ψDAG: Projected Stochastic Approximation Iteration for Linear DAG Structure Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning the structure of Directed Acyclic Graphs (DAGs) presents a significant challenge due to the vast combinatorial search space of possible graphs, which scales exponentially with the number of nodes. Recent advancements have redefined this problem as a continuous optimization task by incorporating differentiable acyclicity constraints. These methods commonly rely on algebraic characterizations of DAGs, such as matrix exponentials, to enable the use of gradient-based optimization techniques. Despite these innovations, existing methods often face optimization difficulties due to the highly non-convex nature of DAG constraints and the per-iteration computational complexity. In this work, we present a novel framework for learning DAGs, employing a Stochastic Approximation approach integrated with Stochastic Gradient Descent (SGD)-based optimization techniques. Our framework introduces new projection methods tailored to efficiently enforce DAG constraints, ensuring that the algorithm converges to a feasible local minimum. With its low iteration complexity, the proposed method is well-suited for handling large-scale problems with improved computational efficiency. We demonstrate the effectiveness and scalability of our framework through comprehensive experiments, which confirm its superior performance across various settings.

1 Introduction

Learning graphical structures from data using Directed Acyclic Graphs (DAGs) is a fundamental challenge in machine learning (Koller & Friedman, 2009; Peters et al., 2016; Arjovsky et al., 2019; Sauer & Geiger, 2021). This task has a wide range of practical applications across fields such as economics, genome research (Zhang et al., 2013; Stephens & Balding, 2009), social sciences (Morgan & Winship, 2015), biology (Sachs et al., 2005a), and causal inference (Pearl, 2009; Spirtes et al., 2000). Learning the graphical structure is essential because the resulting models can often be given causal interpretations or transformed into representations with causal significance, such as Markov equivalence classes. When graphical models cannot be interpreted causally (Pearl, 2009; Spirtes et al., 2000), they can still offer a flexible representation for decomposing the joint distribution.

Structure learning methods are typically categorized into two approaches: score-based algorithms searching for a DAG minimizing a particular loss function and constraint-based algorithms relying on conditional independence tests. Constraint-based methods, such as the PC algorithm (Spirtes & Glymour, 1991) and FCI (Spirtes et al., 1995; Colombo et al., 2012), use conditional independence tests to recover the Markov equivalence class under the assumption of faithfulness. Other approaches, like those described in Margaritis & Thrun (1999) and Tsamardinos et al. (2003), employ local Markov boundary search. On the other hand, score-based methods frame the problem as an optimization of a specific scoring function, with typical choices including BGe (Kuipers et al., 2014), BIC (Chickering & Heckerman, 1997), BDe(u) (Heckerman et al., 1995), and MDL (Bouckaert, 1993). Given the vast

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

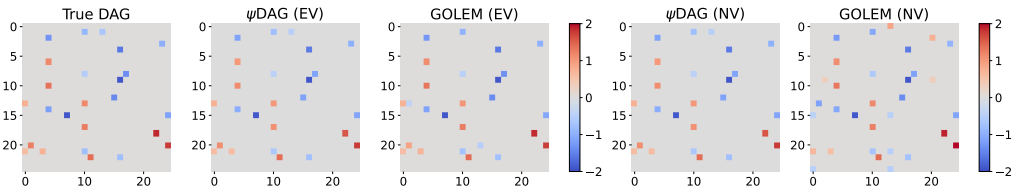


Figure 1: Visual comparison of the learned weighted adjacency matrices on a 25-node ER2 graph under Gaussian noise with equal variances (EV) and non-equal variances (NV, with noise ratio $r = 5$). For both methods ψDAG and GOLEM the L_1 distance in the EV setting is 2.6. In the NV setting, ψDAG maintains an L_1 distance of 2.6, while GOLEM's L_1 distance increases to 10.7, highlighting the robustness and generalization ability of ψDAG across varying noise conditions.

search space of potential graphs, many score-based methods employ local heuristics, such as Greedy Equivalence Search (GES) (Chickering, 2002), to efficiently navigate this complexity.

Recently, Zheng et al. (2018) introduced a smooth formulation for enforcing acyclicity, transforming the structure learning problem from its inherently discrete nature into a continuous, non-convex optimization task. This formulation allows for the use of gradient-based optimization techniques, enabling various extensions and adaptations to various domains, including nonlinear models (Yu et al., 2019; Ng et al., 2022b; Kalainathan et al., 2022), interventional datasets (Brouillard et al., 2020; Faria et al., 2022), unobserved confounders (Bhattacharya et al., 2021; Bellot & Van der Schaar, 2021), incomplete datasets (Gao et al., 2022a; Wang et al., 2020), time series analysis (Sun et al., 2021; Pamfil et al., 2020), multi-task learning (Chen et al., 2021), multi-domain settings (Zeng et al., 2021), federated learning (Ng & Zhang, 2022; Gao et al., 2023), and representation learning (Yang et al., 2021). With the growing interest in continuous structure learning methods (Vowels et al., 2022), a variety of theoretical and empirical studies have emerged. For instance, Ng et al. (2020) investigated the optimality conditions and convergence properties of continuously constrained approaches such as Zheng et al. (2018). In the bivariate case, Deng et al. (2023b) demonstrated that a suitable optimization strategy converges to the global minimum of the least squares objective. Zhang et al. (2022) and Bello et al. (2022) then identified potential gradient vanishing issues with existing DAG constraints (Zheng et al., 2018) and proposed adjustments to overcome these challenges.

Contributions. In this work, we focus on the graphical models represented as Directed Acyclic Graphs (DAGs). Our main contributions can be summarized as follows:

1. **Problem reformulation:** We introduce a new reformulation (9) of the discrete optimization problem for finding DAG as a stochastic optimization problem, and we discuss its properties in detail in Section 3.1. We demonstrate that the solution of this reformulated problem recovers the true DAG (Section 3.1).
2. **Novel algorithm:** Leveraging insights from stochastic optimization, we present a new framework (Algorithm 1) for DAG learning (Section 4) and present a simple yet effective algorithm ψDAG (Algorithm 2) within the framework.
3. **Experimental comparison:** In Section 5, we demonstrate that the method ψDAG scales very well with graph size, handling up to 10000 nodes. At that scale, the primary limitation is not computation complexity but the memory required to store the DAG itself. As a baseline, we compare ψDAG with established DAG learning methods, including NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021) and DAGMA (Bello et al., 2022). We show a significant improvement in scalability, as baseline methods struggle with larger graphs. Specifically, NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021) and DAGMA (Bello et al., 2022) require more than 100 hours for graphs with over 3000 nodes, exceeding the allotted time.

2 Background

In this section, we introduce the necessary graph notation and formalize the linear Structural Equation Model (SEM) framework used for learning Directed Acyclic Graphs (DAGs). For a detailed discussion of related methods and further literature, please refer to Appendix A.

2.1 Graph Notation

Let $\mathcal{G} \stackrel{def}{=} (V, E, w)$ represent a weighted directed graph, where V denotes the set of vertices with cardinality $d \stackrel{def}{=} |V|$, $E \in 2^{V \times V}$ is the set of edges, and $w : V \times V \to \mathbb{R} \setminus \{0\}$ assigns weights to the edges. The *adjacency matrix* $\mathbf{A}(\mathcal{G}) : \mathbb{R}^{d \times d}$ is defined such that $[\mathbf{A}(\mathcal{G})]_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. Similarly, the *weighted adjacency matrix* $\mathbf{W}(\mathcal{G})$ is defined by $[\mathbf{W}(\mathcal{G})]_{ij} = w(i, j)$ if $(i, j) \in E$ and 0 otherwise.

