Optimizing NOTEARS Objectives via Topological Swaps

Chang Deng [1] **Kevin Bello** [1 2] **Bryon Aragam** [1] **Pradeep Ravikumar** [2]

Abstract

Recently, an intriguing class of non-convex optimization problems has emerged in the context of learning directed acyclic graphs (DAGs). These problems involve minimizing a given loss or score function, subject to a non-convex continuous constraint that penalizes the presence of cycles in a graph. In this work, we delve into the optimization challenges associated with this class of non-convex programs. To address these challenges, we propose a bi-level algorithm that leverages the non-convex constraint in a novel way. The outer level of the algorithm optimizes over topological orders by iteratively swapping pairs of nodes within the topological order of a DAG. A key innovation of our approach is the development of an effective method for generating a set of candidate swapping pairs for each iteration. At the inner level, given a topological order, we utilize off-the-shelf solvers that can handle linear constraints. The key advantage of our proposed algorithm is that it is guaranteed to find a local minimum or a KKT point under weaker conditions compared to previous work and finds solutions with lower scores. Extensive experiments demonstrate that our method outperforms state-of-the-art approaches in terms of achieving a better score. Additionally, our method can also be used as a post-processing algorithm to significantly improve the score of other algorithms. Code implementing the proposed method is available at https://github.com/duntrain/topo.

[1]Booth School of Business, University of Chicago, USA. [2]Machine Learning Department, Carnegie Mellon University, USA. Correspondence to: Chang Deng <changdeng@chicagobooth.edu>, Kevin Bello <kbello@cs.cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

We study a class of constrained nonconvex optimization problems defined as follows:

$$\min_{\Theta} Q(\Theta) \;\; \text{subject to} \;\; h(W(\Theta)) = 0, \tag{1}$$

where $\Theta \in \mathbb{R}^l$ corresponds to all model parameters, and $W(\Theta) \in \mathbb{R}^{d\times d}$ is a weighted adjacency matrix — representing the structure of a directed graph of d nodes— induced by Θ. Moreover, $Q : \mathbb{R}^l \to \mathbb{R}$ is a (possibly non-convex) differentiable function that we will refer to as the score or loss function; while $h : \mathbb{R}^{d\times d} \to [0, \infty)$ is a nonnegative **non-convex** differentiable function that penalizes cycles in the weighted adjacency matrix $W(\Theta)$, and whose level set at zero corresponds to directed acyclic graphs (DAGs).

The class of problems (1) arose in the paper by Zheng et al. (2018) in the context of learning the underlying structure of a structural equation model (SEM), typically assumed to be a DAG. In Zheng et al. (2018), the challenges of combinatorial optimization were replaced by those of differentiable non-convex optimization. While global optimality remains intractable in general, the key advantage of the class of problems (1) is that it admits the use of general purpose non-linear optimizers. Due to the latter, several studies have built upon the work of Zheng et al. (2018), usually by either proposing a new characterization of h (e.g., Yu et al., 2019; Bello et al., 2022), or using different score functions Q (e.g., Zheng et al., 2018; 2020; Ng et al., 2020; Yu et al., 2019; Lachapelle et al., 2020). All, however, with a clear lack of optimality guarantees.

Based on these formulations, Wei et al. (2020) and Ng et al. (2022) studied some of the optimization-theoretic curiosities associated with this class of problems. Wei et al. (2020) provides local optimality guarantees assuming *linear models* and a *convex* score Q, while Ng et al. (2022) studies the convergence challenges of (1). The focus of our work is studying optimality for the class of problems (1) in a more general setting, i.e., admitting a possibly non-convex score Q and nonlinear models. We pay close attention to the Karush-Kuhn-Tucker (KKT) optimality conditions, building upon similar results first studied in Wei et al. (2020). The KKT conditions are known to be a necessary first-order characterization of optimal solutions under some regularity

conditions, and form the backbone of nonlinear programming (Bertsekas, 1997; Boyd et al., 2004).

More specifically, we show that by an equivalent reformulation of the KKT conditions, we can find *better* solutions to (1) — that is, KKT points and/or local minima with better (i.e. lower) score — while also relaxing the conditions required in previous work. The key idea is to relate the KKT conditions to an optimal topological sort and leverage the fact that solving the continuous program for a fixed ordering is often tractable. Although not every topological sort corresponds to a local minimum in the continuous formulation, we show that our method can indeed be rigorously interpreted as iteratively selecting better and better local minimizers until no improvement can be found. Our method also avoids explicitly enforcing the acyclicity constraint h, and instead uses the continuous characterization *indirectly* via the KKT conditions.

Contributions. To this end, we make the following specific contributions:

1. We propose a bi-level optimization algorithm, in which the outer level optimizes over topological orders and the inner level optimizes the score given a specific order. To optimize over orders, we use a novel technique for selecting candidate pairs of nodes to be swapped, which is described in detail in Section 4. This approach involves iteratively swapping pairs of nodes within the topological order of a DAG, and utilizes the KKT conditions as a guide for determining which pairs to consider swapping. To optimize the score given a specific order, we utilize state-of-the-art solvers that are able to solve the problems to stationary points.

2. We prove that our method searches between local minima and strictly decreases the score at each iteration (Section 4.3). We furthermore show that our method provably finds local minimizers under strictly weaker conditions compared to previous work (Lemma 4). In particular, we show that the concept of 'irreducibility' introduced in Wei et al. (2020) is not necessary to ensure local optimality, and provide an explicit example as demonstration (Appendix C.2).

3. We conduct a comprehensive set of experiments in multiple settings to evaluate the performance of our algorithm against state-of-the-art methods for solving problem (1). The results of our experiments, summarized in Section 5, demonstrate that our method is able to find minimizers with lower scores (compared to existing algorithms) that are guaranteed to be either local minima or KKT points.

An attractive feature of our method is its flexibility as it can be used both as a standalone algorithm and as a post-processing step when provided with a pre-computed DAG as an initialization. Although the underlying optimization problem is nonconvex and plagued by poor local minima, our results demonstrate that it is still possible to discover suitable local minima with improved scores. This is a noteworthy achievement given that nonconvex problems of this nature are often considered challenging and difficult to optimize.

2. Related Work

Most closely related to our work are methods that build on the non-convex continuous constrained formulation of Zheng et al. (2018), (e.g., Yu et al., 2019; Zheng et al., 2020; Lachapelle et al., 2020; Ng et al., 2020; Zhu et al., 2020; Romain & d'Aspremont, 2020; Bello et al., 2022). In contrast to this previous work, our focus is on *optimality conditions*, i.e. ensuring that we find a DAG that satisfies the KKT optimality conditions (in fact, it will be a local minimizer) of an equivalent formulation to that of Zheng et al. (2018). Similar to our work, recent work (Wei et al., 2020; Ng et al., 2022) has begun to study the optimization-theoretic aspects of this problem. In contrast to Wei et al. (2020), which is only guaranteed to return *some* local minimizer, our method iteratively jumps from one local minimizer to another until a stopping criterion is met. The latter allows our method to seek out for more favorable local minimizers, that is, DAGs that attain lower scores. Ng et al. (2022) studies a different question, namely the convergence of methods for solving these problems.

Although our emphasis is on optimization, it is useful to provide some context from the graphical modeling literature as well. Most algorithms for learning DAGs fall into two main categories: score-based methods that optimize a score function, and constraint-based methods that use independence tests. Since the program (1) is modeled after traditional score-based methods, we only mention a few classical constraint-based algorithms such as: the PC algorithm (Spirtes & Glymour, 1991), a general algorithm that learns the Markov equivalence class; max-min parents and children (MMPC, Tsamardinos et al., 2006); and a variety of algorithms based on local Markov boundary search such as grow-shrink (GS, Margaritis & Thrun, 1999; Margaritis, 2003) and incremental association (IAMB, Tsamardinos et al., 2003).

Score based methods assign a score to a candidate DAG structure based on how well it fits the observed data, and then attempts to find the highest scoring structure. Classical score functions include the log-likelihood based BIC and AIC scores as well as Bayesian scores under different parameter priors (Geiger & Heckerman, 2002). Other related work that study the Gaussian setting are given by Aragam &

Zhou (2015); Ghoshal & Honorio (2017; 2018), and in the non-Gaussian case by Loh & Bühlmann (2014). On the side of approximate algorithms, notable methods include greedy search (Chickering, 2003), order search (Teyssier & Koller, 2005; Scanagatta et al., 2015; Park & Klabjan, 2017), and the LP-relaxation based method proposed by Jaakkola et al. (2010). There are also exact algorithms such as GOBNILP (Cussens, 2012) and bene (Silander & Myllymaki, 2006).

Another line of work (Teyssier & Koller, 2005; Xiang & Kim, 2013; Raskutti & Uhler, 2018; Drton et al., 2018; Ye et al., 2020; Squires et al., 2020; Solus et al., 2021; Wang et al., 2021) studies order-based methods which bear a superficial relationship to our algorithm, but it is worth emphasizing that none of them theoretically analyze optimization properties such as KKT theory, local optimality guarantees or apply to *arbitrary* smooth losses. More specifically, (Ye et al., 2020) is restricted to log-likelihood based scores and (Raskutti & Uhler, 2018; Squires et al., 2020; Solus et al., 2021) require faithfulness (related) assumptions. (Silander & Myllymaki, 2006; Xiang & Kim, 2013) are exact methods that only work with a small number of nodes.

3. Notation and Background

In this section, we establish the notation and provide context for the class of problems (1).

3.1. Nonlinear DAG models

We let $G = (V, E)$ denote a *directed* graph of d nodes, with vertex set $V = [d] := \{1, \dots, d\}$ and edge set $E \subset V \times V$, where $(i, j) \in E$ indicates the presence of a directed edge from node i to node j. For a graph G, we associate each node $i \in V$ to a random variable X_i, and use $X = (X_1, \dots, X_d)$ to denote the d-dimensional random vector.

We consider *structural equation models* (SEMs Peters et al., 2017), in which each node X_j is determined by a function $f_j : \mathbb{R}^d \to \mathbb{R}$ of its parents and independent noise $z = (z_1, \dots, z_d) \in \mathbb{R}^d$ as follows:

$$X_j = f_j(X, z_j), \quad \partial_k f_j = 0 \text{ if } k \notin \mathrm{PA}_j^G, \tag{2}$$

where $\mathrm{PA}_j^G = \{i \in V \mid (i, j) \in E\}$ denotes the set of parents of node j in G. Note that we write f_j as a function of every other variable, and separately impose a restriction on the dependence through the partial derivatives, as in Zheng et al. (2020). This is equivalent to the usual formulation $X_j = f_j(\mathrm{PA}_j^G, z_j)$, and is adopted for mathematical convenience in the sequel. Standard examples of SEMs include linear SEMs (e.g., Peters & Bühlmann, 2014; Loh & Bühlmann, 2014) and additive noise models (Peters et al., 2014).

With this notation, the graphical structure implied by an SCM $f = (f_1, \dots, f_d)$ can be represented by the following $d \times d$ weighted adjacency matrix:

$$W = W(f) = (w_{ij}), \quad w_{ij} = \|\partial_i f_j\|_2. \tag{3}$$

In practice, a family of functions is defined to approximate the nonlinear functions f_j; common examples include multilayer perceptrons (MLP) (Zheng et al., 2020; Lachapelle et al., 2020), and basis expansions (Zheng et al., 2020; Bühlmann et al., 2014). See Appendix A for a detailed discussion on these families of functions.

We use Θ to denote all the model parameters used for approximating f. However, not all of these parameters are utilized for inducing the graphical structure implied by f. To differentiate, we use $\theta \subset \Theta$ to denote the subset of parameters that are used for inducing the weighted adjacency matrix W, and $\tilde{\theta} = \Theta \setminus \theta$ to denote the remaining model parameters. In other words, we have the following relationship: $W(f) = W(\Theta) = W(\theta)$.

To simplify notation and improve the clarity of presentation, we present the case where there is a single parameter θ_{ij} [1] per candidate edge (i, j), i.e., $[W(\theta)]_{ij} = \theta_{ij}$ and $W(\theta) = \theta$. However, note that all of our results hold for the general case and are thoroughly treated in the technical proofs provided in Appendix D.

3.2. Score functions

The class of programs (1) requires a loss/score function Q. We briefly review commonly used scores in the literature. Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_d] \in \mathbb{R}^{n \times d}$ denote the observed data matrix. Let Θ_i denote the parameters used to approximate f_i, we use f_{Θ_i} to denote f_i approximated by Θ_i.

Since the score function depends on the observed data, in this subsection, we use $Q(\Theta; \mathbf{X})$ to denote the score on Θ given $\mathbf{X}$. Then, some possible score functions include:

Least squares. $Q(\Theta; \mathbf{X}) = \frac{1}{2n} \sum_{i=1}^d \|\mathbf{x}_i - f_{\Theta_i}(\mathbf{X})\|_2^2$ for linear SEMs with equal noise variances (Loh & Bühlmann, 2014).

Negative log-likelihood. $Q(\Theta; \mathbf{X}) = \frac{1}{2} \sum_{i=1}^d \log(\|\mathbf{x}_i - f_{\Theta_i}(\mathbf{X})\|_2^2)$ for additive SEMs with Gaussian errors (Bühlmann et al., 2014).

Logistic loss. Let $\mathbf{1}_n$ denote the n-dimensional vector of ones. Then, we have $Q(\Theta; \mathbf{X}) = \frac{1}{n} \sum_{i=1}^d \mathbf{1}_n^\top (\log(\mathbf{1}_n + \exp(f_i(\mathbf{X}))) - \mathbf{x}_i \circ f_{\Theta_i}(\mathbf{X}))$ for generalized linear models with binary variables (Zheng et al., 2020).

In the sequel, we simplify notation by writing $Q(\Theta)$ instead of $Q(\Theta; \mathbf{X})$.

Remark 1. *It is important to emphasize that in practical applications, the choice of score Q is crucial: In order*

[1] θ_{ij} can be a vector, it is required that $[W(\theta)]_{ij} = 0$ if and only if $\theta_{ij} = 0$, see Appendix A for more discussions.

for solutions to this problem to be useful, ideally the minimizer(s) of Q should correspond to the true underlying DAG. This problem has been extensively studied (Geiger & Heckerman, 2002; Chickering, 2003; Van de Geer & Bühlmann, 2013; Loh & Bühlmann, 2014; Nandy et al., 2018; Aragam et al., 2019), so we do not pursue it further here. For example, in recent work, Reisach et al. (2021) show how certain scores are not scale invariant, which may be an issue in practice, but is simply an artifact of the score function, as originally pointed out by Loh & Bühlmann (2014). By contrast, our explicit goal is to study the optimization-theoretic aspects of objectives (1), and not to propose new algorithms for learning causal DAGs.

3.3. Continuous non-convex characterizations of DAGs

To conclude this section, we next provide a brief overview of the existing options for the function h. We remind the reader that for presentation simplicity we have $W(\theta) = \theta$, as discussed at the end of Section 3.1.

Condition 1. *The function h has the following form:*

$$h(B) = \sum_{i=1}^{d} c_i \operatorname{Tr}(B^i),$$

where $c_i > 0$ for any i.

Corollary 1 (Wei et al., 2020 Theorem 1)**.** *If h satisfies Condition 1, then we have that $h(B) = 0$ if and only if B corresponds to a DAG, for any nonnegative matrix B.*

By now the literature contains many different proposals of functions h that satisfy Condition 1; in this paper, we mostly focus on the following three:

1. **The NOTEARS formulation.** Zheng et al. (2018) were the first to propose a differentiable characterization of DAGs given by $h(B) = \operatorname{Tr}(e^B) - d$ for a nonnegative matrix B.
2. **A polynomial formulation.** Yu et al. (2019) proposed the use of $h(B) = \operatorname{Tr}((I + {}^1\!/_d\, B)^d) - d$ for a nonnegative matrix B.
3. **The DAGMA formulation.** Bello et al. (2022) proposed the use of $h(B) = -\log\det(I - B)$ for a nonnegative matrix B with spectral radius less than one.

Note that B above is commonly defined as $B = \theta \circ \theta$, where $\circ$ denotes the Hadamard product. In that case, it has been shown that $\nabla_\theta h(\theta \circ \theta) = 0$ if and only if θ is a DAG (see Wei et al., 2020). The latter implies that all stationary points of h are global minima of h, a property known as invexity, as highlighted by Bello et al. (2022).

