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, which is of independent interest. We combine global simulated annealing over permutations 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 local 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 achieves substantial improvements over existing methods, demonstrating its great potential to learn Bayesian networks from both observational and experimental data.

Index Terms—Bayesian networks, proximal gradient, sparse Cholesky factorization, regularized likelihood, simulated annealing, 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], has been advocated as a promising way to perform global search over the ordering space.

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 [18], can be as high as $O(p^{k+1})$ for $p$ nodes and a prespecified 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 [19]. It is not surprising that the performance of the above global 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 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). Other than using SA solely as a global optimization algorithm, we may also incorporate informative initial orderings, learned from an existing local algorithm, by setting a low starting temperature. Our numerical results demonstrate that this 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 [20, 21]. These methods optimize a regularized likelihood score over the DAG space by blockwise coordinate descent, which is a local search algorithm in nature and thus likely to be trapped in a suboptimal structure. Using DAGs learned by these methods to generate initial orderings, our method can significantly improve the accuracy in structure learning. This highlights the advantages of combining local and global searches over the ordering space under an annealing framework. More recently, Champion et al. [15] developed a genetic algorithm that optimizes an objective function over the DAG space by blockwise coordinate descent. Other than using SA solely as an efficient first-order method, without enumerating possible parent sets for any node, searching over $P$ is done by simulated annealing (SA). Other than using SA solely as a global optimization algorithm, we may also incorporate informative initial orderings, learned from an existing local algorithm, by setting a low starting temperature. Our numerical results demonstrate that this 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 [20, 21]. These methods optimize a regularized likelihood score over the DAG space by blockwise coordinate descent, which is a local search algorithm in nature and thus likely to be trapped in a suboptimal structure. Using DAGs learned by these methods to generate initial orderings, our method can significantly improve the accuracy in structure learning. This highlights the advantages of combining local and global searches over the ordering space under an annealing framework. More recently, Champion et al. [15] developed a genetic algorithm that optimizes an objective function over the DAG space by blockwise coordinate descent. Other than using SA solely as an efficient first-order method, without enumerating possible parent sets for any node, searching over $P$ is done by simulated annealing (SA). Other than using SA solely as a global optimization algorithm, we may also incorporate informative initial orderings, learned from an existing local algorithm, by setting a low starting temperature. Our numerical results demonstrate that this 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.

The paper is organized as follows. Section 2 covers 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 $G$ 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 the likelihood score over the DAG space by blockwise coordinate descent. The SEMs in (2) define a joint Gaussian distribution, $\mathcal{N}(0, \Sigma_0)$, where $\beta_0$ specifies a weight associated with the edge $i \rightarrow j$ and $\beta_0 = 0$ for $i \notin \Pi_j^0$, and $2\Omega_0 := \text{diag}(\omega_0^2)$ as a noise variance matrix. The SEMs in (2) define a joint Gaussian distribution, $X \sim \mathcal{N}(0, \Sigma_0)$, where $\Sigma_0$ is positive definite and given by

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

2.2 Acyclicity and permutations

The support of $B_0$ in (3) defines the structure of $G$, and thus it must satisfy the acyclicity constraint so that $G$ 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 $\pi$ on the set $[p] := \{1, \ldots, p\}$, we associate a permutation matrix $P_\pi$ whose $i^{th}$ row is $e_{\pi(i)}$. For a vector $v = (v_1, \ldots, v_p)^\top$, we have

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

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

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

The paper is organized as follows. Section 2 covers 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 $G$ 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 the likelihood score over the DAG space by blockwise coordinate descent. The SEMs in (2) define a joint Gaussian distribution, $\mathcal{N}(0, \Sigma_0)$, where $\beta_0$ specifies a weight associated with the edge $i \rightarrow j$ and $\beta_0 = 0$ for $i \notin \Pi_j^0$, and $2\Omega_0 := \text{diag}(\omega_0^2)$ as a noise variance matrix. The SEMs in (2) define a joint Gaussian distribution, $X \sim \mathcal{N}(0, \Sigma_0)$, where $\Sigma_0$ is positive definite and given by

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

2.2 Acyclicity and permutations

The support of $B_0$ in (3) defines the structure of $G$, and thus it must satisfy the acyclicity constraint so that $G$ 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 $\pi$ on the set $[p] := \{1, \ldots, p\}$, we associate a permutation matrix $P_\pi$ whose $i^{th}$ row is $e_{\pi(i)}$. For a vector $v = (v_1, \ldots, v_p)^\top$, we have

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

that is, $P_\pi$ permutes the entries of $v$ according to $\pi$. Since $P_\pi^\top P_\pi = I_p$, 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 $\pi$. Then, $B_\pi$ will be a strictly lower triangular matrix if and only if $\pi$ is the reversal of a topological sort of $G$, i.e., $i < j$ in $\pi$ for $j \in \Pi^T_k$. See Figure 1 for an illustration. Under this reparameterization, we translate the acyclicity constraint on $B_0$ into the constraint that $B_\pi$ must be strictly lower triangular for some permutation $\pi$. Define $\Omega_\pi := P_\pi \Omega_0 P_\pi^\top$. Equivalently, one may think of the node $\pi(i)$ relabeled as $i$ in $B_\pi$ and $\Omega_\pi$.

For simplicity, we drop the subscript $\pi$ from $P_\pi$, $B_\pi$ and $\Omega_\pi$ if no confusion arises. Therefore, throughout the paper, $P$ defines a permutation $B$, and $\Omega$ label nodes according to $\pi$ and we write the permuted SEM as

$$PX = B^\top P X + P\varepsilon.$$  

(6)

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

### 3 Regularized likelihood score

In this section, we formulate the objective function to estimate BN structure given data from the Gaussian SEM (2).