When the weight function w is binary, we simplify the notation to $\mathcal{G} \stackrel{def}{=} (V, E)$. Similarly, when the graph $\mathcal{G}$ is clear from context, we shorthand the notation to $\mathbf{A} \stackrel{def}{=} \mathbf{A}(\mathcal{G})$ and $\mathbf{W} \stackrel{def}{=} \mathbf{W}(\mathcal{G})$.
We denote the space of DAGs as $\mathbb{D}$. Since we will be utilizing topological sorting of DAGs[1], we also denote the space of vertex permutations Π.

2.2 Linear DAG and SEM

A Directed Acyclic Graph (DAG) model, defined on a set of n random vectors $\mathbf{X} \in \mathbb{R}^{n \times d}$, where $\mathbf{X} \stackrel{def}{=} (X_1, \ldots, X_n)$ and $X_i \in \mathbb{R}^d$, consists of two components:

1. A DAG $\mathcal{G} = (V, E)$, which encodes a set of conditional independence relationships among the variables.
2. The joint distribution $P(\mathbf{X})$ with density $p(x)$, which is Markov with respect to the DAG $\mathcal{G}$ and factors as $p(x) = \prod_{i=1}^{d} p(x_i \mid x_{\mathrm{PA}_{\mathcal{G}}(i)})$, where $\mathrm{PA}_{\mathcal{G}}(i) = \{j \in V : X_j \to X_i \in E\}$ represents the set of parents of X_i in $\mathcal{G}$.

This work focuses on the linear DAG model, which can be equivalently represented by a set of linear Structural Equation Models (SEMs). In matrix notation, the linear DAG model can be expressed as

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{N}, \tag{1}$$

where $\mathbf{W} = [\mathbf{W}_1 | \cdots | \mathbf{W}_d]$ is a weighted adjacency matrix, and $\mathbf{N} \stackrel{def}{=} (N_1, \ldots, N_n)$ is a matrix where each $N_i \in \mathbb{R}^d$ represents a noise vector with independent components. The structure of graph $\mathcal{G}$ is determined by the non-zero coefficients in $\mathbf{W}$; specifically $X_j \to X_i \in E$ if and only if the corresponding coefficient in $\mathbf{W}_i$ for X_j is non-zero. The classical objective function is based on the least squares loss applied to the linear DAG model,

$$l(\mathbf{W}; \mathbf{X}) \stackrel{def}{=} \tfrac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{W}\|_F^2. \tag{2}$$

3 Stochastic Approximation for DAGs

Our framework is built on a reformulation of the objective function as a stochastic optimization problem, aiming to minimize the stochastic function $F(w)$,

$$\min_{w \in \mathbb{R}^d} \left\{ F(w) \stackrel{def}{=} \mathbb{E}_{\xi} \left[f(w, \xi) \right] \right\}, \tag{3}$$

where $\xi \in \Xi$ is a random variable that follows the distribution Ξ. This formulation is common in stochastic optimization where computing the exact expectation is infeasible, but the values of $f(w, \xi)$ and its stochastic gradients $g(w, \xi)$ can be computed. Linear and logistic regressions are classical examples of such problems.

To address this problem, two main approaches exist: Stochastic Approximation (SA) and Sample Average Approximation (SAA). The SAA approach involves sampling a fixed number n of random variables or data points ξ_i and then minimizing their average $\tilde{F}(w)$:

$$\min_{w \in \mathbb{R}^d} \left\{ \tilde{F}(w) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} f(w, \xi_i) \right\}. \tag{4}$$

[1]Topologial sorting of a graph $\mathcal{G} \stackrel{def}{=} (V, E, w)$ refers to vertex ordering $V_1, V_2, \ldots, V_d$ such that E contains no edges of the form $V_i \to V_j$, where $i \leq j$. Importantly, every DAG has at least one topological sorting.

Algorithm 1 ψDAG framework

Requires: Initial model $\mathbf{W}_0 \in \mathbb{R}^{d\times d}$, such that $\mathrm{diag}(\mathbf{W}_0) = 0$.
for $k = 0, 1, 2 \ldots, K-1$ **do**
  $\mathbf{W}_k^{(1/3)} = \mathcal{A}_1(\mathbf{W}_k)$ $\quad \{\mathbf{W}_k^{(1/3)} \in \mathbb{R}^{d\times d}\}$
  $(\mathbf{W}_k^{(2/3)}, \pi_k) = \psi(\mathbf{W}_k^{(1/3)})$ $\quad \{\mathbf{W}_k^{(2/3)} \in \mathbb{D}\}$
  $\mathbf{W}_{k+1} = \mathcal{A}_2(\mathbf{W}_k^{(2/3)}; \pi_k)$ $\quad \{\mathbf{W}_{k+1} \in \mathbb{D} \subset \mathbb{R}^{d\times d}\}$
end for
Output: $\mathbf{W}_K$.

Now, the problem (4) becomes deterministic and can be solved using various optimization methods, such as gradient descent. However, the main drawback of this approach is that the solution of (4) $\tilde{w}^*$ is not necessarily equal to the solution of the original problem (3). Even with a perfect solution of (4), there will still be a gap $\|\tilde{w}^* - w^*\| = \delta_x$ and $F(\tilde{w}^*) - F^* = \delta_F$ between approximate and true solution. These gaps are dependent on the sample size n.
Stochastic Approximation (SA) minimizes the true function $F(w)$ by utilizing the stochastic gradient $g(w, \xi)$. Below, we provide the formal definition of a stochastic gradient.

Assumption 1. *For all $w \in \mathbb{R}^d$, we assume that stochastic gradients $g(w, \xi) \in \mathbb{R}^d$ satisfy*

$$\mathbb{E}[g(w, \xi) \mid w] = \nabla F(w), \tag{5}$$

$$\mathbb{E}\left[\|g(w, \xi) - \nabla F(w)\|^2 \mid w\right] \leq \sigma_1^2. \tag{6}$$

We use these stochastic gradients in SGD-type methods:

$$w_{t+1} = w_t - h_t g(w_t, \xi_i), \tag{7}$$

where h_t is a step-size schedule. SA originated with the pioneering paper by Robbins & Monro (1951). For convex and L-smooth function $F(w)$, Polyak (1990); Polyak & Juditsky (1992); Nemirovski et al. (2009); Nemirovski & Yudin (1983) developed significant improvements to SA method in the form of longer step-sizes with iterate averaging, and obtained the convergence guarantee

$$\mathbb{E}\left[F(w_T) - F(x^*)\right] \leq \mathcal{O}\left(\frac{\sigma_1 R}{\sqrt{T}} + \frac{L_1 R^2}{T}\right).$$

Lan (2012) developed an optimal method with a guaranteed convergence rate $\mathcal{O}\left(\frac{\sigma_1 R}{\sqrt{T}} + \frac{L_1 R^2}{T^2}\right)$, matching the worst-case lower bounds. The key advantage of SA is that it provides convergence guarantees for the original problem (3). Additionally, methods effective for the SA approach tend to perform well for the SAA approach as well.