Remark 2. *Our results are general and apply to any function h satisfying Condition 1. Thus, our results apply to any of the three h functions mentioned above.*

3.4. Necessary and sufficient conditions for optimality

Wei et al. (2020) first studied (1) from an optimality perspective. The authors argued that the use of the Hadamard product $\theta \circ \theta$ leads to an undesirable property, namely, any feasible Θ in (1) cannot satisfy regularity conditions. Motivated by this negative result, Wei et al. (2020) proposed an alternative, yet equivalent, formulation by replacing $h(\theta \circ \theta)$ by $h(|\theta|)$. Reasoning similarly, we reformulate (1) as

$$\min_{\Theta} \; Q(\Theta) \quad \text{subject to} \quad h(|\theta|) \le 0. \tag{4}$$

By writing $\theta = \theta^+ - \theta^-$, where $\theta^+ = \max\{\theta, 0\}$ and $\theta^- = \max\{-\theta, 0\}$ denote the positive and negative parts of θ, respectively. Then, an equivalent smooth formulation is given by

$$\begin{aligned} &\min_{\theta^+, \theta^-, \tilde{\theta}} \; Q((\theta^+ - \theta^-, \tilde{\theta})) \\ &\text{subject to } h(\theta^+ + \theta^-) = 0, \text{ and } \; \theta^+, \theta^- \ge 0. \end{aligned} \tag{5}$$

For clarity, we remind the reader that in (5) we have $\Theta = (\theta^+, \theta^-, \tilde{\theta})$. Then, the KKT conditions for (5) can be succinctly written as follows:

$$\frac{\partial Q(\Theta)}{\partial \theta^+_{ij}} + \lambda \frac{\partial h(\theta^+ + \theta^-)}{\partial \theta^+_{ij}} = M^+_{ij} \ge 0, \tag{6a}$$

$$-\frac{\partial Q(\Theta)}{\partial \theta^-_{ij}} + \lambda \frac{\partial h(\theta^+ + \theta^-)}{\partial \theta^-_{ij}} = M^-_{ij} \ge 0, \tag{6b}$$

$$\theta^+_{ij} \circ M^+_{ij} = \theta^-_{ij} \circ M^-_{ij} = 0, \tag{6c}$$

$$\frac{\partial Q(\Theta)}{\partial \tilde{\theta}} = 0, \tag{6d}$$

in addition to the feasibility conditions in (5). where $M^\pm$ and λ are the Lagrange multipliers of the constraints on $\theta^\pm$ and h, respectively. Here $\lambda \in \mathbb{R}$, $M^\pm \ge 0$.

Briefly, (6a), (6b) and (6d) results from dual feasibility and the stationarity condition, while (6c) stems from complementary slackness.

The following useful theorem from Wei et al. (2020) establishes the connection between KKT satisfiability in (5) and local minimality in (4) for *linear SEMs* (i.e., $\tilde{\theta} = \emptyset$).

Theorem 1 (Wei et al., 2020, Theorem 7)**.** *Assume that Q is convex, h satisfies the Condition 1, and $\tilde{\theta} = \emptyset$. If (θ^+, θ^-) satisfies the KKT conditions in (6), then $\theta^+ - \theta^-$ is a local minimum of (4).*

A key ingredient of our developments in the sequel is the following alternative characterization of the KKT conditions, which turns out to provide an algorithmically amenable first-order sufficient condition for *local* optimality. We include a proof in Appendix D.

Lemma 1. *If $\Theta = (\theta^+, \theta^-, \tilde{\theta})$ satisfies the following conditions:*

(i) *For* $\{(i,j) \mid [\nabla h(\theta^+ + \theta^-)]_{ij} > 0\} \Rightarrow \theta^{\pm}_{ij} = 0$.

(ii) *For* $\{(i,j) \mid [\nabla h(\theta^+ + \theta^-)]_{ij} = 0\} \Rightarrow \frac{\partial Q(\Theta)}{\partial \theta^{\pm}_{ij}} = 0$.

(iii) $\frac{\partial Q(\Theta)}{\partial \tilde{\theta}} = 0$.

(iv) $\theta^+ \geq 0, \theta^- \geq 0$.

Then, we have that Θ *is a KKT point of* (5)*. Moreover, if the score* Q *is convex, any such* $\Theta = (\theta^+ - \theta^-, \tilde{\theta})$ *is also a local minimum for problem* (4)*.*

Remark 3. *If the score* Q *is smooth but non-convex, then we can no longer use Lemma 1 to automatically promote KKT points to local minima. Thus, in the sequel, whenever the score* Q *is non-convex, all claims about local minima must be demoted to KKT points.*

4. Optimization Algorithm: Topological Swaps

Our key idea is to solve (4) as a two-staged problem: in the inner stage, we solve (4) to an additional constraint that makes the problem tractable, and in the outer stage, we search over the set of constraints. The critical innovation is in using our reformulation of the KKT conditions in guiding this search. Our specific set of constraints relies on imposing an ordering over the variables. We briefly review such order constrained optimization below before formally introducing our overall approach.

4.1. Background: Order-constrained optimization

We leverage the following well-known observation: For a *fixed* topological sort, problem (4), or equivalently (5), can often be solved efficiently. We briefly review this material here for completeness.

Recall that a topological sort (or order) for G is a partial ordering $\prec$ on the vertex set $V = [d]$ such that $X_i \to X_j \implies i \prec j$, here $X_i \to X_j$ means there exists an edge from i to j. A directed graph is acyclic if and only if it has a topological sort, although this sort may not be unique. Equivalently, we can view a topological sort as a permutation on V.

Definition 1 (Topological sort)**.** *A topological sort* $\prec$ *defines a permutation* π *of the vertex set* V *for* G *by letting* $\pi(j)$ *be the* j*-th node in the ordering defined by* $\prec$*. In other words, if* $X_{\pi(i)} \to X_{\pi(j)}$*, then* $i < j$*.*

A similar definition carries over in the obvious way for weighted adjacency matrices θ. We furthermore call G (resp. θ) *consistent* with π if π is a topological sort of G (resp. θ), and write this as $G \sim \pi$ (resp. $\theta \sim \pi$).

Given a permutation π, we then have the following order-constrained optimization problem:

$$\min_{\theta \sim \pi} \; Q(\Theta). \tag{7}$$

Due to the order consistency constraint $\theta \sim \pi$, the acyclicity constraint $h(|\theta|) \leq 0$ is automatically satisfied and hence can be omitted from (7).

We next reformulate (7) with explicit linear constraints. Moreover, in the sequel, we use Θ^*_π to denote any solution to this problem:

$$\Theta^*_\pi = (\theta^*_\pi, \tilde{\theta}^*_\pi) \in \arg\min_{\Theta} \; Q(\Theta) \quad \text{subject to } \theta_{\pi(i),\pi(j)} = 0, \; \forall j < i. \tag{8}$$

Remark 4. *Our results only require solving* (8) *up to stationarity. That is, we can first set* $\theta_{\pi(i),\pi(j)} = 0$*, for all* $j < i$*, and then use any off-the-shelf first-order optimizer (Boyd et al., 2004; Nesterov et al., 2018) for the resulting (non)convex unconstrained problem.*

4.2. Algorithm

Motivated by the observations above, we propose a general bi-level algorithm based on finding the topological sort π of an optimal scoring DAG. For any Θ and $\tau, \xi > 0$, define a set

$$\mathcal{Y}(\Theta, \tau, \xi) \stackrel{\text{def}}{=} \left\{(i,j) \mid [\nabla h\left(|\theta|\right))]_{ij} \leq \tau, \left\| \frac{\partial Q(\Theta)}{\partial \theta_{ij}} \right\|_1 > \xi \right\}. \tag{9}$$

Given this machinery, the four main steps of our approach (Algorithm 1) are as follows:

1. Initialize at an arbitrary sort π, and solve (7).
2. Define a candidate set of possible swaps by $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$ as defined in (9), where (τ_*, ξ^*) are parameters chosen adaptively such that $|\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)| \approx s_{\text{small}}$.
3. Choose the best swap from this set to obtain a new topological sort; i.e., the swap that decreases the score Q the most.
4. Repeat until there is no sufficient improvement in the score.

There are several advantages to this approach:

- Enforcing acyclicity is much simpler: Once a topological sort is fixed, acyclicity is automatically guaranteed and the optimization is straightforward and efficient (cf. Section 4.1). Thus, there is no need to include $h(|\theta|)$ directly in the optimization routines compared to Zheng et al. (2018), which greatly simplifies implementation.

- We will only need to check (ii), (iii), and (iv) in Lemma 1 in order to ensure the KKT conditions are satisfied, and computing the gradients ∇Q, ∇h is easy. Note that Condition (i) is to ensure $|\theta|$ is acyclic, which is always satisfied by the argument in the above item.

It is worth stressing that this is *not the same* as greedily selecting individual edges as in GES (Chickering, 2003): Each swap re-solves (7) *globally*, and hence updates every edge.

Crucially, in the second step, it is not necessary to exhaustively check all possible swaps: By properly exploiting the KKT conditions as in Lemma 1, we are able to limit the set of possible candidate swaps to $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$. This greatly improves the efficiency of the algorithm. Moreover, it is not necessary to find the swap that decreases the score the most in Algorithm 1 line 9. Instead, any swap that decreases Q could be used to accelerate our algorithm. This greedy strategy, which is explored in the appendices, can improve time efficiency while attaining comparable performances.

The main steps of our method are summarized in Algorithm 1; a more comprehensive outline (for reproducibility purposes) can be found in the Appendix B (Algorithm 2). The subroutine FINDPARAMS (detailed in Algorithm 3 in Appendix B) aims to find appropriate values for τ and ξ such that $|\mathcal{Y}(\Theta, \tau, \xi)| \approx s$. In Algorithm 1, the notation s_{small} and s_{large} are used to denote small and large search spaces, respectively.

Remark 5. *It is worth noting how the continuous formulation plays a critical role in Algorithm 1: We use both the KKT conditions and the function h in order to select candidate swaps (cf. (9)).*

4.3. Analysis

Intuitively, the idea behind Algorithm 1 is that it iteratively jumps between better and better local minimizers, until the candidate swaps given by (9) no longer offer any significant improvement in the score. This is achieved by exploiting the KKT conditions (6). In this section, we show that this is not just a heuristic: Under appropriate conditions, Algorithm 1 indeed decreases the score and always terminates at a local minimum or KKT point.

Before proving this, it is worth stressing why this is not obvious *a priori*: Even if we solve (7) to global optimality (i.e., given the order constraint $\theta \sim \pi$), a global solution to (7) need not be a *local* solution to (4). This stems from the fact that a DAG can have more than one topological sort, and the solutions to (7) for each sort need not coincide.

We begin with two important lemmas.

Lemma 2. *If $(i, j) \in \mathcal{Y}(\Theta^*_\pi, 0, 0)$, then $(\theta^*_\pi)_{ij} = 0$.*

Algorithm 1 TOPO

Require: Initial topological sort π, integers s_{small} and s_{large} with $s_{\text{large}} > s_{\text{small}}$, and score function Q.

1: {Here we use π_{ij} to denote the new topological sort by swapping nodes i and j in π.}
2: $(\tau_*, \xi^*) \leftarrow \text{FINDPARAMS}(\theta^*_\pi, s_{\text{small}})$
3: $\mathcal{S} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$
4: **while** $\mathcal{S} \neq \emptyset$ **do**
5: **if** $\exists (i, j) \in \mathcal{S}$ s.t. $Q(\Theta^*_{\pi_{ij}}) < Q(\Theta^*_\pi)$ **then**
6: Update π to be π_{ij} that (most) decreases Q.
7: $\mathcal{S} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$
8: **else**
9: $(\tau^*, \xi_*) \leftarrow \text{FINDPARAMS}(\theta^*_\pi, s_{\text{large}})$
10: $\mathcal{S} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau^*, \xi_*)$ {Try a larger search space}
11: **if** $\exists (i, j) \in \mathcal{S}$ s.t. $Q(\Theta^*_{\pi_{ij}}) < Q(\Theta^*_\pi)$ **then**
12: Update π to be π_{ij} that (most) decreases Q.
13: $\mathcal{S} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$
14: **else**
15: $\mathcal{S} \leftarrow \emptyset$
16: **end if**
17: **end if**
18: **end while**

Ensure: Θ^*_π

Lemma 3. *If the score Q is separable w.r.t θ, i.e. $Q(\Theta) = \sum_j Q_j(\theta_j, \tilde{\theta})$ and $\mathcal{Y}(\Theta^*_\pi, 0, 0) \neq \emptyset$ for some topological sort π, then*

$$Q(\Theta^*_{\pi_{ij}}) < Q(\Theta^*_\pi),$$

for every $(i, j) \in \mathcal{Y}(\Theta^*_\pi, 0, 0)$.

Lemma 3 has an important takeaway message: As long as we can find a pair of nodes $(i, j) \in \mathcal{Y}(\Theta^*_\pi, 0, 0)$—i.e. $\mathcal{Y}(\Theta^*_\pi, 0, 0) \neq \emptyset$—then we can find another topological sort with strictly smaller score. The difficult case is when $\mathcal{Y}(\Theta^*_\pi, 0, 0) = \emptyset$: What Algorithm 1 does is increase the thresholds (τ_*, ξ^*) just enough to make $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*) \neq \emptyset$. Indicated by the previous observation, this suggests that placing node i before node j is likely (but not guaranteed) to decrease the score. There are many strategies for updating the topological sort to make this happen, but we adopt the simplest way, i.e., swapping the node i and node j.

This previous discussion can be made more concrete via the following observation:

Corollary 2. *If $\mathcal{Y}(\Theta^*_\pi, 0, 0) = \emptyset$, then Θ^*_π satisfies the KKT conditions in (6).*

The following definition relates to the score Q and is a relevant property for Theorem 2.

Definition 2 (Connected estimator)**.** *Given a topological sort π, the estimator Θ^*_π is called connected if for any $i < j$ there is a directed path from node $\pi(i)$ to node $\pi(j)$ in θ^*_π.*

Equivalently, for any $i < j$, a connected estimator satisfies $[\nabla h(|\theta^*_\pi|)]_{\pi(j),\pi(i)} > 0$. In general, we expect an estimator to be connected when sparse regularization is not used. It is worth noting that NOTEARS (Zheng et al., 2018) *without* explicit ℓ_1 regularization is observed to return a connected estimator.

Theorem 2. *For any h satisfies the Condition 1. If the score Q is convex (resp. non-convex) and Θ^*_π is connected for all π. Then Algorithm 1 returns a local minimum (resp. KKT point) of problem (4), where the score is decreased at each iteration. Moreover, the solution at each iteration is also a local minimum (resp. KKT point).*

Remark 6. *Although the proof of Theorem 2 is deceptively simple, we stress that it is not* a priori *obvious that swapping pairs of nodes will always decrease the score: Done naïvely, this could increase the score. Our careful use of the KKT conditions precludes this behavior.*

The connected estimator assumption in Theorem 2 can be dropped whenever the score Q is separable (e.g., least squares).

Theorem 3. *For any h satisfies the Condition 1. Assume that the score Q is separable w.r.t θ, i.e., $Q(\Theta) = \sum_j Q_j(\theta_j, \tilde{\theta})$. If the score Q is convex (resp. non-convex), then Algorithm 1 returns a local minimum (resp. KKT point) of problem (4), where the score is decreased at each iteration.*

4.4. Comparison to previous work

Wei et al. (2020) first unveiled the connections between the KKT conditions in (6) and local minimality in (4) by studying a related problem with *explicit edge absence constraints* $\mathcal{Z}$. As such, it is instructive to compare these two approaches since there are some important distinctions. A first clear difference is that the KKTS algorithm by Wei et al. (2020) relies on an assumption they call *irreducibility*, to ensure local minimality.

We provide a complete discussion on the irreducibility assumption of Wei et al. (2020) in Appendix C.2 and focus on the main ideas here. Briefly, KKTS (Wei et al., 2020) uses a set of node pairs $\mathcal{Z}$ to indicate which edges should be absent in the graph. KKTS works by iteratively adding and removing elements to $\mathcal{Z}$, and the algorithm stops once $\mathcal{Z}$ is an irreducible set. Then, Wei et al. (2020) show that when $\mathcal{Z}$ is irreducible, the KKTS solution is a local minimum, provided additional assumptions such as the score being separable and convex.

In Appendix C.2, we show that irreducibility is not a necessary condition for optimality. We prove this by showing a simple example where an optimal solution can correspond to a *reducible* set $\mathcal{Z}$.

Proposition 1. *Irreducibility of the set $\mathcal{Z}$ is sufficient but not necessary for KKTS to find a KKT point of problem (5).*

The above discussion should already mark a clear distinction of Algorithm 1 to KKTS, i.e., our method *does not* rely on the irreducibility assumption. Finally, we note that the irreducibility assumption might seem a mild condition, however, it can have a severe effect on the runtime of KKTS as it will not stop until an irreducible set is found.

A second difference to KKTS is that our approach not only attempts to find an optimal solution but also attempts to find the local optimum with the lowest score possible. This fact is a direct consequence of how Algorithm 1 works, namely, at each iteration we look for a solution with lower score. The fact that KKTS does not use the score Q to guide their search procedure can result in solutions with high scores. We next provide more details. Full details can be found in Appendix C.1.

