DAGs with NO TEARS: Continuous Optimization for Structure Learning

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing

Carnegie Mellon University

November 6, 2018

Abstract

Estimating the structure of directed acyclic graphs (DAGs, also known as Bayesian networks) is a challenging problem since the search space of DAGs is combinatorial and scales superexponentially with the number of nodes. Existing approaches rely on various local heuristics for enforcing the acyclicity constraint. In this paper, we introduce a fundamentally different strategy: We formulate the structure learning problem as a purely *continuous* optimization problem over real matrices that avoids this combinatorial constraint entirely. This is achieved by a novel characterization of acyclicity that is not only smooth but also exact. The resulting problem can be efficiently solved by standard numerical algorithms, which also makes implementation effortless. The proposed method outperforms existing ones, without imposing any structural assumptions on the graph such as bounded treewidth or in-degree. Code implementing the proposed algorithm is open-source and publicly available at `https://github.com/xunzheng/notears`.

1 Introduction

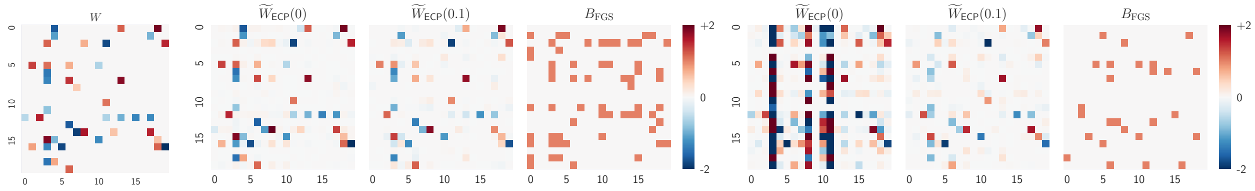
Learning directed acyclic graphs (DAGs) from data is an NP-hard problem (Chickering, 1996; Chickering et al., 2004), owing mainly to the combinatorial acyclicity constraint that is difficult to enforce efficiently. At the same time, DAGs are popular models in practice, with applications in biology (Sachs et al., 2005), genetics (Zhang et al., 2013), machine learning (Koller and Friedman, 2009), and causal inference (Spirtes et al., 2000). For this reason, the development of new methods for learning DAGs remains a central challenge in machine learning and statistics.

In this paper, we propose a new approach for score-based learning of DAGs by converting the traditional *combinatorial* optimization problem (left) into a *continuous* program (right):

$$\begin{array}{cl} \displaystyle\min_{W\in\mathbb{R}^{d\times d}} & F(W) \\ \text{subject to} & \mathsf{G}(W)\in\mathsf{DAGs} \end{array} \quad\iff\quad \begin{array}{cl} \displaystyle\min_{W\in\mathbb{R}^{d\times d}} & F(W) \\ \text{subject to} & h(W)=0, \end{array} \tag{1}$$

where $\mathsf{G}(W)$ is the d-node graph induced by the weighted adjacency matrix W, $F:\mathbb{R}^{d\times d}\to\mathbb{R}$ is a score function (see Section 2.1 for details), and our key technical device $h:\mathbb{R}^{d\times d}\to\mathbb{R}$ is a smooth function over real matrices, whose level set at zero exactly characterizes acyclic graphs. Although the two problems are equivalent, the continuous program on the right eliminates the need for specialized algorithms that are tailored to search over the combinatorial space of DAGs. Instead, we are able to leverage standard numerical algorithms for constrained problems, which makes implementation particularly easy, not requiring any knowledge about graphical models. This is similar in spirit to the situation for undirected graphical models, in which the formulation of a continuous log-det program (Banerjee et al., 2008) sparked a series of remarkable advances in structure learning for undirected graphs (Section 2.2). Unlike undirected models, which can be reduced to a convex program, however, the program (1) is *nonconvex.* Nonetheless, as we will show, even naïve solutions to this program yield state-of-the-art results for learning DAGs.

Contact: `xzheng1@andrew.cmu.edu`, `naragam@cs.cmu.edu`, `pradeepr@cs.cmu.edu`, `epxing@cs.cmu.edu`



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 1: Visual comparison of the learned weighted adjacency matrix on a 20-node graph with $n = 1000$ (large samples) and $n = 20$ (insufficient samples): $\widetilde{W}_{\mathsf{ECP}}(\lambda)$ is the proposed NOTEARS algorithm with ℓ_1-regularization λ, and B_{FGS} is the binary estimate of the baseline (Ramsey et al., 2016). The proposed algorithms perform well on large samples, and remains accurate on small n with ℓ_1 regularization.

Contributions. The main thrust of this work is to re-formulate score-based learning of DAGs so that standard smooth optimization schemes such as L-BFGS (Nocedal and Wright, 2006) can be leveraged. To accomplish this, we make the following specific contributions:

- We explicitly construct a smooth function over $\mathbb{R}^{d\times d}$ with computable derivatives that encodes the acyclicity constraint. This allows us to replace the combinatorial constraint $\mathsf{G} \in \mathbb{D}$ in (4) with a smooth equality constraint.
- We develop an equality-constrained program for simultaneously estimating the structure and parameters of a sparse DAG from possibly high-dimensional data, and show how standard numerical solvers can be used to find stationary points.
- We demonstrate the effectiveness of the resulting method in empirical evaluations against existing state-of-the-arts. See Figure 1 for a quick illustration and Section 5 for details.
- We compare our ouput to the exact global minimizer (Cussens, 2012), and show that our method attains scores that are comparable to the globally optimal score in practice, although our methods are only guaranteed to find stationary points.

Most interestingly, our approach is very simple and can be implemented in about 50 lines of Python code. As a result of its simplicity and effortlessness in its implementation, we call the resulting method NOTEARS: *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning.* The implementation is publicly available at `https://github.com/xunzheng/notears`.

2 Background

The basic DAG learning problem is formulated as follows: Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be a data matrix consisting of n i.i.d. observations of the random vector $X = (X_1, \dots, X_d)$ and let $\mathbb{D}$ denote the (discrete) space of DAGs $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ on d nodes. Given $\mathbf{X}$, we seek to learn a DAG $\mathsf{G} \in \mathbb{D}$ (also called a *Bayesian network*) for the joint distribution $\mathbb{P}(X)$ (Spirtes et al., 2000; Koller and Friedman, 2009). We model X via a structural equation model (SEM) defined by a weighted adjacency matrix $W \in \mathbb{R}^{d\times d}$. Thus, instead of operating on the discrete space $\mathbb{D}$, we will operate on $\mathbb{R}^{d\times d}$, the continuous space of $d \times d$ real matrices.

2.1 Score functions and SEM

Any $W \in \mathbb{R}^{d\times d}$ defines a graph on d nodes in the following way: Let $\mathcal{A}(W) \in \{0,1\}^{d\times d}$ be the binary matrix such that $[\mathcal{A}(W)]_{ij} = 1 \iff w_{ij} \neq 0$ and zero otherwise; then $\mathcal{A}(W)$ defines the adjacency matrix of a directed graph $\mathsf{G}(W)$. In a slight abuse of notation, we will thus treat W as if it were a (weighted) graph. In addition to the graph $\mathsf{G}(W)$, $W = [\, w_1 \,|\, \cdots \,|\, w_d \,]$ defines a linear SEM by $X_j = w_j^T X + z_j$, where

$X = (X_1, \dots, X_d)$ is a random vector and $z = (z_1, \dots, z_d)$ is a random noise vector. We do *not* assume that z is Gaussian. More generally, we can model X_j via a generalized linear model (GLM) $\mathbb{E}(X_j \,|\, X_{\mathrm{pa}(X_j)}) = f(w_j^T X)$. For example, if $X_j \in \{0, 1\}$, we can model the conditional distribution of X_j given its parents via logistic regression.

In this paper, we focus on linear SEM and the least-squares (LS) loss $\ell(W; \mathbf{X}) = \frac{1}{2n}\|\mathbf{X} - \mathbf{X}W\|_F^2$, although everything in the sequel applies to any smooth loss function ℓ defined over $\mathbb{R}^{d\times d}$. The statistical properties of the LS loss in scoring DAGs have been extensively studied: The minimizer of the LS loss provably recovers a true DAG with high probability on finite-samples and in high-dimensions ($d \gg n$), and hence is consistent for both Gaussian SEM (van de Geer and Bühlmann, 2013; Aragam et al., 2016) and non-Gaussian SEM (Loh and Bühlmann, 2014).[2] Note also that these results imply that the faithfulness assumption is not required in this set-up. Given this extensive previous work on statistical issues, our focus in this paper is entirely on the computational problem of finding an SEM that minimizes the LS loss.

This translation between graphs and SEM is central to our approach. Since we are interested in learning a *sparse* DAG, we add ℓ_1-regularization $\|W\|_1 = \|\operatorname{vec}(W)\|_1$ resulting in the regularized score function

$$F(W) = \ell(W; \mathbf{X}) + \lambda\|W\|_1 = \frac{1}{2n}\|\mathbf{X} - \mathbf{X}W\|_F^2 + \lambda\|W\|_1. \tag{2}$$

Thus we seek to solve

$$\begin{aligned} \min_{W\in\mathbb{R}^{d\times d}} \quad & F(W) \\ \text{subject to} \quad & \mathsf{G}(W) \in \mathbb{D}. \end{aligned} \tag{3}$$

Unfortunately, although $F(W)$ is continuous, the DAG constraint $\mathsf{G}(W) \in \mathbb{D}$ remains a challenge to enforce. In Section 3, we show how this discrete constraint can be replaced by a smooth equality constraint.

