Truncated Matrix Power Iteration for Differentiable DAG Learning

Zhen Zhang[*,1], **Ignavier Ng**[*,2], **Dong Gong**[3], **Yuhang Liu**[1], **Ehsan M Abbasnejad**[1], **Mingming Gong**[4], **Kun Zhang**[2,5], and **Javen Qinfeng Shi**[1]

[1]*The University of Adelaide* [2]*Carnegie Mellon University*
[3]*The University of New South Wales* [4]*The University of Melbourne*
[5]*Mohamed bin Zayed University of Artificial Intelligence*
{zhen.zhang02,yuhang.liu01,ehsan.abbasnejad,javen.shi}@adelaide.edu.au
{ignaviernng,kunz1}@cmu.edu dong.gong@unsw.edu.au mingming.gong@unimelb.edu.au

Abstract

Recovering underlying Directed Acyclic Graph (DAG) structures from observational data is highly challenging due to the combinatorial nature of the DAG-constrained optimization problem. Recently, DAG learning has been cast as a continuous optimization problem by characterizing the DAG constraint as a smooth equality one, generally based on polynomials over adjacency matrices. Existing methods place very small coefficients on high-order polynomial terms for stabilization, since they argue that large coefficients on the higher-order terms are harmful due to numeric exploding. On the contrary, we discover that large coefficients on higher-order terms are beneficial for DAG learning, when the spectral radiuses of the adjacency matrices are small, and that larger coefficients for higher-order terms can approximate the DAG constraints much better than the small counterparts. Based on this, we propose a novel DAG learning method with efficient truncated matrix power iteration to approximate geometric series based DAG constraints. Empirically, our DAG learning method outperforms the previous state-of-the-arts in various settings, often by a factor of 3 or more in terms of structural Hamming distance.

1 Introduction

Recovery of Directed Acyclic Graphs (DAGs) from observational data is a classical problem in many fields, including bioinformatics [31, 49], machine learning [19], and causal inference [39]. The graphical model produced by a DAG learning algorithm allows one to decompose the joint distribution over the variables of interest in a compact and flexible manner, and under certain assumptions [27, 39], these graphical models can have causal interpretations.

DAG learning methods can be roughly categorized into constraint-based and score-based methods. Most constraint-based approaches, *e.g.*, PC [37], FCI [11, 38], rely on conditional independence test and thus may require a large sample size [33, 43]. The score-based approaches, including exact methods based on dynamic programming [18, 35, 36], A* search [47, 48], and integer programming [13], as well as greedy methods like GES [9], model the validity of a graph according to some score function and are often formulated and solved as a discrete optimization problem. A key challenge for score-based methods is the combinatorial search space of DAGs [8, 10].

There is a recent interest in developing continuous optimization methods for DAG learning in the past few years. In particular, Zheng et al. [50] develops an algebraic characterization of DAG constraint

[*]Equal Contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

based on a continuous function, specifically matrix exponential, of the weighted adjacency matrix of a graph, and applies it to estimate linear DAGs with equal noise variances using continuous constrained optimization [4, 5]. The resulting algorithm is called NOTEARS. Yu et al. [45] proposes an alternative DAG constraint based on powers of a binomial to improve practical convenience. Wei et al. [44] unifies these two DAG constraints by writing them in terms of a general polynomial one. These DAG constraints have been applied to more complex settings, *e.g.*, with nonlinear relationships [20, 25, 45, 51], time series [26], confounders [6], and flexible score functions [53], and have been shown to produce competitive performance. Apart from the polynomial based constraints, Lee et al. [21], Zhu et al. [52] present alternative formulations of the DAG constraints based on the spectral radius of the weighted adjacency matrix. Instead of solving a constrained optimization problem, Yu et al. [46] proposes NOCURL which employs an algebraic representation of DAGs such that continuous unconstrained optimization can be carried out in the DAG space directly. However, the focus of these works [21, 46, 52] is to improve the scalability, and the performance of the resulting methods may degrade.

To avoid potential numeric exploding that may be caused by high-order terms in the polynomial-based DAG constraints, previous works use very small coefficients for high-order terms in DAG constraints [45, 50]. More precisely, the coefficients decrease quickly to very small values for the relatively higher order terms that may contain useful higher-order information. This may cause difficulties for the DAG constraints to capture the higher-order information for enforcing acyclicity. With such DAG constraints, a possible solution to maintain higher-order information for acyclicity is to apply an even larger penalty weight [45, 50], which, however, causes ill-conditioning issues as demonstrated by Ng et al. [24]. Moreover, these small coefficients will be multiplied on the gradients of higher-order terms during the optimization process, leading to gradient vanishing. In this paper, we show that, on the contrary, larger coefficients may be safely used to construct more informative higher-order terms without introducing numeric exploding. The reason is that the weighted adjacency matrix of a DAG must be nilpotent; thus the candidate adjacency matrix often has a very small spectral radius. As a result, the higher-order power of the matrix may be very close to zero matrix, and larger coefficients may be safely used without causing numeric exploding. We thus propose to use geometric series-based DAG constraint without small coefficients for more accurate and robust DAG learning. To relieve the computational burden from the geometric series, we propose an efficient Truncated Matrix Power Iteration (TMPI) algorithm with a theoretically guaranteed error bound. We summarize our contributions as the following:

- We demonstrate that the key challenge for DAG learning is the gradient vanishing issue instead of gradient and numeric exploding. For sparse graphs, the gradient vanishing issue can be severe.
- We design a DAG constraint based on finite geometric series which is an order-d polynomial over adjacency matrices to escape from gradient vanishing. We show that the relatively large coefficients on the higher-order terms would not result in gradient exploding in practice.
- We show that there exists some constant $k \leqslant d$, *s.t.* our order-d polynomial constraint can be reduced to order-k polynomial without expanding its feasible set. Though finding such value of k requires solving the NP-hard longest simple path problem, we develop a simple heuristic to find such k with bounded error to the exact, order-d, DAG constraint. Based on this result, we propose a DAG constraint based on efficient Truncated Matrix Power Iteration (TMPI), and conduct experiments to demonstrate that the resulting DAG learning method outperforms previous methods by a large margin.
- We provide a systematical empirical comparison of the existing DAG constraints, including the polynomial and spectral radius based constraints, as well as the algebraic DAG representation of NOCURL.

2 Preliminaries

DAG Model and Linear SEM Given a DAG model (a.k.a. Bayesian network) defined over random vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top$ and DAG $\mathcal{G}$, the distribution $P(\mathbf{x})$ and DAG $\mathcal{G}$ are assumed to satisfy the Markov assumption [27, 39]. We say that $\mathbf{x}$ follows a linear Structural Equation Model (SEM) if

$$\mathbf{x} = \mathbf{B}^\top \mathbf{x} + \mathbf{e}, \tag{1}$$

where $\mathbf{B}$ is the weighted adjacency matrix representing the DAG $\mathcal{G}$, and $\mathbf{e} = [e_1, e_2, \ldots, e_d]^\top$ denotes the exogenous noise vector consisting of d independent random variables. With slight abuse of notation, we denote by $\mathcal{G}(\mathbf{B})$ the graph induced by weighted adjacency matrix $\mathbf{B}$. Moreover, we do not distinguish between random variables and vertices or nodes, and use these terms interchangeably.

Our goal is to estimate the DAG $\mathcal{G}$ from n i.i.d. samples of $\mathbf{x}$, indicated by the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. In general, the DAG $\mathcal{G}$ can only be identified up to its Markov equivalence class under the faithfulness [39] or sparsest Markov representation [30] assumption. It has been shown that for linear SEMs with homoscedastic errors, from which the noise terms are specified up to a constant [22], and for linear non-Gaussian SEMs, from which no more than one of the noise term is Gaussian [34], the true DAG can be fully identified. In this work, we focus on the linear SEMs with equal noise variances [28], which can be viewed as an instance of the former.

DAG Constraints for Differentiable DAG Learning Recently, Zheng et al. [50] reformulated the DAG learning problem as the following continuous optimization problem with convex least-squares objective but non-convex constraint,

$$\min_{\mathbf{B} \in \mathbb{R}^{d \times d}} \ \| \mathbf{X} - \mathbf{X}\,\mathbf{B} \|_F^2 + \eta \| \mathbf{B} \|_1, \quad \text{subject to } h_{\text{exp}}(\mathbf{B} \odot \mathbf{B}) = 0, \tag{2}$$

where $\| \cdot \|_F$ and $\| \cdot \|_1$ denote the Frobenius norm and elementwise ℓ_1 norm, respectively, $\eta > 0$ is the regularization coefficient that controls the sparsity of $\mathbf{B}$, and $\odot$ denotes the Hadamard product, which is used to map $\mathbf{B}$ to a positive weighted adjacency matrix with the same structure. Given any $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d \times d}$, the function h_{exp} is defined as

$$h_{\text{exp}}(\tilde{\mathbf{B}}) = \operatorname{tr}(e^{\tilde{\mathbf{B}}}) - d = \operatorname{tr}\left(\sum_{i=1}^{\infty} \frac{1}{i!} \tilde{\mathbf{B}}^i \right), \tag{3}$$

and can be efficiently computed using various approaches [1, 2]. The key here is that $h_{\text{exp}}(\tilde{\mathbf{B}}) = 0$ if and only if $\mathcal{G}(\tilde{\mathbf{B}})$ is a DAG. The formulation (2) involves a hard constraint, and Zheng et al. [50] adopted the augmented Lagrangian method [4, 5] to solve the constrained optimization problem. Ng et al. [24] demonstrated that the augmented Lagrangian method here behaves similarly to the classical quadratic penalty method [15, 29] in practice, and showed that it is guaranteed to return a DAG under certain conditions.

To improve practical convenience, Yu et al. [45] proposed a similar DAG constraint using an order-d polynomial based on powers of a binomial, given by

$$h_{\text{bin}}(\tilde{\mathbf{B}}) = \operatorname{tr}\left(\mathbb{I} + \alpha \tilde{\mathbf{B}} \right)^d - d = \operatorname{tr}\left(\sum_{i=1}^{d} \binom{d}{i} \alpha^i \tilde{\mathbf{B}}^i \right), \tag{4}$$

where $\alpha > 0$ is a hyperparameter and is often set to $1/d$, *e.g.*, in the implementation of DAG-GNN [45]. This function can be evaluated using $O(\log d)$ matrix multiplications using the exponentiation by squaring algorithm. Both $h_{\text{exp}}(\tilde{\mathbf{B}})$ and $h_{\text{bin}}(\tilde{\mathbf{B}})$ are formulated by Wei et al. [44] as a unified one

$$h_{\text{poly}}(\tilde{\mathbf{B}}) = \operatorname{tr}\left(\sum_{i=1}^{d} c_i \tilde{\mathbf{B}}^i \right), \quad c_i > 0, \quad i = 1, 2, \ldots, d. \tag{5}$$

3 DAG Learning with Truncated Matrix Power Iteration

In this section, we first show that gradient vanishing is one main issue for previous DAG constraints and propose a truncated geometric series based DAG constraints to escape from the issue in Section 3.1. Then in Section 3.2 the theoretical properties of our DAG constraints are analyzed. In Section 3.3 we provide an efficient algorithm for our DAG constraints. In Section 3.4, we discuss the relation between our algorithm and previous works.