Example 1. *Consider the following three-node linear SEM with standard Gaussian noise $z_j \sim \mathcal{N}(0,1)$, for $i \in [3]$. Consider also that the score Q is the population least square loss.*

$$X_1 = z_1, \quad X_2 = aX_1 + z_2, \quad X_3 = bX_2 + z_3. \tag{10}$$

In Appendix C.1, we show that for the linear model (10) there exists many values a and b where the solutions from KKTS and NOTEARS produce solutions with higher score w.r.t. Algorithm 1; moreover, NOTEARS produces non-optimal solutions. This is illustrated in Appendix C.1 for $a = 1, b = -0.55$. In each of these examples, our method can always return a solution that satisfies the optimality conditions in Lemma (1) and also attain the lowest score.

5. Experiments

Method	Metric	$d = 20$	$d = 40$	$d = 100$
GOLEM-EV	KKT	0	0	0
	Loss	10.7 ± 0.12	40.7 ± 4.8	68.8 ± 3.9
	SHD	11.4 ± 3.4	51.4 ± 28.3	145.2 ± 52.6
NOTEARS	KKT	0	0	0
	Loss	11.9 ± 0.1	62.1 ± 8.8	73.1 ± 7.6
	SHD	28.6 ± 3.2	129 ± 25.5	140.0 ± 30.1
NOFEARS	KKT	1	1	1
	Loss	11.5 ± 0.3	47.6 ± 1.6	61.2 ± 2.6
	SHD	23.2 ± 4.5	69.8 ± 16.0	87.5 ± 19.2
NOTEARS-TOPO	KKT	1	1	1
	Loss	$\mathbf{9.8 \pm 0.1}$	$\mathbf{38.4 \pm 0.1}$	$\mathbf{47.5 \pm 0.1}$
	SHD	$\mathbf{0.4 \pm 0.2}$	9.2 ± 0.8	$\mathbf{14.2 \pm 1.9}$
RANDOM-TOPO	KKT	1	1	1
	Loss	$\mathbf{9.8 \pm 0.1}$	$\mathbf{38.4 \pm 0.1}$	$\mathbf{47.5 \pm 0.1}$
	SHD	$\mathbf{0.4 \pm 0.2}$	$\mathbf{8.6 \pm 0.9}$	16.3 ± 2.6

Table 1. Experiments on linear DAGs with equal-variance Gaussian noise on ER4 graphs. The score is the least squares, and d is the number of nodes. Our methods are RANDOM-TOPO, and NOTEARS-TOPO.

Method	Metric	$d = 20$	$d = 40$	$d = 100$
GOLEM-NV	KKT	0	0	0
	Loss	9.9 ± 0.6	15.2 ± 1.3	42.7 ± 3.5
	SHD	$\mathbf{2.3 \pm 0.1}$	$\mathbf{23.4 \pm 3.4}$	$\mathbf{82.1 \pm 12.3}$
NOTEARS	KKT	0	0	0
	Loss	13.8 ± 2.1	17.2 ± 1.2	50.6 ± 4.5
	SHD	7.3 ± 0.1	39.2 ± 7.1	138.1 ± 23.6
NOTEARS-TOPO	KKT	1	1	1
	Loss	$\mathbf{8.3 \pm 1.2}$	$\mathbf{13.2 \pm 2.1}$	$\mathbf{35.1 \pm 2.3}$
	SHD	2.7 ± 3.2	26.3 ± 4.2	86.9 ± 6.6
RANDOM-TOPO	KKT	1	1	1
	Loss	8.9 ± 1.3	14.4 ± 1.2	39.2 ± 4.1
	SHD	3.3 ± 0.2	29.1 ± 4.2	106.4 ± 11.6

Table 2. Experiments on linear DAGs with unequal-variance Gaussian noise on ER4 graphs. The score is the log-likelihood with the minimax concave penalty (MCP) penalty, and d is the number of nodes. Our methods are RANDOM-TOPO, and NOTEARS-TOPO.

Method	Metric	$d = 10$	$d = 20$	$d = 50$
GOLEM-EV	KKT	0	0	0
	Loss	4.3 ± 0.1	4616.5 ± 4163.2	$(7.4 \pm 7.4) \cdot 10^{18}$
	SHD	6.5 ± 0.8	85.1 ± 6.4	1152.5 ± 2.6
NOTEARS	KKT	0	0	0
	Loss	6.2 ± 0.2	18.9 ± 1.3	$(1.7 \pm 1.6) \cdot 10^{11}$
	SHD	14 ± 1.1	79.5 ± 2.1	1198.1 ± 5.3
NOTEARS-TOPO	KKT	1	1	1
	Loss	$\mathbf{4.97 \pm 0.1}$	$\mathbf{9.92 \pm 0.1}$	$\mathbf{24.9 \pm 0.3}$
	SHD	$\mathbf{0.1 \pm 0.1}$	$\mathbf{0.7 \pm 0.2}$	$\mathbf{19.4 \pm 5.5}$
RANDOM-TOPO	KKT	1	1	1
	Loss	$\mathbf{4.97 \pm 0.1}$	10.3 ± 0.2	35.8 ± 2.1
	SHD	$\mathbf{0.1 \pm 0.1}$	3.1 ± 1.4	155.6 ± 17.9

Table 3. Experiments on Fully connected linear DAGs with Gaussian noise. The score is least squares, and d is the number of nodes. Our methods are RANDOM-TOPO, and NOTEARS-TOPO.

We compare our method against state-of-the-art solvers for (1), namely, NOTEARS (Zheng et al., 2018; 2020), NOFEARS (KKTS) (Wei et al., 2020), and GOLEM (Ng et al., 2020). For TOPO (Algorithm 1), we consider the case of random initialization (denoted by starting with 'RANDOM'), and initializing at the output of NOTEARS (denoted by starting with 'NOTEARS'). Here, random initialization is conducted by sampling a topological sort π uniformly at random, and solving problem (7) to get Θ^*_π. Details for each experimental setting can be found in Appendix E.

Our main empirical results are shown in Tables 1, 2, 3 and 4. In all the tables, we report: Whether or not the solution of the algorithms satisfies the KKT conditions (1 indicating that the method always returned a KKT point, and 0 indicating that it never returns a KKT point); the score/loss attained by the method; and the structural Hamming distance (SHD) w.r.t. the ground-truth DAG.

In Table 1, we observe that, as expected, NOFEARS and our algorithm are capable of returning a KKT point (and local minimum in this setting since Q is convex). We also note that TOPO with random initialization (Random_Topo) performs competitively in this case, even though the initial topological sort was randomly sampled. Moreover, notice that when initialized at the output of NOTEARS, our method (Notears_Topo) improves the performance of NOTEARS **dramatically**. The latter demonstrates the usability of our method as a post-processing algorithm, as discussed in our contributions.

Method	Metric	$d = 10$	$d = 20$	$d = 40$
NOTEARS-MLP	KKT	0	0	0
	Loss	7.2 ± 0.2	14.4 ± 0.3	28.5 ± 0.4
	SHD	5.6 ± 0.7	29.1 ± 3.1	112.3 ± 20.2
NOTEARS-TOPO	KKT	1	1	1
	Loss	$\mathbf{6.4 \pm 0.1}$	$\mathbf{11.6 \pm 0.1}$	$\mathbf{22.8 \pm 0.6}$
	SHD	$\mathbf{2.7 \pm 0.5}$	$\mathbf{12.1}$	$\mathbf{36.3 \pm 20.4}$
TRUE	KKT	1	1	1
	Loss	6.3 ± 0.1	12.2 ± 0.1	23.4 ± 0.4
	SHD	2.1 ± 0.5	11.6 ± 0.6	36.1 ± 2.2

Table 4. Experiments on Nonlinear Model with Neural Network on ER4 graphs. The score is least squares, and d is the number of nodes. Our method is NOTEARS-TOPO. Here 'True' means the solution of problem (8) using the underlying true topological sort.

In Table 2, for a non-convex score, we observe that TOPO still obtains solutions satisfying KKT optimality and achieves the lowest scores.

In Table 3, we study a very challenging setting where the underlying graph is a fully connected DAG. We observe that existing methods can perform reasonably well when the number of nodes is very small (e.g., 10) but their performance degrade severely for graph with larger number of nodes. In contrast, TOPO works remarkably well in this setting, which should come to no surprise since sparsity assumptions are not required, consistent with our analysis in Section 4.3.

In Table 4, we make explicit comparison to *nonlinear* NOTEARS (Zheng et al., 2020). Comparison against other methods is implicit in previous work (Zheng et al., 2020). We observe that Notears_Topo outperforms all other methods and is close to the solution of problem (8) using the true topological sort.

5.1. Additional experiments

In Appendix E, we provide further experiments. We consider linear models with different noise distributions (e.g., Gaussian, Gumbel and exponential) for {ER1, ER2, ER4, SF1, SF2, SF4} graphs. See Appendix E.4. There, we observe that our methods even with random initialization still outperforms existing methods in terms of score and SHD, also the solutions are guaranteed to be local minimal. Additionally, our results are not specific to certain non-linearities. To illustrate this, we run experiments on a logistic model (binary X_j), and neural networks. See details in the Ap-

pendix E.5. Finally, we also report the runtime and scores of each method for linear and nonlinear models in Appendices E.4 and E.5.

We analyze the sensitivity of the hyperparameters s_{small}, and s_{large} on Algorithm 1 (see Appendix E.3). That is, we thoroughly study the effect of hyperparameters on the cardinality of the search spaces, see eq.(9), and how many times our algorithm searches in a space of large cardinality. Moreover, we test our method when using randomly chosen swapping set to demonstrate the effectiveness of (9) (see Appendix E.6). Finally, we also include an analysis on structural accuracy vs iterations to track the performance of Algorithm 1 (see Appendix E.7).

6. Conclusion

Inspired by the KKT conditions, we developed new insights into the optimization-theoretic properties of NOTEARS objectives, and proposed a new bi-level algorithm with attractive local optimality guarantees. As a by-product, it can also improve the solutions of state-of-the-art solvers for (1) (e.g., NOTEARS, KKTS, GOLEM). Although proving convergence to a global minimizer is expected to be challenging, we have shown that our method has desirable properties for an optimization scheme: (a) It decreases the score in each iteration and (b) It is guaranteed to return a local minimizer (and hence also a KKT point). The key driver behind our approach is the interpretation of the KKT conditions as a proxy for choosing promising node swaps in a topological sort. An important open question for future work is the convergence of Algorithm 1: What is its iteration and computational complexity?

It is also interesting to note that unlike previous methods that rely on explicitly enforcing acyclicity via $h(B)$, our approach only uses $h(B)$ indirectly in order to check the KKT conditions. This idea was already implicit in the KKTS method due to Wei et al. (2020), and could lead to new insights into how to optimize NOTEARS objectives and other acyclicity-constrained problems.

Acknowledgments and Disclosure of Funding

K. B. was supported by NSF under Grant # 2127309 to the Computing Research Association for the CIFellows 2021 Project. B.A. was supported by NSF IIS-1956330, NIH R01GM140467, and the Robert H. Topel Faculty Research Fund at the University of Chicago Booth School of Business. This work was done in part while B.A. was visiting the Simons Institute for the Theory of Computing. P.R. was supported by ONR via N000141812861, and NSF via IIS-1909816, IIS-1955532, IIS-2211907. We also thank the University of Chicago Research Computing Center for assistance with the calculations carried out in this work.

References

Aragam, B. and Zhou, Q. Concave penalized estimation of sparse Gaussian Bayesian networks. *The Journal of Machine Learning Research*, 16(1):2273–2328, 2015.

Aragam, B., Amini, A., and Zhou, Q. Globally optimal score-based learning of directed acyclic graphs in high-dimensions. *Advances in Neural Information Processing Systems*, 32, 2019.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Bello, K., Aragam, B., and Ravikumar, P. K. Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization. In *Advances in Neural Information Processing Systems*, 2022.

Bertsekas, D. P. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Bühlmann, P., Peters, J., and Ernest, J. Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

Chickering, D. M. Optimal structure identification with greedy search. *JMLR*, 3:507–554, 2003.

Cussens, J. Bayesian network learning with cutting planes. *arXiv preprint arXiv:1202.3713*, 2012.

Drton, M., Chen, W., and Wang, Y. S. On causal discovery with equal variance assumption. *arXiv preprint arXiv:1807.03419*, 2018.

Geiger, D. and Heckerman, D. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *Annals of Statistics*, 30: 1412–1440, 2002.

Ghoshal, A. and Honorio, J. Learning identifiable gaussian bayesian networks in polynomial time and sample complexity. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6460–6469, 2017.

Ghoshal, A. and Honorio, J. Learning linear structural equation models in polynomial time and sample complexity. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1466–1475. PMLR, 2018.

Jaakkola, T., Sontag, D., Globerson, A., and Meila, M. Learning bayesian network structure using lp relaxations. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 358–365, 2010.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. Gradient-based neural dag learning. In *International Conference on Learning Representations*, 2020.

Loh, P.-L. and Bühlmann, P. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research*, 15(1):3065–3105, 2014.

Margaritis, D. Learning bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003.

Margaritis, D. and Thrun, S. Bayesian network induction via local neighborhoods. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, pp. 505–511, 1999.

Nandy, P., Hauser, A., and Maathuis, M. H. High-dimensional consistency in score-based and hybrid structure learning. *The Annals of Statistics*, 46(6A):3151–3183, 2018.

Nesterov, Y. et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

Ng, I., Ghassami, A., and Zhang, K. On the role of sparsity and dag constraints for learning linear DAGs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17943–17954. Curran Associates, Inc., 2020.

Ng, I., Lachapelle, S., Ke, N. R., Lacoste-Julien, S., and Zhang, K. On the convergence of continuous constrained optimization for structure learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 8176–8198. PMLR, 2022.

Park, Y. W. and Klabjan, D. Bayesian network learning via topological order. *The Journal of Machine Learning Research*, 18(1):3451–3482, 2017.

Peters, J. and Bühlmann, P. Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *JMLR*, 2014.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

Raskutti, G. and Uhler, C. Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1):e183, 2018.

Reisach, A., Seiler, C., and Weichwald, S. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.

Romain, M. and d'Aspremont, A. A bregman method for structure learning on sparse directed acyclic graphs. *arXiv preprint arXiv:2011.02764*, 2020.

Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. Learning bayesian networks with thousands of variables. In *NIPS*, pp. 1864–1872, 2015.

Silander, T. and Myllymaki, P. A simple approach for finding the globally optimal bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

Solus, L., Wang, Y., and Uhler, C. Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 108(4):795–814, 2021.

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

Squires, C., Wang, Y., and Uhler, C. Permutation-based causal structure learning with unknown intervention targets. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1039–1048. PMLR, 2020.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pp. 584–590, Arlington, Virginia, USA, 2005. AUAI Press. ISBN 0974903914.

Tsamardinos, I., Aliferis, C. F., Statnikov, A. R., and Statnikov, E. Algorithms for large scale markov blanket discovery. In *FLAIRS conference*, volume 2, pp. 376–380, 2003.

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

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

Wang, X., Du, Y., Zhu, S., Ke, L., Chen, Z., Hao, J., and Wang, J. Ordering-based causal discovery with reinforcement learning. *arXiv preprint arXiv:2105.06631*, 2021.

Wei, D., Gao, T., and Yu, Y. DAGs with no fears: A closer look at continuous optimization for learning bayesian networks. In *Advances in Neural Information Processing Systems*, 2020.

Xiang, J. and Kim, S. A* lasso for learning a sparse bayesian network structure for continuous variables. *Advances in neural information processing systems*, 26, 2013.

Ye, Q., Amini, A. A., and Zhou, Q. Optimizing regularized cholesky score for order-based learning of bayesian networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3555–3572, 2020.

Yu, Y., Chen, J., Gao, T., and Yu, M. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pp. 7154–7163. PMLR, 2019.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, pp. 3414–3425. PMLR, 2020.

Zhu, S., Ng, I., and Chen, Z. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*, 2020.

SUPPLEMENTARY MATERIAL
Optimizing NOTEARS Objectives via Topological Swaps

A. Additional Discussion on Family of Approximators

Let $\mathcal{F} = \{f : f_j \in \mathcal{F}_j, \forall j \in [d]\}$ be a family of functions used to approximate the SCM in problem (2). In this section, we focus on the general case and discuss under what conditions that family $\mathcal{F}$ can be used to approximate f_j and how our results apply in this general setting.

We consider approximations $f = (f_1, \dots, f_d) \in \mathcal{F}$ that are parameterized by θ, i.e. $f(x) := f(x; \theta)$. This defines $W(\theta) := W(f(\cdot; \theta))$ as the adjacency matrix defined by (3), which is characterized by θ. Although the following definition is standard, we pause to make this precise since it is crucial in the development that follows:

Definition 3 (sub-vector)**.** *Given a vector $\beta = (\beta_1, \dots, \beta_n) \in \mathbb{R}^n$, and we say that α is a sub-vector of β if and only if there is a subset $J = \{j_1, \dots, j_k\} \subset \{1, 2, \dots, n\}$ such that $\alpha = \beta_J := (\beta_{j_1}, \dots, \beta_{j_k})$.*

Under the following general assumptions, our results and proof in Section D still apply without any modification:

(i) The parametrization is separable in the following sense: $\theta = (\theta_1, \dots, \theta_d)$ and each f_j in (2) is only parameterized by the sub-vector θ_j, i.e., $f_j(x; \theta) = f_j(x; \theta_j)$.