2.2 Previous work

Traditionally, score-based learning seeks to optimize a *discrete score* $Q : \mathbb{D} \to \mathbb{R}$ over the set of DAGs $\mathbb{D}$; note that this is distinct from our score $F(W)$ whose domain is $\mathbb{R}^{d\times d}$ instead of $\mathbb{D}$. This can be written as the following combinatorial optimization problem:

$$\begin{aligned} \min_{\mathsf{G}} \quad & Q(\mathsf{G}) \\ \text{subject to} \quad & \mathsf{G} \in \mathbb{D} \end{aligned} \tag{4}$$

Popular score functions include BDe(u) (Heckerman et al., 1995), BGe (Kuipers et al., 2014), BIC (Chickering and Heckerman, 1997), and MDL (Bouckaert, 1993). Unfortunately, (4) is NP-hard to solve Chickering (1996); Chickering et al. (2004) owing mainly to the nonconvex, combinatorial nature of the optimization problem. This is the main drawback of existing approaches for solving (4): The acyclicity constraint is a combinatorial constraint with the number of acyclic structures increasing superexponentially in d (Robinson, 1977). Notwithstanding, there are algorithms for solving (4) to global optimality for small problems (Ott and Miyano, 2003; Singh and Moore, 2005; Silander and Myllymaki, 2006; Xiang and Kim, 2013; Cussens, 2012; Cussens et al., 2017). There is also a wide literature on approximate algorithms based on order search (Teyssier and Koller, 2005; Schmidt et al., 2007; Scanagatta et al., 2015, 2016), greedy search (Heckerman et al., 1995; Chickering, 2003; Ramsey et al., 2016), and coordinate descent (Fu and Zhou, 2013; Aragam and Zhou, 2015; Gu et al., 2018). By searching over the space of topological orderings, the former order-based methods trade-off the difficult problem of enforcing acyclicity with a search over $d!$ orderings, whereas the latter methods enforce acyclicity one edge at a time, explicitly checking for acyclicity violations each time an edge is added. Other approaches that avoid optimizing (4) directly include constraint-based methods (Spirtes and Glymour, 1991; Spirtes et al., 2000), hybrid methods (Tsamardinos et al., 2006; Gámez et al., 2011), and Bayesian methods (Ellis and Wong, 2008; Zhou, 2011; Niinimäki et al., 2016).

[2] Due to nonconvexity, there may be more than one minimizer: These and other technical issues such as parameter identifiability are addressed in detail in the cited references.

The intractable form of the program (4) has led to a host of heuristic methods, often borrowing tools from the optimization literature, but always resorting to clever heuristics to accelerate algorithms. Here we briefly discuss some of the pros and cons of existing methods. While not all methods suffer from *all* of the problems highlighted below, we are not aware of any methods that simultaneously avoid all of them.

Exact vs. approximate. Broadly speaking, there are two camps: *Approximate* algorithms and *exact* algorithms, the latter of which are guaranteed to return a globally optimal solution. Exact algorithms form an intriguing class of methods, but as they are based around an NP-hard combinatorial optimization problem, these methods remain computationally intractable in general. For example, recent state-of-the-art work (Cussens et al., 2017; Chen et al., 2016) only scale to problems with a few dozen nodes (Van Beek and Hoffmann, 2015).[3] Older methods based on dynamic programming methods (Ott and Miyano, 2003; Singh and Moore, 2005; Silander and Myllymaki, 2006; Xiang and Kim, 2013; Loh and Bühlmann, 2014) also scale to roughly a few dozen nodes. By contrast, state-of-the-art approximate methods can scale to thousands of nodes (Ramsey et al., 2016; Aragam and Zhou, 2015; Scanagatta et al., 2015, 2016).

Local vs. global search. Arguably the most popular approaches to optimizing (4) involve *local* search, wherein edges and parent sets are added sequentially, one node at a time. This is efficient as long as each node has only a few parents, but as the number of possible parents grows, local search rapidly becomes intractable. Furthermore, such strategies typically rely on severe structural assumptions such as bounded in-degree, bounded treewidth, or edge constraints. Since real-world networks often exhibit scale-free and small-world topologies (Watts and Strogatz, 1998; Barabási and Albert, 1999) with highly connected hub nodes, these kinds of structural assumptions are not only difficult to satisfy, but impossible to check. We note here promising work towards relaxing this assumption for discrete data (Scanagatta et al., 2015). By contrast, our method uses *global* search wherein the entire matrix W is updated in each step.

Model assumptions. The literature on DAG learning tends to be split between methods that operate on discrete data vs. methods that operate on continuous data. When viewed from the lens of (3), the reasons for this are not clear since both discrete and continuous data can be considered as special cases of the general score-based learning framework. Nonetheless, many (but not all) of the methods cited already only work under very specific assumptions on the data, the most common of which are categorical (discrete) and Gaussian (continuous). Since (3) is agnostic to the form of the data and loss function, there is significant interest in finding general methods that are not tied to specific model assumptions.

Conceptual clarity. Finally, on a higher level, a significant drawback of existing methods is their conceptual complexity: They are not straightforward to implement, require deep knowledge of concepts from the graphical modeling literature, and accelerating them involves many clever tricks. By contrast, the method we propose in this paper is conceptually very simple, requires no background on graphical models, and can be implemented in just a few lines of code using existing black-box solvers.

2.3 Comparison

It is instructive to compare existing methods for learning DAGs against other methods in the machine learning literature. We focus here on two popular models: Undirected graphical models and deep neural networks. Undirected graphical models, also known as Markov networks, is recognized as a convex problem (Yuan and Lin, 2007; Banerjee et al., 2008) nowadays, and hence can be solved using black-box convex optimizers such as CVX (Grant and Boyd, 2014). However, one should not forget score-based methods based on discrete scores similar to (4) proliferated in the early days for learning undirected graphs (e.g. Koller and Friedman, 2009, §20.7). More recently, extremely efficient algorithms have been developed for this problem using coordinate descent (Friedman et al., 2008) and Newton methods (Hsieh et al., 2014; Schmidt et al., 2009). As another

[3] Cussens (2012) reports experiments with $d > 60$ under a constraint on the maximum parent size.

example, deep neural networks are often learned using various descendants of stochastic gradient descent (SGD) (Bousquet and Bottou, 2008; Kingma and Ba, 2014; Bottou et al., 2016), although recent work has proposed other techniques such as ADMM (Taylor et al., 2016) and Gauss-Newton (Botev et al., 2017). One of the keys to the success of both of these models—and many other models in machine learning—was having a closed-form, tractable program for which existing techniques from the extensive optimization literature could be applied. In both cases, the application of principled optimization techniques led to significant breakthroughs. For undirected graphical models the major technical tool was convex optimization, and for deep networks the major technical tool was SGD.

Unfortunately, the general problem of DAG learning has not benefited in this way, and one of our main goals in the current work is to formulate score-based learning similarly as a closed-form, continuous program. Arguably, the challenges with existing approaches stem from the intractable form of the program (4). One of our main goals in the current work is to formulate score-based learning via a similar closed-form, continuous program. The key device in accomplishing this is a smooth characterization of acyclicity that will be introduced in the next section.

3 A new characterization of acyclicity

In order to make (3) amenable to black-box optimization, we propose to replace the combinatorial acyclicity constraint $\mathsf{G}(W) \in \mathbb{D}$ in (3) with a single smooth equality constraint $h(W) = 0$. Ideally, we would like a function $h : \mathbb{R}^{d \times d} \to \mathbb{R}$ that satisfies the following desiderata:

(a) $h(W) = 0$ if and only if W is acyclic (i.e. $\mathsf{G}(W) \in \mathbb{D}$);

(b) The values of h quantify the "DAG-ness" of the graph;

(c) h is smooth;

(d) h and its derivatives are easy to compute.

Property (b) is useful in practice for diagnostics. By "DAG-ness", we mean some quantification of how severe violations from acyclicity become as W moves further from $\mathbb{D}$. Although there are many ways to satisfy (b) by measuring some notion of "distance" to $\mathbb{D}$, typical approaches would violate (c) and (d). For example, h might be the minimum ℓ_2 distance to $\mathbb{D}$ or it might be the sum of edge weights along all cyclic paths of W, however, these are either non-smooth (violating (c)) or hard to compute (violating (d)). If a function that satisfies desiderata (a)-(d) exists, we can hope to apply existing machinery for constrained optimization such as Lagrange multipliers. Consequently, the DAG learning problem becomes equivalent to solving a numerical optimization problem, which is agnostic about the graph structure.

We proceed in two steps: First, we consider the simpler case of binary adjacency matrices $B \in \{0,1\}^{d \times d}$ (Section 3.1). Note that since $\{0,1\}^{d \times d}$ is a discrete space, we cannot take gradients or do continuous optimization. For this we need the second step, in which we relax the function we originally define on binary matrices to real matrices (Section 3.2).