3.1 Stochastic Reformulation

Using the perspective of Stochastic Approximation, we can rewrite the linear DAG (1) as

$$x = X_i = \left[\mathbf{I} - \mathbf{W}_*^\top\right]^{-1} N_i, \tag{8}$$

where $\mathbf{W}^*$ is a true DAG that corresponds to the full distribution, and our goal is to find DAG $\mathbf{W}$ that is close to $\mathbf{W}^*$. If we assume that $x = X_i$ is a random vector sampled from a distribution $\mathcal{D}$, we can express the objective function as an expectation,

$$\min_{\mathbf{W}\in\mathbb{D}} \mathbb{E}_{x\sim\mathcal{D}}\left[l(\mathbf{W}; x) \overset{def}{=} \tfrac{1}{2}\|x - \mathbf{W}^\top x\|^2\right]. \tag{9}$$

For x from (8) we can calculate $\|x - \mathbf{W}^\top x\| = \|(\mathbf{I} - \mathbf{W}^\top)x\| = \|(\mathbf{I} - \mathbf{W})\left[\mathbf{I} - \mathbf{W}_*^\top\right]^{-1} N_i\|$, which implies that the minimizer of (9) recovers the true DAG. Conversely, this is not the case for methods such as Zheng et al. (2018), Ng et al. (2020), and Bello et al. (2022), which are based on SAA approaches with losses (2), (11), (12), (13).

Algorithm 2 ψDAG

Requires: initial model $\mathbf{W}_0 \in \mathbb{R}^{d\times d}$, numbers or iterations τ_1, τ_2.
for $k = 0, 1, 2 \ldots, K-1$ **do**
  $\mathbf{W}_k^{(1/3)} = \mathsf{SGD}(\mathbf{W}_k)$ {τ_1 iterations over $\mathbb{R}^{d\times d}$}
  $(\mathbf{W}_k^{(2/3)}, \pi_k) =$ Algorithm 3 $(\mathbf{W}_k^{(1/3)})$
  $\mathbf{W}_{k+1} = \mathsf{SGD}_{\pi_k}(\mathbf{W}_k)$ {τ_2 iterations preserving ordering π_k}
end for
Output: $\mathbf{W}_K$

4 Scalable Optimization Framework for DAG Learning

In this section, we present our proposed scalable optimization framework for DAG learning. We begin by showing that using a fixed vertex ordering can lead to suboptimal solutions, as demonstrated in Section 4.1. Motivated by this, we develop a three-stage framework that alternates between unconstrained optimization, projection onto the DAG space, and constrained optimization guided by topological ordering.

Instead of strictly enforcing DAG constraints throughout the entire iteration process, we propose a novel, scalable optimization framework that consists of three main steps:

1. Running an optimization algorithm $\mathcal{A}_1$ without any DAG constraints, only forcing the diagonal to be zero ($\mathrm{diag}(W_k) = 0$), $\mathcal{A}_1 : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times d}$.
2. Finding a DAG that is close to the current iterate using a projection $\psi : \mathbb{R}^{d\times d} \to (\mathbb{D}, \Pi)$, which also returns its topological sorting π.
3. Running the optimization algorithm $\mathcal{A}_2$ while preserving the vertex order, $\mathcal{A}_2 : (\mathbb{D}; \Pi) \to \mathbb{D}$.

This design enables efficient and accurate structure learning while avoiding the computational burden of enforcing DAG constraints at every iteration.

4.1 Optimization for the fixed vertex ordering

Let us clarify how to optimize while preserving the order of the vertices in step 3 of the framework. Given a DAG $\mathcal{G}$, we can construct its topological ordering, denoted as $ord(\mathcal{G})$. In this ordering, for every edge, the start vertex appears earlier in the sequence than the end vertex. In general, this ordering is not unique. In the space of DAGs with d vertices $\mathbb{D}$, there are $d!$ possible topological orderings.

Once we have a topological ordering of the DAG, we can construct a larger DAG, $\hat{\mathcal{G}}$, by performing the transitive closure of $\mathcal{G}$. This new DAG $\hat{\mathcal{G}}$ contains all the edges of the original DAG, and additionally, it includes an edge between vertices V_i and V_j if there exists the path from V_i to V_j in $\mathcal{G}$. Thus, $\hat{\mathcal{G}}$ is an expanded version of $\mathcal{G}$.

Now, the question arises: is it possible to construct an even larger DAG that contains both $\mathcal{G}$ and $\hat{\mathcal{G}}$? The answer is yes! We call this graph the *Full DAG*, denoted by $\tilde{\mathcal{G}}$, which is constructed via full transitive closure[2]. In $\tilde{\mathcal{G}}$, there is an edge from vertex V_i to vertex V_j if $i < j$ is in topological order $ord(\mathcal{G})$. This makes $\tilde{\mathcal{G}}$ the maximal DAG that includes $\mathcal{G}$. Note that for every topological sort, there is a corresponding full DAG. So, there are a total of $d!$ different full DAGs in the space of DAGs with d vertices $\mathbb{D}$.

We are now ready to discuss the optimization part. Let us formulate the following optimization problem

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times d}} \mathbb{E}_{x\sim\mathcal{D}} \left[l(\mathbf{W}\cdot\mathbf{A}; x) = \tfrac{1}{2}\|x - (\mathbf{W}\cdot\mathbf{A})^\top x\|^2 \right], \tag{10}$$

where $(\cdot)$ denotes elementwise matrix multiplication. In this formulation, $\mathbf{A}$ acts as a mask, specifying coordinates that do not require gradient computation. The problem (10) is a quadratic convex

[2]Informally, for set of edges E, the transitive closure E^+ is the smallest set that includes edges (a, b) whenever there is a path from a to b within E. Note that E^+ is the smallest superset of E that satisfies $(a, c) \in E^+$ whenever $(a, b) \in E^+, (b, c) \in E^+$.

Algorithm 3 Projection $\psi(\mathbf{W})$ computing the "closest" vertex ordering (recursive form)

Requires: Model $\mathbf{W} \in \mathbb{R}^{d\times d}$, (optional) weights $\mathbf{L} \in \mathbb{R}^{d\times d}$ with default value $\mathbf{L} = \mathbf{1}\mathbf{1}^\top$.
for $k = 1, \dots, d$ **do**
  Set $r_k = \| (\mathbf{W} \circ \mathbf{L})\,[k][:]\|^2$
  Set $c_k = \| (\mathbf{W} \circ \mathbf{L})\,[:][k]\|^2$
end for
Set $i_c = \arg\min_{k\in\{1,\dots,d\}} c_k$
Set $i_r = \arg\min_{k\in\{1,\dots,d\}} r_k$
if $r_{i_r} <= c_{i_c}$ **then**
  Output: $[\psi(\mathbf{W}(i_c, i_c), \mathbf{L}(i_c, i_c)), i_r]$
else
  Output: $[i_c, \psi(\mathbf{W}(i_c, i_c), \mathbf{L}(i_c, i_c))]$
end if
{By $A(i, j)$ we denote the submatrix $A[1, \dots, i-1, i+1, \dots, d][1, \dots, j-1, j+1, \dots, d]$}

stochastic optimization problem, which can be efficiently solved using stochastic gradient descent (SGD)-type methods. These methods guarantee convergence to the global minimum, with a rate of $\mathcal{O}\left(\frac{\sigma_1 R}{\sqrt{T}} + \frac{L_1 R^2}{T}\right)$.

Assume that $\mathcal{G}^*$ is the true DAG with a weighted adjacency matrix $\mathbf{W}^*$, which is the solution we aim to find. Next, we can have the true ordering $ord(\mathcal{G}^*)$ and the true full DAG $\tilde{\mathcal{G}}^*$ with its adjacency matrix $\mathbf{A}(\tilde{\mathcal{G}}^*)$. The optimization problem (9), with the solution $\mathbf{W}^*$, can be addressed by solving the optimization problem (10) with $\mathbf{A} = \mathbf{A}(\mathcal{G}^*)$. This result indicates that if we know the true topological ordering $ord(\mathcal{G}^*)$, then we can recover the true DAG $\mathbf{W}^*$ with high accuracy. From a discrete optimization perspective, this approach significantly reduces the space of constraints from 2^{d^2-d} to $d!$.