(ii) There are sub-vectors θ_{ij} of θ_j that can reveal if there is no edge from node i to node j (i.e., $[W(\theta)]_{ij} = 0$ if and only if $\theta_{ij} = 0$.) In this case, the general definition in (3) can be replaced with $[W(\theta)]_{ij} = \|\theta_{ij}\|_1$ without loss of generality.

Write $\theta_{ij} = (\theta_{ijr})_r$ for each sub-vector θ_{ij}. Since $[W(\theta)]_{ij} = \|\theta_{ij}\|_1$, we have

$$[W(\theta^+ + \theta^-)]_{ij} = \|\theta^+_{ij} + \theta^-_{ij}\|_1 = \mathbf{1}^\top(\theta^+_{ij} + \theta^-_{ij}) = \sum_r (\theta^+_{ijr} + \theta^-_{ijr})$$

Therefore,

$$\frac{\partial [W(\theta^+ + \theta^-)]_{ij}}{\partial \theta^\pm_{ij}} = \left(\frac{\partial [W(\theta^+ + \theta^-)]_{ij}}{\partial \theta^\pm_{ijr}} \right)_r = (1)_r = \mathbf{1}.$$

In Section 3.4, the KKT conditions for (5) involve the term $\frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^\pm_{ij}}$. By the assumptions above, we see that $h(W(\theta^+ + \theta^-))$ is a function of θ_{ij} through $[W(\theta^+ + \theta^-)]_{ij}$, so that by the chain rule we have

$$\begin{aligned} \frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^\pm_{ij}} &= \frac{\partial h(W(\theta^+ + \theta^-))}{\partial [W(\theta^+ + \theta^-)]_{ij}} \frac{\partial [W(\theta^+ + \theta^-)]_{ij}}{\partial \theta^\pm_{ij}} \\ &= [\nabla h(W(\theta^+ + \theta^-))]_{ij} \, \mathbf{1} \\ &= [\nabla h(W(|\theta|))]_{ij} \mathbf{1}, \end{aligned}$$

this equality is crucial to Lemma 1.

We conclude by discussing three important special cases that satisfy the assumptions above: (1) Linear SEMs, (2) Multilayer perceptrons (MLPs), and (3) Basis expansions.

Linear SEMs. A linear SEM follows the following set of equations:

$$X_j = f_j(X, z_j) = w_j^\top X + z_j, \quad w_j \in \mathbb{R}^d, \quad \forall j \in [d],$$

where $z_j \in \mathbb{R}$ represents the noise following any distribution. Let $W = [w_1 \mid w_2 \mid \cdots \mid w_d] \in \mathbb{R}^{d \times d}$. In this case, all the model parameters are $\theta = W$. The parameters related to node j are $\theta_j = w_j$, thus, each function f_j is only characterized by θ_j. Thus, condition (i) above is clearly satisfied. Furthermore, we have $\theta_{ij} = W_{ij}$ (the (i, j)-th entry of W), where clearly there is no edge from node i to node j if and only if $W_{ij} = 0$. Therefore, condition (ii) above is also satisfied.

Multilayer perceptrons (MLPs). Let a multilayer perceptron (MLP) with h hidden layers and a single activation $\sigma : \mathbb{R} \to \mathbb{R}$ be given by:

$$\mathsf{MLP}(X; A^{(1)}, \dots, A^{(h)}) = \sigma(A^{(h)}\sigma(\dots A^{(2)}\sigma(A^{(1)}x))),$$
$$A^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}, \qquad m_0 = d, \qquad m_h = 1.$$

Then the nonlinear SCM with additive noise can be written as:

$$X_j = f_j(X, z_j) = \mathsf{MLP}(X; A_j^{(1)}, \dots, A_j^{(h)}) + z_j,$$

where $z_j \sim \mathcal{N}(0, 1)$. Let $\theta_j = (A_j^{(1)}, \dots, A_j^{(h)})$ denote the parameters for the j-th MLP, and let $\theta = (\theta_1, \dots, \theta_d)$ denote all model parameters. Define θ_{ij} to be the i-th column of $A_j^{(1)}$. Since $\mathsf{MLP}(X; A_j^{(1)}, \dots, A_j^{(h)})$ is independent of X_i if and only if $\theta_{ij} = 0$ (e.g., Zheng et al., 2020, Proposition 1), we can define $[W(\theta)]_{ij} = \|\theta_{ij}\|_1$. Then, in this case it is easy to check that conditions (i) and (ii) above are satisfied.

Basis expansion. As an alternative to neural networks, we also consider the use of orthogonal basis expansions, as in (Zheng et al., 2020). Let $\{\varphi_r\}_r^\infty$ be an orthonormal basis of functions such that $\mathbb{E}[\varphi_r(X)] = 0$ for each r and

$$f(x) = \sum_{r=1}^{\infty} \alpha_r \varphi_r(x), \qquad \alpha_r = \int_{\mathbb{R}^d} \varphi_r(x) f(x) dx.$$

Consider additive models and one-dimensional expansions as follows:

$$X_j = f_j(X, z_j) = \sum_{i \neq j} f_{ij}(X_i) + z_j = \sum_{i \neq j} \sum_{r=1}^{\infty} \alpha_{ijr} \varphi_r(x_i) + z_j.$$

In this case, we let $\theta = (\alpha_{ijr})_{i,j,r}$ denote all model parameters, $\theta_j = (\alpha_{ijr})_{ir}$ denote all parameters related to node j, and $\theta_{ij} = (\alpha_{ijr})_r$ denote the parameters that model the absence of an edge from node i to node j. Additionally, set $[W(\theta)]_{ij} = \|\theta_{ij}\|_1 = \sum_r |\alpha_{ijr}|$. Similarly, it is easy to check that conditions (i) and (ii) above are both satisfied.

B. Algorithm Details

B.1. Full Algorithm Description

A full and reproducible outline of Algorithm 1 can be found in Algorithm 2. Note that Algorithms 4 (UPDATESORT) and 3 (FINDPARAMS) are subroutines used by Algorithm 2.

B.2. Additional Details on Hyperparameters

In this section, we describe more details of the proposed order-based search method in Algorithm 2. This involves initializing the number of swapping pairs s_{small} to define a small search space, the number of swapping pairs s_{large} to define a large search space, and the maximum number of searches s_0 to perform in the large swapping-pairs space. From line 31 to 32, for each iteration, Algorithm 2 invokes Algorithm 3 to find out the best values for $\tau_*, \xi^*, \tau^*, \xi_*$ to control the number of swapping pairs in the search space. For reference, the predefined T and Ξ that we used in Algorithm 3 are given in Table 5.

By tuning s_{small} and s_{large}, we can control Algorithm 2 to quickly converge to a local minimum, or have the chance to escape to a better local minimum in case it finds a suboptimal one. Specifically, in Line 4, if the running solution θ_π^* does not satisfy the KKT conditions, then, as prescribed by Lemma 3, a better topological sort can be found. In Line 10, although the running solution θ_π^* satisfies the KKT conditions, we use strict positive values for τ_*, and ξ^* to expand the search space $\mathcal{Y}$ and consider potential swapping pairs that can lead us to a better local minimum. Here, the parameter s_0 specifies how many times the algorithm can search in a large swapping-pairs space, with the goal to escape from a region of bad local minima. These hyperparameters are determinant to Algorithm 2 performance and are tuned to balance the trade-off between accuracy and efficiency.

Remark 7. *One merit of Algorithm 2 over prior work is that it does not only search for a local minimum but also tries to escape from bad local minima. Thus, our algorithm usually attains the best scores among all DAG learning*

Algorithm 2 TOPO

Require: Given a topological sort π, two predefined numbers of swapping pairs $s_{\text{small}}, s_{\text{large}}$, number of search in large space s_0 and initialize corresponding $\mathcal{Z}_\pi$. Solve (7) to get Θ^*_π, set $k \leftarrow 0$ and $count \leftarrow 0$.

$(\tau_*, \xi^*) \leftarrow$ FINDPARAMS$(\Theta^*_\pi, s_{\text{small}})$
$(\tau^*, \xi_*) \leftarrow$ FINDPARAMS$(\Theta^*_\pi, s_{\text{large}})$.
while $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*) \neq \emptyset$ **do**
  if $\mathcal{Y}(\Theta^*_\pi, 0, 0) \neq \emptyset$ **then**
    $\mathcal{Y} \leftarrow \mathcal{Y}(\Theta^*_\pi, 0, 0)$
    for $(i, j) \in \mathcal{Y}$ **do**
      $\pi_{ij} \leftarrow$ UPDATESORT$(\theta^*_\pi, (i, j), opt = 2)$.
      Solve (7) to obtain $\Theta^*_{\pi_{ij}}$.
    end for
  else
    $\mathcal{Y} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$
    for $(i, j) \in \mathcal{Y}$ **do**
      $\pi_{ij} \leftarrow$ UPDATESORT$(\theta^*_\pi, (i, j), opt = 1)$.
      Solve (7) to obtain $\Theta^*_{\pi_{ij}}$.
    end for
  end if
  if $\min_{(i,j)\in\mathcal{Y}} Q(\Theta^*_{\pi_{ij}}) \leq Q(\Theta^*_\pi)$ **then**
    update $\pi := \arg\min_{\pi_{ij}} Q(\Theta^*_{\pi_{ij}})$
  else
    if $k < s_0$ **then**
      $\mathcal{Y} \leftarrow \mathcal{Y}(\Theta^*_\pi, \tau^*, \xi_*) \neq \emptyset$
      for $(i, j) \in \mathcal{Y}$ **do**
        $\pi_{ij} \leftarrow$ UPDATESORT$(\theta^*_\pi, (i, j), opt = 1)$.
        Solve (7) to obtain $\Theta^*_{\pi_{ij}}$.
      end for
      if $\min_{(i,j)\in\mathcal{Y}} Q(\Theta^*_{\pi_{ij}}) \leq Q(\Theta^*_\pi)$ **then**
        update $\pi := \arg\min_{\pi_{ij}} Q(\Theta^*_{\pi_{ij}})$, $k \leftarrow k + 1$
      else
        Return Θ^*_π **then break**
      end if
    else
      Return Θ^*_π **then break**
    end if
  end if
  Solve (7) to obtain Θ^*_π
  Update $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$
  $count \leftarrow count + 1$
  $(\tau_*, \xi^*) \leftarrow$ FINDPARAMS$(\Theta^*_\pi, s_{\text{small}})$
  $(\tau^*, \xi_*) \leftarrow$ FINDPARAMS$(\Theta^*_\pi, s_{\text{large}})$.
end while

Algorithm 3 FINDPARAMS

Require: Parameter Θ, integer q that controls the size of the search space.
Create a predefined $T = \{\tau_1, \ldots, \tau_m\}$ and $\Xi = \{\xi_1, \ldots, \xi_l\}$.
Ensure: $(\tau, \xi) = \arg\min_{\tau\in T, \xi\in\Xi} \; |q - |\mathcal{Y}(\Theta, \tau, \xi)||$

algorithms—when comparable. A drawback of Algorithm 2 is the dependence on how large the search space is, which could be computationally intensive. Finally, Algorithm 2 solves (13) repeatedly, whose runtime also heavily determines the efficiency of Algorithm 2. See Section E for runtime comparisons against existing methods.

Algorithm 4 UPDATESORT

Require: Parameter θ or topological sort π, (i, j), opt. Initialize predefined $\epsilon \leftarrow 10^{-8}$ (small).
1: **if** $opt = 1$ **then**
2: Swap nodes i and j, and denote the new topological sort by π_{ij}
3: **else**
4: $\theta' \leftarrow \theta$
5: $\theta'_{ij} \leftarrow \theta_{ij} - \epsilon \frac{\partial Q(\Theta)}{\partial \theta_{ij}}$
6: Find the topological sort of $W(|\theta'|)$, denoted as π_{ij}.
7: **end if**
Ensure: π_{ij}

Parameters	Values						
T	0	1×10^{-8}	1×10^{-7}	1×10^{-6}	1×10^{-5}	1×10^{-4}	1×10^{-3}
Ξ	0	1×10^{-7}	1×10^{-6}	5×10^{-6}	1×10^{-5}	5×10^{-5}	1×10^{-4}
	5×10^{-4}	1×10^{-3}	5×10^{-3}	1×10^{-2}	5×10^{-2}	1×10^{-1}	5×10^{-1}
	1	2	5	10	15	20	40

Table 5. Suggested values for parameters T and Ξ in Algorithm 3.

C. Irreducibility and Comparison with Prior Work

C.1. Three-Node Example where KKTS and NOTEARS Fail

In this section, we expand on Example 1. In particular, we show that our example was **not handpicked** but instead there exists several values a and b where the solutions from KKTS and NOTEARS either are DAGs with incorrect structure, or are non-optimal solutions, or both. Recall that our example follows the following SEM:

$$\begin{aligned} X_1 &= z_1, \\ X_2 &= aX_1 + z_2, \\ X_3 &= bX_2 + z_3, \end{aligned} \tag{11}$$

where $z_i \sim \mathcal{N}(0, 1)$ for $i \in [3]$. For the purposes of this analysis we consider the class of SEMs such that $a^2 > b^2$. Then, the **true** adjacent matrix and topological sort are:

$$W_{\text{true}} = \begin{pmatrix} 0 & a & 0 \\ 0 & 0 & b \\ 0 & 0 & 0 \end{pmatrix}, \qquad \pi_{\text{true}} = [1, 2, 3].$$

Letting $X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$, and $Z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}$. The SEM (11) in vector form can be written as:

$$X = W_{\text{true}}^T X + Z.$$

We will use the population least square (LS) as the score function, which is defined as follows:

$$Q(W) = \mathbb{E}\|X - XW\|_2^2 = \|(I - W_{\text{true}})^{-1}(I - W)\|_2^2$$

The motivation to choose such score is that it was shown by Loh & Bühlmann (2014) that W_{true} is the unique global minimizer of the population LS for linear SEMs with equal noise variances. Next, we provide a closer look as to why our algorithm is capable of learning the correct structure, while KKTS and NOTEARS fail.

C.1.1. The Output of KKTS

In the KKTS algorithm of Wei et al. (2020), consider the set of edge absence constraints to be initialized at:

$$\mathcal{Z}_0 = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)\}.$$

That is, the algorithm is initialized at the empty graph. Then, we have

$$W^*(\mathcal{Z}_0) = 0, \qquad \nabla Q(W^*(\mathcal{Z}_0)) = -2\begin{bmatrix} 1 & a & ab \\ a & a^2+1 & b+a^2b \\ ab & b+a^2b & (ab)^2+b^2+1 \end{bmatrix},$$

and $\{(i,j) \mid [\nabla h(|W^*(\mathcal{Z}_0)|)]_{ij} = 0\} = \{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)\}$. Recall that the KKTS algorithm will remove the pair (i,j) from $\mathcal{Z}_0$ that satisfies the following property:

$$(i,j) = \underset{\{(i,j) \mid [\nabla h(|W^*(\mathcal{Z}_0)|)]_{ij}=0\} \cap \mathcal{Z}_0}{\arg\max} |[\nabla Q(W^*(\mathcal{Z}_0))]_{ij}|.$$

Now, consider the case that $\max\{|a|,|ab|\} < |(a^2+1)b|$, then the pair $(3,2)$ is removed from $\mathcal{Z}_0$ and the resulting set of edge absence constraints is $\mathcal{Z}_1 = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,3)\}$. Then, at the next step we have:

$$W^*(\mathcal{Z}_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{a^2b+b}{(ab)^2+b^2+1} & 0 \end{bmatrix}, \qquad \nabla Q(W^*(\mathcal{Z}_1)) = -2\begin{bmatrix} 1 & \frac{a}{(ab)^2+b^2+1} & ab \\ a & \frac{a^2+1}{(ab)^2+b^2+1} & a^2b+b \\ ab & 0 & (ab)^2+b^2+1 \end{bmatrix},$$

and $\{(i,j) \mid [\nabla h(|W^*(\mathcal{Z}_1)|)]_{ij} = 0\} = \{(1,2),(1,3),(2,1),(3,1),(3,2)\}$. Consider now the case that $\max\{\frac{|a|}{(ab)^2+b^2+1}, |ab|\} < |a|$, then the pair $(2,1)$ is removed from $\mathcal{Z}_1$ and the resulting set of edge absence constraints is $\mathcal{Z}_2 = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,1),(3,3)\}$. Then, at the next step we have:

$$W^*(\mathcal{Z}_2) = \begin{bmatrix} 0 & 0 & 0 \\ \frac{a}{a^2+1} & 0 & 0 \\ 0 & \frac{a^2b+b}{(ab)^2+b^2+1} & 0 \end{bmatrix}, \qquad \nabla Q(W^*(\mathcal{Z}_2)) = -2\begin{bmatrix} \frac{1}{a^2+1} & \frac{a}{(ab)^2+b^2+1} & ab \\ 0 & \frac{a^2+1}{(ab)^2+b^2+1} & a^2b+b \\ 0 & 0 & (ab)^2+b^2+1 \end{bmatrix},$$

and $\{(i,j) \mid [\nabla h(|W^*(\mathcal{Z}_2)|)]_{ij} = 0\} = \{(2,1),(3,1),(3,2)\}$.

