edges, missing edges and reversed edges, respectively, all with respect to CPDAGs. P is the number of edges in the estimated graph. FP is the number of edges in the estimated graph skeleton but not in the true skeleton. M counts the number of edges in the true skeleton but not in the skeleton of the estimated graph. TP reports the number of consistent edges, including edge orientation for directed edges, between the estimated CPDAG and the true CPDAG. Lastly, the number of reversed edges $\mathrm{R} = \mathrm{P} - \mathrm{TP} - \mathrm{FP}$ and so R includes both incorrectly oriented edges and those edges that are oriented in one CPDAG but undirected in the other.

Denote by s_0 the number of edges in the true CPDAG. The overall accuracy of a method is measured by the structural Hamming distance (SHD) and Jaccard index (JI), where $\mathrm{SHD} = \mathrm{R} + \mathrm{FP} + \mathrm{M}$ and $\mathrm{JI} = \mathrm{TP}/(s_0 + \mathrm{P} - \mathrm{TP})$. A method has better performance if it achieves a lower SHD and/or a higher JI.

6.3 Structure learning accuracy

We used large networks, where $p \in (200, 450)$ and $s_0 \in (250, 600]$, to simulate observational data with $n < p$ and $n > p$. For each setting (p, s_0, n), we generated 20 datasets, and ran CD, ARCS(CD), GES, ARCS(GES) and other methods (PC, HC, MMHC and GA) with a maximum time allowance of 10 minutes per dataset. The HC and MMHC algorithms had an upper-bound of the in-degree number as 2. We tried a higher maximum in-degree, but it resulted in a large FP. MMHC and PC were run with a significance level of 0.01 in conditional independence tests. We ran the CD algorithm with an MCP regularized likelihood, in which $\gamma = 2$ and λ was chosen by a default model selection mechanism. GA was run for a maximum of 10^4 iterations, using the default population size and the default rates of mutation and crossover. We tried a larger population size for GA, but it was too time-consuming.

Our methods, ARCS(CD) and ARCS(GES), initialized with permutations from CD and GES estimates, were run for a maximum of $N = 10^4$ iterations, with $T^{(0)} = 1$, $T^{(N)} = 0.1$, and reversal length $m = 4$ (Algorithm 1). The temperature decreased geometrically as $T^{(i)} = \alpha^i T^{(0)}$, where α is determined by $T^{(0)}, T^{(N)}$ and N. A p-value cutoff of 10^{-5} was used in the refinement step (Algorithm 3). For the networks we considered, on average, 500 tests were

TABLE 1: Comparison between ARCS and initial estimates on observational data. ARCS improved GES and CD estimates, and ARCS(GES) achieved the best accuracy for every network.

Network	Method	$n < p$					$n > p$				
(p, s_0)		TP	R	FP	SHD (sd)	JI (sd)	TP	R	FP	SHD (sd)	JI (sd)
Hailfinder	ARCS(GES)	203	42	16	76 (25)	0.64 (0.11)	237	21	15	42 (13)	0.79 (0.06)
(224, 264)	ARCS(CD)	183	58	33	114 (18)	0.51 (0.06)	195	60	41	110 (26)	0.54 (0.07)
	GES	180	53	10	94 (22)	0.56 (0.10)	228	26	18	54 (12)	0.74 (0.05)
	CD	145	93	31	150 (17)	0.38 (0.05)	150	94	34	148 (13)	0.38 (0.04)
Andes	ARCS(GES)	274	33	27	91 (33)	0.69 (0.10)	295	27	26	70 (29)	0.76 (0.08)
(223, 338)	ARCS(CD)	228	75	65	174 (37)	0.48 (0.07)	238	72	66	166 (63)	0.51 (0.10)
	GES	218	36	17	137 (18)	0.56 (0.06)	271	35	36	103 (32)	0.67 (0.09)
	CD	169	112	63	232 (15)	0.33 (0.03)	184	112	70	223 (35)	0.36 (0.05)
Hepar2	ARCS(GES)	300	119	64	255 (27)	0.45 (0.04)	309	121	74	257 (41)	0.45 (0.05)
(280,492)	ARCS(CD)	263	155	82	312 (37)	0.36 (0.04)	284	152	87	294 (55)	0.39 (0.06)
	GES	238	110	62	316 (24)	0.36 (0.04)	299	126	84	277 (28)	0.43 (0.04)
	CD	205	156	100	388 (20)	0.27 (0.03)	229	168	121	384 (31)	0.29 (0.04)
Win95pts	ARCS(GES)	348	69	34	134 (21)	0.63 (0.05)	379	58	38	107 (19)	0.70 (0.04)
(304, 448)	ARCS(CD)	301	104	59	206 (22)	0.49 (0.04)	318	114	111	241 (51)	0.48 (0.07)
	GES	236	78	21	232 (17)	0.43 (0.04)	320	88	64	192 (17)	0.53 (0.03)
	CD	177	169	47	317 (26)	0.27 (0.04)	187	172	39	300 (26)	0.28 (0.04)
Pigs	ARCS(GES)	446	102	27	172 (90)	0.62 (0.13)	466	107	46	172 (40)	0.63 (0.05)
(441, 592)	ARCS(CD)	401	145	47	239 (73)	0.51 (0.10)	437	136	57	212 (55)	0.56 (0.06)
	GES	432	112	38	198 (18)	0.58 (0.03)	467	122	57	182 (23)	0.61 (0.04)
	CD	324	224	106	374 (26)	0.35 (0.02)	334	243	175	433 (43)	0.33 (0.03)
rDAG1	ARCS(GES)	289	7	1	12 (7)	0.94 (0.03)	297	3	1	4 (6)	0.98 (0.03)
(300,300)	ARCS(CD)	253	43	9	56 (14)	0.72 (0.06)	264	35	20	56 (20)	0.75 (0.08)
	GES	274	8	1	27 (6)	0.89 (0.03)	293	4	2	9 (8)	0.96 (0.04)
	CD	189	104	21	132 (18)	0.45 (0.05)	196	98	24	127 (17)	0.47 (0.05)
rDAG2	ARCS(GES)	562	20	12	51 (12)	0.89 (0.03)	584	13	11	27 (14)	0.94 (0.03)
(300,600)	ARCS(CD)	481	90	52	171 (41)	0.65 (0.06)	491	99	106	215 (62)	0.61 (0.06)
	GES	470	29	6	136 (20)	0.74 (0.04)	558	21	17	58 (15)	0.88 (0.03)
	CD	382	159	59	276 (32)	0.47 (0.03)	379	165	61	283 (25)	0.46 (0.03)

In this and all subsequent tables, reported results are rounded averages over 20 datasets.

performed in the refinement step, and the cutoff was chosen by Bonferroni correction to control the familywise error rate at level 0.005. In fact, our results were almost identical for any p-value cutoff between 10^{-3} and 10^{-5}.