To illustrate the specificity of the minimizer of the proposed problem, Figure 2 demonstrates that minimizing (9) over a fixed random vertex ordering does not approach the true solution of (9). The "Correct order" curve demonstrates the convergence of (10) when the true ordering $ord(\mathcal{G}^*)$ is known.

Note that for a fixed vertex ordering and fixed adjacency matrix $\mathbf{A}$, the objective (10) becomes separable, enabling parallel computation for large-scale problems. In this work, we solved the minimization problem (10) for the number of nodes up to $d = 10^4$, at which point the limiting factor was the memory to store $\mathbf{W} \in \mathbb{R}^{d\times d}$. Through parallelization and efficient memory management, it is possible to solve even larger problems.

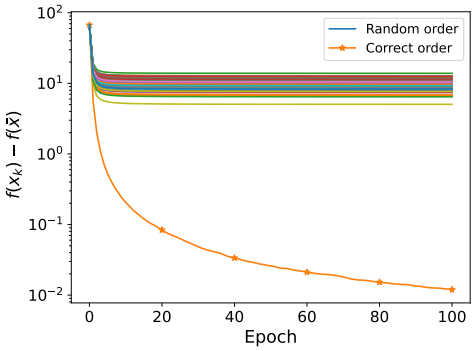
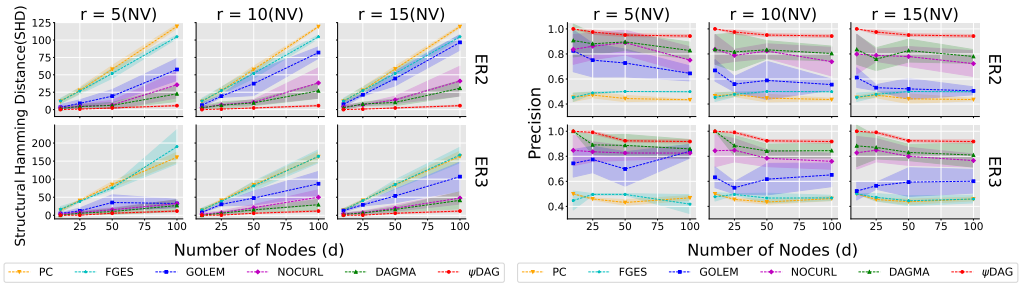


Figure 2: Minimizing (9) using SGD over a fixed topological ordering on ER4 with $d = 100$ and Gaussian noise.

4.2 Methodology

We now introduce the method ψDAG, which implements the framework outlined in Algorithm 1. For simplicity, we select algorithm $\mathcal{A}_1$ as τ_1 steps of Stochastic Gradient Descent (SGD). Similarly, $\mathcal{A}_2$ consists of τ_2 steps SGD, where gradients are projected onto the space spanned by DAG's topological sorting, thus preserving the vertex order. It is important to reiterate that SGD is guaranteed to converge to the neighborhood of the solution. In the implementation, we employed an advanced version of SGD, Universal Stochastic Gradient Method from Rodomanov et al. (2024).

The implementation of the projection method is simple as well. We compute a "closest" topological sorting and remove all edges not permitted by this ordering. The topological sorting is computed by a heuristic that calculates norms of all rows and columns to find the lowest value v_i. The corresponding vertex i is then assigned to the ordering based on the following rule:



(a) SHD (b) Precision

Figure 3: Structure recovery performance under Gaussian noise with non-equal variances (NV) across varying noise ratios ($r \in \{5, 10, 15\}$). Rows correspond to different random graph types, and columns represent increasing noise ratios. Metrics include Structural Hamming Distance (SHD $\downarrow$) and Precision ($\uparrow$). We report the mean values, and standard error is indicated by shaded regions.

- If v_i was the column norm, i is assigned to the beginning of the ordering.
- If v_i was the row norm, i is assigned to the end of the ordering.

This step reduces the number of vertices, and the remaining vertices are topologically sorted using a recursive call. We formalize this procedure in Algorithm 3. Note that this procedure can be efficiently implemented without recursion and with the computation cost $\mathcal{O}(d^2)$.

5 Experiments

We experimentally compare our method, ψDAG[3], with several baselines including PC (Ramsey et al., 2012), FGES (Meek, 1997; Chickering, 2002), NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021) and DAGMA (Bello et al., 2022). As it is established that DAGMA Bello et al. (2022) is an improvement over NOTEARS Zheng et al. (2018), we use mostly the former in our experiments. To ensure a fair comparison, we avoid extensive hyperparameter tuning across all baseline methods. Specifically, we apply the same thresholding procedure as used in Zheng et al. (2018), Ng et al. (2020), Yu et al. (2021), and Bello et al. (2022) across all scenarios.

5.1 Synthetic Data Generation

We generate ground truth DAGs with d nodes and an average of $k \times d$ edges, where $k \in \{2, 3, 4, 6\}$ is a sparsity parameter. The graph structure is based on either the Erdős-Rényi (ER) or the Scale-Free (SF) models. Together with the sparsity level, we denote the graphs as ERk or SFk, respectively. Each edge is assigned a random weight uniformly sampled from the interval $[-2, -0.5] \cup [0.5, 2]$ following the standard practice used in previous work (Zheng et al., 2018; Ng et al., 2020; Yu et al., 2021; Bello et al., 2022) to ensure consistency across methods.

Following the linear Structural Equation Model (SEM), we generate the observed data $\mathbf{X} \in \mathbb{R}^{n \times d}$ using $\mathbf{X} = \mathbf{N}(\mathbf{I} - \mathbf{W})^{-1}$, where $\mathbf{N}$ consists of n independent and identically distributed (i.i.d.) noise samples drawn from Gaussian, exponential, or Gumbel distributions. We evaluate both equal variance (EV) and non-equal variance (NV) Gaussian noise settings. In the EV case, the noise for all variables is scaled by a constant factor of 1.0. In the NV setting, we set the variances of two randomly selected noise variables to be 1 and $r \in \{5, 10, 15\}$, respectively, and the variances of remaining variables are sampled uniformly from $[1, r]$. This setup enables evaluation of robustness under noise heterogeneity. For further details, we refer the reader to Ng et al. (2024). A visual comparison under both EV and NV (with $r = 5$) is shown in Figure 1, highlighting the robustness of ψDAG to non-uniform noise levels. A more detailed description can be found in the Appendix C.

[3]Implementation of the proposed algorithm is available at `https://anonymous.4open.science/r/psiDAG-8F42`. We use the Universal Stochastic Gradient Method (Rodomanov et al., 2024) as the inner optimizer.

5.2 Structure Recovery

We evaluate the structure learning capabilities of our method, as shown in Figure 3, using synthetic data generated from the ER2 and ER3 graphs with varying node counts $d \in \{10, 25, 50, 100\}$. For brevity, we report only the results for Structural Hamming Distance (SHD) and precision across different noise ratios $r \in \{5, 10, 15\}$ in the non-equal variance (NV) Gaussian setting. The complete results for additional metrics, including the F1 score and the recall, are given in the Appendix D.1. Lower SHD and higher precision, F1 score, and recall values indicate better structure recovery. We compare against several representative baselines, including PC, FGES, NOCURL, GOLEM, and DAGMA.