3.1 Gradient Vanishing Issue and Truncated Geometric DAG Constraint

Theoretically, any positive values of c_i suffice to make the DAG constraint term $h_{\text{poly}}(\tilde{\mathbf{B}})$ in Eq. (5) a valid one, *i.e.*, it equals zero if and only if the graph $\mathcal{G}(\tilde{\mathbf{B}})$ is acyclic. Thus previously small

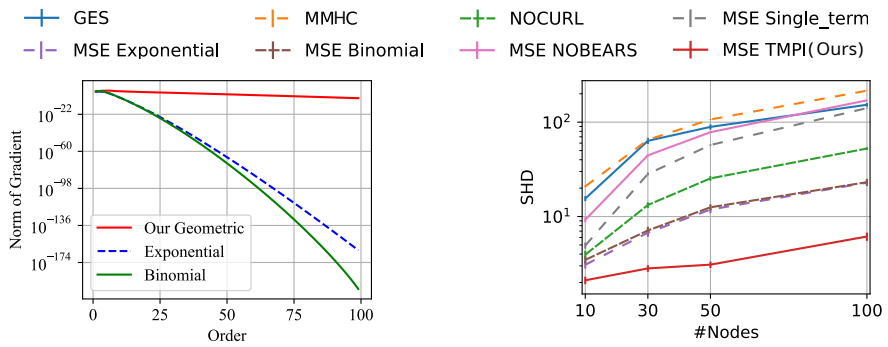


Figure 1: Previous polynomial based DAG constraints, including exponential (Eq. (3)) and binomial (Eq. (4) with $\alpha = 1/d$) constraints, suffer from gradient vanishing. **Left:** The norm of gradient on higher-order terms w.r.t. order on a 100-node graph generated using the ER2 setting in Section 4. Note that the gradients on different order terms refer to $\nabla_{\tilde{\mathbf{B}}} c_i \operatorname{tr}(\tilde{\mathbf{B}}^i)$, where, for different polynomial based DAG constraints, the coefficients c_i are different. **Right:** We propose a geometric series based DAG constraint, named Truncated Matrix Power Iteration (TMPI), to escape from vanished gradients, which leads to a significant performance improvement in terms of Structural Hamming Distance (SHD) on ER2 graphs (see Section 4 for full experimental details and results).

coefficients for higher-order terms are preferred since they are known to have better resistance to numeric exploding of higher-order polynomials [45, 50]. These small coefficients forbid informative higher-order terms and cause severe gradient vanishing, as nilpotent properties of DAG result in candidate adjacency matrix with very small spectral radius (see empirical analysis in Figure 1 **left**). In this case, the weight updates during the optimization process can hardly utilize this higher-order information from the vanishingly small gradients. As a consequence, only lower-order information can be effectively captured, which may be detrimental to the quality of final solution.

To avoid the severe vanishing gradient issue in existing DAG constraints (*e.g.*, Eq. (3), (4)), we propose to discard the coefficients and use a finite geometric series based function:

$$h_{\text{geo}}(\tilde{\mathbf{B}}) = \operatorname{tr}\left(\sum_{i=1}^{d} \tilde{\mathbf{B}}^i\right), \tag{6a}$$

$$\nabla_{\tilde{\mathbf{B}}} h_{\text{geo}}(\tilde{\mathbf{B}}) = \left(\sum_{i=0}^{d-1} (i+1)\tilde{\mathbf{B}}^i\right)^\top. \tag{6b}$$

The gradient of h_{geo} w.r.t. $\tilde{\mathbf{B}}$ denoted by $\nabla_{\tilde{\mathbf{B}}} h_{\text{geo}}(\tilde{\mathbf{B}})$ is an order-$(d-1)$ polynomial of $\tilde{\mathbf{B}}$.

Compared to (3) and (4), Eq. (6a) is a more precise DAG constraint escaped from from severe gradient vanishing. Also it is notable that due to the nilpotent properties of weighted adjacency matrices of DAGs, the higher-order terms in (6a) and (6b) often comes with small norms; thus, our DAG constraint may not suffer from numeric exploding in practice, based on the observations in Section 4.

In Figure 1 we compared the norm of gradients on higher-order terms of different DAG constraints, where we observe that for previous exponential-based (3) and binomial-based (4) DAG constraints, the gradients of higher-order terms converges to 0 very quickly as the order increases. Meanwhile, the gradient of higher-order terms in (6a) converges to 0 with a much slower speed and thus informative higher-order terms are preserved.

We further show that those non-informative higher-order terms that are close to zero in (6a) can be safely dropped to form a more efficient DAG constraint with some $k \leqslant d$, where we often have $k \ll d$ in practice:

$$h^k_{\text{trunc}}(\tilde{\mathbf{B}}) = \operatorname{tr}\left(\sum_{i=1}^{k} \tilde{\mathbf{B}}^i\right). \tag{7}$$

Naively, the DAG constraint (7) requires $\mathcal{O}(k)$ matrix multiplication which is far less than $\mathcal{O}(d)$ for (6a), and under mild conditions the error of (7) is tightly bounded (see Proposition 3 for details). In

Figure 1 **right** we show that our DAG constraint (7) (named TMPI) attains far better performance than previous ones as it escapes from gradient vanishing.

3.2 Theoretical Properties of Truncated Geometric DAG Constraint

We analyze the properties of the power of adjacency matrices, which play an important role in constructing DAG constraints. First notice that, for the kth power of an adjacency matrix $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$, its entry $(\tilde{\mathbf{B}}^k)_{ij}$ is nonzero if and only if there is a length-k path (not necessarily simple) from node X_i to node X_j in graph $\mathcal{G}(\tilde{\mathbf{B}})$. This straightforwardly leads to the following property, which is a corollary of [44, Lemma 1].

Proposition 1. *Let $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$ be the weighted adjacency matrix of a graph $\mathcal{G}$ with d vertices. $\mathcal{G}$ is a DAG if and only if $\tilde{\mathbf{B}}^d = \mathbf{0}$.*

Proposition 1 can be used to construct a single-term DAG constraint term, given by

$$h_{\text{single}}(\tilde{\mathbf{B}}) = \|\tilde{\mathbf{B}}^d\|_p, \tag{8}$$

where $\|\cdot\|_p$ denotes an elementwise ℓ_p norm. However, as the candidate adjacency matrix gets closer to acyclic, the spectral radius of $\tilde{\mathbf{B}}^d$ will become very small. As a result, the value $h_{\text{single}}(\tilde{\mathbf{B}})$, as well as its gradient $\nabla_{\tilde{\mathbf{B}}} h_{\text{single}}(\tilde{\mathbf{B}})$ may become too small to be represented in limited machine precision. This suggests the use of the polynomial based constraints for better performance.

Without any prior information of the graph, one has to compute the dth power of matrix $\tilde{\mathbf{B}}$, either using the DAG constraint terms in Eq. (8) or (5). However, if the length of the longest simple path in the graph is known, we can formulate a lower-order polynomial based DAG constraint.

Proposition 2. *Let $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$ be the weighted adjacency matrix of a graph $\mathcal{G}$ with d vertices, and k be the length of the longest simple path*[1] *in graph $\mathcal{G}$. The following three conditions are equivalent: (1) $\mathcal{G}$ is a DAG, (2) $\tilde{\mathbf{B}}^{k+1} = 0$, and (3) $h_{trunc}^k(\tilde{\mathbf{B}}) = 0$.*

The intuition of the above proposition is as follows. For the graph whose largest simple cycle has a length of k, it suffices to compute the kth power of its adjacency matrix, since computing any power larger than k is simply traveling in the cycles repeatedly. Therefore, Proposition 2 suggests that $h_{\text{trunc}}^k(\tilde{\mathbf{B}}) = 0$ is a valid DAG constraint.

Unfortunately, the longest simple path problem for general graphs is known to be NP-hard [32], and thus we have to find an alternative way to seek the proper k. For a candidate adjacency matrix $\tilde{\mathbf{B}}$, a simple heuristic is to iterate over the power of matrices to find some k *s.t.* matrix $\tilde{\mathbf{B}}^k$ is close enough to a zero matrix. Recall that Proposition 1 implies that the adjacency matrix of a DAG is nilpotent; therefore, when a candidate adjacency matrix $\tilde{\mathbf{B}}$ is close to being acyclic, its spectral radius would be close to zero, which implies that the $\tilde{\mathbf{B}}^k$ would converge quickly to zero as k increases. Thus, if we use the above heuristic to find k, it is highly possible that we find some $k \ll d$.

By using an approximate k instead of the exact one, the DAG constraint in Eq. (7) becomes an approximate one with the following bounds. Denoting by $\|\cdot\|_\infty$ the elementwise infinity norm (*i.e.*, maximum norm), we have the following proposition.

Proposition 3. *Given $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$, if there exists $k < d$ such that $\|\tilde{\mathbf{B}}^k\|_\infty \leqslant \epsilon < \frac{1}{(k+1)d}$, then we have*

$$0 \leqslant h_{geo}(\tilde{\mathbf{B}}) - h_{trunc}^k(\tilde{\mathbf{B}}) \leqslant \frac{1-(d\epsilon)^{d/k-1}}{1-(d\epsilon)^{1/k}} d^{2+1/k}\epsilon^{1+1/k},$$

$$0 \leqslant \|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}}) - \nabla_{\tilde{\mathbf{B}}} h_{trunc}^k(\tilde{\mathbf{B}})\|_F \leqslant (k+1)d\epsilon \|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}})\|_F.$$

Remark 1. *The condition $\epsilon < \frac{1}{(k+1)d}$ in the proposition above is such that the inequalities are non-vacuous, specifically since we must have $\|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}}) - \nabla_{\tilde{\mathbf{B}}} h_{trunc}^k(\tilde{\mathbf{B}})\|_F / \|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}})\|_F < 1$.*

[1] We use simple path to refer to a path that repeats no vertex, except that the first and last may be the same vertex [17, p. 363].

Proposition 3 provides an efficient way to bound the error of truncated constraints $h^k_{\text{trunc}}(\tilde{\mathbf{B}})$ to the geometric series based DAG constraints $h_{\text{geo}}(\tilde{\mathbf{B}})$. However, it also demonstrates a possible limitation for all polynomial based DAG constraints. As we know, when a solution is close to DAG, its spectral radius will be close to zero. In this situation, if the longest simple path in a graph is long, it is unavoidable to use high-order terms to capture this information while such high-order terms might be close to 0. In the worst case, the higher-order terms may underflow due to limited machine precision. Thus some specific numerically stable method may be required, and we leave this for future work.