To perform a complete comparison, we applied the structure refinement step (Section 5.3) to other competing methods as well. It turned out that ARCS, CD, HC and MMHC benefitted from this additional step. Therefore, in this section, we report the results of these methods with the refinement step. Section 6.6 evaluates the effect of this structure refinement step, including results of these methods before refinement.

ARCS versus CD and GES. Table 1 reports the average performance metrics across 20 datasets for 7 networks (5 real and 2 random networks) using CD, ARCS(CD), GES and ARCS(GES). We were interested in the potential improvement of ARCS upon its initial permutations. It is indeed confirmed by the results in the table that ARCS(CD) and ARCS(GES) outperformed CD and GES, respectively, for every network, achieving lower SHDs and higher JIs. The reduction in SHD was close to or above 20% across networks. ARCS always increased TP, while maintaining or slightly reducing FP. The annealing process identified more TP edges, while the refinement step (Algorithm 3) cut down the FP edges given the ordering and parent sets learned through simulated annealing. In the case of $n < p$ for the Pigs network, the high standard deviation of the SHDs of ARCS was caused by a couple of outliers. If we excluded the corresponding data sets, the average SHD of ARCS(GES) and ARCS(CD) would decrease to 153 and 225 with standard deviations of 23 and 41, respectively.

ARCS(GES) versus other methods. We also compared ARCS(GES) with other existing methods, including HC, PC, MMHC and GA. Table 2 summarizes the average performance, where ARCS(GES) outperforms competing algorithms by a great margin. The HC algorithm tended to output a denser DAG than the truth, leading to a large FP. The PC algorithm had a relatively large number of reverse edges, causing a high SHD. The MMHC algorithm had a lower SHD than some other algorithms, but the SHD difference between ARCS(GES) and MMHC was still large. The PC and MMHC algorithms were slow for some networks, and thus are absent in the results for these networks. The GA was formulated in a similar way as ARCS(GES), but the TPs of GA estimates were much lower, resulting in large SHDs for the tested networks.

It is worth mentioning that ARCS(GES) outperformed other methods substantially for larger networks such as

TABLE 2: ARCS against other methods on observational data. ARCS(GES) achieved the best SHD and JI among all methods for every network.

Network	Method	$n < p$					$n > p$				
(p, s_0)		TP	R	FP	SHD (sd)	JI (sd)	TP	R	FP	SHD (sd)	JI (sd)
Hailfinder	ARCS(GES)	203	42	16	76 (25)	0.64 (0.11)	237	21	15	42 (13)	0.79 (0.06)
(224, 264)	HC	151	80	31	144 (13)	0.40 (0.04)	151	87	50	163 (19)	0.38 (0.06)
	PC	41	167	13	236 (7)	0.09 (0.02)	45	191	14	233 (14)	0.10 (0.03)
	GA	51	70	72	284 (14)	0.13 (0.02)	43	68	28	248 (13)	0.12 (0.03)
Andes	ARCS(GES)	274	33	27	91 (33)	0.69 (0.10)	295	27	26	70 (29)	0.76 (0.08)
(223, 338)	HC	142	112	62	258 (19)	0.28 (0.03)	148	121	85	275 (27)	0.27 (0.04)
	PC	84	169	13	267 (10)	0.16 (0.02)	85	202	16	269 (10)	0.15 (0.02)
	GA	57	87	72	352 (14)	0.12 (0.02)	36	84	33	335 (9)	0.08 (0.02)
	MMHC	147	88	3	194 (15)	0.34 (0.05)	171	98	4	171 (16)	0.39 (0.05)
Hepar2	ARCS(GES)	300	119	64	255 (27)	0.45 (0.04)	309	121	74	257 (41)	0.45 (0.05)
(280, 492)	HC	170	149	83	405 (20)	0.24 (0.03)	174	155	95	413 (19)	0.23 (0.02)
	PC	96	154	30	427 (13)	0.14 (0.02)	108	189	33	417 (15)	0.15 (0.02)
	GA	70	107	143	565 (14)	0.09 (0.01)	51	98	77	518 (14)	0.08 (0.02)
	MMHC	86	156	20	426 (15)	0.13 (0.02)	130	158	21	383 (15)	0.19 (0.02)
Win95pts	ARCS(GES)	348	69	34	134 (21)	0.63 (0.05)	379	58	38	107 (19)	0.70 (0.04)
(304, 448)	HC	147	177	54	355 (19)	0.22 (0.03)	148	212	111	410 (17)	0.19 (0.02)
	PC	124	207	28	352 (12)	0.18 (0.02)	134	264	27	341 (12)	0.18 (0.02)
	GA	83	114	57	422 (12)	0.13 (0.02)	41	86	12	419 (7)	0.07 (0.01)
	MMHC	157	147	4	295 (12)	0.26 (0.02)	177	191	3	274 (11)	0.28 (0.02)
Pigs	ARCS(GES)	446	102	27	172 (90)	0.62 (0.13)	466	107	46	172 (40)	0.63 (0.05)
(441, 592)	HC	353	174	102	341 (30)	0.41 (0.04)	367	173	132	357 (41)	0.41 (0.04)
	GA	79	187	91	605 (12)	0.09 (0.01)	46	150	22	569 (10)	0.06 (0.01)
rDAG1	ARCS(GES)	289	7	1	12 (7)	0.94 (0.03)	297	3	1	4 (6)	0.98 (0.03)
(300, 300)	HC	176	108	29	153 (13)	0.40 (0.03)	178	113	45	168 (18)	0.39 (0.04)
	PC	90	202	30	240 (11)	0.17 (0.02)	83	215	31	248 (11)	0.15 (0.02)
	GA	64	81	33	269 (9)	0.15 (0.02)	54	75	6	253 (10)	0.14 (0.03)
	MMHC	193	89	1	108 (8)	0.50 (0.03)	191	103	0	109 (8)	0.47 (0.03)
rDAG2	ARCS(GES)	562	20	12	51 (12)	0.89 (0.03)	584	13	11	27 (14)	0.94 (0.03)
(300, 600)	HC	253	180	51	398 (20)	0.31 (0.02)	248	201	76	428 (17)	0.28 (0.02)
	PC	194	307	7	413 (16)	0.21 (0.02)	175	389	6	431 (13)	0.18 (0.02)
	GA	108	140	63	555 (15)	0.14 (0.02)	42	96	5	563 (7)	0.06 (0.01)
	MMHC	277	156	0	323 (10)	0.37 (0.02)	284	191	0	316 (17)	0.36 (0.03)

If a method is absent for a network, that means, it took more than 10 minutes to run on a singe dataset, and thus is excluded from the comparison.