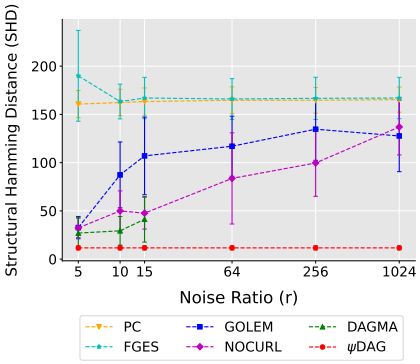


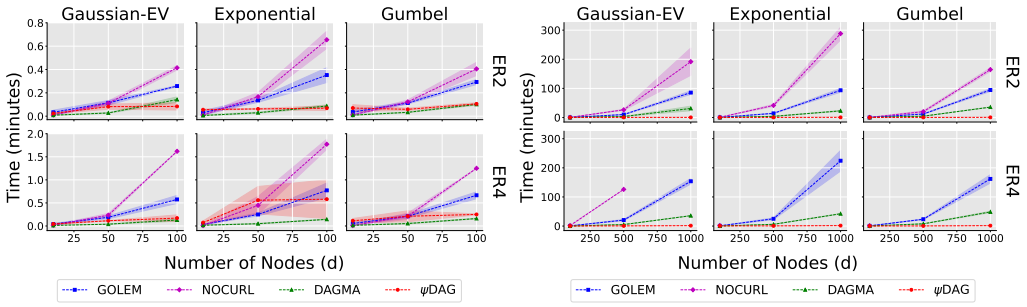
Figure 4: Effect of noise ratio on SHD for ER3 graphs with $d = 100$ in the non-equal variance (NV) Gaussian setting. Error bars denote standard deviation over 3 random seeds. ψDAG remains stable as r increases, while other methods degrade and DAGMA fails to converge for $r > 15$.

Consistent with prior work, all methods perform well in terms of SHD when the number of nodes d and the noise ratio r are small. However, the performance of FGES and PC deteriorates rapidly, even for a moderate number of nodes such as $d = 50$, with SHD increasing significantly. In contrast, our method maintains low SHD across all settings and consistently outperforms baselines as noise heterogeneity increases. Figure 4 shows that the SHD of ψDAG remains stable even as r increases from 5 to 1024, further emphasizing its stability and robustness. Meanwhile, DAGMA fails to converge for $r > 15$, limiting its applicability in high-noise regimes.

5.3 Scalability Comparison

We assess the scalability of the proposed algorithm ,ψDAG, by comparing its runtime against GOLEM, NOCURL, and DAGMA. All methods are run until the objective function converges close to the solution, $f(x_k) - f(\overline{x}) \leq 0.1 \cdot f(\overline{x})$. Figure 5 reports runtime comparisons for ER2 and ER4 graphs under Gaussian, Exponential, and Gumbel noise for graph sizes $d \in \{10, 50, 100, 500, 1000\}$. Due to space constraints, additional results on larger graphs (up to $d = 10,000$) are presented in Appendix D.2, where Figure 9 further highlights the efficiency of ψDAG in high-dimensional settings.

Across all scenarios, ψDAG demonstrates consistently lower runtime compared to baselines, particularly as graph size and density increase. While DAGMA is marginally faster than ψDAG on very small and sparse graphs ($d < 100$), the gap closes quickly with larger graphs. For $d > 100$, ψDAG consistently exhibits superior runtime performance across both sparse and dense graph types. On



(a) Small Graph

(b) Large Graph

Figure 5: Runtime (minutes) of GOLEM, NOCURL, DAGMA, and ψDAG on ER2 and ER4 graphs with increasing number of nodes $d \in \{10, 50, 100, 500, 1000\}$. The columns correspond to different noise distributions: Gaussian (left), exponential (middle), and Gumbel (right). Figure 5a shows results for small graphs ($d \leq 100$) and 5b for large graphs ($d > 100$). ψDAG demonstrates significantly better scalability as the number of nodes increases.

sparse graphs, it converges reliably within a few hours, even at $d = 10,000$, whereas GOLEM and NOCURL exceed a 36-hour runtime for $d \geq 3000$, and DAGMA does so for $d \geq 5000$.

Furthermore, we observe that several baselines fail to meet the convergence criterion even for smaller graphs. For instance, NOCURL does not converge for ER4 graphs with Gaussian-EV noise when $d > 500$, and it fails completely for Exponential and Gumbel noise when $d > 100$. In the ER6 graph, the three baselines GOLEM, NOCURL, and DAGMA do not converge in at least one of the three random seeds. Non-converging runs are excluded from reported statistics. In contrast, ψDAG converges in all runs and maintains competitive runtime performance even at large scale, underscoring both its robustness and practical efficiency.

5.4 Real-world Experiment

We further evaluate ψDAG on a widely used real-world dataset, the *causal protein signaling network*, from Sachs et al. (2005b) and compare it with NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021), and DAGMA (Bello et al., 2022). This dataset captures the expression levels of proteins and phospholipids in human cells under various experimental conditions. It has been extensively used in the literature on causal discovery due to its well-established ground truth and biological relevance. The dataset consists of $n = 853$ observational samples and $d = 11$ variables, with a ground truth DAG containing 17 edges. Despite its small size, it remains a challenging benchmark for causal structure learning algorithms (Zheng et al., 2018; Ng et al., 2020; Gao et al., 2021). We follow the common evaluation setup and apply a threshold of 0.3 across all methods for a fair comparison.

As shown in Table 1, ψDAG achieves superior performance across all metrics: lower Structural Hamming Distance (SHD), higher True Positive Rate (TPR), and lower False Positive Rate (FPR). A more detailed description can be found in Appendix C. We omit the results for DAGMA as it fails to converge on this dataset: its solution $\mathbf{W}$ diverges from the feasible domain in the very first iteration.

Table 1: Performance of top methods on the protein signaling dataset (Sachs et al., 2005b).

	SHD↓	TPR↑	FPR↓
NOTEARS (Zheng et al., 2018)	15	0.29	0.26
GOLEM (Ng et al., 2020)	26	0.29	0.47
NOCURL (Yu et al., 2021)	22	0.35	0.45
ψDAG (Alg.2)	**14**	**0.41**	**0.18**

6 Conclusion

We introduce a novel framework for learning Directed Acyclic Graphs (DAGs) that addresses the scalability and computational challenges of existing methods. Our approach leverages Stochastic Approximation techniques in combination with Stochastic Gradient Descent (SGD)-based methods, allowing for efficient optimization even in high-dimensional settings. A key contribution of our framework is the introduction of new projection techniques that effectively enforce DAG constraints, ensuring that the learned structure adheres to the acyclicity requirement without the need for computationally expensive penalties or constraints seen in prior works.

The proposed framework is theoretically grounded, with convergence guarantees to a feasible local minimum. One of its main advantages is its low iteration complexity, making it highly suitable for large-scale structure learning problems, where traditional methods often struggle with runtime and memory limitations. Through extensive experiments, we show that our approach consistently outperforms strong baselines including NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021), and DAGMA (Bello et al., 2022) in both runtime and structure recovery accuracy. Notably, our method demonstrates robust performance in settings with high noise heterogeneity and varying graph densities.

Limitations and Future Work. In this paper, we have focused on presenting a novel framework for differentiable DAG learning, which integrates a stochastic approach to achieve computational efficiency. While the current results are focused on linear SEMs for simplicity, extending the proposed algorithm to handle nonlinear SEMs (Zheng et al., 2020) is a natural direction for future work. Exploring variance reduction optimization methods is another promising path.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

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

Bello, K., Aragam, B., and Ravikumar, P. DAGMA: Learning DAGs via M-matrices and a log-determinant acyclicity characterization. *Advances in Neural Information Processing Systems*, 35: 8226–8239, 2022.

Bellot, A. and Van der Schaar, M. Deconfounded score method: Scoring DAGs with dense unobserved confounding. *arXiv preprint arXiv:2103.15106*, 2021.