3.3 Dynamic Truncated Matrix Power Iteration

Proposition 3 indicates that it may be safe to discard those matrix powers with index larger than k. In this case, the DAG constraint and its gradient correspond to order-k and order-$(k-1)$ polynomials of $\tilde{\mathbf{B}}$, respectively. A straightforward way to compute the DAG constraint as well as the gradient is provided in Algorithm 1, which we call TMPI. The general polynomial based DAG constraint in (5) requires $\mathcal{O}(d)$ matrix multiplication, while Algorithm 1 reduces the number of multiplication to $\mathcal{O}(k)$. In practice, k can be much smaller than d; for example, when recovering 100-node Erdős–Rényi DAG with expected degree 4, k is often smaller than 20.

Algorithm 1 Truncated Matrix Power Iteration (TMPI)

Input: $\tilde{\mathbf{B}} \in \mathbb{R}^{d\times d}_{\geqslant 0}$, ϵ
Output: $h(\tilde{\mathbf{B}}), \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}})$

1: $i \leftarrow 2, h(\tilde{\mathbf{B}}) \leftarrow \operatorname{tr}(\tilde{\mathbf{B}}), \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}}) \leftarrow \mathbb{I}$
2: **while** $i \leqslant d$ **do**
3: $\quad \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}}) \leftarrow \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}}) + i(\tilde{\mathbf{B}}^{i-1})^\top$
4: $\quad \tilde{\mathbf{B}}^i \leftarrow \tilde{\mathbf{B}}\tilde{\mathbf{B}}^{i-1}$
5: $\quad h(\tilde{\mathbf{B}}) \leftarrow h(\tilde{\mathbf{B}}) + \operatorname{tr}(\tilde{\mathbf{B}}^i)$
6: $\quad$ **if** $\|\tilde{\mathbf{B}}^i\|_\infty \leqslant \epsilon$ **then break**
7: $\quad i \leftarrow i + 1$
8: **end while**
9: **Output** $h(\tilde{\mathbf{B}}), \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}})$

Algorithm 2 Fast TMPI

Input: $\tilde{\mathbf{B}} \in \mathbb{R}^{d\times d}_{\geqslant 0}$, ϵ
Output: $h(\tilde{\mathbf{B}}), \nabla_{\tilde{\mathbf{B}}} h(\tilde{\mathbf{B}})$

1: $i \leftarrow 1$
2: $\tilde{\mathbf{B}}_f \leftarrow \tilde{\mathbf{B}}, \tilde{\mathbf{B}}_g \leftarrow \mathbb{I}, \tilde{\mathbf{B}}_p \leftarrow \tilde{\mathbf{B}}$ {$\tilde{\mathbf{B}}_f$ stores $f_i(\tilde{\mathbf{B}})$, $\tilde{\mathbf{B}}_g$ stores $\nabla_{\tilde{\mathbf{B}}} \operatorname{tr}(f_i(\tilde{\mathbf{B}}))$, and $\tilde{\mathbf{B}}_p$ stores $\tilde{\mathbf{B}}^i$}
3: **while** $i \leqslant d$ **do**
4: $\quad \tilde{\mathbf{B}}_f^{old} \leftarrow \tilde{\mathbf{B}}_f, \tilde{\mathbf{B}}_g^{old} \leftarrow \tilde{\mathbf{B}}_g, \tilde{\mathbf{B}}_p^{old} \leftarrow \tilde{\mathbf{B}}_p$
5: $\quad \tilde{\mathbf{B}}_f \leftarrow \tilde{\mathbf{B}}_f^{old} + \tilde{\mathbf{B}}_p^{old}\tilde{\mathbf{B}}_f^{old}, \tilde{\mathbf{B}}_p \leftarrow \tilde{\mathbf{B}}_p^{old}\tilde{\mathbf{B}}_p^{old}$
6: $\quad \tilde{\mathbf{B}}_g \leftarrow \tilde{\mathbf{B}}_g^{old} + \tilde{\mathbf{B}}_p^\top \tilde{\mathbf{B}}_g^{old} + i(\tilde{\mathbf{B}}_f - \tilde{\mathbf{B}}_f^{old} + \tilde{\mathbf{B}}_p^{old} - \tilde{\mathbf{B}}_p)$
7: $\quad$ **if** $\|\tilde{\mathbf{B}}_p\|_\infty \leqslant \epsilon$ **then break**
8: $\quad i \leftarrow 2i$
9: **end while**
10: **Output** $\operatorname{tr}(\tilde{\mathbf{B}}_f), \tilde{\mathbf{B}}_g$

Fast Truncated Matrix Power Iteration We exploit the specific structure of geometric series to further accelerate the TMPI algorithm from $\mathcal{O}(k)$ to $\mathcal{O}(\log k)$ matrix multiplications. This is acheived by leveraging the following recurrence relations, with a proof provided in Appendix A.4.

Proposition 4. *Given any $d\times d$ real matrix $\tilde{\mathbf{B}}$, let $f_i(\tilde{\mathbf{B}}) = \tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^i$ and $h_i(\tilde{\mathbf{B}}) = \operatorname{tr}(f_i(\tilde{\mathbf{B}}))$. Then we have the following recurrence relations:*

$$
\begin{aligned}
f_{i+j}(\tilde{\mathbf{B}}) =& f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i f_j(\tilde{\mathbf{B}}),\\
\nabla_{\tilde{\mathbf{B}}} h_{i+j}(\tilde{\mathbf{B}}) =& \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_j(\tilde{\mathbf{B}}) + i f_j(\tilde{\mathbf{B}})^\top (\tilde{\mathbf{B}}^{i-1})^\top.
\end{aligned}
$$

This straightfowardly implies the following corollary, with a proof given in Appendix A.5.

Corollary 1. *Given any matrix $d \times d$ real matrix $\tilde{\mathbf{B}}$, let $f_i(\tilde{\mathbf{B}}) = \tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^i$ and $h_i(\tilde{\mathbf{B}}) = \operatorname{tr}(f_i(\tilde{\mathbf{B}}))$. Then we have have the following recurrence relations:*

$$
\begin{aligned}
f_{2i}(\tilde{\mathbf{B}}) =& (\mathbb{I} + \tilde{\mathbf{B}}^i) f_i(\tilde{\mathbf{B}}),\\
\nabla_{\tilde{\mathbf{B}}} h_{2i}(\tilde{\mathbf{B}}) =& \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + i\left(f_{2i}(\tilde{\mathbf{B}}) - f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i - \tilde{\mathbf{B}}^{2i}\right)^\top.
\end{aligned}
$$

Instead of increasing the index i (*i.e.*, power of $\tilde{\mathbf{B}}$) arithmetically by 1, as in line 9 of Algorithm 1, Corollary 1 suggests using its recurrence relations to increase the index i geometrically by a factor of 2, until the condition $\|\tilde{\mathbf{B}}^i\|_\infty \leqslant \epsilon$ is satisfied. The full procedure named Fast TMPI is described in Algorithm 2, which requires $\mathcal{O}(\log k)$ matrix multiplications and $\mathcal{O}(d^2)$ additional storage. An

example implementation is given in Listing 1 in Appendix B.1. Similar strategy can be applied to the original geometric series based constraint (6a) to form a fast algorithm that requires $\mathcal{O}(\log d)$ matrix multiplications. A comparison of the time complexity between our proposed constraint and the existing ones is provided in Appendix B.2.

The Full Optimization Framework For NOTEARS, we apply the same augmented Lagrangian framework [4, 5] adopted by Zheng et al. [50] to convert the constrained optimization problem into the following unconstrained problem

$$\min_{\mathbf{B}\in\mathbb{R}^{d\times d}} \ \|\,\mathbf{X}-\mathbf{X}\,\mathbf{B}\,\|_F^2+\eta\|\,\mathbf{B}\,\|_1+\frac{\rho}{2}h^k_{\text{trunc}}(\mathbf{B}\odot\mathbf{B})^2+\alpha h^k_{\text{trunc}}(\mathbf{B}\odot\mathbf{B}), \tag{9}$$

where the parameters ρ and α are iteratively updated using the same strategy as Zheng et al. [50]. We also apply the proposed DAG constraint to GOLEM [23] that adopts likelihood-based objective with soft constraints, leading to the unconstrained optimization problem

$$\min_{\mathbf{B}\in\mathbb{R}^{d\times d}} \ \frac{d}{2}\log\|\,\mathbf{X}-\mathbf{X}\,\mathbf{B}\,\|_F^2-\log|\det(\mathbb{I}-\mathbf{B})|+\eta_1\|\,\mathbf{B}\,\|_1+\eta_2 h^k_{\text{trunc}}(\mathbf{B}\odot\mathbf{B}), \tag{10}$$

where $\eta_1,\eta_2>0$ are the regularization coefficients. In both formulations above, we use Algorithm 1 or 2 to dynamically compute $h^k_{\text{trunc}}(\mathbf{B}\odot\mathbf{B})$ (and the value of k) during each optimization iteration based on a predefined tolerance ϵ.

3.4 Relation to Previous Works

Our algorithm is closely related to the following DAG constraint considered by Zheng et al. [50].

Lemma 1 (Zheng et al. [50, Proposition 1])**.** *Let $\tilde{\mathbf{B}}\in\mathbb{R}^{d\times d}_{\geqslant 0}$ with spectral radius smaller than one be the weighted adjacency matrix of a graph $\mathcal{G}$ with d vertices. $\mathcal{G}$ is a DAG if and only if*

$$\operatorname{tr}(\mathbb{I}-\tilde{\mathbf{B}})^{-1}=d. \tag{11}$$

When the spectral radius is smaller than one, the inverse of $\mathbb{I}-\tilde{\mathbf{B}}$ is simply the following infinite geometric series:

$$(\mathbb{I}-\tilde{\mathbf{B}})^{-1}=\sum_{i=0}^{\infty}\tilde{\mathbf{B}}^i. \tag{12}$$

Zheng et al. [50] argued that (12) is not well defined for $\tilde{\mathbf{B}}$ with spectral radius larger than 1, and in practice the series (12) may explode very quickly. Based on this argument, they adopted the matrix exponential for constructing the DAG constraint. For the same reason, Yu et al. [45] also used very small coefficients for higher-order terms in polynomial based DAG constraints. However, as we illustrate in Section 3, the DAG constraints proposed by Yu et al. [45], Zheng et al. [50] may suffer from gradient vanishing owing to the small coefficients for higher-order terms. In fact, our constraint (7) serves a good approximation for (11) since we are actually looking for an approximate inverse of $\mathbb{I}-\tilde{\mathbf{B}}$ with a bounded error.