Pigs ($p = 441, s_0 = 592$). MMHC and PC failed to complete a single run on the Pigs network within 10 minutes, while HC and GA had very low accuracies. We suspect that the Pigs network has a certain structure that is particularly difficult to estimate, a hypothesis that merits more investigation.

Additional tests. We also compared ARCS with DAGs with NO TEARS (NOTEARS), a continuous optimization method for structure learning of BNs developed recently [24]. This method is not restricted to continuous data, and thus more general than ARCS. The current version of NOTEARS required a very large amount of memory for large DAGs such as those in Tables 1 and 2. Therefore, we restricted the comparison to relatively small DAGs. To this end, we generated observational data from 9 graphs (with at most 20 nodes), and then applied grid search to choose NOTEARS's tuning parameter since it required a pre-fixed tuning parameter. ARCS achieved higher accuracy than NOTEARS consistently for most graphs (8 out of 9), except Sachs ($p = 11, s_0 = 17$). The advantage of ARCS over NOTEARS grew with the size and complexity of the true DAG. Detailed comparison results are provided in the supplementary material, Section S1.1.

We also tested another order-based algorithm, linear structural equation model learning (LISTEN) [19], which estimates Gaussian DAG structure by a sequential detection of ordering. A key assumption of LISTEN is that the noise variables have equal variances. Moreover, the algorithm requires a prespecified regularization parameter for the score metric. To compare with this algorithm, we adapted our data generation process to satisfy the equal-variance assumption. ARCS(GES) achieved a much higher accuracy, with SHD as small as 13% to 18% of the SHD achieved by LISTEN. Full results are reported in the supplementary material, Section S1.2.

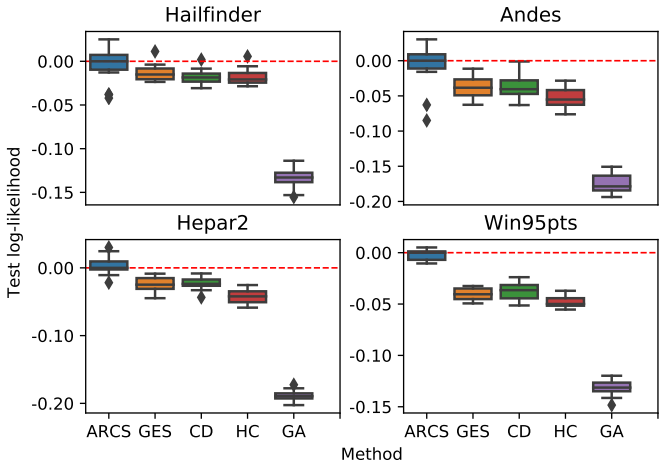


Fig. 3: Test data log-likelihood comparison among BN learning methods. Log-likelihoods are shifted by the median of ARCS (the dashed line).

6.4 Test data likelihood comparison

As discussed in Remark 2, an estimated DAG (or CPDAG) defines a multivariate Gaussian distribution for X via (4). To evaluate the accuracy in estimating the covariance matrix Σ_0, we compared test data log-likelihood as follows. Given an estimated DAG, we applied least-squares regression of each node on its parents to estimate B_0 and Ω_0. Then we simulated test data under the same Gaussian SEM with the true parameters (B_0, Ω_0) and calculated log-likelihood of the test data. A higher test data log-likelihood indicates a better estimate of Σ_0.

Figure 3 shows the test data log-likelihood of a few BN learning methods, including ARCS(GES), GES, CD, HC and GA, on four networks in the case of $n < p$. The case of $n > p$ shows a similar pattern and is thus omitted for brevity. We shifted test data log-likelihoods by the median of ARCS, so that the ARCS medians were always zero. The log-likelihood distributions for all other methods were below zero, indicating that ARCS achieved significantly higher accuracy in estimating Σ_0. Additional results for three other networks (supplementary Figure S3) support exactly the same conclusion here. In all these results, we normalized the log-likelihood by the sample size n and the dimension of the problem p.

6.5 Precision matrix estimation

For a given P, the minimizer of the RC loss $f(L; P)$ (13) is a sparse Cholesky factor of $P\Sigma_0^{-1}P^\top$, the precision matrix Σ_0^{-1} after permuting rows and columns according to P. Instead of joint optimization over (L, P) in (14), one could first estimate a sparse precision matrix $\widehat{\Theta}$ and then learn a DAG structure though Cholesky decomposition of $\widehat{\Theta}$ as in (4). The Cholesky decomposition can be done with a given ordering or by searching for an order that leads to a sparse Cholesky factor. In this section, we compare ARCS(GES), or simply ARCS, against two precision matrix estimation methods, followed by Cholesky decomposition, to clarify their differences.

The first approach estimates a precision matrix via modified Cholesky decomposition (MCD) [30] which requires an input ordering of the variables. We provided MCD with ARCS initial and final orders, which we have called MCD and MCD*, respectively. The initial order of ARCS was found by GES. Given an estimated sparse precision matrix and the input order, we found the corresponding DAG structure encoded by the Cholesky factor, and then refined the DAG structure using Algorithm 3, the same refinement step used by ARCS. The comparison against MCD will show the advantage of joint minimization over (L, P), while the comparison against MCD* demonstrates the effectiveness of ARCS in terms of estimating precision matrices, since the two methods use exactly the same ordering in Cholesky decomposition.