3.1 Special case: Binary adjacency matrices

When does a matrix $B \in \{0,1\}^{d \times d}$ correspond to an acyclic graph? Recall the *spectral radius* $r(B)$ of a matrix B is the largest absolute eigenvalue of B. One simple characterization of acyclicity is the following:

Proposition 1 (Infinite series)**.** *Suppose $B \in \{0,1\}^{d \times d}$ and $r(B) < 1$. Then B is a DAG if and only if*

$$\operatorname{tr}(I - B)^{-1} = d. \tag{5}$$

Proof. It essentially boils down to the fact that $\operatorname{tr} B^k$ counts the number of length-k closed walks in a directed graph. Clearly an acyclic graph will have $\operatorname{tr} B^k = 0$ for all $k = 1, \ldots, \infty$. In other words, B has no cycles if and only if $f(B) = \sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii} = 0$, then

$$\operatorname{tr}(I - B)^{-1} = \operatorname{tr} \sum_{k=0}^{\infty} B^k = \operatorname{tr} I + \sum_{k=1}^{\infty} \operatorname{tr} B^k = d + \sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii} = d + f(B).$$

The desired result follows. □

Unfortunately, the condition that $r(B) < 1$ is strong: although it is automatically satisfied when B is a DAG, it is generally not true otherwise, and furthermore the projection is nontrivial. Alternatively, instead of the infinite series, one could consider the characterization based on *finite* series $\sum_{k=1}^{d} \operatorname{tr} B^k = 0$, which does not require $r(B) < 1$. However, this is impractical for numerical reasons: The entries of B^k can easily exceed machine precision for even small values of d, which makes both function and gradient evaluations highly unstable. Therefore it remains to find a characterization that not only holds for all possible B, but also has numerical stability. Luckily, such function exists.

Proposition 2 (Matrix exponential)**.** *A binary matrix $B \in \{0,1\}^{d \times d}$ is a DAG if and only if*

$$\operatorname{tr} e^B = d. \tag{6}$$

Proof. Similar to Proposition 1 by noting that B has no cycles if and only if $(B^k)_{ii} = 0$ for all $k \geq 1$ and all i, which is true if and only if $\sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii}/k! = \operatorname{tr} e^B - d = 0$. □

It is worth pointing out that matrix exponential is well-defined for all square matrices. In addition to everywhere convergence, this characterization has an added bonus: As the number of edges in B increases along with the number of nodes d, the number of possible closed walks grows rapidly, so the trace characterization $\operatorname{tr}(I - B)^{-1}$ rapidly becomes ill-conditioned and difficult to manage. By re-weighting the number of length-k closed walks by $k!$, this becomes much easier to manage. While this is a useful characterization, it does not satisfy all of our desiderata since—being defined over a discrete space—it is not a smooth function. The final step is to extend Proposition 2 to all of $\mathbb{R}^{d \times d}$.

3.2 The general case: Weighted adjacency matrices

Unfortunately, the characterization (6) fails if we replace B with an arbitrary weighted matrix W. However, we can replace B with any *nonnegative* weighted matrix, and the same argument use to prove Proposition 2 shows that (6) will still characterize acyclicity. Thus, to extend this to matrices with both positive and negative values, we can simply use the Hadamard product $W \circ W$, which leads to our main result.

Theorem 1. *A matrix $W \in \mathbb{R}^{d \times d}$ is a DAG if and only if*

$$h(W) = \operatorname{tr}\left(e^{W \circ W}\right) - d = 0, \tag{7}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential of A. Moreover, $h(W)$ has a simple gradient

$$\nabla h(W) = \left(e^{W \circ W}\right)^T \circ 2W, \tag{8}$$

and satisfies all of the desiderata (a)-(d).

The proof of (7) is similar to (6), and desiderata (c)-(d) follow from (8). To see why desiderata (b) holds, note that the proof of Proposition 1 shows that the power series $\operatorname{tr}(B + B^2 + \cdots)$ simply counts the number of closed walks in B, and the matrix exponential simply re-weights these counts. Replacing B with $W \circ W$ amounts to counting *weighted* closed walks, where the weight of each edge is w_{ij}^2. Thus, larger $h(W) > h(W')$ means either (a) W has more cycles than W' or (b) The cycles in W are more heavily weighted than in W'.

Algorithm 1 NOTEARS algorithm

```
1. Input: Initial guess $(W_0, \alpha_0)$, progress rate $c \in (0,1)$, tolerance $\epsilon > 0$, threshold $\omega > 0$.
2. For $t = 0, 1, 2, \ldots$:
    (a) Solve primal $W_{t+1} \leftarrow \arg\min_W L^\rho(W, \alpha_t)$ with $\rho$ such that $h(W_{t+1}) < ch(W_t)$.
    (b) Dual ascent $\alpha_{t+1} \leftarrow \alpha_t + \rho h(W_{t+1})$.
    (c) If $h(W_{t+1}) < \epsilon$, set $\widetilde{W}_{\mathsf{ECP}} = W_{t+1}$ and break.
3. Return the thresholded matrix $\widehat{W} := \widetilde{W}_{\mathsf{ECP}} \circ 1(|\widetilde{W}_{\mathsf{ECP}}| > \omega)$.
```

Moreover, notice that $h(W) \geq 0$ for all W since each term in the series is nonnegative. This gives another interesting perspective of the space of DAGs as the set of global minima of $h(W)$. However, due to the nonconvexity, this is not equivalent to the first order stationary condition $\nabla h(W) = 0$.

A key conclusion from Theorem 1 is that h and its gradient only involve evaluating the matrix exponential, which is a well-studied function in numerical analysis, and whose $O(d^3)$ algorithm (Al-Mohy and Higham, 2009) is readily available in many scientific computing libraries. Although the connection between trace of matrix power and number of cycles in the graph is well-known Harary and Manvel (1971), to the best of our knowledge, this characterization of acyclicity has not appeared in the DAG learning literature previously. We defer the discussion of other possible characterizations in the appendix. In the next section, we apply Theorem 1 to solve the program (3) to stationarity by treating it as an equality constrained program.

4 Optimization

Theorem 1 establishes a smooth, algebraic characterization of acyclicity that is also computable. As a consequence, the following equality-constrained program (ECP) is equivalent to (3):

$$(\mathsf{ECP}) \qquad \begin{array}{ll} \displaystyle\min_{W \in \mathbb{R}^{d \times d}} & F(W) \\ \text{subject to} & h(W) = 0. \end{array} \tag{9}$$

The main advantage of (ECP) compared to both (3) and (4) is its amenability to classical techniques from the mathematical optimization literature. Nonetheless, since $\{W : h(W) = 0\}$ is a nonconvex constraint, (9) is a nonconvex program, hence we still inherit the difficulties associated with nonconvex optimization. In particular, we will be content to find stationary points of (9); in Section 5.3 we compare our results to the global minimizer and show that the stationary points found by our method are close to global minima in practice.

In the follows, we outline the algorithm for solving (9). It consists of three steps: (i) converting the *constrained* problem into a sequence of *unconstrained* subproblems, (ii) optimizing the unconstrained subproblems, and (iii) thresholding. The full algorithm is outlined in Algorithm 1.

4.1 Solving the ECP with augmented Lagrangian

We will use the augmented Lagrangian method (e.g. Nemirovski, 1999) to solve (ECP), which solves the original problem augmented by a quadratic penalty:

$$\begin{array}{ll} \displaystyle\min_{W \in \mathbb{R}^{d \times d}} & F(W) + \dfrac{\rho}{2}|h(W)|^2 \\ \text{subject to} & h(W) = 0 \end{array} \tag{10}$$

with a penalty parameter $\rho > 0$. A nice property of the augmented Lagrangian method is that it approximates well the solution of a *constrained* problem by the solution of *unconstrained* problems *without* increasing the penalty parameter ρ to infinity (Nemirovski, 1999). The algorithm is essentially a dual ascent method for (10). To begin with, the dual function with Lagrange multiplier α is given by

$$D(\alpha) = \min_{W \in \mathbb{R}^{d \times d}} L^\rho(W, \alpha), \tag{11}$$

$$\text{where} \quad L^\rho(W, \alpha) = F(W) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{12}$$

is the augmented Lagrangian. The goal is to find a local solution to the dual problem

$$\max_{\alpha \in \mathbb{R}} \quad D(\alpha). \tag{13}$$

Let $W_\alpha^\star$ be the local minimizer of the Lagrangian (11) at α, i.e. $D(\alpha) = L^\rho(W_\alpha^\star, \alpha)$. Since the dual objective $D(\alpha)$ is linear in α, the derivative is simply given by $\nabla D(\alpha) = h(W_\alpha^\star)$. Therefore one can perform dual gradient ascent to optimize (13):

$$\alpha \leftarrow \alpha + \rho h(W_\alpha^\star), \tag{14}$$

where the choice of step size ρ comes with the following convergence rate:

Proposition 3 (Corollary 11.2.1, Nemirovski, 1999)**.** *For ρ large enough and the starting point α_0 near the solution $\alpha^\star$, the update (14) converges to $\alpha^\star$ linearly.*

In our experiments, typically fewer than 10 steps of the augmented Lagrangian scheme are required.

4.2 Solving the unconstrained subproblem

The augmented Lagrangian converts a *constrained* problem (10) into a sequence of *unconstrained* problems (11). We now discuss how to solve these subproblems efficiently. Let $\boldsymbol{w} = \text{vec}(W) \in \mathbb{R}^p$, with $p = d^2$. The unconstrained subproblem (11) can be considered as a typical minimization problem over real vectors:

$$\min_{\boldsymbol{w} \in \mathbb{R}^p} f(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1, \tag{15}$$

$$\text{where} \quad f(\boldsymbol{w}) = \ell(W; \mathbf{X}) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{16}$$

is the smooth part of the objective. Our goal is to solve the above problem to high accuracy so that $h(W)$ can be sufficiently suppressed.

In the special case of $\lambda = 0$, the nonsmooth term vanishes and the problem simply becomes an unconstrained smooth minimization, for which a number of efficient numerical algorithms are available, for instance the L-BFGS (Byrd et al., 1995). To handle the nonconvexity, a slight modification (Nocedal and Wright, 2006, Procedure 18.2) needs to be applied.