Now we note that $\{(i,j) \mid [\nabla h(|W^*(\mathcal{Z}_2)|)]_{ij} = 0\} \cap \mathcal{Z}_2 = \{(3,1)\}$. However, we have $[\nabla Q(W^*(\mathcal{Z}_2))]_{31} = 0$, that is, even if we remove the pair $(3,1)$ from $\mathcal{Z}_2$, the corresponding $\mathcal{Z}_3 = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$ leads to

$$W^*(\mathcal{Z}_3) = W^*(\mathcal{Z}_2), \qquad \nabla Q(W^*(\mathcal{Z}_3)) = \nabla Q(W^*(\mathcal{Z}_2)).$$

Combining all the considerations on a and b, we can conclude that as long as the following is satisfied:

$$|ab| < |a| < |(a^2+1)b|,$$

KKTS will find a DAG with incorrect structure, in fact, a DAG where all edges are reversed. Finally, one can easily see that there are infinitely many a and b satisfying the above condition. For example, let $a = 1$ and $b = -0.55$. We want to emphasize here that our result is also consistent with the result returned by the Python program provided by Wei et al. (2020).

C.1.2. The Output of NOTEARS

Since the NOTEARS implementation by Zheng et al. (2018) uses the augmented Lagrangian method and solve each inner unconstrained subproblem using the Quasi-Newton L-BFGS method, it is impossible to derive an analytical solution. Instead, we directly verify the DAG solution returned by NOTEARS by setting the ground-truth SEM $a = 1$, $b = -0.55$.

$$W_{\text{notears}} = \begin{bmatrix} 0 & 1.49 \times 10^{-4} & -7.00 \times 10^{-7} \\ 0.16 & 0 & \text{-1.55} \\ \text{-0.22} & -1.59 \times 10^{-5} & 0 \end{bmatrix}$$

We can note that W_{notears} is not 'exactly' a DAG. Thus, we use a threshold to remove small entries in W_{notears}. The resulting adjacency matrix is now a DAG and is denoted by $W_{\text{notears\_thres}}$. One can clearly see that the NOTEARS solution does not recover the true structure.

$$W_{\text{notears\_thres}} = \begin{bmatrix} 0 & 0 & 0 \\ 0.16 & 0 & \text{-1.55} \\ \text{-0.22} & 0 & 0 \end{bmatrix}$$

Let us now check the KKT conditions given in Lemma 1. We have $\{(i,j) \mid [\nabla h(|W_{\text{notears\_thres}}|)]_{ij} = 0\} = \{(3,1),(2,3),(2,1)\}$, and $[\nabla Q(W_{\text{notears\_thres}})]_{21} \neq 0$, $[\nabla Q(W_{\text{notears\_thres}})]_{23} \neq 0$ and $[\nabla Q(W_{\text{notears\_thres}})]_{31} \neq 0$. We can then observe that NOTEARS fails at both outputting the true DAG structure, and a DAG that is a local minimum.

C.1.3. The Output of TOPO (Algorithm 2)

To study how TOPO works for this 3-node example, we first calculate the loss of all possible topological sorts in the following table.

Topological sort π	Score
$(1,2,3)$	3
$(1,3,2)$	$2+b^2+\frac{1}{1+b^2}$
$(2,1,3)$	$2+a^2+\frac{1}{a^2+1}$
$(3,1,2)$	$1+b^2+(ab)^2+\frac{1}{1+b^2}+\frac{1+b^2}{1+b^2+(ab)^2}$
$(2,3,1)$	$2+a^2+\frac{1}{a^2+1}$
$(3,2,1)$	$\frac{1}{1+a^2}+\frac{1+a^2}{1+(ab)^2+b^2}+1+b^2+(ab)^2$

Now, rewriting eq.(8) for the case of linear models, we have:

$$W_\pi^* \in \arg\min_{W \sim \pi} Q(W).$$

To understand why TOPO is capable of returning the correct true structure, we next define adjacent topological sorts.

Definition 4. *For two topological sorts π_1 and π_2, we say that π_1 and π_2 are* adjacent *if there exists a pair of nodes in π_1 such that when swapped the resulting topological ordering is π_2.*

Recall that $\pi_{\text{true}} = (1,2,3)$. The following statement explains precisely the success of Algorithm 2.

Corollary 3. *Assume that $a \neq 0$, $b \neq 0$, and $a^2 > b^2$, then for any topological sort $\pi \neq \pi_{true}$, we have that $Q(W_\pi^*) > Q(W_{\pi_{true}}^*) = Q(W_{true}) = 3$. Moreover, there always exists an adjacent topological sort π_{adj} such that $Q(W_\pi^*) > Q(W_{\pi_{adj}}^*)$. In other words, for any initial topological sort $\pi \neq (1,2,3)$, TOPO (Algorithm 2) can always return π_{true} and W_{true} at last.*

All the situations are summarized in the Table 6.

Current order π	Adjacent order π_{adj}	Current loss		Better loss
$(1,2,3)$	$(1,2,3)$	3	$\geq$	3
$(1,3,2)$	$(1,2,3)$	$2+b^2+\frac{1}{1+b^2}$	$\geq$	3
$(2,1,3)$	$(1,2,3)$	$2+a^2+\frac{1}{a^2+1}$	$\geq$	3
$(3,1,2)$	$(1,3,2)$	$1+b^2+(ab)^2+\frac{1}{1+b^2}+\frac{1+b^2}{1+b^2+(ab)^2}$	$\geq$	$2+b^2+\frac{1}{1+b^2}$
$(2,3,1)$	$(1,3,2)$	$2+a^2+\frac{1}{a^2+1}$	$\geq$	$2+b^2+\frac{1}{1+b^2}$
$(3,2,1)$	$(1,2,3)$	$\frac{1}{1+a^2}+\frac{1+a^2}{1+(ab)^2+b^2}+1+b^2+(ab)^2$	$\geq$	3

Table 6. There always exists an adjacent topological sort whose score is strictly less than the current topological sort.

C.1.4. Analysis

In this section we aim to provide more intuition as to why KKTS and NOTEARS fail in the above example. Regarding the KKTS algorithm, it removes a pair (i,j) from the set of edge absence constraints at each iteration. Loosely speaking,

this is equivalent to adding an edge $X_i \to X_j$ at each iteration, which implies that node i must appear before node j in the topological sort, and such relative ordering in the topological sort *will never be reversed* in later iterations of the algorithm. Therefore, once a wrong pair (i, j) is removed from the set of edge absence constraints, the KKTS algorithm has no ability to correct such erroneous step. Although KKTS ensures that a local minimum is returned, it can learn a completely erroneous DAG structure, as shown in our example above. Regarding NOTEARS, as indicated by Wei et al. (2020), the algorithm does not guarantee to return a local minimum even under the right formulation, and in most cases the NOTEARS solution is neither the correct structure nor a local minimum.

In contrast to KKTS and NOTEARS, TOPO can return correct structure in the example above regardless of the initial topological sort. Our swapping strategy allows TOPO to change the topological sort in each iteration; importantly, while TOPO checks the optimality conditions, it uses the score function as the only criterion to find another topological ordering with better score, thus, jumping from one local optimum to a better local optimum.

C.2. Detailed Discussion About Irreducibility

In this section, we discuss the differences between our method and the KKTS algorithm (Wei et al., 2020). One obvious difference is the type of constraint used. Another important difference is that the KKTS algorithm proposed by Wei et al. (2020) relies on an assumption they call *irreducibility*, which we will show by example is **not needed** in general.

In this section, we consider one of special case of (2): Linear SEMs, which is studied by previous work (Zheng et al., 2018; Wei et al., 2020).

$$X_j = w_j^\top X + z_j, \quad w_j \in \mathbb{R}^d. \tag{12}$$

Let $W = [w_1, \ldots, w_d]$. In this case, θ is equivalent to W and θ_{ij} is equivalent to W_{ij}. Therefore, to be consistent with previous works (Zheng et al., 2018; Wei et al., 2020), we use W to replace θ.

To connect the KKT conditions and local minimiality, Wei et al. (2020) used a related problem with explicit edge absence constraints, which correspond to zero-value constraints on the matrix W. Specifically, given a set $\mathcal{Z} \subset V \times V$, their explicit edge absence constrained problem is given by:

$$\min_W \; Q(W; \mathbf{X}) \quad \text{subject to} \quad W_{ij} = 0, \forall\, (i,j) \in \mathcal{Z}. \tag{13}$$

Following the notation in Wei et al. (2020), we denote its optimal solution by $W^*(\mathcal{Z})$. As with (7), this problem can be solved efficiently—in fact, (7) is just a special case of (13) with $\mathcal{Z} = \mathcal{Z}_\pi$, where

$$\mathcal{Z}_\pi := \{(\pi(j), \pi(i)) \mid i < j\}.$$

Driven by Theorem 1, the KKT-informed local search (KKTS) algorithm in Wei et al. (2020) repeatedly solves the edge absence problem (13) for different $\mathcal{Z}$. The KKTS algorithm stops once an *irreducible* set $\mathcal{Z}$ is found, and the output $W^*(\mathcal{Z})$ is guaranteed to be a local minimum for problem (4). Wei et al. (2020) define irreducibility of a set $\mathcal{Z}$ as follows:

Definition 5 (Irreducibility, Wei et al., 2020)**.** *A set $\mathcal{Z}$ is called irreducible if $(i,j) \in \mathcal{Z} \Rightarrow (\nabla h(|W^*(\mathcal{Z})|))_{ij} > 0$.*

Although irreduciblity of $\mathcal{Z}$ is a sufficient condition for a feasible solution to be a KKT point (Theorem 8, Wei et al., 2020), it is not necessary, as the following example shows.

Fix indices $i_0 < j_0$ and define a ground truth DAG $W^\dagger$ by

$$(W^\dagger)_{i,j} = \begin{cases} 1 & \text{if } i = i_0, j = j_0, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\Omega = [d] \times [d]$ denote all pairs of indices, we initialize $\mathcal{Z}_0 = \Omega \setminus \{(i_0, j_0)\}$. Recall that the KKTS algorithm repeatedly removes elements from $\mathcal{Z}$ until it is irreducible. Let us assume KKTS takes m steps, the elements removed in order are $(i_1, j_1), (i_2, j_2), \ldots (i_m, j_m)$ and define $\mathcal{Z}_k = \Omega \setminus \{(i_0, j_0), (i_1, j_1) \ldots, (i_k, j_k)\}$.

Lemma 4. *Assume Q is separable and that $W^\dagger$ is the unique global minimum of (4). Then initializing KKTS with $\mathcal{Z}_0 = \Omega \setminus \{(i_0, j_0)\}$, we have the following:*

1. *$W^*(\mathcal{Z}_k) = W^\dagger$ for each $k = 0, 1, \ldots m$,*

2. $\mathcal{Z}_0, \ldots, \mathcal{Z}_{m-1}$ *are not irreducible,*

3. $\mathcal{Z}_m = \{(j_0, i_0)\} \cup \{(i,i) | i = 1, \ldots, d\}$ *is irreducible.*

In other words, the global minimum $W^\dagger$ is a KKT point that is not always irreducible, although it can be written in terms of an irreducible $\mathcal{Z}$. It is easy to construct models (12) and score functions Q such that $W^\dagger$ is a global minimizer: Simply choose the population least squares score with $z_j \sim \mathcal{N}(0,1)$ for each j; see Loh & Bühlmann (2014) for details.

Corollary 4. *Irreduciblity of $\mathcal{Z}$ is sufficient but not necessary for KKTS to find a KKT point of problem 5.*

Lemma 4 has direct implications when the underlying DAG is sparse. If the initial $\mathcal{Z}_0 = \Omega$, KKTS needs to remove most of the elements in $\mathcal{Z}_0$ to reach an irreducible $\mathcal{Z}$, thus, it can be computationally intense and inefficient. Moreover, the score function in KKTS has a sparse regularization, and it can return the wrong $W^*(\mathcal{Z})$ even if the current $\mathcal{Z}$ characterizes the edge absences of the ground truth exactly.

D. Proofs of Technical Results

In this section, we present the proofs of lemmas and theorems in detail. First, let us discuss more on how to solve problem (8) in the algorithm 2 which is helpful for our proof. In problem (8), we can eliminate the constraint $\theta_{\pi(i),\pi(j)} = 0, \forall i > j$ by plugging them back to the objective function, then it is equivalent to the following unconstrained optimization problem, Note that we can write $\Theta = (\theta_{\pi(i),\pi(j)}, \theta_{\pi(m),\pi(n)}, \tilde{\theta})$, where $i > j$ and $n > m$. In this case, $\Theta = (0, \theta_{\pi(m),\pi(n)}, \tilde{\theta})$

$$(\{\theta^*_{\pi(m),\pi(n)}\}_{n>m}, \tilde{\theta}^*) = \arg\min Q(\Theta) = \arg\min Q((0, \theta_{\pi(m),\pi(n)}, \tilde{\theta}))$$

Therefore, we can use any off-shelf optimizer that can solve such unconstrained optimization to a stationary point that will be suitable for our purpose, i.e., gradient descent or Adam (Kingma & Ba, 2014). Throughout the proof, we repeatedly use the fact that $\frac{\partial Q(\Theta)}{\partial \tilde{\theta}^*} = 0$ and $\frac{\partial Q(\Theta)}{\partial \theta^*_{\pi(m),\pi(n)}} = 0, \forall\, n > m, \forall\, \pi$. At last, we can construct θ^+, θ^- from θ by $\theta^+ = \max\{\theta, 0\}$ and $\theta^- = \max\{-\theta, 0\}$.

D.1. The Extension of Theorem 1

The proof of Theorem 1 in (Wei et al., 2020) is for the case where the adjacency matrix W does not have any parametrization. For completeness and ease of reference, we state the generalization to general parametrizations here:

Theorem 4 (Theorem (1) in our content)**.** *Assume that $Q(\theta)$ is convex. Then if $(\theta^+, \theta^-, \tilde{\theta})$ satisfies the KKT condition in (6), $(\theta^*, \tilde{\theta})$ is a local minimum for problem (4), where $\theta^* = \theta^+ - \theta^-$.*

Although the proof is similar, we include a proof for completeness in Appendix D.10.

D.2. Proof of Lemma 1

Proof. Let us denote θ_{ij} as $(\theta_{ijr})_r$, here note θ_{ij} is a vector and its each component is denoted as θ_{ijr}, where $r = 1, \ldots$. Therefore,

$$\frac{\partial Q(\Theta)}{\partial \theta^\pm_{ij}} = \left(\frac{\partial Q(\Theta)}{\partial \theta^\pm_{ijr}}\right)_r$$

Let us simplify term $\frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^\pm_{ij}}$. First, note that

$$[W(\theta^+ + \theta^-)]_{ij} = \|\theta^+_{ij} + \theta^-_{ij}\|_1 = \mathbf{1}^\top(\theta_{ij} + \theta^-_{ij}) = \sum_r (\theta_{ijr} + \theta^-_{ijr})$$

(here we use the fact $\theta^+_{ij} \geq 0, \theta^-_{ij} \geq 0$). Remember $h(W(\theta^+ + \theta^-))$ is a function of θ_{ij} through $[W(\theta^+ + \theta^-)]_{ij}$, we can use chain rule

$$\frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^\pm_{ij}} = \frac{\partial h(W(\theta^+ + \theta^-))}{\partial [W(\theta^+ + \theta^-)]_{ij}} \frac{\partial [W(\theta^+ + \theta^-)]_{ij}}{\partial \theta^\pm_{ij}}$$

$$
\begin{aligned}
&= [\nabla h(W(\theta^+ + \theta^-))]_{ij}\, \mathbf{1} \\
&= [\nabla h(W(|\theta|))]_{ij}\, \mathbf{1}
\end{aligned}
$$

First, for any (i, j) such that

$$
[\nabla h(W(|\theta|))]_{ij} = \left[\nabla h(W(\theta^+ + \theta^-))\right]_{ij} > 0,
$$

we set

$$
\lambda > \max_{(i,j):[\nabla h(W(|\theta|))]_{ij}>0} \frac{\|\partial Q(\Theta)/\partial \theta^{\pm}_{ij}\|_1}{[\nabla h(W(|\theta|))]_{ij}}.
$$

Therefore, (6a) and (6b) are satisfied with $M^+_{ij} > 0$ and $M^-_{ij} > 0$. From condition (i), we have $\theta_{ij} = 0$, that is, $\theta^{\pm}_{ij} = 0$, thus, (6c) is satisfied since $\theta^+_{ij} \circ M^+_{ij} = \theta^-_{ij} \circ M^-_{ij} = 0$.