#### 3.1 Cholesky loss

Let $X := [X_1, \ldots, X_p] \in \mathbb{R}^{n \times p}$ 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:

$$XP^\top = XP^\top B + EP^\top,$$

(7)

where each row of $E \in \mathbb{R}^{n \times p}$ is an i.i.d. error vector from $N(0, \Omega_0)$. In (7), $XP^\top$ and $EP^\top$ are $X$ and $E$ with columns permuted according to $P = P_\pi$. It then follows that each row of $XP^\top$ is an i.i.d. observation from $N(0, \Sigma)$ with $\Sigma^{-1} = (I - B)\Omega^{-\frac{1}{2}}(I - B)^\top$, and thus the negative log-likelihood of (7) is

$$\ell(B, \Omega, P \mid X)$$

$$= \frac{1}{2} \text{tr} \left[ (XP^\top B)^\top (I - B)\Omega^{-\frac{1}{2}}(I - B) \right] + \frac{n}{2} \log |\Omega|. $$

(8)

Recall that $B$ and $\Omega = \text{diag}(\omega_j^2)$ are defined by permuting the rows and columns of $B_0$ and $\Omega_0$ 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 $L_j = (\epsilon_j - \beta_j)/\omega_j$ is a weighted coefficient vector for node $\pi(j)$. We define what we call the Cholesky loss function

$$\mathcal{L}_{\text{cho}}(L; A) := \frac{1}{2} \text{tr} \left[ (ALL^\top)^\top \right] - \log |L|,$$

(9)

where $|L|$ denotes the determinant of $L$. Noting that $|L| = |(I - B)\Omega^{-\frac{1}{2}}| = |\Omega|^{-\frac{n}{2}}$ and denoting by $\hat{\Sigma} := \frac{1}{n}X^\top X$ 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

$$\ell(L, P) = n \cdot \mathcal{L}_{\text{cho}}(L; P\hat{\Sigma}P^\top)$$

$$= \frac{n}{2} \text{tr} \left[ (P\hat{\Sigma}P^\top LL^\top)^\top \right] - n \log |L|,$$

(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 $C(M)$ be its unique Cholesky factor, i.e., the unique lower triangular matrix $L$ with positive diagonal entries such that $M = LL^\top$.

**Proposition 1.** For any positive definite matrix $A \in \mathbb{R}^{p \times p}$, we have

$$\arg \min_{L \in \mathcal{L}_p} \mathcal{L}_{\text{cho}}(L; A) = \{C(A^{-1})\}$$

with optimal value

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

**Consequently,** $\mathcal{L}_{\text{cho}}^*(A) = \mathcal{L}_{\text{cho}}(PAP^\top)$ for any permutation matrix $P$.

The proof is given in Appendix A.1. Proposition 1 states that $C(A^{-1})$ is the unique minimizer of $\mathcal{L}_{\text{cho}}(\cdot; A)$.

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 $\ell^*$ be the optimal value of this problem, i.e.,

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

Then, Proposition 1 implies

$$\ell^*(P) = n \cdot \mathcal{L}_{\text{cho}}(P\hat{\Sigma}P^\top) = n \cdot \mathcal{L}_{\text{cho}}^*(\hat{\Sigma}),$$

(11)

showing that $\ell^*$ 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.

#### 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 [23], the true DAG \( G \) in (2) and its equivalence class are the sparsest among all DAGs that can parameterize the joint distribution \( P(0, \Sigma_0) \). 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 \( f(L; P) \) over \( L \) is the Cholesky factor of \((P\Sigma P^\top)^{-1} = P\Sigma^{-1} P^\top\). For a sparse \( \Sigma^{-1} \), 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 [24].

From a statistical perspective, however, \( \Sigma^{-1} \) is, in general, not sparse (due to noise) even if the inverse of population covariance matrix \( \Sigma = \mathbb{E}[\Sigma] \) 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 loss function. The right derivative of MCP at zero is

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

where \( \lambda \geq 0 \) and \( \gamma > 1 \). Parameter \( \lambda \) measures the penalty level, while \( \gamma \) controls the concavity of the function. For a fixed value of \( \lambda \), the MCP approaches the \( \ell_1 \) penalty as \( \gamma \to \infty \), and the \( \ell_0 \) penalty as \( \gamma \to 0^+ \).

Figure 2 compares the MCP with \((\gamma, \lambda) = (2, 1)\) and the \( \ell_1 \) penalty. The right derivative of MCP at zero is \( \lambda \), which is the same as the derivative of the \( \ell_1 \) penalty. The MCP function flattens out when \( |x| \geq \gamma \lambda \).

**Remark 1.** Aragam and Zhou [21] 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.

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

![Fig. 2: A comparison between the MCP (solid line) and the \( \ell_1 \) penalty (dashed line).](image)
where \( x_{kj} \) is the value of the \( j \)th variable in the \( h \)th observation and \( p_{a_{hj}} \) 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

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

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

\[
\ell_\mathcal{O}(L, P) = \sum_{j=1}^p |\mathcal{O}_{\pi(j)}| |\mathcal{L}| \log \left( L_j^{-1} P \hat{\Sigma}^j P^\top \right),
\]

where \( L_j = (\epsilon_j - \beta_j)/\omega_j \in \mathbb{R}^p, L = (L_j) \in \mathcal{L}_p \), and \( |L_j| := L_{jj} \in \mathcal{L}_p \). (9)

See Appendix A.2 for the proof. 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 the next section can be applied on both observational and experimental data.

### 4 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 for an ordering that minimizes 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).

#### 4.1 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 \( \alpha \) given in line 7. The probability is determined by the difference between the proposed and current scores \( f(P^*) - f(P) \) 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 \( \hat{\pi} \) and \( \pi^\star \) be the permutations associated with \( \hat{P} \) and \( P^* \) as in (5). We propose \( \pi^\star \) by flipping (i.e., reversing the order of) a random interval of length \( m \) in the current permutation \( \hat{\pi} \). For example, with \( m = 3 \) we may flip \( \hat{\pi} = (1, 2, 3, 4, \ldots, p) \) to \( \pi^\star = (1, 4, 3, 2, \ldots, p) \) in the proposal.

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).

#### 4.2 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 [28].

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

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

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 \( \hat{L} \) and optimize the resulting approximation to \( f(u; P) \) to get a new estimate \( L^+ \):

\[
L^+ = \arg \min_{u \in \mathcal{L}_p} \ell(L) + \nabla \ell(L)^\top (u - L) + \frac{1}{2\lambda} ||u - L||^2 + \rho(u)
\]

\begin{align}
& = \arg \min_{u \in \mathcal{L}_p} \frac{1}{2\lambda} ||L - \nabla \ell(L) - u||^2 + \rho(u),
\end{align}

Algorithm 1: Annealing on regularized Cholesky score (ARCS).

**Input:** Dataset \( \mathbf{X} \), initial permutation matrix \( P_0 \), a temperature schedule \( \{T^{(i)}, i = 0, \ldots, N\} \), constant \( m \).