When $\lambda > 0$, the problem becomes composite minimization, which can also be efficiently solved by the proximal quasi-Newton (PQN) method (Zhong et al., 2014). At each step k, the key idea is to find the descent direction through a quadratic approximation of the smooth term:

$$\boldsymbol{d}_k = \arg\min_{\boldsymbol{d} \in \mathbb{R}^p} \ \boldsymbol{g}_k^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T B_k \boldsymbol{d} + \lambda \|\boldsymbol{w}_k + \boldsymbol{d}\|_1, \tag{17}$$

where $\boldsymbol{g}_k$ is the gradient of $f(\boldsymbol{w})$ and B_k is the L-BFGS approximation of the Hessian. Note that for each coordinate j, problem (17) has a closed form update $\boldsymbol{d} \leftarrow \boldsymbol{d} + z^\star e_j$ given by

$$z^\star = \arg\min_z \ \frac{1}{2} \underbrace{B_{jj}}_{a} z^2 + \underbrace{(\boldsymbol{g}_j + (B\boldsymbol{d})_j)}_{b} z + \lambda |\underbrace{\boldsymbol{w}_j + \boldsymbol{d}_j}_{c} + z| = -c + S\left(c - \frac{b}{a}, \frac{\lambda}{a}\right). \tag{18}$$

Moreover, the low-rank structure of B_k enables fast computation for coordinate update. As we describe in Appendix A, the precomputation time is only $O(m^2 p + m^3)$ where $m \ll p$ is the memory size of L-BFGS, and each coordinate update is $O(m)$. Furthermore, since we are using sparsity regularization, we can further speed up the algorithm by aggressively shrinking the active set of coordinates based on their subgradients (Zhong et al., 2014), and exclude the remaining dimensions from being updated. With the updates restricted to the active set $\mathcal{S}$, all dependencies of the complexity on $O(p)$ becomes $O(|\mathcal{S}|)$, which is substantially smaller. Hence the overall complexity of L-BFGS update is $O(m^2|\mathcal{S}| + m^3 + m|\mathcal{S}|T)$, where T is the number of inner iterations, typically $T = 10$.

4.3 Thresholding

In regression problems, it is known that post-processing estimates of coefficients via hard thresholding provably reduces the number of false discoveries (Zhou, 2009; Wang et al., 2016). Motivated by these encouraging results, we threshold the edge weights as follows: After obtaining a stationary point $\widetilde{W}_{\mathsf{ECP}}$ of (10), given a fixed threshold $\omega > 0$, set any weights smaller than ω in absolute value to zero. This strategy also has the important effect of "rounding" the numerical solution of the augmented Lagrangian (10), since due to numerical precisions the solution satisfies $h(\widetilde{W}_{\mathsf{ECP}}) \leq \epsilon$ for some small tolerance ϵ near machine precision (e.g. $\epsilon = 10^{-8}$), rather than $h(\widetilde{W}_{\mathsf{ECP}}) = 0$ strictly. However, since $h(\widetilde{W}_{\mathsf{ECP}})$ explicitly *quantifies* the "DAG-ness" of $\widetilde{W}_{\mathsf{ECP}}$ (see desiderata (b), Section 3), a small threshold ω suffices to rule out cycle-inducing edges.

5 Experiments

We compared our method against greedy equivalent search (GES) (Chickering, 2003; Ramsey et al., 2016), the PC algorithm (Spirtes et al., 2000), and LiNGAM (Shimizu et al., 2006). For GES, we used the fast greedy search (FGS) implementation from Ramsey et al. (2016). Since the accuracy of PC and LiNGAM was significantly lower than either FGS or NOTEARS, we only report the results against FGS here. This is consistent with previous work on score-based learning (Aragam and Zhou, 2015), which also indicates that FGS outperforms other techniques such as hill-climbing and MMHC (Tsamardinos et al., 2006). FGS was chosen since it is a state-of-the-art algorithm that scales to large problems.

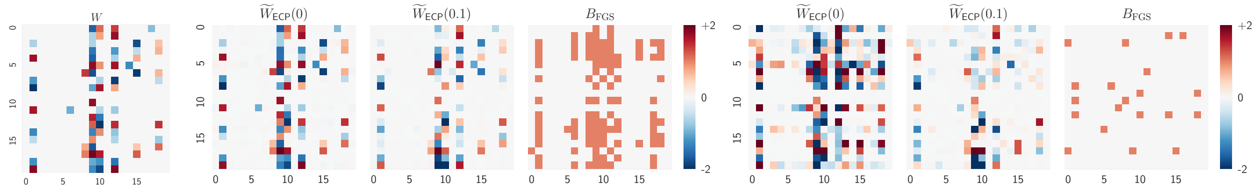
For brevity, we outline the basic set-up of our experiments here; precise details of our experimental set-up, including all parameter choices and more detailed evaluations, can be found in Appendix D. In each experiment, a random graph G was generated from one of two random graph models, Erdös-Rényi (ER) or scale-free (SF). Given G, we assigned uniformly random edge weights to obtain a weight matrix W. Given W, we sampled $X = W^T X + z \in \mathbb{R}^d$ from three different noise models: Gaussian (Gauss), Exponential (Exp), and Gumbel (Gumbel). Based on these models, we generated random datasets $\mathbf{X} \in \mathbb{R}^{n \times d}$ by generating rows i.i.d. according to one of these three models with $d \in \{10, 20, 50, 100\}$ and $n \in \{20, 1000\}$. Since FGS outputs a CPDAG instead of a DAG or weight matrix, some care needs to be taken in making comparisons; see Appendix D.1 for details.

5.1 Parameter estimation

We first performed a qualitative study of the solutions obtained by NOTEARS *without thresholding* by visualizing the weight matrix $\widetilde{W}_{\mathsf{ECP}}$ obtained by solving (ECP) (i.e. $\omega = 0$). This is illustrated in Figures 1 (ER-2) and 2 (SF-4). The key takeaway is that our method provides (empirically) consistent parameter estimates of the true weight matrix W. The final thresholding step in Algorithm 1 is only needed to ensure accuracy in structure learning. It also shows how effective is ℓ_1-regularization in small n regime.

5.2 Structure learning

We now examine our method for structure recovery, which is shown in Figure 3. For brevity, we only report the numbers for the structural Hamming distance (SHD) here, but complete figures and tables for additional



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 2: Parameter estimates of $\widetilde{W}_{\mathsf{ECP}}$ on a scale-free graph. Without the additional thresholding step in Algorithm 1, NOTEARS still produces consistent estimates of the true graph. The proposed method estimates the weights very well with large samples even without regularization, and remains accurate on insufficient samples when ℓ_1-regularization is introduced. See also Figure 1.

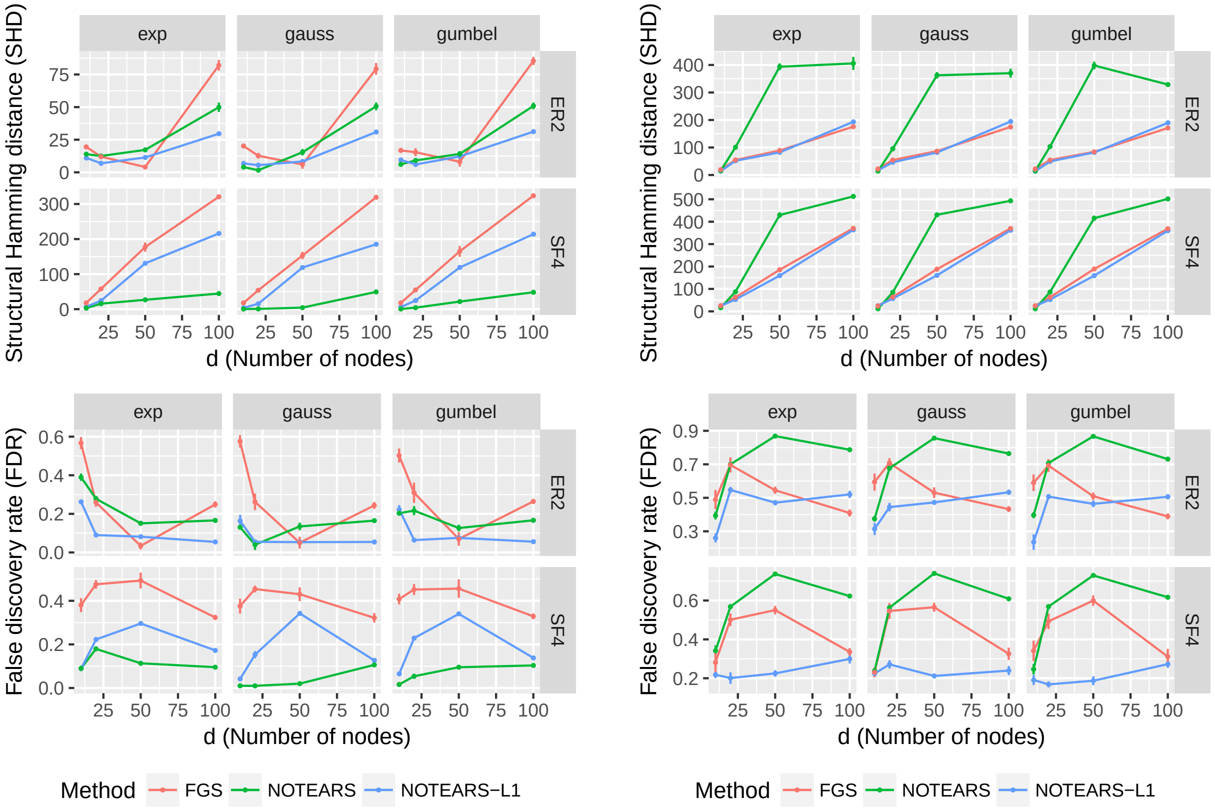
Table 1: Comparison of NOTEARS vs. globally optimal solution. $\Delta(W_{\mathsf{G}}, \widehat{W}) = F(W_{\mathsf{G}}) - F(\widehat{W})$.