Bertsekas, D., Hager, W., and Mangasarian, O. Nonlinear programming. *Athena Scientific*, 1999.

Bhattacharya, R., Nagarajan, T., Malinsky, D., and Shpitser, I. Differentiable causal discovery under unmeasured confounding. In *International Conference on Artificial Intelligence and Statistics*, pp. 2314–2322. PMLR, 2021.

Birgin, E., Castillo, R., and Martínez, J. Numerical comparison of augmented Lagrangian algorithms for nonconvex problems. *Computational Optimization and Applications*, 31(1):31–55, 2005.

Bouckaert, R. Probabilistic network construction using the minimum description length principle. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pp. 41–48. Springer, 1993.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33: 21865–21877, 2020.

Broyden, C. G. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21:368–381, 1967. doi: 10.2307/2003239. URL `http://www.jstor.org/stable/2003239`.

Chen, W., Drton, M., and Wang, Y. S. On causal discovery with an equal-variance assumption. *Biometrika*, 106(4):973–980, September 2019. ISSN 1464-3510. doi: 10.1093/biomet/asz049. URL `http://dx.doi.org/10.1093/biomet/asz049`.

Chen, X., Sun, H., Ellington, C., Xing, E., and Song, L. Multi-task learning of order-consistent causal graphs. *Advances in Neural Information Processing Systems*, 34:11083–11095, 2021.

Chickering, D. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554, 2002.

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

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

Deng, C., Bello, K., Aragam, B., and Ravikumar, P. Optimizing notears objectives via topological swaps, 2023a. URL `https://arxiv.org/abs/2305.17277`.

Deng, C., Bello, K., Ravikumar, P., and Aragam, B. Global optimality in bivariate gradient-based DAG learning. *Advances in Neural Information Processing Systems*, 36:17929–17968, 2023b.

Faria, G., Martins, A., and Figueiredo, M. Differentiable causal discovery under latent interventions. In *Conference on Causal Learning and Reasoning*, pp. 253–274. PMLR, 2022.

Gao, E., Ng, I., Gong, M., Shen, L., Huang, W., Liu, T., Zhang, K., and Bondell, H. MissDAG: Causal discovery in the presence of missing data with continuous additive noise models. *Advances in Neural Information Processing Systems*, 35:5024–5038, 2022a.

Gao, E., Chen, J., Shen, L., Liu, T., Gong, M., and Bondell, H. FedDAG: Federated DAG structure learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=MzWgBjZ6Le.

Gao, M., Tai, W. M., and Aragam, B. Optimal estimation of gaussian dag models, 2022b. URL https://arxiv.org/abs/2201.10548.

Gao, Y., Shen, L., and Xia, S.-T. DAG-GAN: Causal structure learning with generative adversarial nets. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3320–3324. IEEE, 2021.

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

Kalainathan, D., Goudet, O., Guyon, I., Lopez-Paz, D., and Sebag, M. Structural agnostic modeling: Adversarial learning of causal graphs. *Journal of Machine Learning Research*, 23(219):1–62, 2022.

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

Kuipers, J., Moffa, G., and Heckerman, D. Addendum on the scoring of Gaussian directed acyclic graphical models. *The Annals of Statistics*, 42(4):1689–1691, 2014.

Lam, W.-Y., Andrews, B., and Ramsey, J. Greedy relaxations of the sparsest permutation algorithm, 2022. URL https://arxiv.org/abs/2206.05421.

Lan, G. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133: 365–397, 2012. ISSN 1436-4646. doi: 10.1007/s10107-010-0434-y. URL https://doi.org/10.1007/s10107-010-0434-y.

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

Margaritis, D. and Thrun, S. Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems*, 12, 1999.

Meek, C. *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie Mellon University, 1997.

Morgan, S. and Winship, C. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009. doi: 10.1137/070704277. URL https://doi.org/10.1137/070704277.

Nemirovski, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983.

Ng, I. and Zhang, K. Towards federated Bayesian network structure learning with continuous optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 8095–8111. PMLR, 2022.

Ng, I., Ghassami, A., and Zhang, K. On the role of sparsity and DAG constraints for learning linear DAGs. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

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

Ng, I., Zhu, S., Fang, Z., Li, H., Chen, Z., and Wang, J. Masked gradient-based causal structure learning. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pp. 424–432. SIAM, 2022b.

Ng, I., Huang, B., and Zhang, K. Structure learning with continuous optimization: A sober look and beyond. In *Causal Learning and Reasoning*, pp. 71–105. PMLR, 2024.

Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Georgatzis, K., Beaumont, P., and Aragam, B. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pp. 1595–1605. Pmlr, 2020.

Pearl, J. *Causality*. Cambridge University Press, 2009.

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

Peters, J., Bühlmann, P., and Meinshausen, N. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Polyak, B. T. A new method of stochastic approximation type. *Avtomatika i Telemekhanika*, 51: 98–107, 1990.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 1992. doi: 10.1137/0330046. URL https://doi.org/10.1137/0330046.

Ramsey, J., Zhang, J., and Spirtes, P. L. Adjacency-faithfulness and conservative causal inference, 2012. URL https://arxiv.org/abs/1206.6843.

Raskutti, G. and Uhler, C. Learning directed acyclic graphs based on sparsest permutations, 2019. URL https://arxiv.org/abs/1307.0366.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951. ISSN 0003-4851. URL http://www.jstor.org/stable/2236626.

Rodomanov, A., Kavis, A., Wu, Y., Antonakopoulos, K., and Cevher, V. Universal gradient methods for stochastic convex optimization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Wnhp34K5jR.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D., and Nolan, G. Causal protein-signaling networks derived from multiparameter single-cell data. *Science (New York, N.Y.)*, 308(5721):523—529, April 2005a. ISSN 0036-8075. doi: 10.1126/science.1105809. URL https://doi.org/10.1126/science.1105809.

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

Sauer, A. and Geiger, A. Counterfactual generative networks. *arXiv preprint arXiv:2101.06046*, 2021.

Scheines, R., Spirtes, P., Glymour, C., Meek, C., and Richardson, T. The tetrad project: Constraint based aids to causal model specification. *Multivariate Behavioral Research*, 33(1):65–117, 1998.

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

Spirtes, P., Meek, C., and Richardson, T. Causal inference in the presence of latent variables and selection bias. In *Conference on Uncertainty in Artificial Intelligence*, 1995. URL https://api.semanticscholar.org/CorpusID:11987717.

Spirtes, P., Glymour, C., Scheines, R., and Heckerman, D. *Causation, prediction, and search*. MIT Press, 2000.

Squires, C., Amaniampong, J., and Uhler, C. Efficient permutation discovery in causal dags, 2020. URL https://arxiv.org/abs/2011.03610.

Stephens, M. and Balding, D. Bayesian statistical methods for genetic association studies. *Nature Reviews Genetics*, 10(10):681–690, 2009.

Sun, X., Schulte, O., Liu, G., and Poupart, P. NTS-NOTEARS: Learning nonparametric DBNS with prior knowledge. *arXiv preprint arXiv:2109.04286*, 2021.

Tsamardinos, I., Aliferis, C., Statnikov, A., and Statnikov, E. Algorithms for large scale Markov blanket discovery. In *FLAIRS*, volume 2, pp. 376–81, 2003.

Vowels, M., Camgoz, N., and Bowden, R. D'ya like DAGs? A survey on structure learning and causal discovery. *ACM Computing Surveys*, 55(4):1–36, 2022.

Wang, Y., Menkovski, V., Wang, H., Du, X., and Pechenizkiy, M. Causal discovery from incomplete data: A deep learning approach. *arXiv preprint arXiv:2001.05343*, 2020.