**Output:** Adjacency matrix \( \hat{B} \).

1: Select tuning parameters \((\gamma, \lambda)\) for \( f(L; P) \) according to Algorithm 4 (Section 4.4).
2: \( \hat{P} \leftarrow P_0, \hat{L} \leftarrow \arg \min_{L \in \mathcal{L}_p} f(L; \hat{P}) \) by Algorithm 2, \( f(\hat{P}) \leftarrow f(\hat{L}, \hat{P}) \).
3: for \( i = 0, \ldots, N \) do
4: \( T \leftarrow T^{(i)} \).
5: Propose \( P^* \) by flipping a random length-\( m \) interval in the permutation defined by \( \hat{P} \).
6: \( L^* \leftarrow \arg \min_{L \in \mathcal{L}_p} f(L; P^*) \) using Algorithm 2,
7: \( f(P^*) \leftarrow f(L^*; P^*) \).
8: \( \alpha \leftarrow \min \{1, \exp \left(-\frac{1}{T} [f(P^*) - f(\hat{P})] \right) \} \).
9: Set \( (\hat{P}, \hat{L}, f(\hat{P})) \leftarrow (P^*, L^*, f(P^*)) \) with prob. \( \alpha \).
10: Refine adjacency matrix \( \hat{B} \) given \( (\hat{P}, \hat{L}) \) by Algorithm 3 (Section 4.3).

**Algorithm 2** Generating new permutation

**Input:** Target matrix \( \hat{B} \), current permutation \( \hat{P} \), and \( \pi^\star \), an initial permutation.

1: \( \hat{P} \leftarrow \hat{P} \).
2: \( \hat{P} \leftarrow \hat{P} \).
3: for \( \alpha \leftarrow 0 \) do
4: \( \alpha \leftarrow \alpha + \delta \alpha \).
5: Propose \( \pi^\star \) by flipping a random length-\( m \) interval in the current permutation defined by \( \hat{P} \).
6: \( \hat{L} \leftarrow \hat{L} \).
7: Set \( \hat{L} \leftarrow \hat{L} \).
8: for \( i = 1 \) to \( N \) do
9: \( L \leftarrow L \).
10: \( L \leftarrow L \).
11: if \( L \leftarrow L \) then 
12: \( L \leftarrow L \).
13: \( L \leftarrow L \).
14: end if
15: end for
16: Refine adjacency matrix \( \hat{B} \) given \( (\hat{P}, \hat{L}) \) by Algorithm 3 (Section 4.3).

**Algorithm 3** Refinement of adjacency matrix

**Input:** \( \hat{B} \), \( \hat{P} \), \( \pi^\star \), an initial permutation.

1: \( \hat{B} \leftarrow \hat{B} \).
2: \( \hat{B} \leftarrow \hat{B} \).
3: for \( j = 1 \) to \( p \) do
4: \( j \leftarrow j \).
5: \( j \leftarrow j \).
6: if \( j \leftarrow j \) then 
7: \( j \leftarrow j \).
8: \( j \leftarrow j \).
9: end if
10: end for
11: Refine adjacency matrix \( \hat{B} \) given \( (\hat{P}, \pi^\star) \) by Algorithm 3 (Section 4.3).
Algorithm 2 Compute the RC score by proximal gradient.

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

Output: $L$.

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

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

$$\text{prox}_\rho(x) := \arg\min_{u \in \mathbb{L}_p} \left( \rho(u) + \frac{1}{2}\|x-u\|^2 \right),$$

where $x \in \mathbb{L}_p$ and $\|\cdot\|$ is the usual Euclidean norm. Then, (19) is equivalent to

$$L^+ = \text{prox}_{\rho}(L - t \nabla \ell(L)), \quad (20)$$

where $\text{prox}_{\rho}(\cdot)$ 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 $x \in \mathbb{L}_p$,

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

The proximal operators on the RHS are univariate, and the univariate $\text{prox}_{\rho}(\cdot)$, both of which have nice closed-form expressions:

**Lemma 3.** The gradient of $\ell(L,P)$ in (10) w.r.t. $L$ is

$$\nabla \ell(L) = n \left( P L P^T - \text{diag} \left\{ L_{ii} \right\} \right),$$

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

**Lemma 4.** The gradient of $\ell(O)(L,P)$ in (17) w.r.t. $L_j$ is

$$\nabla_{L_j} \ell(O)(L,P) = \left| O_{\pi(j)} \right| \left( \Pi_j \left( P \hat{S}^\top P^\top L_j \right) - e_j^L \right)_{L_{jj}}.$$
Algorithm 3 Constraint-based structure refinement.

Input: Dataset $X$, permutation $\widehat{\pi}$, adjacency matrix $\widehat{B}$, significance level $\alpha$.
Output: Adjacency matrix $B$.

1: $Z_0 \leftarrow \Phi^{-1}(1 - \frac{\alpha}{2})$, where $\Phi(x)$ is the CDF of $N(0,1)$.
2: for $j = 1, \ldots, p$ do
3:   $\widehat{p}_{\text{ind}} \leftarrow \{i : \hat{B}_{ij} \neq 0\}$.
4: for $k \in \widehat{p}_{\text{ind}}$ do
5:   $s \leftarrow \widehat{p}_{\text{ind}} \setminus \{k\}$.
6:   $X^j \leftarrow$ observations for which $j$ is not intervened
7:   $n \leftarrow$ number of rows in $X^j$.
8:   $r_{j,k} \leftarrow$ sample partial correlation between $X_j$ and $X_k$ given $X_s$ based on $X^j$.
9:   $z \leftarrow \frac{1}{\sqrt{n}} \left| r_{j,k} \right| - 3 \log \left( \frac{1}{1 + r_{j,k}^2} \right)$.
10: Remove $k$ from $\widehat{p}_{\text{ind}}$, if $|z| < Z_0$.
11: end for
12: $\widehat{B}_{ij} \leftarrow 1$ if $i \in \widehat{p}_{\text{ind}}$ and $\widehat{B}_{ij} \leftarrow 0$ otherwise.
13: 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.

1: Define $\bar{L}(\theta) := \arg\min_{L \in L_\theta} f_\theta(L; P_0)$ computed by Algorithm 2.
2: Let $\text{BIC}(\theta) := 2\ell(\bar{L}(\theta); P_0) + \|\bar{L}(\theta)\|_0 \log(\max\{n, p\})$.
3: Output $i^* = \arg\min_{i} \text{BIC}(\theta(i))$.