n	λ	Graph	$F(W)$	$F(W_{\mathsf{G}})$	$F(\widehat{W})$	$F(\widetilde{W}_{\mathsf{ECP}})$	$\Delta(W_{\mathsf{G}}, \widehat{W})$	$\lVert\widehat{W} - W_{\mathsf{G}}\rVert$	$\lVert W - W_{\mathsf{G}}\rVert$
20	0	ER2	5.11	3.85	5.36	3.88	-1.52	0.07	3.38
20	0.5	ER2	16.04	12.81	13.49	12.90	-0.68	0.12	3.15
1000	0	ER2	4.99	4.97	5.02	4.95	-0.05	0.02	0.40
1000	0.5	ER2	15.93	13.32	14.03	13.46	-0.71	0.12	2.95
20	0	SF4	4.99	3.77	4.70	3.85	-0.93	0.08	3.31
20	0.5	SF4	23.33	16.19	17.31	16.69	-1.12	0.15	5.08
1000	0	SF4	4.96	4.94	5.05	4.99	-0.11	0.04	0.29
1000	0.5	SF4	23.29	17.56	19.70	18.43	-2.13	0.13	4.34

metrics can be found in the supplement. Consistent with previous work on greedy methods, FGS is very competitive when the number of edges is small (ER-2), but rapidly deteriorates for even modest numbers of edges (SF-4). In the latter regime, NOTEARS shows significant improvements. This is consistent across each metric we evaluated, and the difference grows as the number of nodes d gets larger. Also notice that our algorithm performs uniformly better for each noise model (Exp, Gauss, and Gumbel), without leveraging any specific knowledge about the noise type. Again, ℓ_1-regularizer helps significantly in the small n setting.

5.3 Comparison to exact global minimizer

In order to assess the ability of our method to solve the original program given by (3), we used the GOBNILP program (Cussens, 2012; Cussens et al., 2017) to find the exact minimizer of (3). Since this involves enumerating all possible parent sets for each node, these experiments are limited to small DAGs. Nonetheless, these small-scale experiments yield valuable insight into how well NOTEARS performs in actually solving the original problem. In our experiments we generated random graphs with $d = 10$, and then generated 10 simulated datasets containing $n = 20$ samples (for high-dimensions) and $n = 1000$ (for low-dimensions). We then compared the scores returned by our method to the exact global minimizer computed by GOBNILP along with the estimated parameters. The results are shown in Table 1. Surprisingly, although NOTEARS is only guaranteed to return a local minimizer, in many cases the obtained solution is very close to the global minimizer, as evidenced by deviations $\lVert\widehat{W} - W_{\mathsf{G}}\rVert$. Since the general structure learning problem is NP-hard, we suspect that although the models we have tested (i.e. ER and SF) appear amenable to fast solution, in the worst-case there are graphs which will still take exponential time to run or get stuck in a local minimum. Furthermore, the problem becomes more difficult as d increases. Nonetheless, this is encouraging evidence that the nonconvexity of (9) is a minor issue in practice. We leave it to future work to investigate these problems further.



(a) SHD with $n = 1000$

(b) SHD with $n = 20$

Figure 3: Structure recovery in terms of SHD and FDR to the true graph (lower is better). Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.

5.4 Real-data

We also compared FGS and NOTEARS on a real dataset provided by Sachs et al. (2005). This dataset consists of continuous measurements of expression levels of proteins and phospholipids in human immune system cells ($n = 7466$ $d = 11$, 20 edges). This dataset is a common benchmark in graphical models since it comes with a known *consensus network*, that is, a gold standard network based on experimental annotations that is widely accepted by the biological community. In our experiments, FGS estimated 17 total edges with an SHD of 22, compared to 16 for NOTEARS with an SHD of 22.

6 Discussion

We have proposed a new method for learning DAGs from data based on a continuous optimization program. This represents a significant departure from existing approaches that search over the discrete space of DAGs, resulting in a difficult optimization program. We also proposed two optimization schemes for solving the resulting program to stationarity, and illustrated its advantages over existing methods such as greedy equivalence search. Crucially, by performing global updates (e.g. all parameters at once) instead of local updates (e.g. one edge at a time) in each iteration, our method is able to avoid relying on assumptions about the local structure of the graph. To conclude, let us discuss some of the limitations of our method and possible directions for future work.

First, it is worth emphasizing once more that the equality constrained program (9) is a nonconvex program. Thus, although we overcome the difficulties of *combinatorial* optimization, our formulation still inherits the difficulties associated with *nonconvex* optimization. In particular, black-box solvers can at best find stationary points of (9). With the exception of exact methods, however, existing methods suffer from this drawback as well.[4] The main advantage of NOTEARS then is *smooth*, *global search*, as opposed to combinatorial, local search; and furthermore the search is delegated to standard numerical solvers.

Second, the current work relies on the smoothness of the score function, in order to make use of gradient-based numerical solvers to guide the graph search. However it is also interesting to consider non-smooth, even discrete scores such as BDe (Heckerman et al., 1995). Off-the-shelf techniques such as Nesterov's smoothing (Nesterov, 2005) could be useful, however more thorough investigation is left for future work.

Third, since the evaluation of the matrix exponential is $O(d^3)$, the computational complexity of our method is cubic in the number of nodes, although the constant is small for sparse matrices. In fact, this is one of the key motivations for our use of second-order methods (as opposed to first-order), i.e. to reduce the number of matrix exponential computations. By using second-order methods, each iteration make significantly more progress than first-order methods. Furthermore, although in practice not many iterations ($t \sim 10$) are required, we have not established any worst-case iteration complexity results. In light of the results in Section 5.3, we expect there are exceptional cases where convergence is slow. Notwithstanding, NOTEARS already outperforms existing methods when the in-degree is large, which is known difficult spot for existing methods. We leave it to future work to study these cases in more depth.

Lastly, in our experiments, we chose a fixed, suboptimal value of $\omega > 0$ for thresholding (Section 4.3). Clearly, it would be preferable to find a data-driven choice of ω that adapts to different noise-to-signal ratios and graph types. It is an intersting direction for future to study such choices.

The code is publicly available at https://github.com/xunzheng/notears.

References

A. H. Al-Mohy and N. J. Higham. A New Scaling and Squaring Algorithm for the Matrix Exponential. *SIAM Journal on Matrix Analysis and Applications*, 2009.

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

B. Aragam, A. A. Amini, and Q. Zhou. Learning directed acyclic graphs with penalized neighbourhood regression. *Submitted*, arXiv:1511.08963, 2016.

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

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

A. Botev, H. Ritter, and D. Barber. Practical gauss-newton optimisation for deep learning. *arXiv preprint arXiv:1706.03662*, 2017.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

R. R. Bouckaert. Probabilistic network construction using the minimum description length principle. In *European conference on symbolic and quantitative approaches to reasoning and uncertainty*, pages 41–48. Springer, 1993.

O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[4] GES (Chickering, 2003) is known to find the global minimizer in the limit $n \to \infty$ under certain assumptions, but this is not guaranteed for finite samples.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 1995.

E. Y.-J. Chen, Y. Shen, A. Choi, and A. Darwiche. Learning bayesian networks with ancestral constraints. In *Advances in Neural Information Processing Systems*, pages 2325–2333, 2016.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from data*, pages 121–130. Springer, 1996.

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

D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

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

J. Cussens, D. Haws, and M. Studenỳ. Polyhedral aspects of score equivalence in bayesian network structure learning. *Mathematical Programming*, 164(1-2):285–324, 2017.

B. Ellis and W. H. Wong. Learning causal Bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482), 2008.

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

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

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2): 106–148, 2011.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

J. Gu, F. Fu, and Q. Zhou. Penalized estimation of directed acyclic graphs from discrete data. *Statistics and Computing*, DOI: 10.1007/s11222-018-9801-y, 2018.

F. Harary and B. Manvel. On the number of cycles in a graph. *Matematickỳ časopis*, 1971.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Quic: quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(1):2911–2947, 2014.

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

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

J. Kuipers, G. Moffa, and D. Heckerman. Addendum on the scoring of gaussian directed acyclic graphical models. *The Annals of Statistics*, pages 1689–1691, 2014.

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

A. Nemirovski. Optimization II: Standard Numerical Methods for Nonlinear Continuous Optimization. 1999.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 2005.

T. Niinimäki, P. Parviainen, and M. Koivisto. Structure discovery in bayesian networks by sampling partial orders. *Journal of Machine Learning Research*, 17(1):2002–2048, 2016.

J. Nocedal and S. J. Wright. *Numerical Optimization.* 2006.

S. Ott and S. Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14: 124–133, 2003.

J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, pages 1–9, 2016.

R. W. Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial mathematics V*, pages 28–43. Springer, 1977.

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

M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon. Learning bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1864–1872, 2015.

M. Scanagatta, G. Corani, C. P. de Campos, and M. Zaffalon. Learning treewidth-bounded bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1462–1470, 2016.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using L1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

M. Schmidt, E. Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Artificial Intelligence and Statistics*, pages 456–463, 2009.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

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

A. P. Singh and A. W. Moore. Finding optimal bayesian networks by dynamic programming. 2005.

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

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pages 2722–2731, 2016.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Uncertainty in Artifical Intelligence (UAI)*, 2005.

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

P. Van Beek and H.-F. Hoffmann. Machine learning of bayesian networks using constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 429–445. Springer, 2015.