The second approach is graphical Lasso [44] followed by a Cholesky decomposition. Given a sparse precision matrix $\widehat{\Theta}$ estimated by graphical Lasso, we used the approximate minimum degree ordering algorithm [45], a heuristic common in numerical linear algebra, to find a permutation P such that the Cholesky factor of $P\widehat{\Theta}P^\top$ is sparse. We call this approach GC. We also applied the Cholesky decomposition to $\widehat{P}\widehat{\Theta}\widehat{P}^\top$, where $\widehat{P}$ is the permutation found by ARCS. We call this approach GC*. The comparison between ARCS and GC will highlight the importance of imposing sparsity on L in our formulation (14), rather than on Σ^{-1} as in GC. The difference between GC* and GC will show the effectiveness of the ordering obtained by ARCS, relative to the minimum degree ordering, in achieving a sparse Cholesky factorization.

We used R package `CovTools` to run MCD, the `glasso` package to run graphical Lasso, and MATLAB function `chol` for the approximate minimum degree algorithm. Based on a grid search on a sample dataset, we chose the tuning parameter of 0.001 for graphical Lasso. The upper bound on the bandwidth of the precision matrix, as required by `CovTools`, was set to 10.

Table 3 compares the structure recovery accuracy among ARCS, GC*, GC, MCD* and MCD. ARCS showed the best performance with a great margin, followed by GC* which used the permutation found by ARCS to complete the Cholesky decomposition. The observation that GC* consistently outperformed GC indicates that ARCS found a better topological sort than the minimum degree algorithm. Both MCD and MCD* suffered from high SHDs and low JIs, which shows that imposing sparsity on the precision matrix and decoupling its estimation from the order search would be suboptimal for DAG structure learning.

We also compared test data log-likelihood among these methods in supplementary Figure S4. Interestingly, although not designed for precision matrix estimation, ARCS achieved a much higher test data likelihood compared to all the precision matrix estimation methods. In particular, the fact that ARCS outperformed MCD* implies that our method did a better job at estimating precision matrices compared to MCD, even though the latter was provided with the same ordering finally used by ARCS.

6.6 Effectiveness of refinement

Recall that we employed the constraint-based refinement step in ARCS after annealing (Section 5.3). To analyze the

TABLE 3: ARCS against precision matrix estimation methods

Network (p, s_0)	Method	$n < p$ SHD(sd)	$n < p$ JI(sd)	$n > p$ SHD(sd)	$n > p$ JI(sd)
Hailfinder	ARCS	76 (25)	0.64 (0.11)	42 (13)	0.79 (0.06)
(224, 264)	GC*	128 (15)	0.53 (0.06)	82 (15)	0.64 (0.06)
	GC	297 (10)	0.09 (0.01)	236 (19)	0.16 (0.05)
	MCD*	290 (7)	0.03 (0.02)	304 (11)	0.06 (0.02)
	MCD	285 (10)	0.06 (0.02)	300 (12)	0.10 (0.03)
Andes	ARCS	91 (33)	0.69 (0.10)	70 (29)	0.76 (0.08)
(223, 338)	GC*	98 (24)	0.67 (0.08)	169 (19)	0.51 (0.05)
	GC	597 (114)	0.04 (0.05)	266 (75)	0.29 (0.17)
	MCD*	335 (16)	0.07 (0.05)	345 (25)	0.11 (0.04)
	MCD	322 (14)	0.09 (0.04)	321 (15)	0.14 (0.03)
Hepar2	ARCS	255 (27)	0.45 (0.04)	257 (41)	0.45 (0.05)
(280, 492)	GC*	392 (24)	0.32 (0.03)	328 (35)	0.38 (0.04)
	GC	546 (27)	0.16 (0.02)	497 (28)	0.19 (0.02)
	MCD*	594 (16)	0.02 (0.01)	659 (22)	0.02 (0.01)
	MCD	557 (13)	0.07 (0.02)	596 (15)	0.09 (0.02)
Win95pts	ARCS	134 (21)	0.63 (0.05)	107 (19)	0.70 (0.04)
(304, 448)	GC*	160 (17)	0.58 (0.04)	402 (11)	0.10 (0.02)
	GC	390 (14)	0.16 (0.02)	428 (7)	0.06 (0.01)
	MCD*	450 (4)	0.01 (0.01)	462 (7)	0.03 (0.01)
	MCD	446 (5)	0.03 (0.01)	453 (9)	0.06 (0.01)

benefit of this post-processing step, we applied the same refinement step to other methods except PC and GA. We excluded PC because as a constraint-based method, it has already performed all conditional independence tests. The GA estimated graphs had the lowest TPs in Table 2, and thus further removing edges by the refinement step would not improve its overall accuracy. Note that GES outputs a CPDAG, so we applied the refinement step on a randomly chosen DAG in the equivalence class of the estimated CPDAG.

We summarize the mean SHDs before and after the refinement step for each method in Table 4. The refinement step reduced the SHDs for most of the methods included in this comparison. It worked particularly well for HC with a substantial decrease in SHD. The improvements for ARCS, CD and MMHC were quite substantial for some datasets while marginal for other datasets. There was no change in the GES estimates after the refinement step. It is observed that ARCS(GES) achieved the smallest SHD, after the refinement step was applied to all the methods. Moreover, ARCS(GES) without refinement had already outperformed the majority of the competing methods, except for the case Hailfinder ($n < p$) when compared against GES.

6.7 Effectiveness of BIC selection

Given an initial permutation, we used the BIC to choose tuning parameters (γ, λ) before applying the ARCS algorithm (Section 5.4). In Tables 2, the number of predicted edges (TP+R+FP) by ARCS(GES) is closer to s_0 than any other competing method in every network. This observation signifies the effectiveness of our parameter selection method by BIC. To further study its effect, we compared DAGs estimated with all values on a grid of (γ, λ) by ARCS(GES), or simply ARCS.

Figure 4 shows the histograms of the SHDs of ARCS estimates for a grid of tuning parameters. We used the Andes datasets with $(p, s_0) = (223, 338)$ and $n \in \{200, 400\}$ here. The shaded part of a histogram reports SHDs achieved with

TABLE 4: Comparison on SHDs before (B) and after (A) the refinement step.