We use the Bayesian information criterion (BIC) [29] 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 $\|\bar{L}(\cdot)\|_0$ is the number of nonzero entries in $\bar{L}(\cdot)$, and then we output the one with the lowest BIC score. The regularization parameter in BIC($\theta$) 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 $\gamma$ 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 $\gamma \geq 100$ is essentially the same as the $\ell_1$ penalty. For the regularization parameter $\lambda$, 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.

5 Results

5.1 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 [30]. We duplicated some of them to further increase the network size. Synthetic DAG structures were constructed using the sparsebn package [31]. Given a DAG structure, we sampled the edge coefficients $\beta_{ij}$ 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 Heparfinder (224, 264), 1 copy of Andes (223, 338), 2 copies of Pigs (441, 592), and random DAGs, rDAG1 (300, 300) and rDAG2 (300, 600).

In the observational data setting, we compared our algorithm with the following BN learning algorithms: the coordinate descent (CD) algorithm [21], 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 $(\beta_{ij}, \beta_{ji})$ 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 [22].

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, and used the following R packages for other methods: sparsebn [31] for the CD algorithm, rcausal [32] for
the GES, GIES (for experimental data) and PC algorithms, bnlearn [33] for the MMHC and HC algorithms, and GADAG [15] for the GA.

5.2 Accuracy metrics

Among all methods applied on 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. Define \( P \), TP, FP, M, R as the numbers of estimated edges, true positive edges, false positive edges, missing edges and reverse edges, respectively. To standardize the performance metrics in observational data setting, we consider both directed and undirected edges in the definitions of these metrics as follows.

\[
P = \text{the number of edges in the estimated graph.}
\]

\[
\text{FP} = \text{the number of edges in the true skeleton but not in the estimated skeleton.}
\]

\[
\text{M} = \text{the number of edges in the true skeleton but not in the estimated skeleton.}
\]

\[
\text{R} = \text{the number of reversed edges.}
\]

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

5.3 Comparison on observational data

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 \( \lambda \) 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 \( 10^4 \) iterations, with initial temperature \( T^{(0)} = 1 \) and reversal length \( m = 4 \) (Algorithm 1). 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 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 \( 10^{-5} \).

ARCS versus CD and GES. Table 1 reports the average performance metrics across 20 datasets for 7 networks (5 real and 2 random networks) in the high-dimensional setting \( n < p \) 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 very substantial, close to or above 40%, for many networks, such as Win95pts, Pigs1, rDAG1 and rDAG2.

The same comparison was done for low-dimensional settings with \( n \geq p \), reported in Table 2. We observe similar improvements of ARCS(CD) and ARCS(GES) over CD and GES, consistent with the results for high-dimensional data. It is seen from both tables that ARCS always increased TP, while maintaining or slightly reducing FP. The annealing process often identified more TP edges, while the refinement

### Table 1: Comparison between ARCS and initial estimates on observational data \((n < p)\).

| network    | method              | P   | TP  | R   | FP  | SHD | JI  |
|------------|---------------------|-----|-----|-----|-----|-----|-----|
| Hailfinder | ARCS(CD)            | 270 | 154 | 84  | 32  | 142 | 0.41|
| Hailfinder | ARCS(GES)           | 274 | 187 | 54  | 33  | 110 | 0.53|
| Hailfinder | CD                  | 243 | 184 | 49  | 10  | 89  | 0.57|
| Hailfinder | GES                 | 261 | 215 | 30  | 16  | 65  | 0.70|
| Andes      | ARCS(CD)            | 352 | 176 | 108 | 68  | 230 | 0.34|
| Andes      | ARCS(GES)           | 368 | 233 | 71  | 65  | 170 | 0.50|
| Andes      | CD                  | 272 | 220 | 35  | 17  | 135 | 0.57|
| Andes      | GES                 | 334 | 276 | 31  | 27  | 89  | 0.70|
| Hepar2     | ARCS(CD)            | 495 | 245 | 120 | 130 | 377 | 0.33|
| Hepar2     | ARCS(GES)           | 500 | 311 | 107 | 82  | 264 | 0.46|
| Hepar2     | CD                  | 410 | 253 | 96  | 62  | 302 | 0.39|
| Hepar2     | GES                 | 483 | 324 | 96  | 64  | 232 | 0.50|
| Win95pts   | ARCS(CD)            | 399 | 195 | 154 | 50  | 302 | 0.30|
| Win95pts   | ARCS(GES)           | 465 | 317 | 88  | 59  | 190 | 0.53|
| Win95pts   | CD                  | 335 | 244 | 71  | 21  | 225 | 0.45|
| Win95pts   | GES                 | 415 | 361 | 57  | 34  | 121 | 0.67|
| Pigs1      | ARCS(CD)            | 715 | 342 | 207 | 166 | 417 | 0.36|
| Pigs1      | ARCS(GES)           | 587 | 421 | 123 | 43  | 214 | 0.55|
| Pigs1      | CD                  | 581 | 431 | 112 | 38  | 199 | 0.58|
| Pigs1      | GES                 | 579 | 454 | 94  | 27  | 165 | 0.63|
| rDAG1      | ARCS(CD)            | 319 | 208 | 85  | 25  | 117 | 0.51|
| rDAG1      | ARCS(GES)           | 305 | 260 | 36  | 9   | 49  | 0.76|
| rDAG1      | CD                  | 283 | 268 | 14  | 1   | 33  | 0.85|
| rDAG1      | GES                 | 297 | 283 | 13  | 1   | 18  | 0.90|
| rDAG2      | ARCS(CD)            | 715 | 407 | 136 | 171 | 364 | 0.50|
| rDAG2      | ARCS(GES)           | 623 | 498 | 73  | 52  | 154 | 0.75|
| rDAG2      | CD                  | 505 | 474 | 26  | 6   | 132 | 0.75|
| rDAG2      | GES                 | 594 | 564 | 18  | 13  | 49  | 0.89|

In this and all subsequent tables, reported results are rounded averages over 20 datasets; the best SHD and JI for each network are highlighted in boldface.

1. Results for Pigs1 are averaged over 17 datasets because CD estimates were too dense using the default model selection criterion for the other three datasets.
Figure 1: A comparison of CD, ARCS(CD), GES and ARCS(GES) on observational data (n > p).