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

X. Wang, D. Dunson, and C. Leng. No penalty no tears: Least squares in high-dimensional linear models. In *International Conference on Machine Learning*, pages 1814–1822, 2016.

D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440, 1998.

J. Xiang and S. Kim. A* Lasso for learning a sparse Bayesian network structure for continuous variables. In *Advances in Neural Information Processing Systems*, pages 2418–2426, 2013.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1): 19–35, 2007.

B. Zhang, C. Gaiteri, L.-G. Bodea, Z. Wang, J. McElwee, A. A. Podtelezhnikov, C. Zhang, T. Xie, L. Tran, R. Dobrin, et al. Integrated systems approach identifies genetic nodes and networks in late-onset alzheimer's disease. *Cell*, 153(3):707–720, 2013.

K. Zhong, I. E.-H. Yen, I. S. Dhillon, and P. K. Ravikumar. Proximal quasi-newton for computationally intensive l1-regularized m-estimators. In *Advances in Neural Information Processing Systems*, pages 2375–2383, 2014.

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

S. Zhou. Thresholding procedures for high dimensional variable selection and statistical estimation. In *Advances in Neural Information Processing Systems*, pages 2304–2312, 2009.

Algorithm 2 Proximal Quasi-Newton for unconstrained problem (Zhong et al., 2014)

```
1. Input: $\boldsymbol{w}_0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{w}_0)$, active set $\mathcal{S} = [p]$.
2. For $k = 0, 1, 2, \ldots$:
    (a) Shrink $\mathcal{S}$ to rule out $j$ with $w_j = 0$ or small subgradient $|\partial_j L(\boldsymbol{w})|$
    (b) If shrinking stopping criteria is satisfied
        i. Reset $\mathcal{S} = [p]$ and L-BFGS memory
        ii. Update shrinking stopping criteria and continue
    (c) Solve (17) for descent direction $\boldsymbol{d}_k$ using coordinate update (18) on active set
    (d) Line search for step size $\eta \in (0, 1]$ until Armijo rule is satisfied:
            $$f(\boldsymbol{w}_k + \eta \boldsymbol{d}_k) \le f(\boldsymbol{w}_k) + \eta c_1 (\lambda \|\boldsymbol{w}_k + \boldsymbol{d}_k\|_1 - \lambda \|\boldsymbol{w}_k\| + \boldsymbol{g}_k^T \boldsymbol{d}_k), \quad (20)$$
        where $c_1$ is some small constant, typically set to $10^{-3}$ or $10^{-4}$.
    (e) Generate new iterate $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \eta \boldsymbol{d}_k$
    (f) Update $\boldsymbol{g}, \boldsymbol{s}, \boldsymbol{y}, Q, R, \widehat{Q}$ restricted to $\mathcal{S}$
```

A Details of Proximal Quasi-Newton

Recall $B_k \in \mathbb{R}^{p \times p}$ is the low-rank approximation of the Hessian matrix given by L-BFGS updates. Let the memory size of L-BFGS be m, which is taken to be $m \ll p$. The compact form of L-BFGS update can be written as

$$B_k = \gamma_k I - Q\widehat{Q}, \tag{19}$$

where

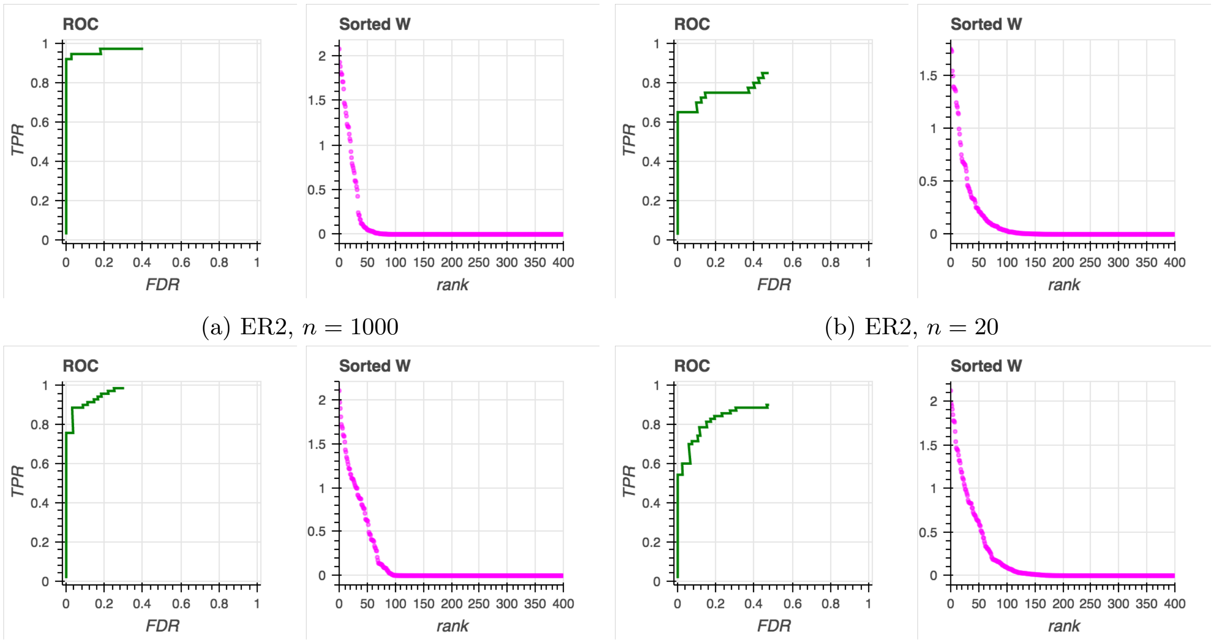
$$Q = \begin{bmatrix} \gamma_k S_k & Y_k \end{bmatrix}, \ R = \begin{bmatrix} \gamma_k S_k^T S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1}, \ \widehat{Q} = RQ^T,$$
$$S_k = \begin{bmatrix} \boldsymbol{s}_{k-m} & \cdots & \boldsymbol{s}_{k-1} \end{bmatrix}, \ Y_k = \begin{bmatrix} \boldsymbol{y}_{k-m} & \cdots & \boldsymbol{y}_{k-1} \end{bmatrix},$$
$$\boldsymbol{s}_k = \boldsymbol{w}_{k+1} - \boldsymbol{w}_k, \ \boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k, \ \gamma_k = \boldsymbol{y}_{k-1}^T \boldsymbol{y}_{k-1} / \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1},$$
$$D_k = \text{diag} \begin{bmatrix} \boldsymbol{s}_{k-m}^T \boldsymbol{y}_{k-m} & \cdots & \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}, \ (L_k)_{ij} = \begin{cases} \boldsymbol{s}_{k-m+i-1}^T \boldsymbol{y}_{k-m+j-1} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}.$$

The low rank structure of B_k enables fast computation of subsequent coordinate descent procedure. Specifically, notice that all $Q, R, \widehat{Q}$, and $\text{diag}(B)$ can be precomputed in $O(m^2 p + m^3)$ time, which is significantly smaller than naive Hessian inversion $O(p^3)$. After precomputation, in each coordinate update, both a and c in (18) can be computed and updated in $O(1)$ time. Moreover, let $\widehat{\boldsymbol{d}} = \widehat{Q}\boldsymbol{d} \in \mathbb{R}^{2m}$, we have $(B\boldsymbol{d})_j = \gamma \boldsymbol{d}_j - Q_{j,:}\widehat{\boldsymbol{d}}$, which suggests b in (18) only requires $O(m)$ to compute and update. Therefore each coordinate update is $O(m)$.

The detailed procedure of PQN is outlined in Algorithm 2.

B Sensitivity of threshold

We demonstrate the effect of threshold in Figure 4. For each setting, we computed the "ROC" curve for FDR and TPR with varying level of threshold, while ensuring the resulting graph is indeed a DAG. On the right,



(a) ER2, $n = 1000$

(b) ER2, $n = 20$

(c) SF4, $n = 1000$

(d) SF4, $n = 20$

Figure 4: Illustration of the effect of the threshold with $d = 20$ and $\lambda = 0.1$. For each subfigure, ROC curve (left) shows FDR and TPR with varying level of threshold, and sorted weights (right) plots the entries of $\widetilde{W}_{\mathsf{ECP}}$ in decreasing order.

we also present the estimated edge weights of $\widetilde{W}_{\mathsf{ECP}}$ in decreasing order. One can first observe that in all cases most of the edge weights are equal or close to zero as expected. The remaining question is how to choose a threshold that separates out these (near zero) from signals (away from zero) so that best performance can be achieved. With enough samples, one can often notice a sudden change in the weight distribution as in Figure 4(a)(c). With insufficient samples, the breakpoint is less clear, and the optimal choice that balances between TPR and FDR is depends on the specific settings. Nonetheless, the predictive performance is less sensitive to threshold value as one can see from the slope of the decrease in the weights before getting close to zero. Indeed, in our experiments, we found a fixed threshold $\omega = 0.3$ is a suboptimal yet reasonable choice across many different settings.

C Sensitivity of weight scale

We investigate the effect of weight scaling to the NOTEARS algorithm in Figure 5. In particular, we run experiments with $w_{ij} \in \alpha \cdot [0.5, 2] \cup -\alpha \cdot [0.5, 2]$ with $\alpha \in \{1.0, 0.9, 0.8, \ldots, 0.1\}$. On the left, we plot the smallest threshold ω required to obtain a DAG (see Section 4.3) for different scale α. Overall, across different values of α, the variation in the smallest ω required is minimal. We also hasten to point out that this also decreases the signal to noise ratio (SNR), which more directly affects the accuracy. Indeed, in the figure on the right, we can observe (as expected) some performance drop when using smaller value of α.