SHD	ARCS(GES) B	ARCS(GES) A	ARCS(CD) B	ARCS(CD) A	CD B	CD A	HC B	HC A	MMHC B	MMHC A	GES
Hailfinder	129	76	165	114	151	150	344	144	–	–	94
Andes	125	91	256	174	237	232	356	258	203	194	137
Hepar2	290	255	425	312	411	388	513	405	430	426	316
Win95pts	205	134	293	206	319	317	544	355	316	295	232
Pigs	193	172	263	239	974	374	547	341	–	–	198
rDAG1	13	12	62	56	137	132	437	153	143	108	27
rDAG2	68	51	200	171	389	276	483	398	328	323	136
Hailfinder	43	42	112	110	148	148	335	163	–	–	54
Andes	71	70	174	166	224	223	341	275	181	171	103
Hepar2	273	257	304	294	385	384	505	413	389	383	277
Win95pts	110	107	244	241	300	300	519	410	295	274	192
Pigs	173	172	214	212	436	433	518	357	–	–	182
rDAG1	5	4	56	56	128	127	420	168	146	109	9
rDAG2	27	27	221	215	283	283	481	428	320	316	58

The top panel shows results for $n < p$ and the bottom for $n > p$.

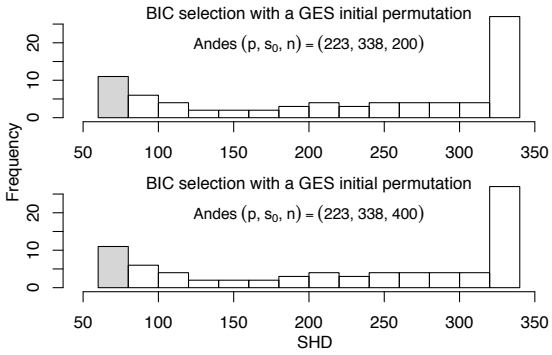


Fig. 4: Performance of the BIC selected parameter among a grid of (γ, λ) given an initial permutation. Tuning parameters that lead to lower SHDs than the BIC selection are shown in gray.

choices of tuning parameters on the grid that were smaller than the SHD corresponding to the BIC selected parameters. Each histogram has a high spike of large SHD, corresponding to large values of λ that generate almost empty graphs. The SHDs of ARCS with BIC selection corresponded to the 16^{th} and 3^{rd} percentiles in low and high sample sizes, respectively. These low percentiles confirm that the BIC selection works well for choosing the tuning parameters. Moreover, in our tests, the BIC usually selected $\gamma^* = 2$, the smallest provided value for γ. Since for small γ, the MCP is closer to the ℓ_0 penalty and far from the ℓ_1 norm, this choice of γ indicates the preference of concave penalties over ℓ_1 in estimating sparse DAGs. Some of CD's and GA's inferior performances, such as CD on the Pigs network (Table 1) and the overall performance of GA (Tables 2), were potentially caused by a bad choice of their tuning parameters. This demonstrates the importance of our data-driven selection scheme for a regularized likelihood method.

6.8 Empirical loss evaluation

To quantify the empirical loss and global search ability of our ARCS algorithm, we compared the BIC score of an ARCS estimate against the global minimum BIC score. A global minimum can be identified by integer linear programming (ILP) [46]. With a properly chosen upper bound on the parent size, and given sufficient computing budget, this method is guaranteed to find a global minimizer. We

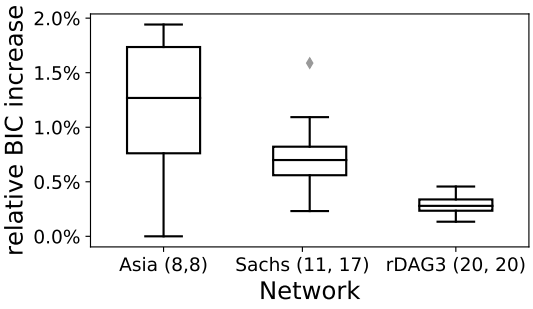


Fig. 5: Distributions of the relative BIC increase. The numbers following each DAG report its (p, s_0).

set the upper bound to the maximum parent size in the DAG estimated by ARCS(GES), and then calculated local BIC scores of all possible parent sets for each node, which were input to ILP implemented in the `GOBNILP` software package [46].

We chose relatively small DAGs ($p \leq 20$) in this comparison and simulated datasets with sample size $n = 5p$. For each dataset, we computed the BIC score of the DAG estimated by ARCS(GES) and found the minimum BIC score by ILP. Define the relative BIC increase of ACRS compared to ILP as (BIC(ARCS)$-$BIC(ILP))/BIC(ILP). Figure 5 shows the relative increase distribution across 20 datasets generated from each of the selected DAGs. Since an upper bound on parent set size has been given to ILP, its minimum score may not be best possible BIC score. As expected, the BIC score of ARCS was slightly higher than the minimum score found by ILP. However, it is comforting to see the relative BIC increase of ARCS was less than 2% for almost all the datasets. Note that the loss function of ARCS may be quite different from the BIC when n is small. This is the reason why ARCS had the largest relative BIC increase for the smallest 8-node graph `Asia` for which the sample size $n = 40$.

Note that ILP is computationally intensive. In our tests, we allocated 20 hours to ILP for each single dataset generated from four graphs, the three reported in Figure 5 and `rDAG4` ($p = 20, s_0 = 40$). Yet it still failed to find an optimum within the time constraint for the largest graph `rDAG4`. Therefore, we excluded this graph from the figure. Meanwhile ARCS took less than one minute to complete every single run for these graphs.

7 Results on Experimental Data

To generate experimental datasets, we generated p blocks of observations, in each of which a single variable was under intervention. For each block, we generated 5 observations, and thus $n = 5p$. Networks in this experiment were smaller, with $p \leq 50$ and $s_0 \leq 100$ (see Table 5). Using these networks, we also simulated observational data of the same sample size, $n = 5p$, to study the effect of experimental interventions. Since the number of conditional independence tests were smaller in this setting, we used a p-value cutoff of 10^{-3} in each refinement test of the ARCS algorithm (Algorithm 3) to control the overall false discovery rate.

To assess the accuracy on experimental data, we compare an estimated DAG with the true one to calculate the numbers of false positives (FP), missing edges (M), reverse edges (R) and true positives (TP). FP and M follow the same calculations as in the observational settings. R counts the number of edges whose orientations are opposite between the two DAGs, and TP $=$ P$-$R$-$FP. Note that the definitions of R and TP are different from those for observational data, because under the intervention setting used in this comparison, the true causal DAG is identifiable [22]. The structural Hamming distance (SHD) and the Jaccard index (JI) are then calculated as in the observational case.