Table 2: Comparison between ARCS and initial estimates on observational data (n > p).

| Network  | Method  | P  | TP  | R  | FP  | SHD | JI  |
|----------|---------|----|-----|----|-----|-----|-----|
| Andes    | CD      | 278| 156 | 88 | 34  | 142 | 0.41|
|          | ARCS(CD)| 296| 196 | 59 | 41  | 109 | 0.54|
|          | GES     | 272| 201 | 53 | 18  | 81  | 0.60|
|          | ARCS(GES)|273| 232 | 63 | 15  | 47  | 0.76|
| Hailfinder| CD     | 366| 189 | 107| 70  | 220 | 0.37|
|          | ARCS(CD)| 376| 242 | 69 | 66  | 162 | 0.53|
|          | GES     | 342| 273 | 33 | 36  | 102 | 0.67|
|          | ARCS(GES)|348| 296 | 26 | 26  | 69  | 0.76|
| Win95pts | CD      | 519| 267 | 130| 122 | 347 | 0.36|
|          | ARCS(CD)| 523| 330 | 106| 87  | 248 | 0.49|
|          | GES     | 509| 312 | 113| 84  | 264 | 0.45|
|          | ARCS(GES)|505| 336 | 95 | 75  | 231 | 0.51|
| Pigs     | CD      | 755| 353 | 224| 179 | 418 | 0.36|
|          | ARCS(CD)| 631| 451 | 123| 57  | 198 | 0.59|
|          | GES     | 472| 332 | 76 | 64  | 180 | 0.56|
|          | ARCS(GES)|475| 390 | 47 | 38  | 96  | 0.73|
| rDAG1    | CD      | 319| 211 | 84 | 24  | 113 | 0.52|
|          | ARCS(CD)| 319| 264 | 36 | 20  | 56  | 0.74|
|          | GES     | 299| 283 | 14 | 2   | 19  | 0.90|
|          | ARCS(GES)|300| 287 | 12 | 1   | 14  | 0.92|
| rDAG2    | CD      | 605| 403 | 140| 62  | 258 | 0.50|
|          | ARCS(CD)| 696| 510 | 79 | 106 | 196 | 0.65|
|          | GES     | 596| 559 | 20 | 17  | 57  | 0.88|
|          | ARCS(GES)|608| 583 |14  |11   | 28  | 0.93|

Figure 3: Boxplots of SHDs for CD, ARCS(CD), GES and ARCS(GES) on observational data from two networks.

Table 3: ARCS against others on observational data (n < p).

| Network  | Method  | P  | TP  | R  | FP  | SHD | JI  |
|----------|---------|----|-----|----|-----|-----|-----|
| Andes    | ARCS(GES)|261| 215 | 30 | 16  | 65  | 0.70|
|          | HC      | 414| 120 |116 |179  | 324 | 0.21|
|          | PC      | 227| 127 | 83 | 17  | 154 | 0.35|
|          | GA      | 193| 68  | 54 | 71  | 266 | 0.18|
| Hailfinder| ARCS(GES)|334| 276 | 31 | 27  | 89  | 0.70|
|          | HC      | 422| 140 |125 |158  | 356 | 0.23|
|          | MMHC    | 257| 146 | 98 | 13  | 205 | 0.33|
|          | PC      | 276| 120 |137 |18  | 236 | 0.24|
|          | GA      | 214| 77  | 67 | 70  | 331 | 0.16|
| Win95pts | ARCS(GES)|483| 324 | 96 | 64  | 232 | 0.50|
|          | HC      | 540| 153 |175 |212  | 551 | 0.17|
|          | PC      | 310| 108 |149 |53   | 436 | 0.16|
|          | MMHC    | 269| 108 |135 |26  | 410 | 0.17|
|          | GA      | 434| 103 | 85 | 246 | 636 | 0.10|
| Pigs     | ARCS(GES)|575| 454 | 94 | 27  | 165 | 0.63|
|          | HC      | 856| 367 |168 |322  | 546 | 0.34|
|          | GA      | 437| 135 |127 |175  | 632 | 0.15|
| rDAG1    | ARCS(GES)|297| 283 | 13 | 1   | 18  | 0.90|
|          | HC      | 562| 135 |155 |273  | 438 | 0.19|
|          | MMHC    | 316| 143 |144 |28   | 188 | 0.30|
|          | PC      | 328| 166 |126 |36   | 170 | 0.36|
|          | GA      | 175| 89  | 56 | 30  | 242 | 0.23|
| rDAG2    | ARCS(GES)|594| 564 | 18 | 13  | 49  | 0.89|
|          | HC      | 577| 277 |170 |131  | 454 | 0.31|
|          | MMHC    | 435| 307 |143 |5  | 298 | 0.41|
|          | PC      | 512| 162 |341 |9    | 446 | 0.17|
|          | GA      | 313| 141 |106 |67   | 526 | 0.18|

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

HC, PC, MMHC and GA. Table 3 summarizes the average performance on high-dimensional observational data. Among all the methods, ARCS(GES) achieved the lowest SHD and the highest JI for all networks. 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.

Table 4 summarizes the results for the low-dimensional setup, n > p. It confirms that ARCS(GES) outperforms competing algorithms by a great margin. In general, HC had a larger P and GA had a smaller P than the true DAG, which indicates that their estimates were either too dense or too sparse. That the GA estimates were too sparse is more pronounced in this setting compared to the high-dimensional case. This might be related to its tuning parameter choice.
TABLE 4: ARCS against others on observational data (n > p).