Proposition 5. *Given $\tilde{\mathbf{B}}\in\mathbb{R}^{d\times d}_{\geqslant 0}$, if there exists k such that $\|\tilde{\mathbf{B}}^k\|_\infty\leqslant\epsilon<1/d$, then we have*

$$\left\|(\mathbb{I}-\tilde{\mathbf{B}})^{-1}-\sum_{i=0}^{k}\tilde{\mathbf{B}}^k\right\|_F\leqslant d\epsilon\left\|(\mathbb{I}-\tilde{\mathbf{B}})^{-1}\right\|_F.$$

Previous works [24, 44] pointed out that for DAG constraints with form $h(\mathbf{B}\odot\mathbf{B})$, the Hadamard product would lead to vanishing gradients in the sense that the gradient $\nabla_{\mathbf{B}}h(\mathbf{B}\odot\mathbf{B})=0$ if and only if $h(\mathbf{B}\odot\mathbf{B})=0$. Ng et al. [24] further showed that this may lead to ill-conditioning issue when solving the constrained optimization problem. One may replace the Hadamard product with absolute value, as in Wei et al. [44], but the non-smoothness of absolute value may make the optimization problem difficult to solve and lead to poorer performance [24]. As pointed out by Ng et al. [24], Quasi-Newtown methods, *e.g.*, L-BFGS [7], still perform well even when faced with the ill-conditioning issue caused by vanishing gradients of constraints with form $h(\mathbf{B}\odot\mathbf{B})$. In this work, we identify another type of gradient vanishing issue caused by (1) the higher-order terms and small coefficients in function h itself and (2) the small spectral radius of candidate matrix $\mathbf{B}$, and further provide an effective solution by constructing a Truncated Matrix Power Iteration based DAG constraint.

4 Experiments

We compare the performance of different DAG learning algorithms and DAG constraints to demonstrate the effectiveness of the proposed TMPI based DAG constraint. We provide the main experimental results in this section, and further empirical studies can be found in Appendix C.[2]

4.1 DAG Learning

We first conduct experiments on synthetic datasets using similar settings as previous works [23, 46, 50]. In summary, random Erdös-Rényi graphs [14] are generated with d nodes and kd expected edges (denoted by ERk). Edge weights generated from uniform distribution over the union of two intervals $[-2, -0.5] \cup [0.5, 2.0]$ are assigned to each edge to form a weighted adjacency matrix $\mathbf{B}$. Then n samples are generated from the linear SEM $\mathbf{x} = \mathbf{B}\,\mathbf{x} + \mathbf{e}$ to form an $n \times d$ data matrix $\mathbf{X}$, where the noise $\mathbf{e}$ are i.i.d. sampled from Gaussian, Exponential, or Gumbel distribution. We set the sample size $n = 1000$ and consider 4 different number of nodes $d = 10, 30, 50, 100$ unless otherwise stated. For each setting, we conduct 100 random simulations to obtain an average performance.

Methods and Baselines We consider continuous DAG learning methods based on both Mean Squared Error (MSE) loss (see Eq. (9)) and likelihood-based (see Eq. (10)) loss from GOLEM [23]. For both losses, we include the following DAG constraints: (1) MSE/Likelihood loss+Exponential DAG constraint (3), *i.e.* NOTEARS [50]/GOLEM[23] ; (2) MSE/Likelihood loss+Binomial DAG constraint (4) from DAG-GNN [45]; (3) MSE/Likelihood loss+Spectral radius based DAG constraint from NOBEARS [21]; (4) MSE/Likelihood loss+Single Term DAG constraint (8); (5) MSE/Likelihood loss+Truncated Matrix Power Iteration (TMPI) based DAG constraint (7). For the NOCURL method [46], since it is a two stage methods where the initialization stage rely on MSE loss and Binomial DAG constraint, we include it as a baseline, but do not combine it with likelihood loss. The other baselines include GES [9] and MMHC [41, 42][3].

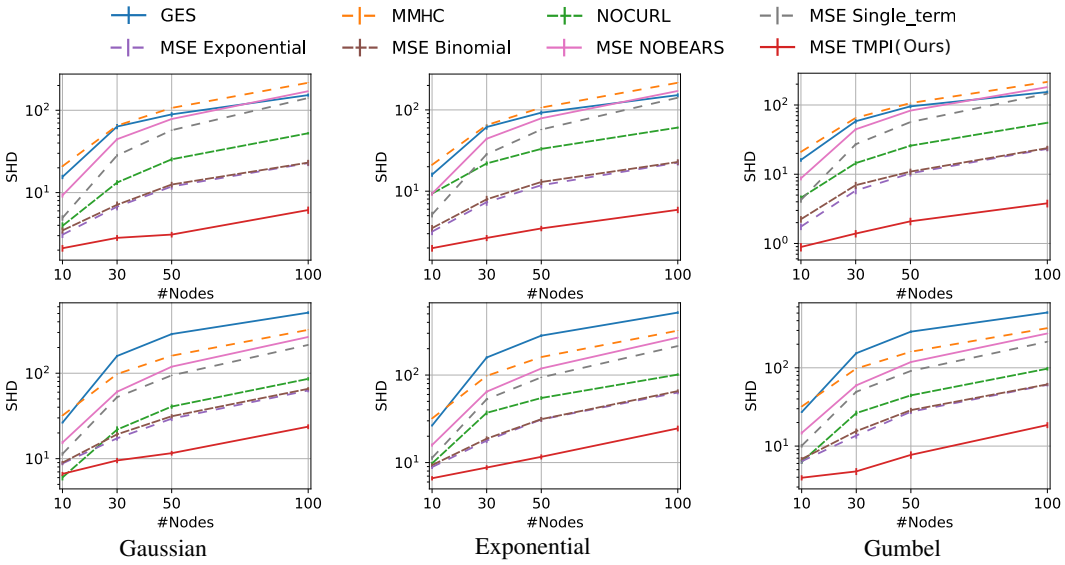


Figure 2: DAG learning results of MSE loss (*e.g.* Eq. (9)) based algorithms and baselines in terms of SHD (lower is better) on ER2 (**top**) and ER3 (**bottom**) graphs, where our algorithm consistently outperforms others almost by an order of magnitude. Error bars represent standard errors over 100 simulations.

For MSE loss based DAG learning methods (9), we set $\eta = 0.1$ and use the same strategy as Zheng et al. [50] to update the parameters ρ and α. For likelihood loss based DAG learning methods (10), we set $\eta_1 = 0.02$ and $\eta_2 = 5.0$ as Ng et al. [23]. For the Binomial DAG constraints (4), the parameter α is set to $1/d$ as Yu et al. [45]. For our TMPI DAG constraint, we set the parameter $\epsilon = 10^{-6}$. For the NOCURL method, we use their public available implementation and default

[2]The source code is avaiable at here.

[3]The implementation is provided here.

parameter setting[4]. For GES and MMPC, we use the implementation and default parameter setting in the `CausalDiscoveryToolbox` package [16].

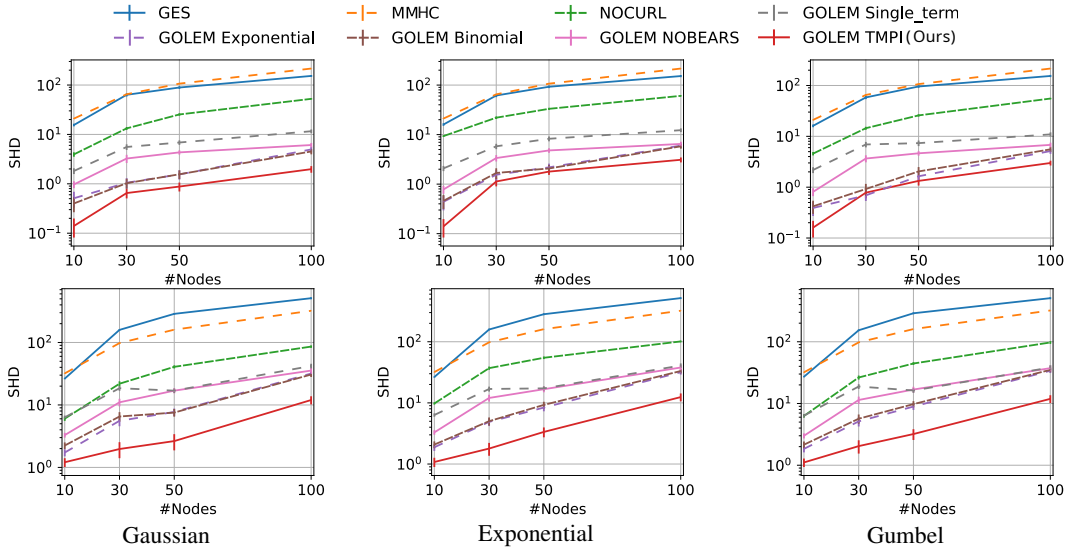


Figure 3: DAG learning results of likelihood loss (*i.e.* Eq. (10)) based algorithms and baselines in terms of SHD (lower is better) on ER2 (**top**) and ER3 (**bottom**) graphs, where our algorithm consistently outperforms others in a large margin in almost all cases. Error bars represent standard errors over 100 simulations.

Results The results for MSE/likelihood based DAG learning algorithm are shown in Figure 2/3, where we compare the Structural Hamming Distance (SHD) of different algorithms. The complete results for other metrics are reported in Appendix C.1. The original NOTEARS (*i.e.* MSE+Exponential) / GOLEM (*i.e.* GOLEM+Exponential) has similar performance as MSE/GOLEM+Binomial. This may not be surprising because the coefficients of each order term in the Exponential and Binomial constraints are actually very close (see Figure 1 for an example). The performance of algorithms using single-term DAG constraint is often not comparable to the other polynomial-based constraints since it suffers more from gradient vanishing. The performance of algorithms using the spectral radius based constraints (BEARS) are also worse than the ones using Exponential and Binomial constraints. The algorithms using our TMPI constraints outperforms all others in almost all cases, which demonstrate the effectiveness of our DAG constraint.

Additional Experiments We conduct additional experiments with scale-free graphs, smaller sample size (*i.e.*, $n = 200$ samples), larger graphs (*i.e.*, 500-node graphs), and denser graphs (*i.e.*, ER6 graphs). Due to the space limit, the results of these experiments are provided in Appendices C.2, C.3, C.4, and C.6, respectively, which further demonstrate the effectiveness of our proposed DAG constraint. We also compare our method with NOFEARS [44] in Appendix C.5.

4.2 Extension to Nonlinear Case

Although the description of our algorithm in Section 3 focuses on the linear case, it can be straightforwardly extended to the nonlinear case. In this section, we extend our DAG constraint to nonlinear case by replacing the binomial DAG constraint in DAG-GNN [45], NOTEARS-MLP [51] and GRAN-DAG [20] with our TMPI constraint.

Synthetic Data We compare our algorithm and DAG-GNN on 50-node ER1 graph with nonlinear SEM $\mathbf{x} = \mathbf{B}\cos(\mathbf{x}+1) + \mathbf{e}$ over 5 simulations. Our algorithm achieves SHD of 22.2 ± 4.2 while DAG-GNN achieves SHD of 25.2 ± 4.5. We simulate a nonlinear dataset with 50-node ER1 graphs and each function being an MLP as Zheng et al. [51]. The original NOTEARS-MLP has SHD 16.9 ± 1.5, while NOTEARS-MLP with our TMPI constraint achieves an SHD of 14.9 ± 1.3.