Second, for any (i, j) such that

$$
[\nabla h(W(|\theta|))]_{ij} = \left[\nabla h(W(\theta^+ + \theta^-))\right]_{ij} = 0,
$$

we have from (6a) and (6b)

$$
\frac{\partial Q(\Theta)}{\partial \theta^+_{ij}} = M^+_{ij} \geq 0. \quad -\frac{\partial Q(\Theta)}{\partial \theta^-_{ij}} = M^-_{ij} \geq 0.
$$

It is also known that

$$
\frac{\partial Q(\Theta)}{\partial \theta^+_{ij}} = \frac{\partial Q(\Theta)}{\partial \theta^-_{ij}}
$$

From condition (ii), we set corresponding $M^{\pm}_{ij} = 0$, then (6a) is satisfied. We also have $\theta^{\pm}_{ij} \circ M^{\pm}_{ij} = 0$, hence (6c) is satisfied. From (iii), (6d) is satisfied. From (iv), we know $\theta^+ \geq, \theta^- \geq 0$. Also, it is obvious that $\forall (i, j)$, we have $\theta \circ \nabla h(\theta^+ + \theta^-) = 0$, it is equivalent to $h(\theta^+ + \theta^-) = 0$ (Wei et al., 2020, Lemma 4). The feasibility conditions in (5) are also satisfied. Thus, (θ^+, θ^-) satisfies the KKT conditions in (6).

Finally, from Theorem 4, $(\theta^+ - \theta^-, \tilde{\theta})$ is a local minimum for problem (4) if $Q(\Theta)$ is convex. □

D.3. Proof of Lemma 2

Proof. Assume $p < q$, node $\pi(p)$ comes before $\pi(q)$ in π by the definition of topological sort, so there is no directed walk from $\pi(q)$ to $\pi(p)$, which implies $(\nabla h(W(|\theta^*_\pi|)))_{\pi(p),\pi(q)} = 0$ (Wei et al., 2020, Lemma 7) and $(W(|\theta^*_\pi|))_{\pi(q),\pi(p)} = 0$. By the optimality conditions of (8), $\frac{\partial Q(\Theta^*_\pi)}{\partial \theta_{\pi(p),\pi(q)}} = 0$. In other word, possible elements in $\mathcal{Y}(\Theta^*_\pi, 0, 0)$ must has formula $(\pi(q), \pi(p))$ where $p < q$. Therefore, $(\theta^*_\pi)_{\pi(q),\pi(p)} = 0$. By the definition of $[W(|\theta^*_\pi|)]_{\pi(q),\pi(p)} = \|(\theta^*_\pi)_{\pi(q),\pi(p)}\|_1 = 0$ □

D.4. Proof of Lemma 3

Proof. Let any $(i, j) \in \mathcal{Y}(\Theta^*_\pi, 0, 0)$, then $(\nabla h(W(|\theta^*_\pi|)))_{ij} = 0$ and $\frac{\partial Q(\Theta^*_\pi)}{\partial \theta_{ij}} \neq 0$, it indicates there is no directed walk from $\boldsymbol{j}$ to such $\boldsymbol{i}$. From Lemma 2, $(\theta^*_\pi)_{ij} = 0$. Changing the value of $(\theta^*_\pi)_{ij}$ introduces new edge which can create a cycle, however, from Lemma 6 in Wei et al. (2020), changing the value of $(\theta^*_\pi)_{ij}$ cannot create directed walks from j to i, by the assumption of separability of $Q(\Theta)$ and following the same argument of proof of Lemma 8 in Wei et al. (2020), changing the value of $(\theta^*_\pi)_{ij}$ will not create cycle. Therefore, $(\theta^*_\pi)_{ijr}$ can be increased or decreased ($\frac{\partial Q(\theta^*_\pi)}{\partial \theta_{ijr}} < 0$ or $\frac{\partial Q(\theta^*_\pi)}{\partial \theta_{ijr}} > 0$) to reduce the loss function while maintaining feasible, which implies $W(|\widetilde{\theta}|)$ in **Algorithm 4** is still a DAG and $Q(\widetilde{\theta}) < Q(\theta^*_\pi)$. $W(|\widetilde{\theta}|)$ follows the topological sort π_{ij}, so $Q(\Theta^*_{\pi_{ij}}) \leq Q(\widetilde{\theta}) < Q(\theta^*_\pi)$. □

D.5. Proof of Lemma 4

Proof. For $\mathcal{Z}_0 = \Omega \backslash \{(i_0, j_0)\}$, $W^*(\mathcal{Z}_0)$ is obviously a DAG, $W^\dagger$ is global minimum of problem (4), then $Q(W^\dagger) \leq Q(W^*(\mathcal{Z}_0))$. $W^\dagger$ is also a feasible solution for problem (13) with $\mathcal{Z}_0$, then $Q(W^*(\mathcal{Z}_0)) \leq Q(W^\dagger)$. $W^\dagger$ is unique by

assumption, thus $W^\dagger = W^*(\mathcal{Z}_0)$. For $\mathcal{Z}_1 = \Omega\backslash\{(i_0, j_0), (i_1, j_1)\}$, we can use the same arguments. KKTS continues until $Z_l = \Omega\backslash\{(i_0, j_0), (i_1, j_1), \ldots, (i_l, j_l)\}$ can not guarantee the solution $W^*(\mathcal{Z}_l)$ to be a DAG. For example, if

$$\begin{aligned}\mathcal{Z}_{l-1} =& \Omega\backslash\{(i,j)|i<j, i=1,\ldots,d\}\\ \mathcal{Z}_l =& \mathcal{Z}_{l-1}\backslash\{(i_m, j_m)\}\end{aligned}$$

The only requirement for (i_m, j_m) is $i_m > j_m$. Followed by the same argument, we know $W^*(\mathcal{Z}_{l-1}) = W^\dagger$. Using Lemma 8 from Wei et al. (2020), $W^*(\mathcal{Z}_l)$ is also a DAG, hence $Q(W^\dagger) \leq Q(W^*(\mathcal{Z}_l))$. Besides, $W^\dagger$ is also a feasible solution for problem (13) with $\mathcal{Z}_l$, thus $Q(W^*(\mathcal{Z}_l)) \leq Q(W^\dagger)$. $W^\dagger$ is unique by assumption, so $W^\dagger = W^*(\mathcal{Z}_l)$. By the same arguments, KKTS continues until an irreducible $Z_m = \{(j_0, i_0)\} \cup \{(i,i)|i = 1,\ldots,d\}$ is returned. □

D.6. Proof of Corollary 2

Proof. Because $\mathcal{Y}((\theta^*_\pi), 0, 0) = \emptyset$, we know for any (i, j) such that $[\nabla h(W(|\theta^*_\pi|))]_{ij} = 0$, we have $\frac{\partial Q(\Theta^*_\pi)}{\partial \theta_{ij}} = 0$, (ii) in Lemma 1 is satisfied. Therefore, we only need prove for any (i, j) such that $[\nabla h(W(|\theta^*_\pi|))]_{ij} > 0$, then $(\theta^*_\pi)_{ij} = 0$, i.e. $[W(|\theta^*_\pi|)]_{ij} = 0$. Because $[\nabla h(W(|\theta^*_\pi|))]_{ij} > 0$ implies there exist a directed walk from j to i, which means node j appear before node i in topological sort, so $\theta_{ij} = 0$. Thus, (i) in Lemma 1 is also satisfied. The explanation given at the start of the Section D fulfills condition (iii). (iv) is satisfied naturally by our construction. Therefore, Θ^*_π is a KKT point by Lemma 1. □

D.7. Proof of Corollary 4

Proof. Follows from Lemma 4. □

D.8. Proof of Theorem 2

Proof. For any $p < q$, $[\nabla h(W(|\theta^*_\pi|))]_{\pi(q),\pi(p)} > 0$ by definition of connected estimator. Because $\pi(p)$ appears before $\pi(q)$ in the topological sort, $[W(|\theta^*_\pi|)]_{\pi(q),\pi(p)} = 0$, i.e., $(\theta^*_\pi)_{\pi(q),\pi(p)} = 0$. All pairs $(\pi(q), \pi(p))$ for $p < q$ satisfies Lemma 1 condition (i). By the same argument from proof of corollary 2, all pairs $(\pi(p), \pi(q))$ for $p < q$ satisfies Lemma 1 condition (ii). Condition (iii) is satisfied by the reasoning presented at the beginning of Section D. (iv) is satisfied naturally by our construction. Therefore, Θ^*_π is KKT point, by Theorem 1, it is also a local minimum if Q is convex. Under the connected estimator assumption, the solution at each iteration is a local minimum if Q is convex. □

D.9. Proof of Theorem 3

Proof. If $\mathcal{Y}(\theta^*_\pi, 0, 0) \neq \emptyset$, we can always construct a new topological sort π_{ij} by Lemma 3 and strictly decreases score function. Otherwise, Algorithm searches in space $\mathcal{Y}(\Theta^*_\pi, \tau_*, \xi^*)$ or $\mathcal{Y}(\Theta^*_\pi, \tau^*, \xi_*)$ to find better topological sort until it cannot. Note that at last iteration, it must be that $\mathcal{Y}(\Theta^*_\pi, 0, 0) = \emptyset$, such θ^*_π is KKT point, i.e. local minimum if Q is convex by Theorem 4. □

D.10. Proof of Theorem 4

Before we jump into the proof, let us first consider the problem

$$\min_{\Theta} \quad Q(\Theta) \qquad \text{subject to} \quad \theta_{ij} = 0, \quad (i,j) \in \mathcal{Z} \tag{14}$$

Remember the definition $\tilde{\theta} = \Theta \setminus \theta$ .The necessary conditions of optimality for (14) are

$$\frac{\partial Q(\Theta)}{\partial \theta_{ij}} = 0, \quad (i,j) \notin \mathcal{Z} \tag{15a}$$

$$\theta_{ij} = 0, \quad (i,j) \in \mathcal{Z} \tag{15b}$$

$$\frac{\partial Q(\Theta)}{\partial \tilde{\theta}} = 0 \tag{15c}$$

Given a KKT point $(\theta^+, \theta^-, \tilde{\theta})$ in (6), we can define the set

$$\mathcal{P} := \{(i,j) : [\nabla h(W(\theta^+ + \theta^-))]_{ij} > 0\} \tag{16}$$

Although set $\mathcal{P}$ doesn't appear in Theorem 4 explicitly, but it appears in Lemma 5 which is key to prove the Theorem 4.

Lemma 5. *If $(\theta^+, \theta^-, \tilde{\theta})$ satisfies the KKT conditions in (6), then $\Theta^* = (\theta^*, \tilde{\theta})$ satisfies the optimality conditions in (15) for $\mathcal{Z} = \mathcal{P}$ which is defined in (16), where $\theta^* = \theta^+ - \theta^-$. If in addition $Q(\Theta)$ is convex, then Θ^* is a minimizer of (14) for $\mathcal{Z} = \mathcal{P}$.*

Proof of Theorem 4. Let θ be feasible solution (i.e. $W(|\theta|)$ is a DAG) to (4) with $\|\theta - \theta^*\|_F < \epsilon$ (the Frobenius norm is used for concreteness). Since $\nabla h(W(|\theta|))$ is a continuous function of θ, there exists a sufficiently small $\epsilon > 0$ such that $[\nabla h(W(|\theta|))]_{ij} > 0$ whenever $[\nabla h(W(|\theta^*|))]_{ij} > 0$, in other words for (i, j) in the set $\mathcal{P}$. Then for feasible θ within such an ϵ-ball around θ^*, it follows from the same argument in proof of Lemma 5, $\theta_{ij} = 0$ for $(i, j) \in \mathcal{P}$. θ is therefore a feasible solution to (14) for $\mathcal{Z} = \mathcal{P}$. By Lemma 5 and the convexity of Q, we then have $Q(\Theta^*) \leq Q(\Theta)$ for all feasible θ such that $\|\theta - \theta^*\|_F < \epsilon$. □

D.11. Proof of Lemma 5

Proof. For $(i, j) \notin \mathcal{Z} = \mathcal{P}$, we have $[\nabla h(W(|\theta|))]_{ij} = 0$, because $\frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^{\pm}_{ij}} = [\nabla h(W(|\theta|))]_{ij}\mathbf{1}$, then $\frac{\partial h(W(\theta^+ + \theta^-))}{\partial \theta^{\pm}_{ij}} = \mathbf{0}$. From (6a) and (6b),

$$\frac{\partial Q(\Theta)}{\partial \theta^+_{ij}} = M^+_{ij} \geq 0 \quad -\frac{\partial Q(\Theta)}{\partial \theta^-_{ij}} = M^-_{ij} \geq 0$$

It is also know that $\frac{\partial Q(\Theta)}{\partial \theta^+_{ij}} = \frac{\partial Q(\Theta)}{\partial \theta^-_{ij}}$, so $\frac{\partial Q(\Theta)}{\partial \theta^{\pm}_{ij}} = 0$, it is equivalent to $\frac{\partial Q(\Theta)}{\partial \theta_{ij}} = 0$. It means (15a) is satisfied.

For $(i, j) \in \mathcal{Z} = \mathcal{P}$, we have $[\nabla h(W(|\theta|))]_{ij} > 0$. Since (θ^+, θ^-) is feasible solution, which means $W(|\theta|)$ is a DAG. Moreover, $[\nabla h(W(|\theta|))]_{ij} > 0$ indicates there is a directed path from node j to i, then it implies there is no edge from node i to node j. Hence, $[W(|\theta|)]_{ij} = \|\theta_{ij}\|_1 = \|\theta^+_{ij}\|_1 + \|\theta^-_{ij}\|_1 = 0$, i.e. $\theta^+_{ij} = \theta^-_{ij} = 0$. we conclude $\theta^*_{ij} = \theta^+_{ij} - \theta^-_{ij} = 0$. Now (15b) is satisfied. From (6d), it is obvious (15c) is satisfied.

Therefore, $(\theta^*, \tilde{\theta})$ satisfies the optimality conditions in (15) for $\mathcal{Z} = \mathcal{P}$, where $\theta^* = \theta^+ - \theta^-$. If $Q(\theta)$ is convex function, conditions in (15) is also sufficient for optimality in (14). □

E. Detailed Experiments

E.1. Experimental Setting

Here we describe the details about how to generate graphs and data for Linear SEMs with different noise distributions, fully connected graphs, logistic models and nonlinear models with neural networks. For each model, a random graph $\mathcal{G}$ was generated from one of two random graph models, Erdős-Rényi (ER) or scale-free (SF) with kd edges ($k \in \{1, 2, 4\}$) on average, denoted by ERk or SFk.

- *Erdős-Rényi* (ER), Random graphs whose edges are add independently with equal probability. We simulated models with $d, 2d$ and $4d$ edges (in expectation) each, denoted by $ER1, ER2$, and $ER4$ respectively.
- Scale-free network(SF). Network simulated according to the preferential attachment process (Barabási & Albert, 1999). We simulated scale-free network with $d, 2d$ and $4d$ edges and $\beta = 1$, where β is the exponent used in the preferential attachment process.

Linear SEMs. Given a random DAG $B \in \{0, 1\}^{d\times d}$ from one of these two graphs, we assigned edge weights independently from Unif$([-2, -0.5] \cup [0.5, 2])$ to obtain a weight matrix $W \in \mathbb{R}^{d\times d}$. Given W, we sampled $X = W^\top X + z \in \mathbb{R}^d$ according to the following three noise models:

- *Gaussian noise* with equal variance(`Gauss-EV`). $z \sim \mathcal{N}(0, I_{d\times d})$
- *Gaussian noise* with unequal variance (`Gauss-NV`): $z_i \sim \mathcal{N}(0, \sigma_i^2), i = 1, \ldots, d$ where $\sigma_i \sim$ Unif$[1, 2]$
- *Exponential noise* (`Exp`). $z_j \sim$ Exp(1), $j = 1, \ldots, d$

- *Gumbel noise* (`Gumbel`). $z_j \sim \text{Gumbel}(0, 1)$, $j = 1 \ldots, d$

Based on these models, we generated random datasets $\mathbf{X} \in \mathbb{R}^{n \times d}$ by generating the rows i.i.d. according to one of the models above. For each simulation, we generated $n = 1000$ samples for graphs with $d \in \{10; 20; 50; 100\}$ nodes. For each dataset, we run FGS, PC, NOTEARS, KKTS with NOTEARS as initialization, TOPO with random initialization, TOPO with NOTEARS as initialization, and GOLEM-EV(equal variance), GOLEM-NV(Unequal variance). Here random initialization means a topological sort π is randomly sampled, the solve (7) to obtain θ^*_π as initialization. Finally, a post-processing threshold of $\omega = 0.3$ is applied on W, following (Zheng et al., 2018). Since FGS outputs a CPDAG instead of a DAG or weight matrix, we orient the undirected edges favorably when making comparisons. In linear model with unequal variance Gaussian noise, the minimax concave penalty (MCP) is used to approximate ℓ_0 penalty,

$$p(w) = \begin{cases} \lambda|w| - \frac{w^2}{2\beta} & \text{if } |w| \le \beta\lambda \\ \frac{\beta\lambda^2}{2} & \text{otherwise} \end{cases}$$

and set $\lambda = 0.005$ and $\beta = 10$.