| network (p, q, n) | method    | P  | TP | R  | FP | SHD | JI  |
|------------------|-----------|----|----|----|----|-----|-----|
| Hallfinder (224, 264, 400) | ARCS(GES) | 273 | 232 | 26 | 15 | 47 | 0.76 |
|                   | HC        | 409 | 118 | 121 | 170 | 317 | 0.21 |
|                   | PC        | 256 | 167 | 70  | 18  | 115 | 0.48 |
|                   | GA        | 138 | 63  | 47  | 27  | 228 | 0.19 |
| Andes (223, 338, 400) | ARCS(GES) | 348 | 296 | 26  | 26  | 69 | 0.76 |
|                   | HC        | 419 | 145 | 125 | 148 | 341 | 0.24 |
|                   | MMHC      | 286 | 169 | 104 | 14  | 184 | 0.37 |
|                   | PC        | 308 | 154 | 135 | 19  | 203 | 0.31 |
|                   | GA        | 153 | 64  | 56  | 33  | 307 | 0.15 |
| Hepar2 (280, 492, 400) | ARCS(GES) | 505 | 336 | 95  | 75  | 251 | 0.51 |
|                   | HC        | 535 | 152 | 180 | 203 | 544 | 0.17 |
|                   | PC        | 535 | 125 | 180 | 49  | 416 | 0.17 |
|                   | MMHC      | 317 | 137 | 152 | 28  | 384 | 0.20 |
|                   | GA        | 416 | 114 | 84  | 218 | 597 | 0.10 |
| Win95pts (304, 448, 500) | ARCS(GES) | 475 | 390 | 47  | 38  | 96 | 0.73 |
|                   | HC        | 576 | 140 | 223 | 213 | 522 | 0.16 |
|                   | MMHC      | 394 | 164 | 208 | 23  | 307 | 0.24 |
|                   | PC        | 433 | 153 | 247 | 33  | 328 | 0.21 |
|                   | GA        | 138 | 71  | 56  | 11  | 389 | 0.14 |
| Pigs (441, 592, 600)  | ARCS(GES) | 619 | 471 | 102 | 46  | 166 | 0.64 |
|                   | HC        | 843 | 377 | 164 | 302 | 517 | 0.36 |
|                   | GA        | 218 | 112 | 83  | 23  | 503 | 0.16 |
| rDAG1 (300, 300, 450) | ARCS(GES) | 300 | 287 | 12  | 1   | 14 | 0.92 |
|                   | HC        | 549 | 186 | 105 | 247 | 371 | 0.28 |
|                   | MMHC      | 324 | 209 | 85  | 29  | 120 | 0.51 |
|                   | PC        | 335 | 210 | 88  | 37  | 127 | 0.50 |
|                   | GA        | 135 | 81  | 47  | 7   | 226 | 0.20 |
| rDAG2 (300, 600, 600) | ARCS(GES) | 608 | 583 | 14  | 11  | 28 | 0.93 |
|                   | HC        | 577 | 274 | 177 | 126 | 452 | 0.30 |
|                   | MMHC      | 574 | 314 | 165 | 5   | 291 | 0.41 |
|                   | PC        | 574 | 268 | 298 | 7   | 339 | 0.30 |
|                   | GA        | 144 | 88  | 51  | 5   | 516 | 0.13 |

It is worth mentioning that ARCS(GES) outperformed other methods substantially for larger networks such as Pigs with p = 441 and q0 = 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.

We also tested another order-based algorithm, linear structural equation model learning (LISTEN) [34], 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 14% to 18% of the SHD achieved by LISTEN. Full results are reported in the supplementary material.

5.4 Comparison 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 ≤ 50 and q0 ≤ 100, including real DAGs (p, q0): Asia (8, 8), Sachs (11, 17), Ins. (27, 52), Alarm (37, 46), and Barley (48, 84), and random DAGs: rDAG3 (20, 20), rDAG4 (20, 40), rDAG5 (50, 50), and rDAG6 (50, 100). Using these networks, we also simulated observational data of the same sample size, n = 5p, to study the effect of experimental interventions. Since networks in the experimental setting were smaller, we used a p-value cutoff of 10^-3 in the refinement step of the ARCS algorithm (Algorithm 3).

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 definition of R is different from that for observational data, because under the intervention setting used in this comparison, the true causal DAG is identifiable [20]. The structural Hamming distance (SHD) and the Jaccard index (JI) are then calculated as in the observational case.

In this setting, we compared ARCS with the CD algorithm [20] and the greedy interventional equivalence search (GES) algorithm [35], 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 each type of networks. It is seen that 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 and 2), ARCS(CD) usually had better performance than ARCS(GIES) on experimental data.

**Experimental versus observational.** We also compared the performance of our method ARCS(CD) on experimental and observational data with the same sample size n = 5p. Figure 4 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, 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 4 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, q0 = 8), ARCS(CD) output 16 estimated DAGs with P = 8 and R = 1, resulting in a very thin
TABLE 5: Performance comparison on experimental data.

| network     | method      | P  | TP | R  | FP | SHD | JI  |
|-------------|-------------|----|----|----|----|-----|-----|
| Asia (8,8,40) | CD          | 14 | 6  | 1  | 7  | 8   | 0.47|
|             | 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 (11,17,55) | CD          | 23 | 9  | 5  | 10 | 17  | 0.34|
|             | 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. (27,52,135) | CD         | 56 | 30 | 12 | 14 | 36  | 0.39|
|             | 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 (37,46,185) | CD         | 51 | 35 | 9  | 7  | 16  | 0.57|
|             | 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 (48,84,240) | CD         | 85 | 52 | 17 | 16 | 48  | 0.45|
|             | 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 (20,20,100) | CD          | 25 | 15 | 5  | 6  | 11  | 0.49|
|             | 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 (20,40,100) | CD          | 43 | 25 | 8  | 10 | 24  | 0.45|
|             | 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|
| rDAGS (50,50,250) | CD          | 55 | 39 | 10 | 6  | 17  | 0.60|
|             | 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 (50,100,250) | CD          | 101| 70 | 17 | 13 | 43  | 0.54|
|             | 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|

interquantile range in the boxplots of Asia in Figure 4.

5.5 Effectiveness of BIC selection

Given an initial permutation, we use the BIC to choose tuning parameters \((\gamma, \lambda)\) before applying the ARCS algorithm (Section 4.4). In Tables 3 and 4, the number of predicted edges 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 \((\gamma, \lambda)\) by ARCS(CD) and ARCS(GES).

Figure 5 shows the histograms of the SHDs for all values of \((\gamma, \lambda)\) on Andes datasets with \((p, s_0) = (223, 338)\) and \(n \in \{200, 400\}\). The grey area in a histogram corresponds to values of \((\gamma, \lambda)\) that led to a lower SHD than the \((\gamma^*, \lambda^*)\) chosen by the BIC, i.e., it indicates the percentile of the SHD of the BIC selection among all choices of the tuning parameters. The smaller this percentile, the better the BIC selection performance.