[4] `https://github.com/fishmoon1234/DAG-NoCurl/blob/master/main_efficient.py` from commit 4e2a1f4 on 14 Jun 2021. Better performance may be attained by tuning the hyper-parameters, but the current best reported performance is slightly worse than NOTEARS in most settings.

Real Data We also run an experiment on the Protein Signaling Networks dataset [31] consisting of $n = 7466$ samples with $d = 11$ nodes. The dataset is widely used to evaluate the performance of DAG learning algorithms; however, since the identifiability of the DAG is not guaranteed, we compare both SHD and SHDC (SHD of CPDAG), following Lachapelle et al. [20]. In this experiment, we replace the original DAG constraints in DAG-GNN [45] and Gran-DAG [20] with our TMPI DAG constraints. The results are shown in Table 1, showing that by replacing the DAG constraint with our TMPI constraint, there is a performance improvement on SHD and SHDC for almost all cases.

4.3 Ablation Study

We use the ER3 dataset in previous section to conduct an ablation study to compare the performance of the Geometric constraint (6a) and our TMPI constraint (7), with both the naive implementation (*i.e.*, linear complexity) and fast implementation (*i.e.*, logarithmic complexity) from Algorithms 1 and 2, respectively. The results are shown in Figure 4. The SHD between Geometric constraint and TMPI constraint in both the naive and fast implementations are similar, and our Fast TMPI algorithm runs significantly faster than the other three, especially when the graph size is large. We also conduct further empirical study to illustrate the numeric difference between different k for our DAG constraint; due to limited space, a detailed discussion is provided in Appendix C.7.

Table 1: Experimental results on Sachs dataset. Our TMPI DAG constraint consistently outperforms other DAG constraints.

	SHD	SHDC
DAG-GNN	16	21
DAG-GNN/TMPI (Ours)	16	17
Gran-DAG	13	11
Gran-DAG/TMPI (Ours)	**12**	**9**
Gran-DAG++	13	10
Gran-DAG++/TMPI (Ours)	**12**	**9**

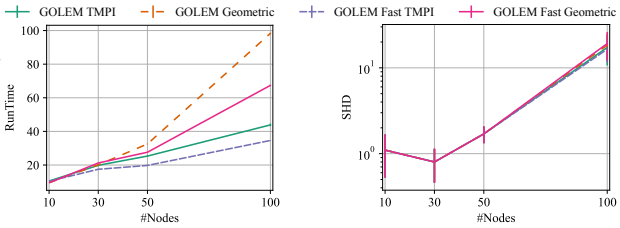


Figure 4: Comparison of Geometric and TMPI constraints, all algorithm achieves similar performance but our fast algorithm runs significantly faster. **Left:** Running time; **Right:** SHD. Error bars represent standard errors over 100 simulations.

5 Conclusion and Future Work

While polynomial constraints-based DAG learning algorithms achieve promising performance, they often suffer from gradient vanishing in practice. We identified one important source of gradient vanishing: improper small coefficients for higher-order terms, and proposed to use the geometric series based DAG constraints to escape from gradient vanishing. We further proposed an efficient algorithm that can evaluate the constraint with bounded error in $\mathcal{O}(\log k)$ complexity, where k is often far smaller than the number of nodes. By replacing the previous DAG constraints with ours, the performance of DAG learning can often be significantly improved in various settings.

A clear next step would be finding strategies to adaptively adjust the coefficients for higher-order terms in polynomial based constraints. In a nutshell, polynomial-based constraints may explode on candidate graphs that are far from acyclic, and also suffer from gradient vanishing problem when the graph is nearly acyclic. Fixed coefficients are not able to solve both problems at the same time. Another possible direction is to use the sparse properties in DAGs to make DAG learning more efficient. Since the adjacency matrix of DAG must be nilpotent, there might exist more compact representation of the adjacency matrix, such as a low-rank CANDECOMP/PARAFAC (CP) decomposition, which can potentially be used to derive an efficient algorithm for very large graphs.

Acknowledgements

We thank the anonymous reviewers, especially Reviewer iGwL, for their helpful comments. ZZ, DG, YL, EA and JQS were partially supported by Centre for Augmented Reasoning at the Australian Institute for Machine Learning. IN and KZ were partially supported by the National Institutes of Health (NIH) under Contract R01HL159805, by the NSF-Convergence Accelerator Track-D award #2134901, by a grant from Apple Inc., and by a grant from KDDI Research Inc.. MG was supported by ARC DE210101624.

References

[1] A. Al-Mohy and N. Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31, 2009.

[2] P. Bader, S. Blanes, and F. Casas. Computing the matrix exponential with an optimized taylor polynomial approximation. *Mathematics*, 7(12):1174, 2019.

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

[4] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[6] R. Bhattacharya, T. Nagarajan, D. Malinsky, and I. Shpitser. Differentiable causal discovery under unmeasured confounding. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[7] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[8] D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer, 1996.

[9] D. M. Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[10] M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5, 2004.

[11] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

[12] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. ISSN 0747–7171. Computational algebraic complexity editorial.

[13] J. Cussens. Bayesian network learning with cutting planes. In *Conference on Uncertainty in Artificial Intelligence*, 2011.

[14] P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

[15] R. Fletcher. *Practical Methods of Optimization*. Wiley-Interscience, 1987.

[16] D. Kalainathan and O. Goudet. Causal discovery toolbox: Uncover causal relationships in python. *arXiv preprint arXiv:1903.02278*, 2019.

[17] D. E. Knuth. *The Art of Computer Programming, Volume 1: (3rd Ed.) Sorting and Searching*. 1997.

[18] M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5(Dec):549–573, 2004.

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

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

[21] H.-C. Lee, M. Danieletto, R. Miotto, S. T. Cherng, and J. T. Dudley. Scaling structural learning with no-bears to infer causal transcriptome networks. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2020*, pages 391–402. World Scientific, 2019.

[22] P. L. Loh and P. Bühlmann. High-dimensional learning of linear causal networks via inverse covariance estimation. *Journal of Machine Learning Research*, 2013.

[23] I. Ng, A. Ghassami, and K. Zhang. On the role of sparsity and DAG constraints for learning linear DAGs. *Advances in Neural Information Processing Systems*, 33, 2020.

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

[25] I. Ng, S. Zhu, Z. Fang, H. Li, Z. Chen, and J. Wang. Masked gradient-based causal structure learning. In *SIAM International Conference on Data Mining*, 2022.

[26] R. Pamfil, N. Sriwattanaworachai, S. Desai, P. Pilgerstorfer, P. Beaumont, K. Georgatzis, and B. Aragam. DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[27] J. Pearl. Models, reasoning and inference. *Cambridge, UK: Cambridge University Press*, 19, 2000.

[28] J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2013.

[29] M. J. D. Powell. Nonlinear programming—sequential unconstrained minimization techniques. *The Computer Journal*, 12(3), 1969.

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

[31] 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.

[32] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[33] R. D. Shah and J. Peters. The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3):1514–1538, 2020.

[34] 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(Oct):2003–2030, 2006.

[35] T. Silander and P. Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Conference on Uncertainty in Artificial Intelligence*, 2006.

[36] A. P. Singh and A. W. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

[37] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

[38] P. Spirtes, C. Meek, and T. Richardson. Causal inference in the presence of latent variables and selection bias. In *Conference on Uncertainty in Artificial Intelligence*, 1995.

[39] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[40] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[41] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, 2003.

[42] 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.

[43] M. J. Vowels, N. C. Camgoz, and R. Bowden. D'ya like dags? a survey on structure learning and causal discovery. *arXiv preprint arXiv:2103.02582*, 2021.

[44] D. Wei, T. Gao, and Y. Yu. DAGs with no fears: A closer look at continuous optimization for learning bayesian networks. *arXiv preprint arXiv:2010.09133*, 2020.

[45] Y. Yu, J. Chen, T. Gao, and M. Yu. DAG-GNN: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[46] Y. Yu, T. Gao, N. Yin, and Q. Ji. DAGs with no curl: An efficient dag structure learning approach. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[47] C. Yuan and B. Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48(1):23–65, 2013.

[48] C. Yuan, B. Malone, and X. Wu. Learning optimal Bayesian networks using A* search. In *International Joint Conference on Artificial Intelligence*, 2011.

[49] 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.

[50] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. DAGs with No Tears: Continuous optimization for structure learning. *arXiv preprint arXiv:1803.01422*, 2018.

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

[52] R. Zhu, A. Pfadler, Z. Wu, Y. Han, X. Yang, F. Ye, Z. Qian, J. Zhou, and B. Cui. Efficient and scalable structure learning for bayesian networks: Algorithms and applications. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2613–2624. IEEE, 2021.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendices
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendices
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendices
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendices
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We use publicly available datasets
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendices

A Proofs

A.1 Proof of Proposition 1

Proposition 1. *Let $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d \times d}$ be the weighted adjacency matrix of a graph $\mathcal{G}$ with d vertices. $\mathcal{G}$ is a DAG if and only if $\tilde{\mathbf{B}}^d = \mathbf{0}$.*

Proof. Any path in a d-node DAG cannot be longer than $d-1$, and thus we must have $(\tilde{\mathbf{B}}^d)_{ij} = 0$ for all i, j, which indicates for DAG we must have $\tilde{\mathbf{B}}^d = \mathbf{0}$. On the other side, a nilpotent adjacency matrix must form a DAG [44, Lemma 1]. □

A.2 Proof of Proposition 2

Proposition 2. *Let $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d \times d}$ be the weighted adjacency matrix of a graph $\mathcal{G}$ with d vertices, let k be the length of longest simple path in $\mathcal{G}$, the following 3 conditions are equivalent: (1) $\mathcal{G}$ is a DAG, (2) $\tilde{\mathbf{B}}^{k+1} = 0$, and (3) $h_{trunc}^k(\tilde{\mathbf{B}}) = 0$.*

Proof. First notice that $(\tilde{\mathbf{B}}^k)_{j\ell} > 0$ if and only if there is at least a directed walk with length k from node j to node ℓ in $\mathcal{G}$. We consider the following cases:

- (1) $\Longrightarrow$ (2): Since $\mathcal{G}$ is a DAG, the diagonal entries of $\tilde{\mathbf{B}}^{k+1}$ are zero. Therefore, it suffices to consider its non-diagonal entries. For a DAG, any directed path in a DAG must be a simple path without loop. If it is not, then we can find a directed path, where a node appears twice, which means that there is a loop in the graph and it is in contradiction with the acyclic property. Thus if $\mathcal{G}$ is a DAG, there will be no path with length larger than k, and so the non-diagonal entries of $\tilde{\mathbf{B}}^{k+1}$ are zero. Combining both cases, we have $\tilde{\mathbf{B}}^{k+1} = 0$.
- (1) $\Longrightarrow$ (3): Since $\mathcal{G}$ is a DAG, the diagonal entries of $\tilde{\mathbf{B}}^i, i = 1, \ldots, d$ are zeros. Therefore, we must have $\operatorname{tr}\left(\sum_{i=1}^k \tilde{\mathbf{B}}^i\right) = 0$.
- (2) $\Longrightarrow$ (1): Since $\tilde{\mathbf{B}}^{k+1} = 0$, $\tilde{\mathbf{B}}$ is nilpotent and $\mathcal{G}$ must be acyclic.
- (3) $\Longrightarrow$ (1): It is clear that a directed graph is a DAG if and only if it does not contain simple cycle. Since $\operatorname{tr}\left(\sum_{i=1}^k \tilde{\mathbf{B}}^i\right) = 0$, all path with length smaller than or equal to k must not be a (simple) cycle. If there exists any path with length larger than k that forms a simple cycle, then there exists a simple path that is longer than k, which contradicts our assumption that k is the longest simple path. Combining both cases, $\mathcal{G}$ does not contain simple cycle and thus must be acyclic.