Wei, D., Gao, T., and Yu, Y. Dags with no fears: A closer look at continuous optimization for learning bayesian networks, 2020. URL `https://arxiv.org/abs/2010.09133`.

Yang, M., Liu, F., Chen, Z., Shen, X., Hao, J., and Wang, J. CausalVAE: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9593–9602, 2021.

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

Yu, Y., Gao, T., Yin, N., and Ji, Q. Dags with no curl: An efficient dag structure learning approach, 2021. URL `https://arxiv.org/abs/2106.07197`.

Zeng, Y., Shimizu, S., Cai, R., Xie, F., Yamamoto, M., and Hao, Z. Causal discovery with multi-domain LiNGAM for latent factors. In *Causal Analysis Workshop Series*, pp. 1–4. PMLR, 2021.

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

Zhang, Z., Ng, I., Gong, D., Liu, Y., Abbasnejad, E., Gong, M., Zhang, K., and Shi, J. Q. Truncated matrix power iteration for differentiable DAG learning. *Advances in Neural Information Processing Systems*, 35:18390–18402, 2022.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. DAGs with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Contents

A Related Work

A significant body of research on DAG learning revolves around non-convex continuous optimization frameworks, such as NOTEARS (Zheng et al., 2018), GOLEM (Ng et al., 2020), NOCURL (Yu et al., 2021), and DAGMA (Bello et al., 2022). These approaches address the DAG constraint using either smooth approximations or novel penalty functions, but they are often computationally expensive and lack scalability.

Zheng et al. (2018) addressed the constrained optimization problem

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times d}} \ell(\mathbf{W};\mathbf{X})_{\mathsf{NOTEARS}} \overset{def}{=} \frac{1}{2n}\|\mathbf{X}-\mathbf{X}\mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_1 \quad \text{subject to} \quad h(\mathbf{W}) = 0, \tag{11}$$

where $\ell(\mathbf{W};\mathbf{X})$ represents the least squares objective and $h(\mathbf{W}) := \text{tr}(e^{\mathbf{W}\odot\mathbf{W}}) - d$ enforces the DAG constraint. Additionally, an ℓ_1 regularization term $\lambda\|\mathbf{W}\|_1$, where $\|\cdot\|_1$ is the element-wise ℓ_1-norm and λ is a hyperparameter incorporated into the objective function. This formulation addresses the linear case with equal noise variances, as discussed in Loh & Bühlmann (2014) and Peters & Bühlmann (2014). This constrained optimization problem is solved using the augmented Lagrangian method (Bertsekas et al., 1999), followed by thresholding the obtained edge weights. However, since this approach computes the acyclicity function via the matrix exponential, each iteration incurs a computational complexity of $\mathcal{O}(d^3)$, which significantly limits the scalability of the method.

Ng et al. (2020) introduced the GOLEM method, which enhances the scoring function by incorporating an additional log-determinant term, $\log|\det(\mathbf{I}-\mathbf{W})|$ to align with the Gaussian log-likelihood,

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times d}} \ell(\mathbf{W};\mathbf{X})_{\mathsf{GOLEM}} \overset{def}{=} \frac{d}{2}\log\|\mathbf{X}-\mathbf{X}\mathbf{W}\|_F^2 - \log|\det(\mathbf{I}-\mathbf{W})| + \lambda_1\|\mathbf{W}\|_1 + \lambda_2 h(\mathbf{W}), \tag{12}$$

where λ_1 and λ_2 serve as regularization hyperparameters within the objective function. Although the newly added log-determinant term is zero when the current model $\mathbf{W}$ is a DAG, this score function does not provide an exact characterization of acyclicity. Specifically, the condition $\log|\det(\mathbf{I}-\mathbf{W})| = 0$ does not imply that $\mathbf{W}$ represents a DAG.

Bello et al. (2022) introduces a novel acyclicity characterization for DAGs using a log-determinant function,

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times d}} \ell(\mathbf{W};\mathbf{X})_{\mathsf{DAGMA}} \overset{def}{=} \frac{1}{2n}\|\mathbf{X}-\mathbf{X}\mathbf{W}\|_F^2 + \lambda_1\|\mathbf{W}\|_1 \quad \text{subject to} \quad h^s_{ldet}(\mathbf{W}) = 0, \tag{13}$$

where $h^s_{ldet}(\mathbf{W}) \overset{def}{=} -\log\det(s\mathbf{I}-\mathbf{W}\circ\mathbf{W}) + d\log s$, and it is both exact and differentiable.

In practice, the augmented Lagrangian method enforces the hard DAG constraint by increasing the penalty coefficient toward infinity, which requires careful parameter fine-tuning and can lead to numerical difficulties and ill-conditioning (Birgin et al., 2005; Ng et al., 2022a). As a result, existing methods face challenges in several aspects of optimization, including careful selection of constraints, high computational complexity, and scalability issues.

Yu et al. (2021) introduce a novel formulation for DAG structure learning by expressing the weighted adjacency matrix as the Hadamard product of a skew-symmetric matrix and the gradient of a potential function on graph nodes. This representation avoids explicit acyclicity constraints and enables a continuous, constraint-free optimization framework. However, although NOCURL avoids direct constraints through this parameterization and Hodge decomposition, it still relies on repeated L-BFGS (Broyden, 1967) optimization steps , which can become computationally expensive for large graphs. In contrast, ψDAG avoids both acyclicity constraints and expensive optimization procedures, allowing for more efficient scaling in high-dimensional settings.

Other works, such as Chen et al. (2019), proposed variance ordering procedures for estimating topological orderings under equal error variances. Although these methods naturally extend to high-dimensional settings, their reliance on controlling the maximum in-degree of the graph becomes computationally intensive as graph density increases. In contrast, ψDAG avoids these assumptions and demonstrates scalability on graphs with up to $10,000$ nodes. Gao et al. (2022b) focused on theoretical guarantees for Gaussian DAG models, obtaining minimax optimal bounds for structural recovery. Although their work offers valuable insights into sample efficiency, it does not address the computational challenges of large-scale DAG learning.

Wei et al. (2020) examined optimization challenges in NOTEARS by analyzing the KKT conditions and proposed the KKTS algorithm as a post-processing enhancement. While this method improves the structural Hamming distance (SHD), its reliance on specific constraints and post-hoc refinements limits its applicability. In contrast, ψDAG reformulates DAG learning as a stochastic optimization problem, seamlessly integrating gradient-based methods for large-scale graphs.

Additionally, Deng et al. (2023a) introduced a bilevel algorithm that iteratively refines topological orders through node swaps, achieving local minima or KKT points. However, this approach is

constrained by a specific function $h(B) = \sum_{i=1}^{d} c_i \mathrm{Tr}(B^i)$, which is computationally expensive and limits its scalability to applications that involve larger graphs. Consequently, their experiments are restricted to synthetic datasets with graphs containing up to $d = 100$ nodes. Moreover, the algorithm initializes the $\mathbf{W}$ matrix using linear regression coefficients in the least squares case, resulting in a different starting point for optimization, which makes direct comparisons with other methods challenging. Our method addresses these limitations by generalizing the DAG learning framework and demonstrating superior scalability and performance on both synthetic and real datasets.

Compared to permutation-based methods such as SP (Raskutti & Uhler, 2019), Efficient Permutation Discovery (Squires et al., 2020), and GRaSP (Lam et al., 2022), ψDAG avoids exhaustive or greedy searches over permutations. Instead, it leverages a novel projection technique that efficiently infers causal orders without the computational overhead associated with permutation-based algorithms. This design choice allows ψDAG to maintain accuracy while offering superior scalability and efficiency in learning DAG structures.