D Experiments

D.1 Experiment details

We used simulated graphs from two well-known ensembles of random graphs:

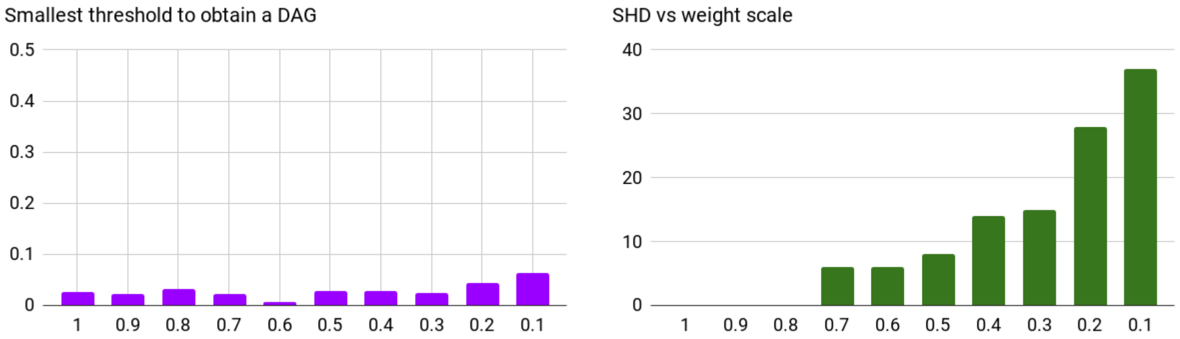


Figure 5: Varying weight scale $\alpha \in \{1.0, \dots, 0.1\}$ with $d = 20$ and $n = 1000$ on an ER-2 graph. (Left) Smallest threshold ω such that $\widehat{W}$ is a DAG. (Right) SHD between ground truth and NOTEARS, lower the better. The minimum ω remains stable, while the accuracy of NOTEARS drops as expected since the SNR decreases with α.

- *Erdös-Rényi (ER).* Random graphs whose edges are added independently with equal probability p. We simulated models with d, $2d$, and $4d$ edges (in expectation) each, denoted by ER-1, ER-2, and ER-4, respectively.
- *Scale-free networks (SF).* Networks simulated according to the preferential attachment process described in Barabási and Albert (1999). We simulated scale-free networks with $4d$ edges and $\beta = 1$, where β is the exponent used in the preferential attachment process.

Scale-free graphs are popular since they exhibit topological properties similar to real-world networks such as gene networks, social networks, and the internet. Given a random acyclic graph $B \in \{0,1\}^{d\times d}$ from one of these two ensembles, we assigned edge weights independently from Unif $\big([-2,-0.5] \cup [0.5,2]\big)$ to obtain a weight matrix $W = [w_1 \,|\, \cdots \,|\, w_d] \in \mathbb{R}^{d\times d}$. Given W, we sampled $X = W^T X + z \in \mathbb{R}^d$ according to the following three noise models:

- *Gaussian noise* (Gauss). $z \sim \mathcal{N}(0, I_{d\times d})$.
- *Exponential noise* (Exp). $z_j \sim \text{Exp}(1)$, $j = 1, \dots, d$.
- *Gumbel noise* (Gumbel). $z_j \sim \text{Gumbel}(0,1)$, $j = 1, \dots, d$.

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 samples for graphs with $d \in \{10, 20, 50, 100\}$ nodes. To study both high- and low-dimensional settings, we used $n \in \{20, 1000\}$.

For each dataset, we ran FGS, PC, and LinGAM and NOTEARS to compare the performance in reconstructing the DAG B. We used the following implementations:

- FGS and PC were implemented through the `py-causal` package, available at https://github.com/bd2kccd/py-causal. Both of these methods are written in highly optimized Java code.
- LinGAM was implemented using the author's Python code: https://sites.google.com/site/sshimizu06/lingam.

Since the accuracy of PC and LiNGAM was significantly lower than either FGS or NOTEARS, we only report the results against FGS. A few comments on FGS are in order: 1) FGS estimates a graph, so it does not output any parameter estimates; 2) Instead of returning a DAG, FGS returns a CPDAG (Chickering, 2003), which contains undirected edges; 3) FGS has a single tuning parameter that controls the strength of regularization. Thus, in our evaluations, we treated FGS favourably by treating undirected edges as true positives as long as the true graph had a directed edge in place of the undirected edge. For tuning parameters, we used the values suggested by the authors of the FGS code.

Denote the estimate returned by FGS by B_{FGS}. As discussed in Appendix B, we fix the threshold at $\omega = 0.3$. Having fixed ω, when there is no regularization, NOTEARS requires no tuning. With ℓ_1-regularization, NOTEARS-ℓ_1 requires a choice of λ which wes selected as follows: Based on the estimate returned by FGS, we tuned λ so that the selected graph (after thresholding) had the same number of edges as B_{FGS} (or as close as possible). This ensures that the results are not influenced by hyperparameter tuning, and fairly compares each method on graphs of roughly the same complexity. Denote this estimate by $\widehat{W}$ and the resulting adjacency matrix by $\widehat{B} = \mathcal{A}(\widehat{W})$.

D.2 Metrics

We evaluated the learned graphs on four common graph metrics: 1) False discovery rate (FDR), 2) True positive rate (TPR), 3) False positive rate (FPR), and 4) Structural Hamming distance (SHD). Recall that SHD is the total number of edge additions, deletions, and reversals needed to convert the estimated DAG into the true DAG. Since we consider directed graphs, a distinction between True Positives (TP) and Reversed edges (R) is needed: the former is estimated with correct direction whereas the latter is not. Likewise, a False Positive (FP) is an edge that is not in the undirected skeleton of the true graph. In addition, Positive (P) is the set of estimated edges, True (T) is the set of true edges, False (F) is the set of non-edges in the ground truth graph. Finally, let (E) be the extra edges from the skeleton, (M) be the missing edges from the skeleton. The four metrics are then given by:

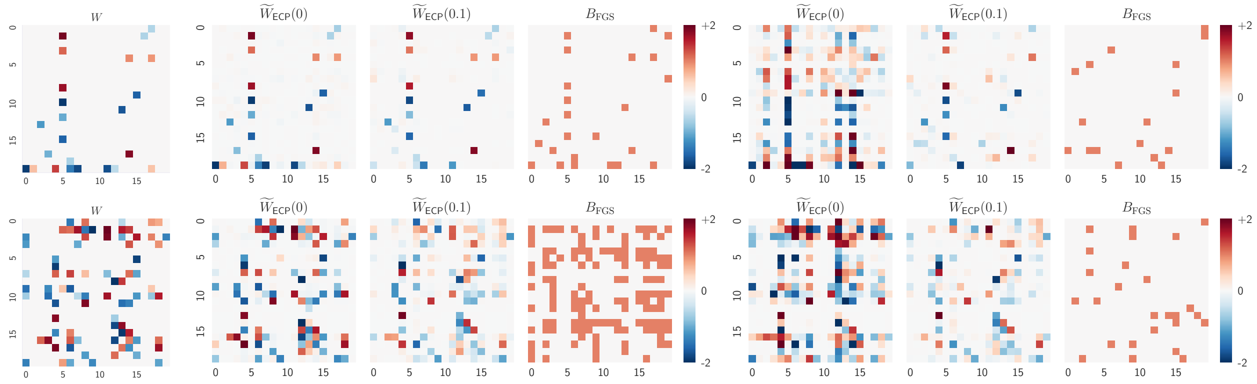
1. FDR $= (R + FP)/P$
2. TPR $= TP/T$
3. FPR $= (R + FP)/F$
4. SHD $= E + M + R$.

D.3 Further evaluations

Figure 6 shows learned weighted adjacency matrices for ER1 and ER4. One can observe the same trend: with large n, both regularized and unregularized NOTEARS works well compared to FGS, and with small n, due to identifiability, the unregularized NOTEARS suffers significantly, yet with the help of ℓ_1-regularization we can still accurately recover the true underlying graph.

Figure 7 and Figure 8 shows structure recovery results for $n = 1000$ and $n = 20$ for various random graphs and SEM noise types. Other than fixed ω as in the main paper, we also included the optimal choice of thresholding, marked as “best”. The trend is consistent with the main text: our method in general outperforms FGS, without tuning ω to the optimum for each setting.

Table 2 extends the global minimizer result for various random graph types. For each random graph and samples, we computed exact local scores as inputs to GOBNILP program, which finds the globally optimal structure for the given score. We can again observe that the difference between our estimate $\widehat{W}$ and global minimizer W_{G} is small across all cases.



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 6: Visual comparison of the learned weighted adjacency matrix on a 20-node graph with $n = 1000$ (large samples) and $n = 20$ (insufficient samples): $\widetilde{W}_{\mathsf{ECP}}(\lambda)$ is the proposed NOTEARS algorithm with ℓ_1-regularization λ, and B_{FGS} is the binary estimate of the baseline (Ramsey et al., 2016). Top row: ER1, bottom row: ER4.