Here we proved that (1) and (2) are equivalent; (1) and (3) are equivalent. Thus the three conditions are equivalent under the assumption of the proposition. □

A.3 Proof of Proposition 3

We first describe a property of the polynomials of matrices in Lemma 2, which will be used to prove Proposition 3.

Lemma 2. *Given $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$, assume that we have two series $[\alpha_0, \alpha_1, \ldots, \alpha_k] \in \mathbb{R}_{\geqslant 0}^{k+1}$ and $[\beta_0, \beta_1, \ldots, \beta_k] \in \mathbb{R}_{\geqslant 0}^{k+1}$. If we have $\alpha_i \geqslant \beta_i$ for $i = 0, 1, \ldots k$, then the following inequalities hold:*

$$\left\| \sum_{i=0}^{k} \alpha_i \tilde{\mathbf{B}}^i \right\|_F \geqslant \left\| \sum_{i=0}^{k} \beta_i \tilde{\mathbf{B}}^i \right\|_F, \tag{13a}$$

$$\left\| \sum_{i=0}^{k} \alpha_i \tilde{\mathbf{B}}^i \right\|_\infty \geqslant \left\| \sum_{i=0}^{k} \beta_i \tilde{\mathbf{B}}^i \right\|_\infty. \tag{13b}$$

Proof. By the assumption that $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$, we have $\tilde{\mathbf{B}}^i \in \mathbb{R}_{\geqslant 0}^{d\times d}$ for $i = 0, 1, \ldots, k$. Then, using the assumption that $\alpha_i \geqslant \beta_i$ for $i = 0, 1, \ldots k$, straightly we can see that each entry of $\sum_{i=0}^{k} \alpha_i \tilde{\mathbf{B}}^i$ should be no less than the corresponding entry of $\sum_{i=0}^{k} \beta_i \tilde{\mathbf{B}}^i$. Thus (13) must hold. □

With the lemma above, we now present the proof of Proposition 3.

Proposition 3. *Given $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d\times d}$, if there exists $k < d$ such that $\|\tilde{\mathbf{B}}^k\|_\infty \leqslant \epsilon < \frac{1}{(k+1)d}$, then we have*

$$0 \leqslant h_{geo}(\tilde{\mathbf{B}}) - h_{trunc}^k(\tilde{\mathbf{B}}) \leqslant \frac{1-(d\epsilon)^{d/k-1}}{1-(d\epsilon)^{1/k}} d^{2+1/k}\epsilon^{1+1/k},$$
$$0 \leqslant \|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}}) - \nabla_{\tilde{\mathbf{B}}} h_{trunc}^k(\tilde{\mathbf{B}})\|_F \leqslant (k+1)d\epsilon \|\nabla_{\tilde{\mathbf{B}}} h_{geo}(\tilde{\mathbf{B}})\|_F.$$

Proof. Let $[\lambda_1, \lambda_2, \ldots, \lambda_d]$ be the d eigen values of $\tilde{\mathbf{B}}$. We have

$$\max_i |\lambda_i^k| \leqslant \|\tilde{\mathbf{B}}^k\|_F \leqslant d\|\tilde{\mathbf{B}}^k\|_\infty \leqslant d\epsilon, \tag{14}$$

which indicates that

$$\max_i |\lambda_i| \leqslant (d\epsilon)^{1/k}. \tag{15}$$

As all entries of $\tilde{\mathbf{B}}$ are non-negative, for all $j > 1$ we have

$$\mathrm{tr}(\tilde{\mathbf{B}}^{k+j}) = |\mathrm{tr}(\tilde{\mathbf{B}}^{k+j})| = \left| \sum_{i=1}^{d} \lambda_i^{k+j} \right| \leqslant \sum_{i=1}^{d} |\lambda_i^{k+j}| \leqslant d \max_i |\lambda_i^{k+j}| = d(\max_i |\lambda_i|)^{k+j} \leqslant d^{2+j/k}\epsilon^{1+j/k},$$

which implies that

$$h_{\text{geo}}(\tilde{\mathbf{B}}) - h_{\text{trunc}}^k(\tilde{\mathbf{B}}) = \sum_{j=1}^{d-k} \mathrm{tr}(\tilde{\mathbf{B}}^{k+j}) \leqslant \sum_{j=1}^{d-k} d^{2+j/k}\epsilon^{1+j/k} = \frac{1-(d\epsilon)^{d/k-1}}{1-(d\epsilon)^{1/k}} d^{2+1/k}\epsilon^{1+1/k}.$$

This finishes the proof of the first part of the proposition. For the second part, we have

$$
\begin{aligned}
&\|\nabla_{\tilde{\mathbf{B}}}[h_{\text{geo}}(\tilde{\mathbf{B}}) - h^k_{\text{trunc}}(\tilde{\mathbf{B}})]\|_F \\
=&\underbrace{\left\|\sum_{j=1}^{d-k}(k+j)\,\mathbf{B}^{k+j-1}\right\|_F \leqslant \left\|\sum_{j=1}^{d-k}(kj+j)\,\mathbf{B}^{k+j-1}\right\|_F}_{kj+j\geqslant k+j \text{ for } j=1,2,\ldots d-k, \text{ and then by Lemma 2}} \\
=&\underbrace{(k+1)\left\|\tilde{\mathbf{B}}^k\sum_{j=1}^{d-k}j\tilde{\mathbf{B}}^{j-1}\right\|_F \leqslant (k+1)\|\tilde{\mathbf{B}}^k\|_F\left\|\sum_{j=1}^{d-k}j\tilde{\mathbf{B}}^{j-1}\right\|_F}_{\text{Submultiplicativity of Frobenius norm}} \\
=&\underbrace{(k+1)\|\tilde{\mathbf{B}}^k\|_F\left\|\sum_{j=1}^{d-k}j\tilde{\mathbf{B}}^{j-1}+\sum_{j=d-k+1}^{d}0\tilde{\mathbf{B}}^{j-1}\right\|_F \leqslant (k+1)\|\tilde{\mathbf{B}}^k\|_F\left\|\sum_{j=1}^{d}j\tilde{\mathbf{B}}^{j-1}\right\|_F}_{\text{By Lemma 2}} \\
\leqslant&\underbrace{(k+1)d\epsilon\left\|\sum_{j=1}^{d}j\tilde{\mathbf{B}}^{j-1}\right\|_F}_{\text{By Eq. (14)}} \\
=&(k+1)d\epsilon\|\nabla_{\tilde{\mathbf{B}}}h_{\text{geo}}(\tilde{\mathbf{B}})\|_F.
\end{aligned}
$$

□

A.4 Proof of Proposition 4

Proposition 4. *Given any $d\times d$ real matrix $\tilde{\mathbf{B}}$, let $f_i(\tilde{\mathbf{B}}) = \tilde{\mathbf{B}}+\tilde{\mathbf{B}}^2+\cdots+\tilde{\mathbf{B}}^i$ and $h_i(\tilde{\mathbf{B}}) = \mathrm{tr}(f_i(\tilde{\mathbf{B}}))$. Then we have the following recurrence relations:*

$$
\begin{aligned}
f_{i+j}(\tilde{\mathbf{B}}) =& f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i f_j(\tilde{\mathbf{B}}), \\
\nabla_{\tilde{\mathbf{B}}}h_{i+j}(\tilde{\mathbf{B}}) =& \nabla_{\tilde{\mathbf{B}}}h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top\nabla_{\tilde{\mathbf{B}}}h_j(\tilde{\mathbf{B}}) + if_j(\tilde{\mathbf{B}})^\top(\tilde{\mathbf{B}}^{i-1})^\top.
\end{aligned}
$$

Proof. Substituting $i+j$ for i in the definition of $f_i(\tilde{\mathbf{B}})$, we have

$$
\begin{aligned}
f_{i+j}(\tilde{\mathbf{B}}) &= \tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^i + \tilde{\mathbf{B}}^{i+1} + \tilde{\mathbf{B}}^{i+2} + \cdots + \tilde{\mathbf{B}}^{i+j} \\
&= (\tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^i) + (\tilde{\mathbf{B}}^{i+1} + \tilde{\mathbf{B}}^{i+1} + \tilde{\mathbf{B}}^{i+2} + \cdots + \tilde{\mathbf{B}}^{i+j}) \\
&= f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i(\tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^j) \\
&= f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i f_j(\tilde{\mathbf{B}}).
\end{aligned}
$$

Similarly, to establish the recurrence relation for $\nabla_{\tilde{\mathbf{B}}}h_{i+j}(\tilde{\mathbf{B}})$, we have

$$
\begin{aligned}
\nabla_{\tilde{\mathbf{B}}}h_{i+j}(\tilde{\mathbf{B}}) &= \nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(f_{i+j}(\tilde{\mathbf{B}})) \\
&= \nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(f_i(\tilde{\mathbf{B}})) + \nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(\tilde{\mathbf{B}}^i f_j(\tilde{\mathbf{B}})) \\
&= \nabla_{\tilde{\mathbf{B}}}h_i(\tilde{\mathbf{B}}) + \nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(\tilde{\mathbf{B}}^i f_j(\tilde{\mathbf{B}})) \\
&= \nabla_{\tilde{\mathbf{B}}}h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top\nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(f_j(\tilde{\mathbf{B}})) + f_j(\tilde{\mathbf{B}})^\top\nabla_{\tilde{\mathbf{B}}}\mathrm{tr}(\tilde{\mathbf{B}}^i) \\
&= \nabla_{\tilde{\mathbf{B}}}h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top\nabla_{\tilde{\mathbf{B}}}h_j(\tilde{\mathbf{B}}) + if_j(\tilde{\mathbf{B}})^\top(\tilde{\mathbf{B}}^{i-1})^\top
\end{aligned}
$$

where the fourth equality follows from the product rule. □

A.5 Proof of Corollary 1

Corollary 1. *Given any matrix $d \times d$ real matrix $\tilde{\mathbf{B}}$, let $f_i(\tilde{\mathbf{B}}) = \tilde{\mathbf{B}} + \tilde{\mathbf{B}}^2 + \cdots + \tilde{\mathbf{B}}^i$ and $h_i(\tilde{\mathbf{B}}) = \mathrm{tr}(f_i(\tilde{\mathbf{B}}))$. Then we have have the following recurrence relations:*

$$\begin{aligned} f_{2i}(\tilde{\mathbf{B}}) =& (\mathbb{I} + \tilde{\mathbf{B}}^i) f_i(\tilde{\mathbf{B}}), \\ \nabla_{\tilde{\mathbf{B}}} h_{2i}(\tilde{\mathbf{B}}) =& \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + i \left(f_{2i}(\tilde{\mathbf{B}}) - f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i - \tilde{\mathbf{B}}^{2i} \right)^\top . \end{aligned}$$