TABLE 5: Performance comparison on experimental data.

Network (p, s_0, n)	Method	P	TP	R	FP	SHD	JI
`Asia`	CD	14	6	1	7	8	0.47
$(8, 8, 40)$	ARCS(CD)	5	5	1	0	**4**	0.53
	GIES	9	1	6	2	9	0.07
	ARCS(GIES)	5	5	1	0	**4**	**0.53**
`Sachs`	CD	23	9	5	10	17	0.34
$(11, 17, 55)$	ARCS(CD)	12	10	2	1	8	0.50
	GIES	17	4	10	4	17	0.13
	ARCS(GIES)	12	10	2	1	**8**	**0.51**
`Ins.`	CD	56	30	12	14	36	0.39
$(27, 52, 135)$	ARCS(CD)	54	40	7	7	**18**	**0.63**
	GIES	72	14	34	24	62	0.13
	ARCS(GIES)	57	38	10	10	24	0.56
`Alarm`	CD	51	35	9	7	16	0.57
$(37, 46, 185)$	ARCS(CD)	48	43	3	2	**5**	**0.86**
	GIES	70	6	40	24	65	0.05
	ARCS(GIES)	51	40	6	6	12	0.70
`Barley`	CD	85	52	17	16	48	0.45
$(48, 84, 240)$	ARCS(CD)	103	67	13	23	**41**	**0.58**
	GIES	146	21	58	67	129	0.10
	ARCS(GIES)	122	44	35	44	84	0.28
`rDAG3`	CD	25	15	5	6	11	0.49
$(20, 20, 100)$	ARCS(CD)	20	18	1	1	**2**	**0.84**
	GIES	29	4	15	10	25	0.09
	ARCS(GIES)	20	18	2	1	3	0.83
`rDAG4`	CD	43	25	8	10	24	0.45
$(20, 40, 100)$	ARCS(CD)	39	34	3	2	8	0.76
	GIES	48	6	30	11	46	0.07
	ARCS(GIES)	38	35	2	1	**6**	**0.82**
`rDAG5`	CD	55	39	10	6	17	0.60
$(50, 50, 250)$	ARCS(CD)	51	48	2	1	**3**	**0.90**
	GIES	87	7	43	37	80	0.05
	ARCS(GIES)	52	47	3	2	5	0.86
`rDAG6`	CD	101	70	17	13	43	0.54
$(50, 100, 250)$	ARCS(CD)	106	94	5	7	**12**	**0.86**
	GIES	155	15	81	58	143	0.06
	ARCS(GIES)	159	59	36	64	105	0.34

The best SHD and JI for each network are highlighted in boldface.

7.1 Comparison on experimental data

In this setting, we compared ARCS with the CD algorithm [22] and the greedy interventional equivalence search (GIES) algorithm [47], both of which can handle experimental interventions. We initialize ARCS with CD and GIES estimates and call them ARCS(CD) and ARCS(GIES), respectively.

Table 5 compares the CD, ARCS(CD), GIES and ARCS(GIES) algorithms, averaging over 20 datasets for

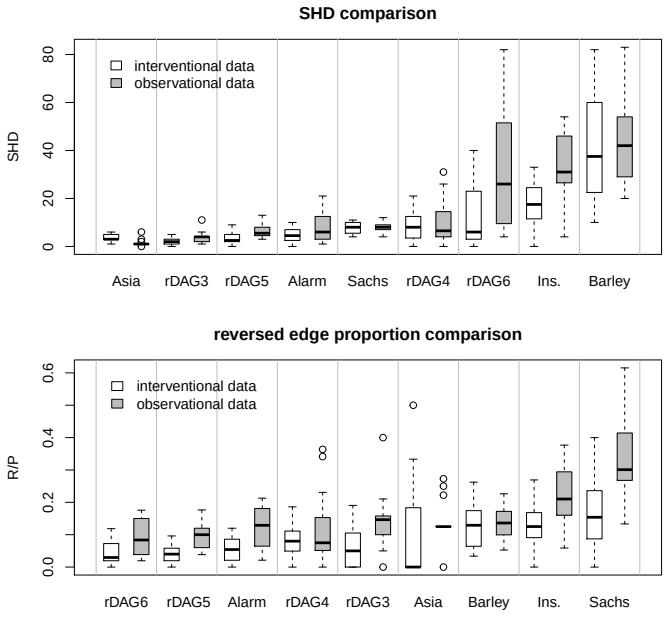


Fig. 6: Comparison of SHD and reversed edge proportion between experimental and observational data with ARCS(CD).

each type of networks. Both ARCS(GIES) and ARCS(CD) achieved dramatic improvements upon GIES and CD algorithms for every single network. This observation is consistent with the findings from the observational data and further confirms that the ARCS algorithm is a powerful tool for improving local estimates. Different from the observational data results (Table 1), ARCS(CD) usually had better performance than ARCS(GIES) on experimental data. The standard deviation, relative to the mean, of the results here was comparable to that of the observational data results, and thus is not reported in Table 5 for brevity.

We also compared the performance of our method ARCS(CD) on experimental and observational data with the same sample size $n = 5p$. Figure 6 plots the SHDs of ARCS(CD) on 20 datasets, with a side-by-side comparison between observational and experimental data. For some networks, such as `rDAG6` and `Ins.`, the estimated DAGs using experimental data had much lower SHDs than using the observational data. For some small networks, such as `Asia` ($p = 8, s_0 = 8$), ARCS(CD) achieved a low SHD on observational data and the improvement when using experimental data was not substantial.

Because estimated DAGs did not have the same number of predicted edges, we further compared the reversed edge proportion (R/P). The DAGs estimated by ARCS(CD) had a lower R/P on the experimental than the observational data for all networks. The decrease in R/P with experimental interventions was remarkable, as Figure 6 shows. This finding supports the idea that experimental interventions help correct the reversed edges and distinguish equivalent DAGs. Note that for the 20 observational datasets generated from `Asia` ($p = 8$, $s_0 = 8$), ARCS(CD) output 16 estimated DAGs with P $= 8$ and R $= 1$, resulting in a very thin interquantile range in the boxplots of `Asia` in Figure 6.