For TOPO, we use the least-square loss $Q(W, \mathbf{X}) = \frac{1}{2n}\|\mathbf{X} - \mathbf{X}W\|_F^2$ without any regularization for all noise type. We also use the polynomial acyclicity penalty $h(A) = \text{Tr}((I + A/d)^d) - d$ (Yu et al., 2019) and $h(A) = -\log\det(I - A)$ (Bello et al., 2022), because it is faster and more accurate than $h(A) = \text{Tr}(e^A) - d$ (Zheng et al., 2018). For the choices of $s_{\text{small}}, s_{\text{large}}, s_0$, Table 7 summarizes the suggested hyerparameters. The basic idea is to increase $s_{\text{small}}, s_{\text{large}}, s_0$ when d grows or graph get denser.

# node	s_{small}	s_{large}	s_0
$d = 10$	30	45	1
$d = 20$	50	150	1
$d = 50$	100	1000	10
$d = 100$	150	2500	15

Table 7. Suggested hyperparamters for $s_{\text{small}}, s_{\text{large}}, s_0$

Logistic Models. Given $\mathcal{G}$, we assigned edge weights independently from $\text{Unif}([-2, -0.5] \cup [0.5, 2])$ to obtain a weight matrix $W \in \mathbb{R}^{d \times d}$. Given W, we sample X_j according to following

$$X_j = \text{Bernoulli}(\exp(w_j^\top X)/(1 + \exp(w_j^\top X))) \quad j = 1, \ldots, d$$

Based on these models, we generated random datasets $\mathbf{X} \in \mathbb{R}^{n \times d}$ by generating the rows i.i.d. according to one of the models above. For each simulation, we generated $n = 10000$ samples for graphs with $d \in \{10; 20; 30; 40; 50\}$ nodes. For each dataset, we run FGS, PC, NOTEARS, TOPO with random initialization, TOPO with NOTEARS as initialization. We use penalized log-likelihood as score function, i.e.

$$Q(f, \mathbf{X}) = \frac{1}{n}\sum_{i=1}^{d} \mathbf{1}_n^\top (\log(\mathbf{1}_n + \exp(f_i(\mathbf{X}))) - \mathbf{x}_i \circ f_i(\mathbf{X})) + \lambda\|W\|_1$$

where $\lambda = 0.01$.

Fully Connected Graphs. We randomly generate a topological sort π, and generated a fully connected graph that is consistent with topological sort π. Other setting is the same as Linear SEM. Because this is a really hard problem, we increase $s_{\text{small}}, s_{\text{large}}, s_0$ compared to Table 7.

Nonlinear Models with Neural Networks. We mainly follow the nonlinear setting in Zheng et al. (2020). Given $\mathcal{G}$, we simulate the SEM:

$$X_j = f_j(X_{\text{pa}(j)}) + z_j \qquad \forall j \in [d]$$

where $z_j \sim \mathcal{N}(0, 1)$. Here f_j is a randomly initialized MLP as described in Section 3.3

For TOPO, the score function is

$$Q(f, \mathbf{X}) = \frac{1}{2n} \sum_{i=1}^{d} \|\mathbf{x}_i - \hat{f}_i(\mathbf{X})\|_2^2$$

Here each $\hat{f}_i$ is chosen as MLP with one hidden layer of size 30 and sigmoid activation.

Implementation The implementation details of baseline are listed below:

- FGS and PC are standard baseline for structure learning. The implementation is based on the `py-causal` package, available at https://github.com/bd2kccd/py-causal. For PC algorithm, use Fisher Z test. For GES, we use `cg-bic-scores` and `maxDegree=50`.
- NOTEARS (NOTERAS_MLP) was implemented using Python code: https://github.com/xunzheng/notears. Its score function is least square loss with ℓ_1 regularization. We use default threshold $\omega = 0.3$.
- KKTS was implemented using Python code: https://github.com/skypea/DAG_No_Fear. We allow KKTS to reverse edges in each iteration to achieve best performance.
- GOLEM was implemented using Python and Tensorflow code: https://github.com/ignavierng/golem. We use default parameters.

In the experiments, we use default hyperparameters for those baseline unless otherwise stated.

E.2. Metrics

We evaluate the performance of each algorithm with the following three metrics:

- Structure Hamming distance (SHD): A standard benchmark in the structure learning literature that counts the total number of edges additions, deletions, and reversals needed to convert the estimated graph into the true graph. For PC and GES, they all return CPDAG that may contain undirected edges, in which case we evaluate them favorably by assuming correct direction for undirected edges whenever possible.
- Score: the value of least square score function.
- KKT: Whether solution satisfies the KKT conditions, 1 stands for Yes and 0 stands for No. Define a KKT matrix for θ, denoted as $K(\theta)$.

$$[K(\theta)]_{ij} = \begin{cases} |\frac{\partial Q(\Theta)}{\partial \theta_{ij}}| & \text{if } \nabla h(W(\theta)) = 0 \\ |W(\theta)_{ij}| & \text{if } \nabla h(W(\theta)) > 0 \end{cases}$$

$$\text{KKT} = \begin{cases} 1 & \text{if } \max_{ij} [K(\theta)]_{ij} = 0 \\ 0 & \text{if } \max_{ij} [K(\theta)]_{ij} \neq 0 \end{cases}$$

- Timing: how much time the algorithm takes to run, we use it to measure the speed of the algorithms.

E.3. Sensitivity of $s_{\text{small}}, s_{\text{large}}, s_0$

In Tables 2, 3, 4, and 5, we investigate the effect of sizes of search space and the number of searching times in larger spaces on Algorithm 2. Here we focus on two cases: (1) Simple case: $ER1$ graphs with Gaussian noise and $d = 100$. (2) Hard case: $SF4$ graphs with Guassian noise and $d = 100$. Columns represent different $s_{\text{small}} = 50, 150, 200$. Rows represent different $s_{\text{large}} = 1000, 2000, 3000$. Blank implies algorithm has stopped at current iteration. Here we use n_0 to indicate how many large searches has been used. Generally speaking, for sparser graphs, using small search space and small s_0 are enough to return a good solution. While for denser graphs, the performance of Algorithm 2 is more sensitive to the choice of $s_{\text{small}}, s_{\text{large}}, s_0$.

	$n_0 = 0$			$n_0 = 1$			$n_0 = 2$			$n_0 = 3$		
	50	150	200	50	150	200	50	150	200	50	150	200
1000	136	136	136	20	11	0	8	6		5	0	
2000	136	136	136	19	11	0	8	6		4	0	
3000	136	136	136	19	11	0	8	6		2	0	

Table 8. Structural Hamming Distance (SHD) for different $s_{\text{small}}, s_{\text{large}}, n_0$ with $d = 100$ and $n = 1000$ on an Gaussian ER1 graph

	$n_0 = 0$			$n_0 = 1$			$n_0 = 2$			$n_0 = 3$		
	50	150	200	50	150	200	50	150	200	50	150	200
1000	113.017	113.017	113.017	49.570	48.291	47.215	47.731	47.874		47.451	47.219	
2000	113.017	113.017	113.017	49.281	48.291	47.215	48.141	47.874		47.438	47.219	
3000	113.017	113.017	113.017	49.281	48.291	47.215	48.141	47.874		47.369	47.219	

Table 9. Score for different $s_{\text{small}}, s_{\text{large}}, n_0$ with $d = 100$ and $n = 1000$ on an Gaussian ER1 graph

	$n_0 = 0$			$n_0 = 1$			$n_0 = 2$			$n_0 = 3$		
	50	150	200	50	150	200	50	150	200	50	150	200
1000	776	405	322	672	244	144	568	185	143	295	58	143
2000	774	405	349	693	311	40	455	112	38	56	0	38
3000	779	405	366	574	119	144	272	118	71	44	0	50

Table 10. Structural Hamming Distance (SHD) for different $s_{\text{small}}, s_{\text{large}}, n_0$ with $d = 100$ and $n = 1000$ on an Gaussian SF4 graph

	$n_0 = 0$			$n_0 = 1$			$n_0 = 2$			$n_0 = 3$		
	50	150	200	50	150	200	50	150	200	50	150	200
1000	194.871	67.834	63.498	162.679	57.024	50.124	82.946	55.102	49.199	58.244	48.113	49.198
2000	189.561	67.834	62.848	157.953	61.346	48.028	83.159	50.351	47.800	47.905	47.695	47.799
3000	187.662	67.843	62.79	106.329	54.097	51.45	56.241	49.924	49.70	47.925	47.694	47.71

Table 11. Loss for different $s_{\text{small}}, s_{\text{large}}, n_0$ with $d = 100$ and $n = 1000$ on an Gaussian SF4 graph

E.4. Linear Models

SHD comparisons: ER and SF graphs without FGES and PC

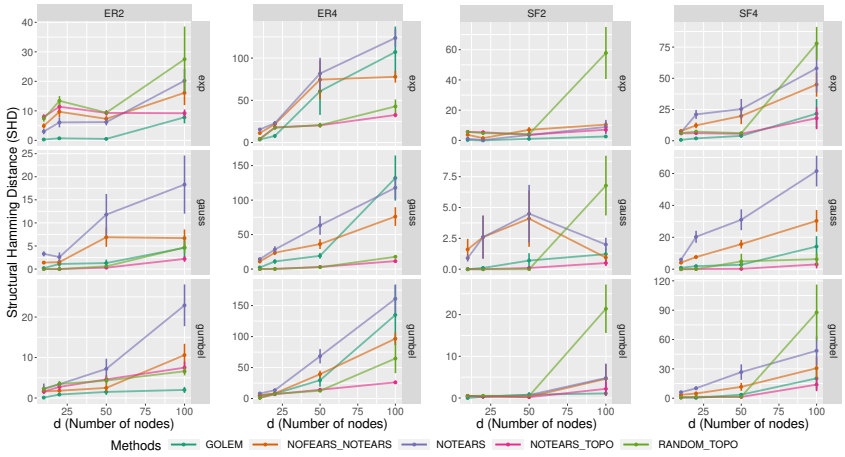


Figure 1. Stuctural Hamming distance (SHD) (lower is better). Row: noise type of SEM. Columns: random graph types, {SF, ER}-k = {Scale-Free,Erdős-Rényi } graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

SHD comparisons: ER graphs

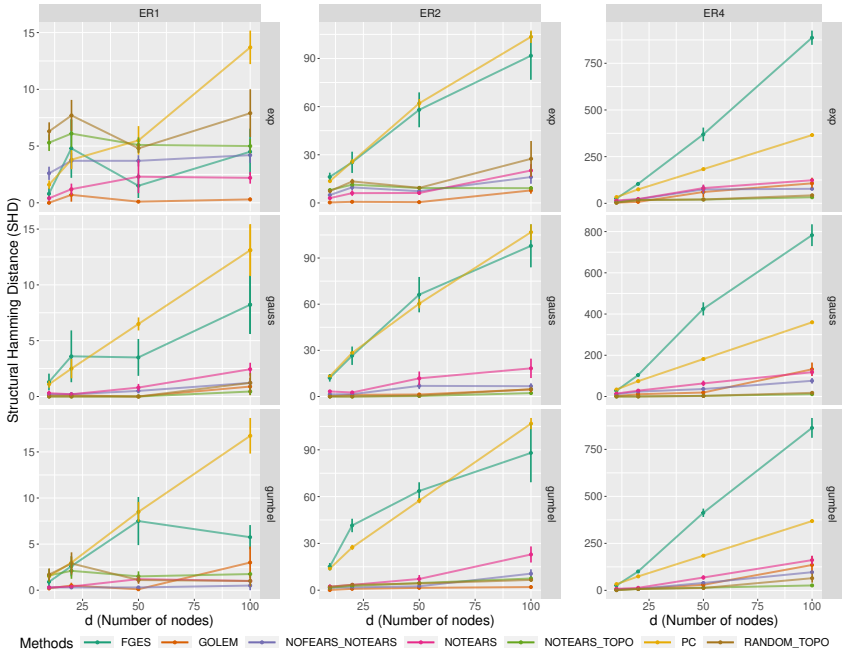


Figure 2. Stuctural Hamming distance (SHD) (lower is better). Row: noise type of SEM. Columns: random graph types, {ER}-k = {Erdős-Rényi } graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

SHD comparisons: SF graphs

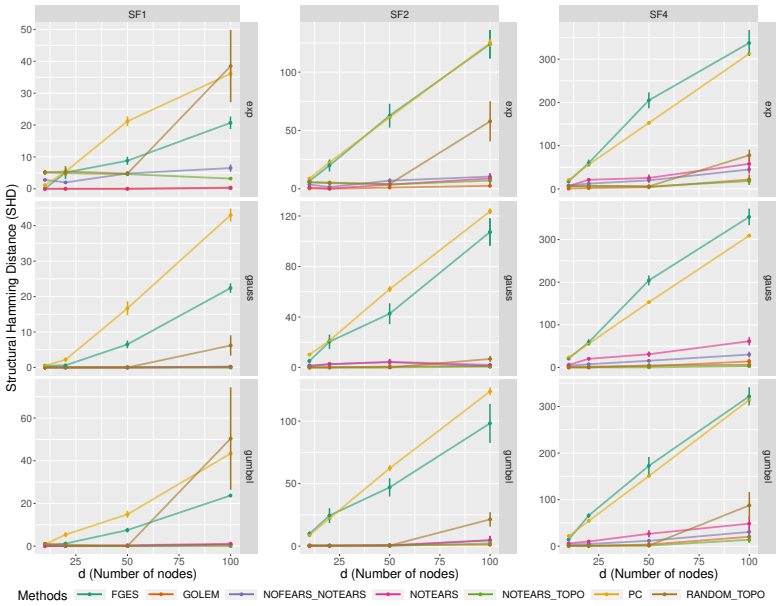


Figure 3. Structural Hamming distance(SHD) (lower is better). Row: noise type of SEM. Columns: random graph types, {SF}-k = {scale free} graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

Running time comparisons: ER graphs

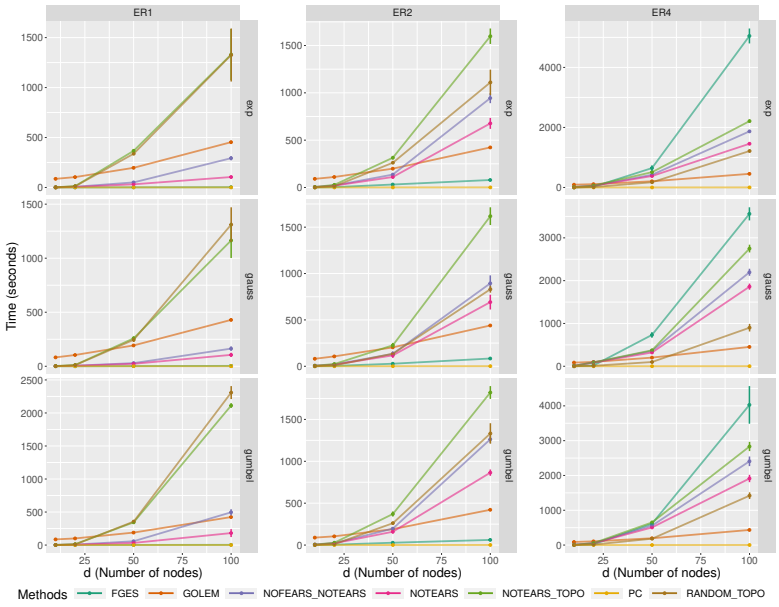


Figure 4. Runtime. Row: noise type of SEM. Columns: random graph types, {ER}-k = {Erdős-Rényi } graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

Running time comparisons: SF graphs

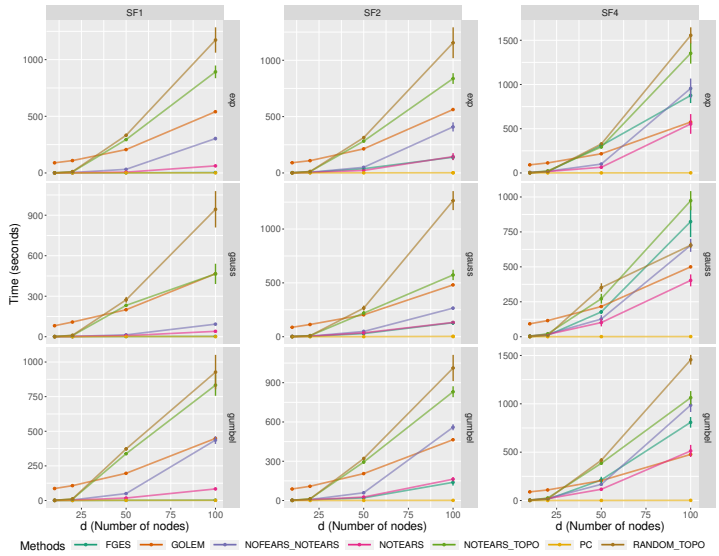


Figure 5. Structural Hamming distance(SHD) (lower is better). Row: noise type of SEM. Columns: random graph types, {SF}-k = {scale free} graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