Each histogram has a high spike of large SHD, corresponding to large values of \(\lambda\) that generate almost empty graphs. The SHDs of ARCS(CD) with BIC selection corresponded to the 14th and 16th percentiles in low and high dimensions, respectively, and for ARCS(GES) the 16th and 3rd percentiles. 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 \(\gamma\). Since for small \(\gamma^*\) the MCP is closer to the \(\ell_1\) penalty and far from the \(\ell_1\) norm, this choice of \(\gamma^*\) indicates the preference of concave penalties over \(\ell_1\) 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 3 and 4), 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.

5.6 Random initialization with a high temperature

Recall that we started ARCS(CD) and ARCS(GES) 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 nine networks. We chose ARCS(CD) due to its superior performance on these networks in the experimental setting (Table 5). As shown in Figure 6, ARCS(RND) and ARCS(CD) had comparable performances on all net-
works. 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, 2 and 5. This study suggests that, by combining effective local and global searches under a regularized likelihood framework, our ARCS algorithm is a promising approach to the challenging problem of DAG structure learning.

6 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. 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 [36] or develop a continuous regularizer for multinomial likelihood.

For the simulated annealing step, there are several interesting aspects to investigate. 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. This paper focuses on numerical results, and shows the advantage and potential application of local search and global annealing in learning BNs. Left as future work are theoretical properties of our method, such as the consistency of the regularized Cholesky score and the convergence of ARCS with a good initial permutation.

APPENDIX A

A.1 Proof of Proposition 1

Since $A$ is positive definite, $A^{-1}$ is positive definite as well. Let us apply the Cholesky decomposition on $A^{-1}$, i.e., $A^{-1} = C C^\top$ where $C = L(A^{-1})$. Therefore, $A = C^{-\top} C^{-1}$.

Recall that $L_p$ is the set of lower triangular matrices with positive diagonals. The goal is to show that the minimizer of $\mathcal{L}_{\text{chal}}(L; A)$ over $L_p$, denoted by $L^*$, equals $C$.

Letting $R = C^{-\top} L$, we have $\log |R| = - \log |C| + \log |L|$, where $\log |C|$ is a constant. Since $C \in L_p$ and $L \in L_p$, we have $R \in L_p$ as well. Also,

$$\text{tr}(ALL^\top) = \text{tr}(C^{-\top} C^{-1} LL^\top) = \text{tr}(RR^\top) = ||R||_F^2.$$  

Then,

$$\min_{L \in L_p} \mathcal{L}_{\text{chal}}(L; A) = \min_{L \in L_p} \left[ \frac{1}{2} \text{tr}(ALL^\top) - \log |L| \right]$$

$$= \min_{R \in L_p} \frac{1}{2} \left[ ||R||_F^2 - 2 \log |R| \right] - \log |C|$$

$$= \min_{R \in L_p} \frac{1}{2} \left[ \sum_{(i,j) \succ j} R_{ij}^2 + \frac{p}{p} \left( R_{ii}^2 - 2 \log R_{ii} \right) \right] - \log |C|.$$
The problem is separable over entries. Minimizing over off-diagonal entries, the unique minimizer is $R^{*}_{i,j} = 0$ for all $i > j$ (and clearly $i < j$). Optimizing over $R_{ii}$, we are minimizing $x \rightarrow x^2 - 2 \log x$ over $x > 0$. The unique minimizer is attained at $x = 1$, i.e. $R^{*}_{i,i} = 1$ for all $i$. Therefore the unique minimizer is $L^* = I_p$, i.e. $C^{-1}L^* = I_p$. In terms of the original variable, the minimizer is $L^* = C = C(A^{-1})$.

Substituting $L^* = C(A^{-1})$ into $\mathcal{L}_{chol}(L; A)$, we obtain the minimum value:

$$
\mathcal{L}_{chol}^*(A) := \mathcal{L}_{chol}(L^*; A) = \frac{1}{2} \text{tr} (AA^{-1}) - \log |A^{-1}|.
$$

The assertion $\mathcal{L}_{chol}^*(A) = \mathcal{L}_{chol}^*(PAP^T)$ follows by noting that $|PSP^T| = |P||\Sigma||P^T| = |\Sigma|$.

### A.2 Proof of Lemma 2

Recall $B = (\beta_j) \in \mathbb{R}^{p \times p}$, $\Omega = \text{diag}(\omega^2_j) \in \mathbb{R}^{p \times p}$, we have the following experimental data log-likelihood:

$$
\ell_{\Omega}(B, \Omega, P, \pi | X) = \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{2} \left[ \left| X_{\Omega^{(i)}:\pi^{(j)}} - X_{\Omega^{(j)}P^T \beta} \right|^2 + \frac{1}{2} |\Omega_{\pi^{(j)}}| \log \omega_j^2 \right].
$$

For the second term, $\ell_2^2 = -\log \Pi_{j \in I} \log L_{ij}$, and thus $\nabla \ell_2^2(L) = n \Pi_{j \in I} (Z^{T} Z L, \delta)$, where $(A, B) = \text{tr}(A^T B)$ for two matrices $A$ and $B$. Be defined, $\Pi_\Omega$ maps a $p \times p$ matrix to its lower triangular projection. Since $\delta \in \mathbb{R}^{p \times p}$, $(Z^{T} Z L, \delta)$, and thus $\nabla \ell_2^2(L) = n \Pi_\Omega (Z^{T} Z L, \delta)$.

For the second segment of $\ell_2$, it is also easy to see that $\nabla L_j \ell_2^1 = -\frac{\partial}{\partial L_{ij}} \ell_2^1$. Putting the pieces together gives the desired result.

### A.4 Proof of Lemma 4

Recall the MCP function is defined as

$$
\rho(x) = \begin{cases} 
\frac{1}{2} \gamma x^2, & x < \gamma \\
\frac{1}{2} \gamma x^2 + \frac{2}{\gamma} x, & x \geq \gamma
\end{cases}
$$

It is not hard to see that $\rho(\gamma x) = \gamma^2 \rho_1(x)$. It follows that

$$
\text{pro}x_{\ell_2^1}(\gamma \gamma x) = \gamma \rho_1(x).
$$

### A.5 Proof of Lemma 5

Recall the MCP function is defined as

$$
\rho(x) = \begin{cases} 
\frac{1}{2} \gamma x^2, & x < \gamma \\
\frac{1}{2} \gamma x^2 + \frac{2}{\gamma} x, & x \geq \gamma
\end{cases}
$$

It is not hard to see that $\rho(\gamma x) = \gamma^2 \rho_1(x)$. It follows that

$$
\text{pro}x_{\ell_2^1}(\gamma \gamma x) = \gamma \rho_1(x).
$$

where the second equality uses the change of variable $u = \gamma v$ and the fourth uses the fact that arg min is invariant to rescaling the objective. This establishes (21).