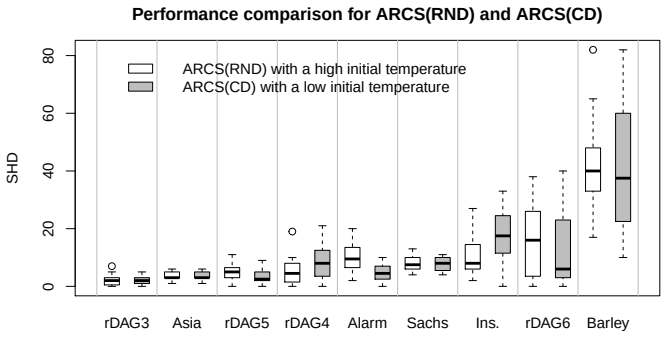


Fig. 7: A comparison between ARCS(RND) with a high initial temperature and ARCS(CD) on experimental data.

7.2 Random initialization with a high temperature

Recall that we started ARCS(CD) and ARCS(GIES) with $T^{(0)} = 1$. To test its global search ability over the permutation space, we may initialize the annealing process with a random permutation and a high temperature, which we denote by ARCS(RND). For a random initial permutation, we do not need to preserve its properties, so we use a high initial temperature $T^{(0)} = 100$ to help the algorithm traverse the search space.

We compared ARCS(RND) and ARCS(CD) on experimental data for 9 networks. We chose ARCS(CD) due to its superior performance on these networks in the experimental setting (Table 5). As shown in Figure 7, ARCS(RND) and ARCS(CD) had comparable performances on all networks. Both of them learned DAGs with small SHDs. ARCS(RND) was slightly better on `rDAG4` and `Ins.`, and slightly worse on `Alarm` and `rDAG6`. For the other networks, the two methods were quite comparable, demonstrating the effectiveness of ARCS(RND).

The networks in this experiment had $p \leq 50$. For $p = 50$, there are $50! \approx 3 \times 10^{64}$ possible permutations. ARCS(RND) managed to learn a network structure within 10^4 iterations, which is much smaller than $p!$. However, the performance of ARCS(RND) was not competitive on large networks. The reason is that for large p, it takes much more time for the annealing to thoroughly search the huge permutation space. Therefore, for large networks, it is better to initialize the ARCS algorithm with estimates from other local methods and choose a low temperature. Given a good initial estimate, ARCS searches over the permutation space and improves the accuracy of the initial estimate as demonstrated in Tables 1 and 5. This study suggests that, by searching over the permutation space under a regularized likelihood framework, our ARCS algorithm is a promising approach to the challenging problem of DAG structure learning.

8 Discussion

In this paper, we developed a method to learn Gaussian BN structures by minimizing the MCP regularized Cholesky score over topological sorts, through a joint iterative update on a permutation matrix and a lower triangular matrix. We search over the permutation space and optimize the network structure encoded by a lower triangular matrix

given a topological sort. This approach relates BN learning problem to sparse Cholesky factorization, and provides an alternative formulation for the order-based search. The proposed regularized Cholsky score is shown to be consistent for estimating topological sorts and DAG structures. Our method can serve as an improvement of a local search or a stand-alone method with a best-guess initial permutation. Although we formulated this order-based search for Gaussian BNs, it can potentially be extended to discrete BNs and other scoring functions. A main difference in the extension to discrete data is the proximal gradient step, where we can borrow the regularized multi-logit model in [48] or develop a continuous regularizer for multinomial likelihood.

With a proper temperature schedule, simulated annealing may search over the permutation space effectively, and there are several interesting aspects to investigate in the annealing process. For instance, various operators of moving from one permutation to another have been proposed for greedy order-based search, which could better guide the annealing process in traversing the search space. The numerical results in this paper demonstrate the advantage and potential application of local search and global annealing in learning BNs. Left as future work are theoretical properties of our method in high-dimensional settings, such as the consistency of the regularized Cholesky score when $p \gg n$.

Acknowledgment

This work was supported by NSF grant IIS-1546098.

References

[1] P. Spirtes and C. Glymour, "An Algorithm for Fast Recovery of Sparse Causal Graphs," *Social Science Computer Review*, vol. 9, no. 1, pp. 62–72, 1991.

[2] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Machine Learning*, vol. 20, pp. 197–243, 1995.

[3] J. Suzuki, "A Construction of Bayesian Networks from Databases Based on an MDL Scheme," in *Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence*, pp. 266–273, 1993.

[4] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm," *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.

[5] J. A. Gámez, J. L. Mateo, and J. M. Puerta, "Learning Bayesian Networks by Hill Climbing: Efficient Methods Based on Progressive Restriction of the Neighborhood," *Data Mining and Knowledge Discovery*, vol. 22, pp. 106–148, 2011.

[6] D. M. Chickering, "Optimal Structure Identification with Greedy Search," *Journal of Machine Learning Research*, vol. 3, pp. 507–554, 2002.

[7] P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers, "Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 912–926, 1996.

[8] M. Teyssier and D. Koller, "Ordering-Based Search: A Simple and Effective Algorithm for Learning Bayesian Networks," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pp. 584–590, 2005.

[9] J. I. Alonso-Barba, L. delaOssa, and J. M. Puerta, "Structural Learning of Bayesian Networks Using Local Algorithms Based on the Space of Orderings," *Soft Computing*, vol. 15, no. 10, pp. 1881–1895, 2011.

[10] M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon, "Learning Bayesian Networks with Thousands of Variables," in *Advances in Neural Information Processing Systems*, pp. 1864–1872, 2015.

[11] M. Scanagatta, G. Corani, and M. Zaffalon, "Improved Local Search in Bayesian Networks Structure Learning," in *Proceedings of Machine Learning Research*, vol. 73, pp. 45–56, 2017.

[12] T. Silander and P. Myllymäki, "A Simple Approach for Finding the Globally Optimal Bayesian Network Structure," in *Proceedings of the 22nd Conference Annual Conference on Uncertainty in Artificial Intelligence*, pp. 445–452, 2006.

[13] M. Bartlett and J. Cussens, "Advances in Bayesian Network Learning using Integer Programming," in *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pp. 182–191, 2013.

[14] C. Lee and P. van Beek, "Metaheuristics for Score-and-Search Bayesian Network Structure Learning," in *Proceedings of the 30th Canadian Conference on Artificial Intelligence*, pp. 129–141, 2017.