Table 2: Comparison of NOTEARS vs. globally optimal solution. $\Delta(W_{\mathsf{G}}, \widehat{W}) = F(W_{\mathsf{G}}) - F(\widehat{W})$.

n	λ	Graph	$F(W)$	$F(W_{\mathsf{G}})$	$F(\widehat{W})$	$F(\widetilde{W}_{\mathsf{ECP}})$	$\Delta(W_{\mathsf{G}}, \widehat{W})$	$\|\widehat{W} - W_{\mathsf{G}}\|$	$\|W - W_{\mathsf{G}}\|$
20	0.00	ER1	5.01	3.69	5.19	3.73	-1.50	0.09	3.54
20	0.50	ER1	12.43	9.90	10.69	9.88	-0.78	0.11	2.76
1000	0.00	ER1	4.96	4.93	4.97	4.92	-0.04	0.03	0.35
1000	0.50	ER1	12.37	10.53	11.01	10.58	-0.48	0.11	2.47
20	0.00	ER2	5.11	3.85	5.36	3.88	-1.52	0.07	3.38
20	0.50	ER2	16.04	12.81	13.49	12.90	-0.68	0.12	3.15
1000	0.00	ER2	4.99	4.97	5.02	4.95	-0.05	0.02	0.40
1000	0.50	ER2	15.93	13.32	14.03	13.46	-0.71	0.12	2.95
20	0.00	ER4	4.76	3.66	5.23	3.88	-1.57	0.08	4.25
20	0.50	ER4	28.24	16.38	19.81	16.82	-3.44	0.15	6.66
1000	0.00	ER4	5.03	5.00	5.50	4.97	-0.50	0.00	0.46
1000	0.50	ER4	28.51	18.29	29.91	18.69	-11.61	0.13	5.76
20	0.00	SF4	4.99	3.77	4.70	3.85	-0.93	0.08	3.31
20	0.50	SF4	23.33	16.19	17.31	16.69	-1.12	0.15	5.08
1000	0.00	SF4	4.96	4.94	5.05	4.99	-0.11	0.04	0.29
1000	0.50	SF4	23.29	17.56	19.70	18.43	-2.13	0.13	4.34

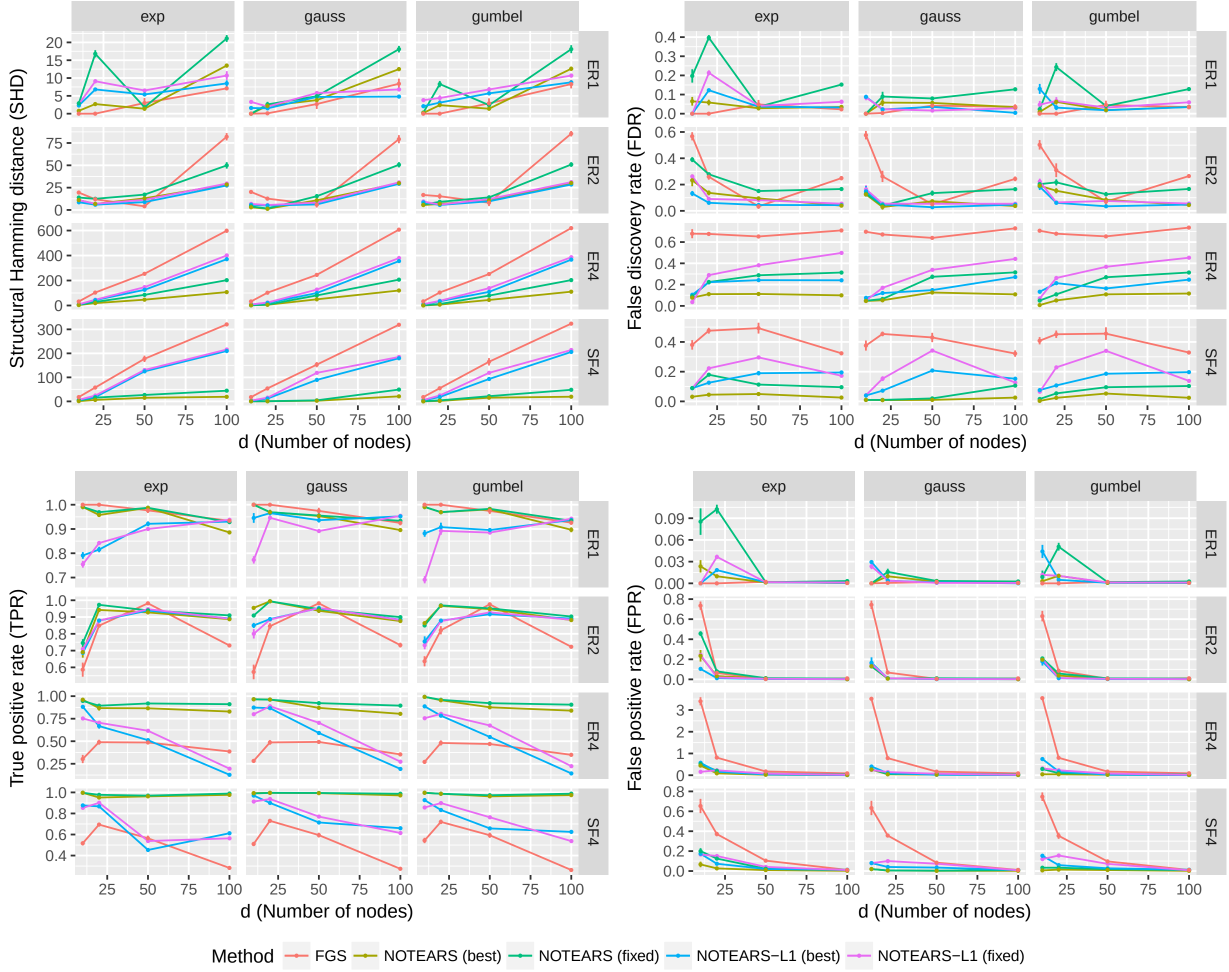


Figure 7: Structure recovery results for $n = 1000$. Lower is better, except for TPR (lower left), for which higher is better. Rows: random graph types, $\{\text{ER},\text{SF}\}\text{-}k = \{$Erdös-Rényi, scale-free$\}$ graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.

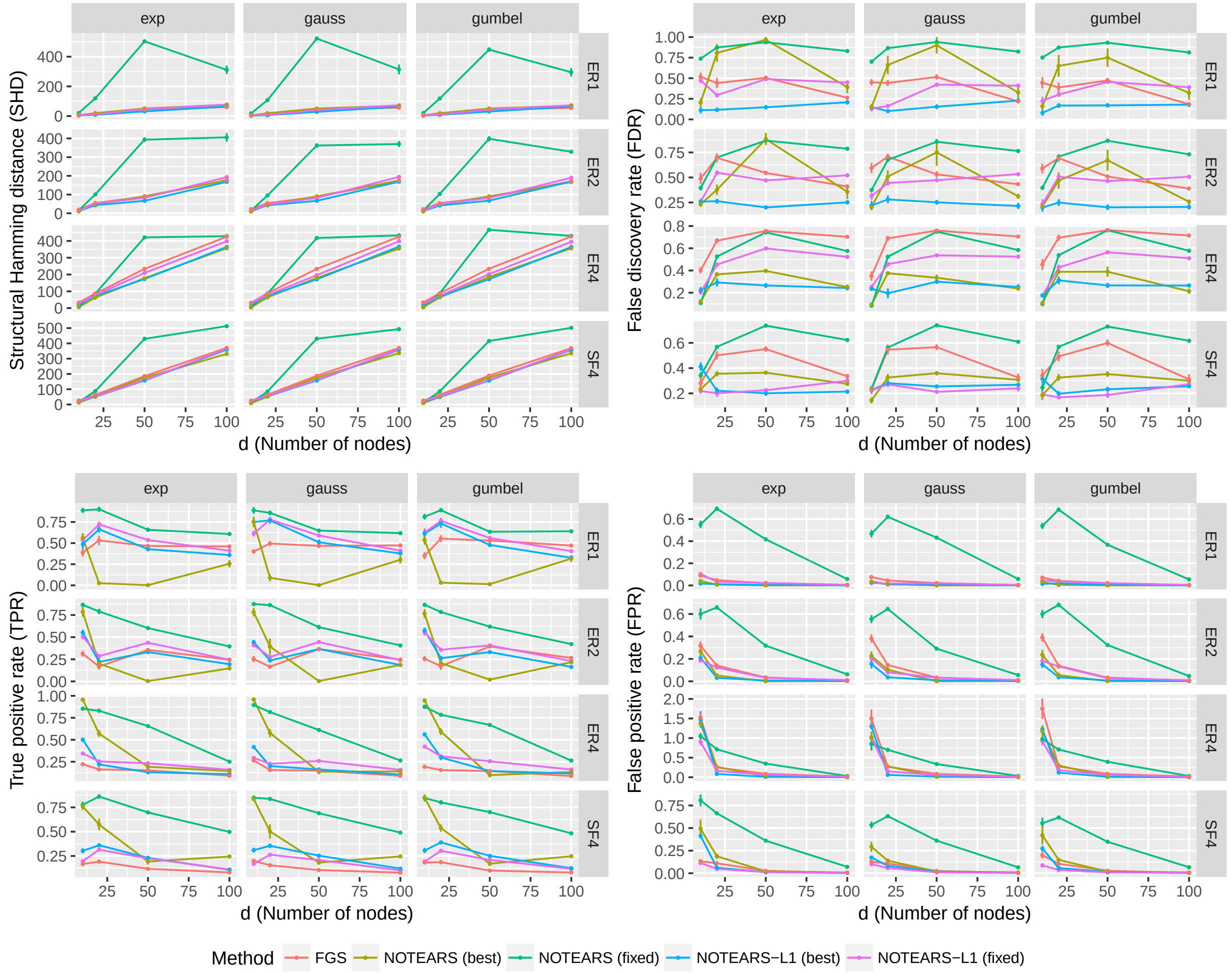


Figure 8: Structure recovery results for $n = 20$. Lower is better, except for TPR (lower left), for which higher is better. Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.