Due to the symmetry of $\rho_1$, we have $\text{pro}x_{\ell_2^1}(-x) = -\text{pro}x_{\ell_2^1}(x)$, which is easy to see by changing a variable $u = -v$ in the defining optimization. Thus, it is enough to consider $x \geq 0$ which we assume in the following.

The function $h(u) = \rho_1(u) + \frac{1}{2} (u - x)^2$ is continuous and decreasing on $(-\infty, 0]$, hence it achieves its minimum of $\frac{x^2}{2\gamma}$ over this interval at $u = 0$. Over $(0, \infty)$, the function is differentiable with derivative $h'(u) = (1 - u) + \frac{1}{\gamma} (u - x)$ which is piecewise linear (or affine) with a break at $u = 1$. The first segment of $h'$ is a line connecting $(0, 1 - x/\alpha)$ to
The next segment is an always-increasing section starting at \((1, (1-x)/\alpha)\) and increasing with slope \(1/\alpha\). See Figure 7(a). The behavior of \(h'\) for \(u \in (0, 1)\) determines the minimizer. We have four cases:

1) When both \(1 - x/\alpha > 0\) and \((1-x)/\alpha > 0\), that is \(x < \min(\alpha, 1)\), \(h'\) is increasing in \([0, 1]\). Hence, \(h' > 0\) over \([0, \infty)\) and the overall minimum of \(h\) occurs at \(u_1 = 0\).

2) When \(1 - x/\alpha \leq 0 < (1-x)/\alpha\), that is, \(\alpha \leq x < 1\), then \(h\) has a single critical point at \(u_2 = (x-\alpha)/(1-\alpha)\) before which it decreases and after which it increases. Hence, this is its unique minimizer.

3) When both \(1 - x/\alpha < 0\) and \((1-x)/\alpha < 0\), that is, \(x > \max(\alpha, 1)\), then, the only critical point occurs in the second linear segment and is \(u_3 = x\) which is the unique minimizer.

4) When \((1-x)/\alpha \leq 0 < 1 - x/\alpha\), that is, \(1 < x \leq \alpha\), then both of the points \(u_2\) and \(u_3\) in cases 2 and 3 are critical points. The function drops in \((-\infty, u_1)\) where \(u_1 = 0\), increases in \((u_1, u_2)\), drops in \((u_2, u_3)\) and increases in \((u_3, \infty)\). Thus, both \(u_1\) and \(u_3\) are local minima (while \(u_2\) is a local maximum). The global minimum is determined by comparing \(h(u_1) = x^2/(2\alpha)\) and \(h(u_3) = 1/2\). That is, if \(x < \sqrt{\alpha}\), the global minimum is \(u_1 = 0\) (Figure 7b); if \(x > \sqrt{\alpha}\), the global minimum is \(u_3 = x\) (Figure 7c); if \(x = \sqrt{\alpha}\), minimum is \(\{u_1, u_3\} = \{0, x\}\), which is not unique.

What left is the special case when both \(1 - x/\alpha = 0\) and \((1-x)/\alpha = 0\), i.e. \(\alpha = x = 1\), indicating the first segment of \(h'(u), u \geq 0\), is flat as zero. Hence, minimizer of \(h\) is any value in the interval \([0, 1]\). We merge case 4 into cases 1 and 3 by revising the domains of \(x\), and thus complete the derivation of (22).

**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, 1993.

[4] I. Tsalavardinis, 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. Kuipers, “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, 2005, pp. 584–590.

[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. Scagnatga, C. P. de Campos, G. Corani, and M. Zaffalon, “Learning Bayesian Networks with Thousands of Variables,” in Advances in Neural Information Processing Systems, 2015, pp. 1864–1872.

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

[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, 2006, pp. 445–452.

[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, 2013, pp. 182–191.

[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, 2017, pp. 129–141.

[15] M. Champion, V. Ficheny, and M. Vignes, “Inferring Large Graphs Using \(\ell_1\)-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] 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.

[19] D. M. Chickering, “Learning Bayesian Networks is NP-Complete,” in Learning from Data, ser. Lecture Notes in
[20] 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.

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

[22] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least Angle Regression,” The Annals of Statistics, vol. 32, no. 2, pp. 407–499, 2004.

[23] P. Spirtes, C. Glymour, and R. Scheines, Causation, Prediction, and Search. Springer-Verlag, New York, 1993.

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

[25] C.-H. Zhang, “Nearly Unbiased Variable Selection under Minimax Concave Penalty,” The Annals of Statistics, vol. 38, no. 2, pp. 894–942, 2010.

[26] J. Pearl, “Causal Diagrams for Empirical Research,” Biometrika, vol. 82, pp. 669–710, 1995.

[27] 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.

[28] N. Parikh and S. Boyd, “Proximal Algorithms,” Foundations and Trends in Optimization, vol. 1, no. 3, pp. 123–231, 2013.

[29] G. Schwarz, “Estimating the Dimension of a Model,” The Annals of Statistics, vol. 6, no. 2, pp. 461–464, 1978.

[30] M. Scutari. Bayesian network repository. Accessed: 2019-01-21. [Online]. Available: http://www.bnlearn.com/bnrepository/

[31] B. Aragam, J. Gu, and Q. Zhou, “Learning Large-Scaled Bayesian Networks with the sparsebn Package,” Journal of Statistical Software, 2019, arXiv:1703.04025.

[32] J. D. Ramsey, “Scaling up Greedy Causal Search for Continuous Variables,” arXiv:1507.07749, 2015.

[33] M. Scutari, “Learning Bayesian Networks with the bnlearn R Package,” Journal of Statistical Software, vol. 35, no. 3, pp. 1–22, 2010.

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

[35] A. Hauser and P. Bühlmann, “Characterization and Greedy Learning of Intervenational Markov Equivalence Classes of Directed Acyclic Graphs,” The Journal of Machine Learning Research, vol. 13, pp. 2409–2464, 2012.

[36] 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.