[15] M. Champion, V. Picheny, and M. Vignes, "Inferring Large Graphs Using ℓ_1-Penalized Likelihood," *Statistics and Computing*, vol. 28, pp. 905–921, 2018.

[16] N. Friedman and D. Koller, "Being Bayesian about Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks," *Machine Learning*, vol. 50, pp. 95–125, 2003.

[17] B. Ellis and W. H. Wong, "Learning Causal Bayesian Network Structures from Experimental Data," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 778–789, 2008.

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

[19] A. Ghoshal and J. Honorio, "Learning Linear Structural Equation Models in Polynomial Time and Sample Complexity," in *Proceedings of Machine Learning Research*, vol. 84, pp. 1466–1475, 2018.

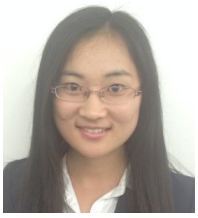
[20] G. F. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data," *Machine Learning*, vol. 9, pp. 309–347, 1992.

[21] D. M. Chickering, "Learning Bayesian Networks is NP-Complete," in *Learning from Data*, ser. Lecture Notes in Statistics. Springer, 1996.

[22] F. Fu and Q. Zhou, "Learning Sparse Causal Gaussian Networks with Experimental Intervention: Regularization and Coordinate Descent," *Journal of the American Statistical Association*, vol. 108, pp. 288–300, 2013.

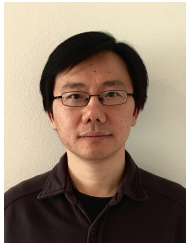
[23] B. Aragam and Q. Zhou, "Concave Penalized Estimation of Sparse Gaussian Bayesian Networks," *Journal of Machine Learning Research*, vol. 16, pp. 2273–2328, 2015.

[24] X. Zheng, B. Aragam, P. Pavikumar, and E. P. Xing, "Dags with NO TEARS: Continuous Optimization for Structure Learning," in *Advances in Neural Information Processing Systems*, 2018.
[25] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least Angle Regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[26] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. Springer-Verlag, New York, 1993.
[27] L. Vandenberghe and M. S. Andersen, "Chordal Graphs and Semidefinite Optimization," *Foundations and Trends in Optimization*, vol. 1, no. 4, pp. 241–433, 2014.
[28] C.-H. Zhang, "Nearly Unbiased Variable Selection under Minimax Concave Penalty," *The Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.
[29] Z. Chen and C. Leng, "Local Linear Estimation of Covariance Matrices via Cholesky Decomposition," *Statistica Sinica*, vol. 15, no. 1249-1263, 2015.
[30] K. Lee and J. Lee, "Estimating Large Precision Matrices via Modified Cholesky Decomposition," *Statistica Sinica Preprint No: SS-2018-0476*, 2018.
[31] S. Touchette, W. Gueaieb, and E. Lanteigne, "Efficient Cholesky Factor Recovery for Column Reordering in Simultaneous Localisation and Mapping," *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 859–875, 2016.
[32] N. Verzelen, "Adaptive Estimation of Covariance Matrices via Cholesky Decomposition." *Electronic Journal of Statistics*, vol. 4, pp. 1113–1150, 2010.
[33] Q. Li and X. S. Zhang, "Bayesian Estimation of Large Precision Matrix Based on Cholesky Decomposition," *Acta Mathematica Sinica, English Series*, vol. 35, pp. 619–631, 2019.
[34] M. Pourahmadi, "Covariance Estimation: The GLM and Regularization Perspectives," *Statistical Science*, vol. 26, pp. 369–387, 2011.
[35] J. Pearl, "Causal Diagrams for Empirical Research," *Biometrika*, vol. 82, pp. 669–710, 1995.
[36] J. Robins, "A New Approach to Causal Inference in Mortality Studies with a Sustained Exposure Period - Application to Control of the Healthy Worker Survivor Effect," *Mathematical Modelling*, vol. 7, pp. 1393–1512, 1986.
[37] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.
[38] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.
[39] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.
[40] M. Scutari. Bayesian network repository. Accessed: 2019-01-21. [Online]. Available: http://www.bnlearn.com/bnrepository/
[41] B. Aragam, J. Gu, and Q. Zhou, "Learning Large-Scaled Bayesian Networks with the `sparsebn` Package," *Journal of Statistical Software*, vol. 91, no. 11, pp. 1–38, 2019.
[42] J. D. Ramsey, "Scaling up Greedy Causal Search for Continuous Variables," *arXiv:1507.07749*, 2015.
[43] M. Scutari, "Learning Bayesian Networks with the bnlearn R Package," *Journal of Statistical Software*, vol. 35, no. 3, pp. 1–22, 2010.
[44] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse Inverse Covariance Estimation with the Graphical Lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.
[45] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An Approximate Minimum Degree Ordering Algorithm," *SIAM Journal on Matrix Analysis and Application*, vol. 17, no. 4, pp. 886–905, Dec. 1996.
[46] J. Cussens, M. J´arvisalo, J. H. Korhonen, and M. Bartlett, "Bayesian Network Structure Learning with Integer Programming: Polytopes, Facets and Complexity," *Journal of Artificial Intelligence Research*, vol. 58, pp. 185–229, 2017.
[47] A. Hauser and P. Bühlmann, "Characterization and Greedy Learning of Interventional Markov Equivalence Classes of Directed Acyclic Graphs," *The Journal of Machine Learning Research*, vol. 13, pp. 2409–2464, 2012.
[48] J. Gu, F. Fu, and Q. Zhou, "Penalized Estimation of Directed Acyclic Graphs from Discrete Data," *Statistics and Computing*, vol. 29, pp. 161–176, 2019.



Qiaoling Ye received her B.S. degree from UCLA Department of Mathematics in 2014. She has been working on causal graphical models for her Ph.D. in Statistics at UCLA since 2015.

Arash A. Amini received his Ph.D. in electrical engineering from University of California, Berkeley, in 2011. He is currently an Assistant Professor of Statistics at UCLA. His research interests lie in high-dimensional statistics, functional and nonparametric estimation, network data analysis, optimization and graphical models.



Qing Zhou received his Ph.D. in Statistics from Harvard University in 2006. He is currently Professor of Statistics at UCLA. His research interests include causal inference, graphical models, machine learning, high-dimensional statistics, Monte Carlo methods, and bioinformatics. He received an NSF Career Award in 2011.