Score comparisons: ER graphs

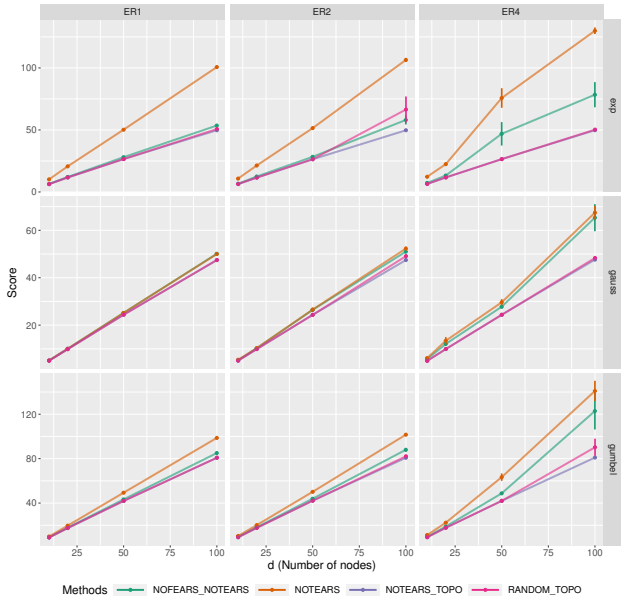


Figure 6. least square score (lower is better). Row: noise type of SEM. Columns: random graph types, {ER}-k = {Erdős-Rényi } graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

Score comparisons: SF graphs

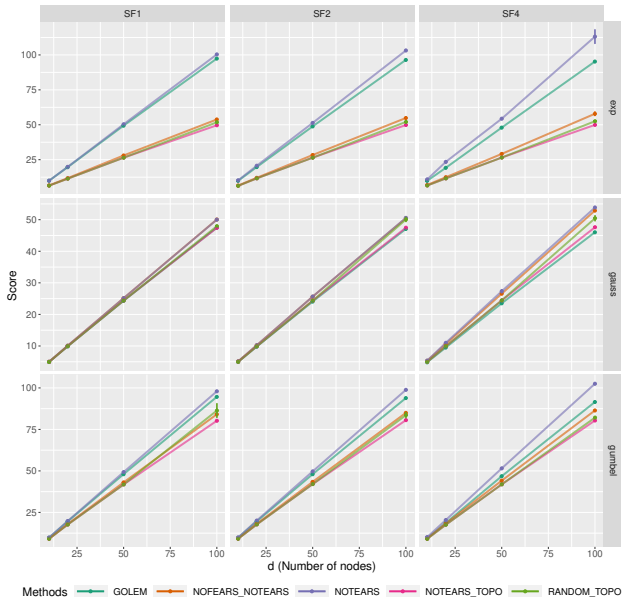


Figure 7. least square score (lower is better). Row: noise type of SEM. Columns: random graph types, {SF}-k = {scale free} graphs with kd expected edges. Here, nofears_notears (KKTS algorithm (Wei et al., 2020) uses NOTEARS solution as initial point). Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

E.5. Nonlinear Models

E.5.1. Logistic Model

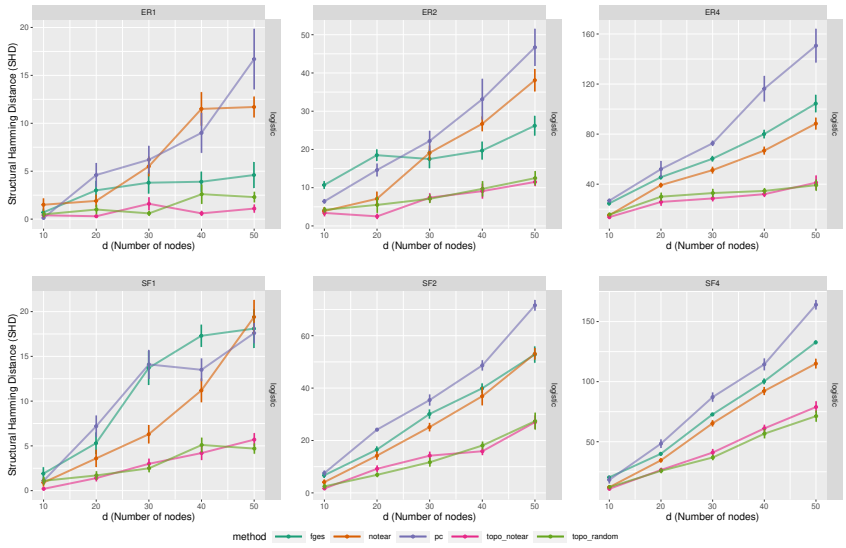


Figure 8. Structural Hamming distance(SHD) for Logistic Model, Row: random graph types, {SF, ER}-k = {Scale-Free,Erdős-Rényi } graphs. Columns: kd expected edges. Our methods are Random_Topo (random initialization), and Notears_Topo (using NOTEARS solution as initial point.) Error bars represent standard errors over 10 simulations.

E.5.2. NEURAL NETWORKS

SHD comparison

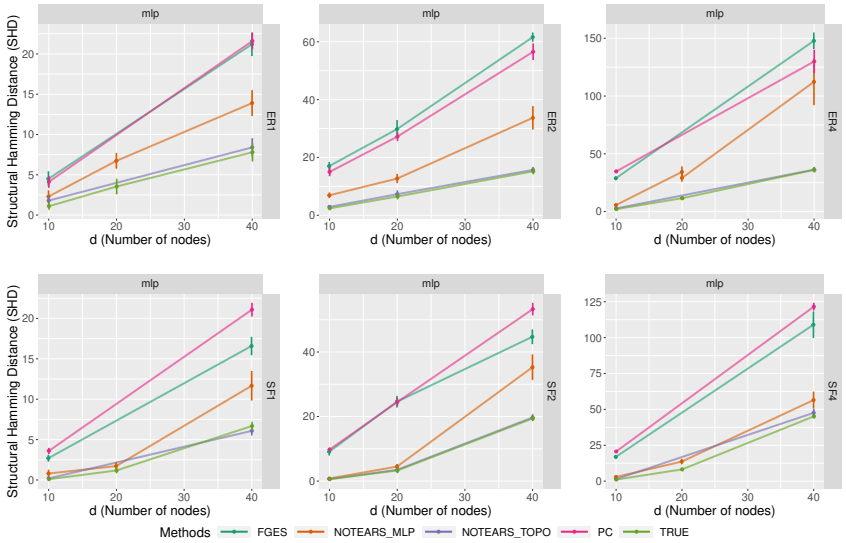


Figure 9. Structural Hamming distance(SHD) for Nonlinear Model with Neural Network, Row: random graph types, {SF, ER} = {Scale-Free,Erdős-Rényi } graphs. Columns: kd expected edges. Our methods are $\mathrm{Random\_Topo}$ (random initialization), and $\mathrm{Notears\_Topo}$ (using NOTEARS solution as initial point.) True(baseline): solution to (8) with true topological sort using Neural Network. Error bars represent standard errors over 10 simulations.

Score comparison

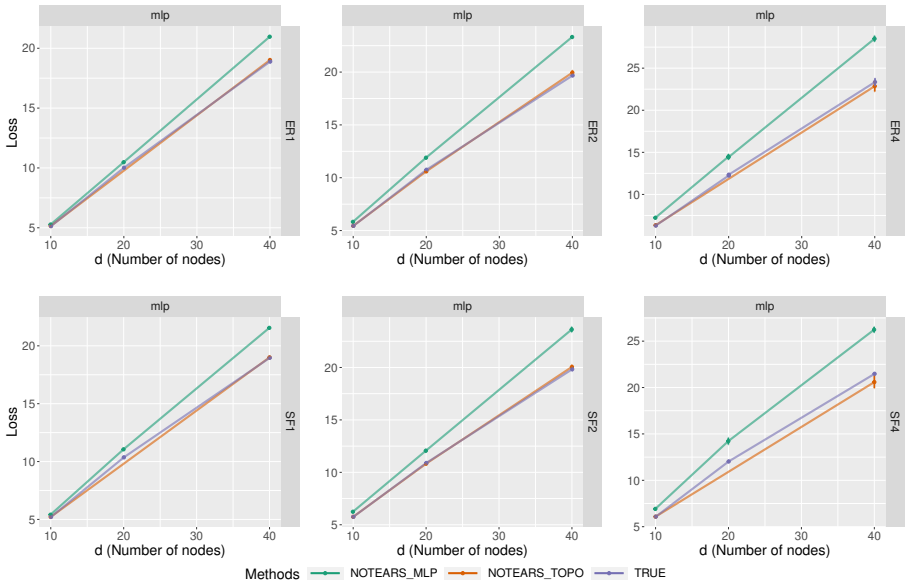


Figure 10. Score for Nonlinear Model with Neural Network, Row: random graph types, {SF, ER} = {Scale-Free,Erdős-Rényi } graphs. Columns: kd expected edges. Our methods are $\mathrm{Random\_Topo}$ (random initialization), and $\mathrm{Notears\_Topo}$ (using NOTEARS solution as initial point.) True(baseline): solution to (8) with true topological sort using Neural Network. Error bars represent standard errors over 10 simulations.

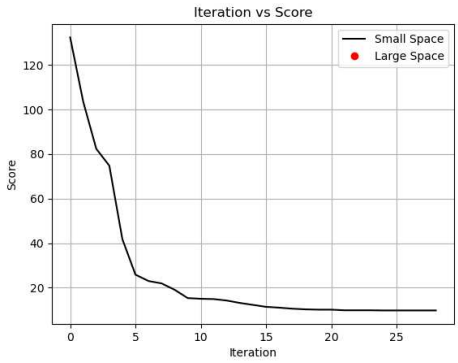
E.6. Comparison against randomly chosen swapping set

			TOPO		Random	
n	d	# edge	SHD	loss	SHD	loss
1000	20	80	0.1	9.85	32.5	26.85
1000	50	200	3	24.33	126.7	57.33
1000	100	400	13.75	47.45	286.9	107.95

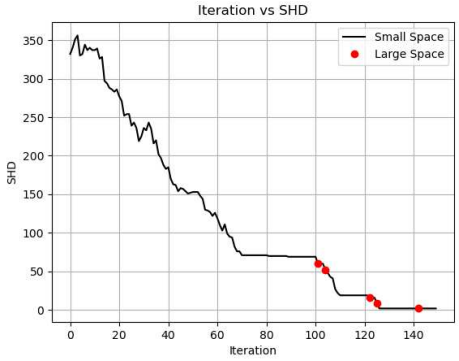
Table 12. TOPO: the candidate swapping set $\mathcal{Y}(\theta, \tau, \xi)$ by (9) ."Random": the TOPO algorithm chooses the candidate swapping set $\mathcal{Y}(\theta, \tau, \xi)$ randomly. Model: Linear model with Gaussian noise. Graph type: ER4 graphs. It justifies choosing swapping set $\mathcal{Y}(\theta, \tau, \xi)$ by (9) can significantly improve the performance of TOPO Algorithm.

E.7. Accuracy vs iteration

(a) $d = 20$

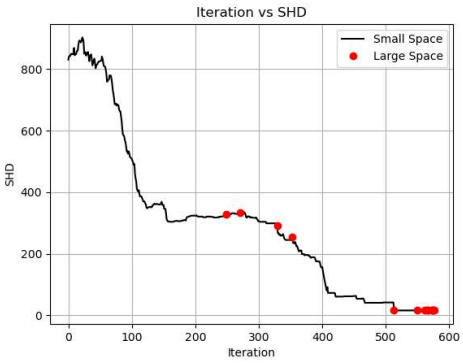


(b) $d = 20$

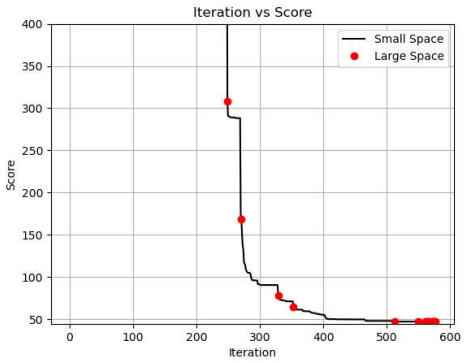


(c) $d = 50$

(d) $d = 50$



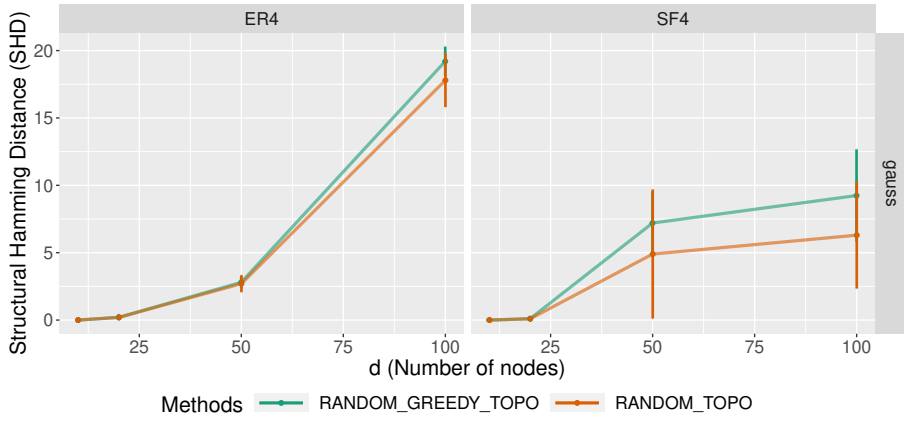
(e) $d = 100$



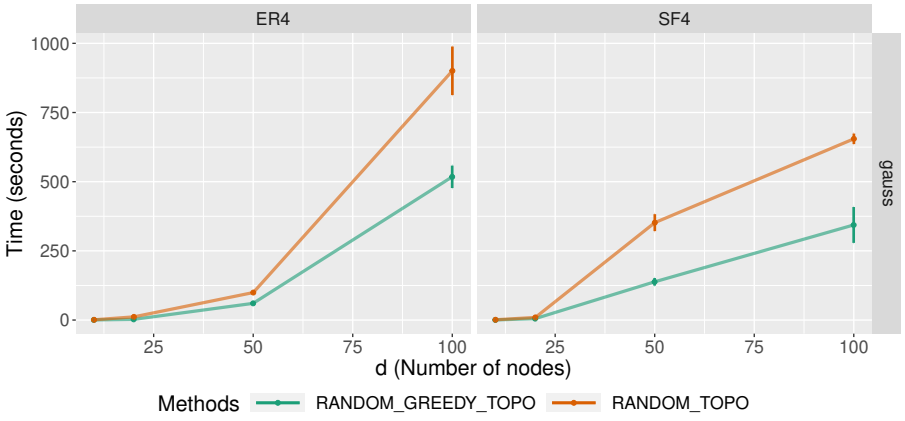
(f) $d = 100$

Figure 11. Iteration vs SHD (left)/Score (right). Model: linear model with Gaussian noise. Graph type: ER4 graphs. Black: search in small space. Red: search in large space. When graph is small, searching in small space is enough for finding a good local optimal, but when graph gets larger, searching in large space helps to jump out of local point and decrease the score.

E.8. Greedy Strategy



(a) Structural Hamming Distance (SHD)



(b) Run-time (seconds)

Figure 12. Comparison between greedy scheme and non-greedy scheme by SHD & running time. Random_Topo (TOPO starts with random initialization and uses the swap that decreases the score the most at each iteration), and Random_Greedy_Topo (TOPO starts with random initialization and uses the swap once it is found to decrease score.) Model: linear model with Gaussian noise. Graph type: ER4 graphs. Greedy scheme significantly improves time efficiency by sacrificing just a little accuracy.

F. Broader Impacts

Bayesian networks are fundamental models that represent the probabilistic relationship about how data are generated by a set of random variables. Our work contributes to the most fundamental questions: What is the underlying structure that generates data? Specifically speaking, how can one recover such structure accurately and efficiently? We propose an algorithm with theoretical guarantees to address them. The significant contribution of this work is about better solving a nonconvex continuous score-based structure learning formulation. The dramatic improvements in accuracy means better structure recovery and more accurate discovery about the underlying probabilistic relationships.

A potential negative impact of this work is that errors in structure learning may compound into potentially more serious downstream errors. For example, a false discovery about causality may result in a company investing tons of money and efforts to remedy an incorrectly detected cause to a problem, resulting in immeasurable losses. How to prevent incorrect causation under this continuous framework is a crucial and exciting future research direction.