Proof. The first equality is straightforwardly obtained by substituting $i = j$ in the first equality of Proposition 4. Similarly, for the second equality, substituting $i = j$ in the second equality of Proposition 4 yields

$$\begin{aligned} \nabla_{\tilde{\mathbf{B}}} h_{2i}(\tilde{\mathbf{B}}) &= \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + i f_i(\tilde{\mathbf{B}})^\top (\tilde{\mathbf{B}}^{i-1})^\top \\ &= \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + i \left(\tilde{\mathbf{B}}^i + \tilde{\mathbf{B}}^{i+1} + \cdots + \tilde{\mathbf{B}}^{2i-1} \right)^\top \\ &= \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + (\tilde{\mathbf{B}}^i)^\top \nabla_{\tilde{\mathbf{B}}} h_i(\tilde{\mathbf{B}}) + i \left(f_{2i}(\tilde{\mathbf{B}}) - f_i(\tilde{\mathbf{B}}) + \tilde{\mathbf{B}}^i - \tilde{\mathbf{B}}^{2i} \right)^\top . \end{aligned}$$

□

A.6 Proof of Proposition 5

Proposition 5. *Given $\tilde{\mathbf{B}} \in \mathbb{R}_{\geqslant 0}^{d \times d}$, if there exists k such that $\|\tilde{\mathbf{B}}^k\|_\infty \leqslant \epsilon < 1/d$, then we have*

$$\left\| (\mathbb{I} - \tilde{\mathbf{B}})^{-1} - \sum_{i=0}^{k} \tilde{\mathbf{B}}^k \right\|_F \leqslant d\epsilon \left\| (\mathbb{I} - \tilde{\mathbf{B}})^{-1} \right\|_F .$$

Proof. Under the assumption of the proposition, the spectral radius of $\tilde{\mathbf{B}}$ is upper bounded by $(d\epsilon)^{1/k}$ with similar reasoning in (15), which is smaller than one; therefore, we can write $(\mathbb{I} - \tilde{\mathbf{B}})^{-1} = \sum_{j=0}^{\infty} \tilde{\mathbf{B}}^j$. Thus we have

$$\begin{aligned} \left\| (\mathbb{I} - \tilde{\mathbf{B}})^{-1} - \sum_{i=0}^{k} \tilde{\mathbf{B}}^k \right\|_F &= \left\| \sum_{j=1}^{\infty} \tilde{\mathbf{B}}^{k+j} \right\|_F = \left\| \tilde{\mathbf{B}}^k \sum_{j=1}^{\infty} \tilde{\mathbf{B}}^j \right\|_F \\ \leqslant \|\tilde{\mathbf{B}}^k\|_F \left\| \sum_{j=0}^{\infty} \tilde{\mathbf{B}}^j \right\|_F &\leqslant d \|\tilde{\mathbf{B}}^k\|_\infty \left\| (\mathbb{I} - \tilde{\mathbf{B}})^{-1} \right\|_F = d\epsilon \left\| (\mathbb{I} - \tilde{\mathbf{B}})^{-1} \right\|_F . \end{aligned}$$

□

B Fast Truncated Matrix Power Iteration

B.1 Example Implementation

We provide an example implementation of Algorithm 2 in Listing 1.

```python
import numpy as np


def h_fast_tmpi(B, eps=1e-6):
    d = B.shape[0]
    global old_g
    global old_B
    global sec_g
    if old_g is None:
        old_g = np.zeros_like(B)
        old_B = np.zeros_like(B)
        sec_g = np.zeros_like(B)
    if old_g.shape[0] != d:
        old_g = np.zeros_like(B)
        old_B = np.zeros_like(B)
        sec_g = np.zeros_like(B)
    _B = np.copy(B)
    _g = np.copy(B)
    _grad = np.eye(d)

    i = 1
    while i <= d:
        np.copyto(old_B, _B)
        np.copyto(old_g, _g)
        np.matmul(_B, _g, out=_g)
        np.add(old_g, _g, out=_g)

        np.copyto(sec_g, _grad)
        np.matmul(_B.T, _grad, out=_grad)
        np.matmul(_B, _B, out=_B)
        np.add(_grad, sec_g, out=_grad)
        np.copyto(sec_g, _g)
        sec_g -= old_g
        sec_g += old_B
        sec_g -= _B
        sec_g *= i

        _grad += sec_g.T

        if np.max(np.abs(_B)) <= eps:
            break
        i *= 2

    return np.trace(_g), _grad
```

Listing 1: An example implementation of Algorithm 2 in Python.

B.2 Comparison of Time Complexity of Different DAG Constraints

The time complexity for computing the proposed DAG constraints, along with the existing ones, is provided in Table 2. Note that the complexity is described in terms of matrix multiplications, whose efficient algorithms have received much attention in the field of numerical linear algebra in the past decades. Coppersmith–Winograd [12] algorithm and Strassen algorithm [40] are widely used and require $\mathcal{O}(d^{2.376})$ and $\mathcal{O}(d^{2.807})$ operations, respectively.

C Additional Experimental Results

C.1 Other Performance Metrics

The False Discovery Rate (FDR), True Positive Rate (TPR), and False Positive Rate (FPR) are provided in this section, and our TMPI based algorithm often achieves best performance for all these metrics.

Table 2: Complexity of different DAG constraints terms and their gradients in terms of matrix multiplications.

DAG constraint term (and its gradient)	Complexity
Matrix exponential [50][i]	$\mathcal{O}(d)$
Binomial [45] [ii]	$\mathcal{O}(\log d)$
Polynomial [44]	$\mathcal{O}(d)$
NOBEARS [21]	$\mathcal{O}(1)$
Single term	$\mathcal{O}(\log d)$
Geometric	$\mathcal{O}(d)$
Fast geometric	$\mathcal{O}(\log d)$
TMPI	$\mathcal{O}(k)$
Fast TMPI	$\mathcal{O}(\log k)$

[i] More precisely, the complexity of matrix exponential depends on the desired accuracy of the computation, for which a wide variety of algorithms are available [1, 2]. Here we use a fixed complexity since the first d terms of the Taylor expansion are sufficient to form a DAG constraint. [ii] Based on exponentiation by squaring.

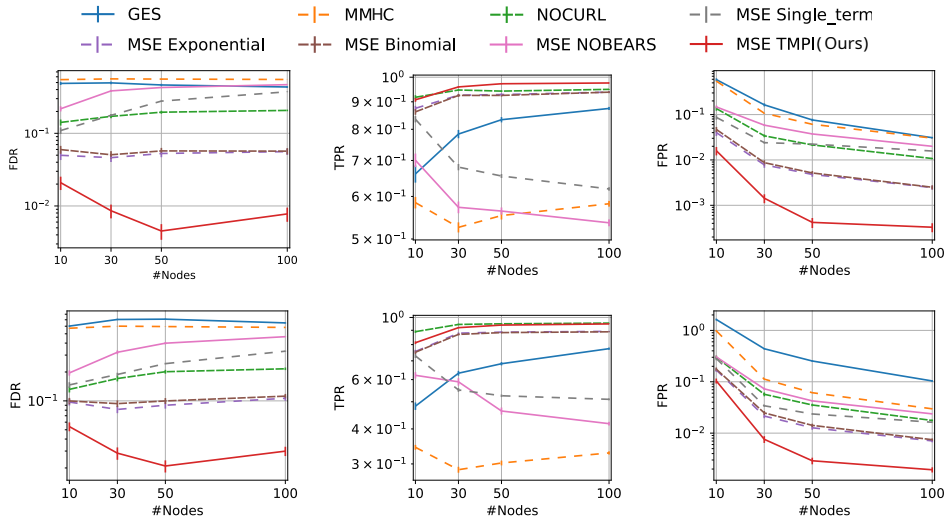


Figure 5: False Discovery Rate (FDR, lower is better), True Positive Rate (TPR, higher is better) and False Positive Rate (FPR, lower is better) of MSE based algorithms on ER2 (**top**) and ER3 (**bottom**) graphs.

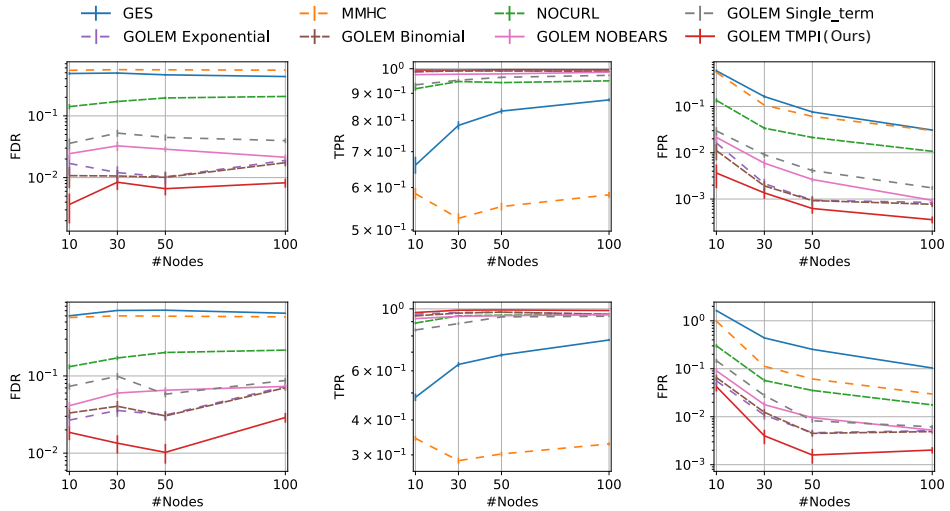
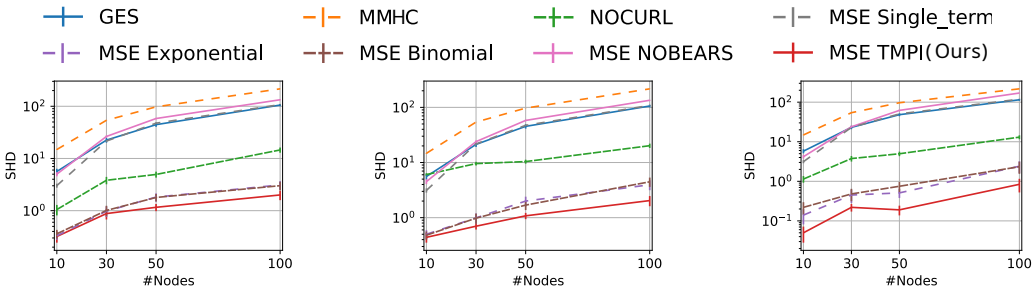


Figure 6: False Discovery Rate (FDR, lower is better), True Positive Rate (TPR, higher is better) and False Positive Rate (FPR, lower is better) of likelihood loss based algorithms on ER2 (**top**) and ER3 (**bottom**) graphs.

C.2 Experiments on Scale-Free Graphs

We consider the scale-free graph model [3] and the results are shown in Figures 7, where we can see that with different graph structures our algorithm consistently achieves better results.

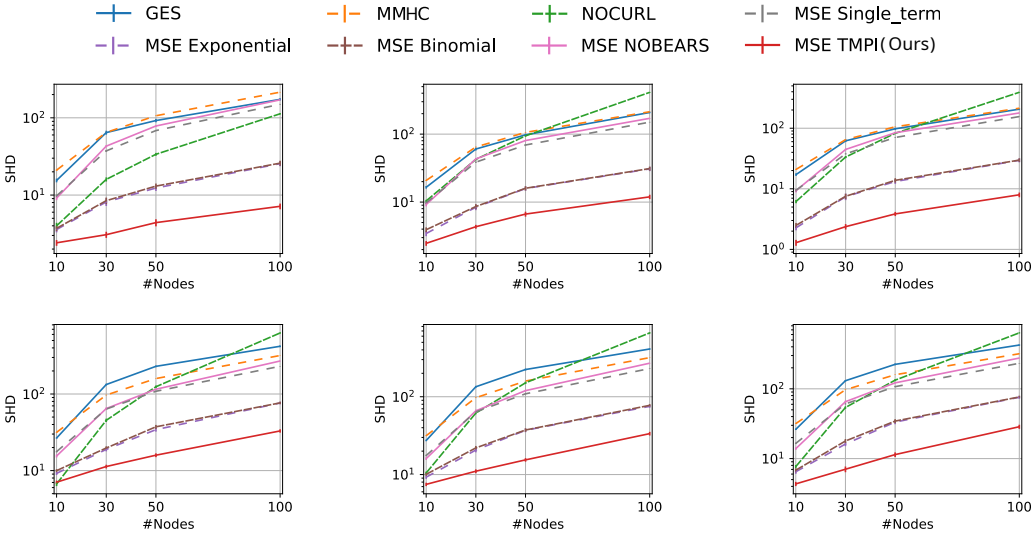


(a) Gaussian. (b) Exponential. (c) Gumbel.

Figure 7: SHD of different algorithms on SF2 graphs.

C.3 Experiments with Smaller Sample Size

We experiment with a smaller sample size, *i.e.*, $n = 200$ samples, on the linear SEM, and the results are shown in Figure 8, where our algorithm outperforms all others in most of the cases.



Gaussian Exponential Gumbel

Figure 8: Results with 200 samples on ER2 (**top**) and ER3 (**bottom**) graphs.

C.4 Experiments on Larger Graphs

We conduct experiments on ER2 graph with 500 nodes with Gaussian noises. For these kind of graphs, the GOLEM algorithm is known to perform well [23]. Thus we combine the GOLEM algorithm with different DAG constraints, where for all DAG constraints, the parameters η_1 and η_2 for GOLEM algorithm are set to 0.02 and 5.0, respectively, and the number of iterations is set to 70000. The results are shown in Table 3 and our algorithm achieves the best SHD and fastest running time. The running time are measured on a server with V100 GPU.

Table 3: Experiments on 500-node ER2 graph with Gaussian noise. Best results are in bold. All results are averaged from 20 simulations, along with the standard errors.

	GOLEM + Exponetial	GOLEM + Binomial	GOLEM + TMPI
SHD ↓	45.20 ± 6.44	46.75 ± 7.41	$\mathbf{14.45} \pm \mathbf{2.27}$
Running Time(s) ↓	882.22 ± 1.08	795.24 ± 1.24	$\mathbf{670.26} \pm \mathbf{0.76}$

C.5 Comparison with NOFEARS

Wei et al. [44] and Ng et al. [24] mentioned that the operation $\mathbf{B} \odot \mathbf{B}$ is one source of gradient vanishing. Based on the observation, Wei et al. [44] proposed a KKT condition based local search strategy (named NOFEARS) to avoid gradient vanishing caused by $\mathbf{B} \odot \mathbf{B}$. As shown by Wei et al. [44], to attain best performance the NOFEARS method is applied as a post-processing procedure for NOTEARS[50]. Our method identified a different source of gradient vanishing caused by the small coefficients for higher-order terms in DAG constraints. Thus our work is in parallel with previous work [24, 44], and the local search strategy by Wei et al. [44] can be combined with our technique. We compare the performance of our algorithm and NOFEARS [44] in Figure 9, where our method outperforms the original NOFEARS (NOTEARS+KKTS search) in most cases. The method that combines our method and NOFEARS attains the best performance (*i.e.*, lowest SHD and False Discovery Rate (FDR), highest True Positive Rate (TPR)).

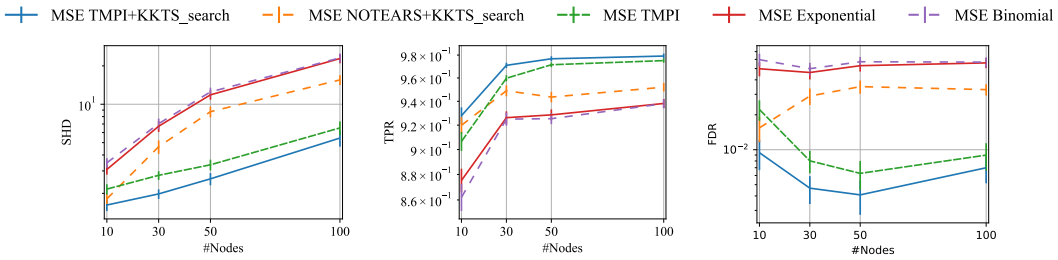


Figure 9: Comparison with KKT based local search strategy [44] on ER2 graphs with linear Gaussian SEMs. The results are averaged from 100 simulations. The NOFEARS method (NOTEARS+KKTS search) achieves better performance tham that of the original NOTEARS (MSE Exponential), but it performs worse than our TMPI method in most cases. The method that combines TMPI and local search (TMPI+KKTS search) achieves the best performance.

C.6 Experiments on Denser Graphs

We have conducted a simple experiment on ER6 graphs to compare the performance of different polynomial constraints on ER6 graphs (where the expected degree of node is 12). The dataset generating procedure for ER6 graph is the same as Section 4, but the number of samples n is set to 5000. The result is shown in Figure 10, where our TMPI method achieves best performance (*i.e.*, lowest SHD and False Detection Rate (FDR), highest True Positive Rate (TPR)).

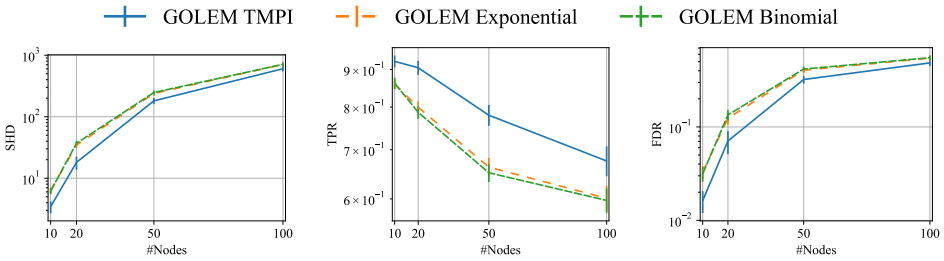


Figure 10: Comparison of different polynomial based DAG constraints on ER6 graphs. All results are averaged from 20 simulations.

C.7 Additional Ablation Study

We provide an ablation study to illustrate the numeric difference between different k for our DAG constraint, as depicted in Figure 11. One observes that typically the error of our truncated DAG constraints against the geometric series based DAG constraint appears to drop exponentially as k increases. Furthermore, this error may often exceed the machine precision. Therefore, for our TMPI algorithm (*i.e.*, the $\mathcal{O}(k)$ implementation described in Algorithm 1), the value of k is often not large, and thus the TMPI algorithm is usually faster than both the geometric and fast geometric method, as demonstrated by the empirical study in Figure 4. Our fast TMPI algorithm (i.e., the $\mathcal{O}(\log k)$ implementation described in Algorithm 2) further accelerates the procedure and its running time is the shortest in most settings.

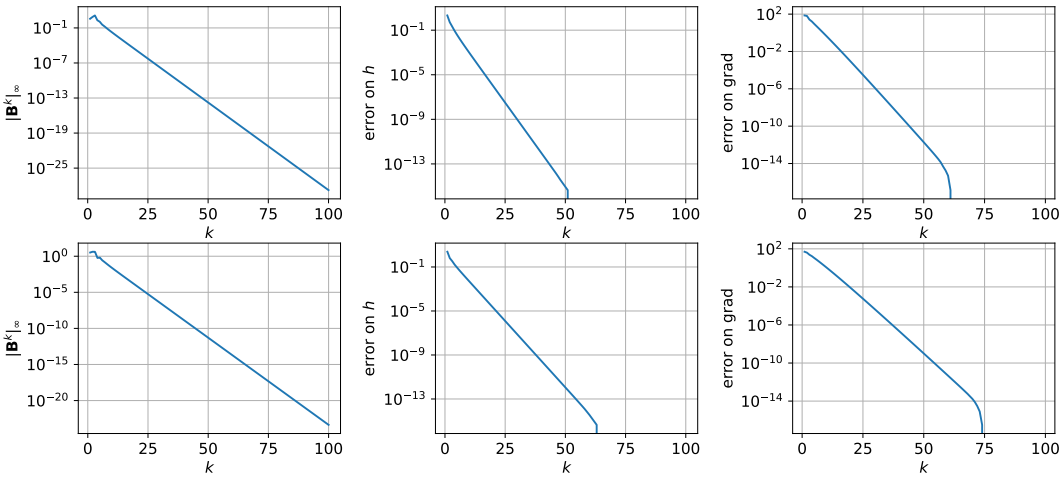


Figure 11: Typical example of $\|\tilde{\mathbf{B}}^k\|_\infty$, error of $h^k_{\text{trunc}}(\tilde{\mathbf{B}})$ against $h_{\text{geo}}(\tilde{\mathbf{B}})$, and error of $\nabla_{\tilde{\mathbf{B}}} h^k_{\text{trunc}}(\tilde{\mathbf{B}})$ against $\nabla_{\tilde{\mathbf{B}}} h_{\text{geo}}(\tilde{\mathbf{B}})$ (in terms of infinity norm). The analysis is based on ER2 (**top**) and ER3 (**bottom**) graphs.