While many of these works focus on specific assumptions, penalty terms, or theoretical guarantees, our framework prioritizes scalability, flexibility, and applicability. To overcome these challenges, we propose a novel framework for enforcing the acyclicity constraint, utilizing a low-cost projection method. This approach significantly reduces iteration complexity and eliminates the need for expensive hyperparameter tuning.

B Theoretical Results

In this section, we present some theoretical properties of the DAG set and analyze the convergence of the proposed method.

Lemma 2. *The DAG set $\mathbb{D}$ is a conic set. Specifically, for any $\mathbf{W} \in \mathbb{D}$ and $\alpha \geq 0$, we have $\alpha\mathbf{W} \in \mathbb{D}$. Additionally, the DAG set $\mathbb{D}$ includes the entire line, meaning that for any $\mathbf{W} \in \mathbb{D}$ and $\alpha \in \mathbb{R}$, $\alpha\mathbf{W} \in \mathbb{D}$.*

Proof. We begin by observing that $\mathbf{0} \in \mathbb{D}$, as a graph with no edges is trivially a DAG. Next, consider any $\mathbf{W} \in \mathbb{D}$ and $\alpha \in \mathbb{R} \setminus \{0\}$. Scaling $\mathbf{W}$ by α does not alter the structure of the graph; it only changes the edge weights. Since the graph remains acyclic, $\alpha\mathbf{W} \in \mathbb{D}$. Thus, the DAG set $\mathbb{D}$ satisfies the stated properties. □

Now, let us move to the subsets of DAG, which are based on a topological ordering π.

Definition 3. *A topological ordering π of a directed graph is a linear ordering of its vertices such that, for every directed edge (u, v) from vertex u to vertex v, u comes before v in the ordering. We call $Ord(\mathbf{W})$ a set of all possible topological orderings for DAG $\mathbf{W}$ and $ord(\mathbf{W})$ is one of the orderings.*

For the graphs with d vertices, there are exactly $d!$ distinct topological orderings.
Every topological ordering π corresponds to subspace of all DAGs which can have this topological ordering, we call it π-subspace DAG.

Definition 4. *A π-subspace $\mathbb{D}_\pi$ is a set of all DAGs $\mathbf{W}$ such that $\pi \in Ord(\mathbf{W})$.*

Let us prove that π-subspace $\mathbb{D}_\pi$ is a linear subspace.

Lemma 5. *$\mathbb{D}_\pi$ is a linear subspace, meaning for any $\mathbf{W}_1 \in \mathbb{D}_\pi, \mathbf{W}_2 \in \mathbb{D}_\pi$, $\alpha \in \mathbb{R}, \beta \in \mathbb{R}$, $\mathbf{W} = \alpha\mathbf{W}_1 + \beta\mathbf{W}_2 \in \mathbb{D}_\pi$.*

Proof. We should simply note that any non-zero value in $\mathbf{W}_1$ corresponds to an edge between vertices u and v such that v is after u in the ordering π. The same holds for $\mathbf{W}_2$. Hence, any non-zero value in $\mathbf{W}$ holds the ordering π. □

Next, we highlight that the DAG set $\mathbb{D}$ is a union of π-subspaces for all possible orderings π.

Lemma 6. *The DAG set $\mathbb{D}$ is a union of all π-subspaces.*

$$\mathbb{D} = \cup_\pi \mathbb{D}_\pi.$$

Proof. For any DAG $\mathbf{W} \in \mathbb{D}$ there exists a topological ordering π, hence $\mathbf{W} \in \mathbb{D}_\pi \in \cup_\pi \mathbb{D}_\pi$. On the other side, all elements of $\cup_\pi \mathbb{D}_\pi$ are DAGs by definition and belongs to $\mathbb{D}$.

□

Now, we move to the proposed method.

Theorem 7. *For an L_1-smooth function $F(\mathbf{W}) = \mathbb{E}_{x\sim\mathcal{D}}\left[l(\mathbf{W}; x)\right]$ restricted in a domain of radius R, $\|x - y\| \leq R$, $\forall x, y \in$ dom F, consider $\mathcal{A}_2$ in the Algorithm 1 be chosen as Universal Stochastic Gradient Method (Rodomanov et al., 2024). Running $\mathcal{A}_2$ for T SGD-type steps accessing σ_1-stochastic gradients (Theorem 1) in the π-subspace $\mathbb{D}_\pi$ converges to a minimum of problem* (9) *with additional subspace constraints at the rate*

$$\mathbb{E}\left[F(\mathbf{W}_T) - \underset{\substack{\mathbf{W}\in\mathbb{D}_\pi,\\ ord(\mathbf{W})=\pi}}{\arg\min}\ F(\mathbf{W})\right] \leq \mathcal{O}\left(\frac{\sigma_1 R}{\sqrt{T}} + \frac{L_1 R^2}{T}\right).$$

Proof. A direct consequence of the convergence guarantees of the Universal Stochastic Gradient Method, Theorem 4.2 of Rodomanov et al. (2024). □

Theorem 8. *For an L_1-smooth function $F(\mathbf{W}) = \mathbb{E}_{x\sim\mathcal{D}}\left[l(\mathbf{W}; x)\right]$ restricted in a domain of radius R, $\|x - y\| \leq R$, $\forall x, y \in$ dom F, Algorithm 2 with Universal Stochastic Gradient Method (Rodomanov et al., 2024) as $\mathcal{A}_1$ and $\mathcal{A}_2$ with converges to a local minimum of problem* (9)*.*

Proof. We denote a subspace minimum of problem (9) as $\mathbf{W}^*_{\pi_k} = \arg\min_{\substack{\mathbf{W}\in\mathbb{D}_\pi,\\ ord(\mathbf{W})=\pi}} F(\mathbf{W})$. There are two cases. The first case, $\mathbf{W}_{\pi^*_k}$ is a local minimum of a general problem (9). Then, by Theorem 7, the method converges to the local minimum. The second case, the subspace minimum $\mathbf{W}^*_{\pi_k}$ is not a local minimum as there exists an orthogonal subspace $\mathbb{D}_{\tilde{\pi}} \perp \mathbb{D}_\pi$ such that $\mathbf{W}^*_{\pi_k} \in \mathbb{D}_{\tilde{\pi}}$ and $F(\mathbf{W}^*_{\pi_k}) > F(\mathbf{W}^*_{\tilde{\pi}})$. By steps from $\mathcal{A}_1^{k+1}$, the method decreases towards $F(\mathbf{W}^*_{\tilde{\pi}})$. To fully guarantee the convergences, one can memorize the visited subspaces and forbid projecting on them. Then, $\mathbb{D}_{\pi_{k+1}}$ is not visited subspace. □

C Detailed Experiment Description

Computing. Our experiments were carried out on a machine equipped with 80 CPUs and one NVIDIA Quadro RTX A6000 48GB GPU. Each experiment was allotted a maximum wall time of 36 hours as in DAGMA Bello et al. (2022).

Graph Models. In our experimental simulations, we generate graphs using two established random graph models:

- **Erdős-Rényi (ER) graphs:** These graphs are constructed by independently adding edges between nodes with a uniform probability. We denote these graphs as ER_k, where kd represents the expected number of edges.
- **Scale-Free (SF) graphs**: These graphs follow the preferential attachment process as described in Barabási & Albert (1999). We use the notation SF_k to indicate a scale-free graph with expected kd edges and an attachment exponent of $\beta = 1$, consistent with the preferential attachment process. Since we focus on directed graphs, this model corresponds to Price's model, a traditional framework used to model the growth of citation networks.

It is important to note that ER graphs are inherently undirected. To transform them into Directed Acyclic Graphs (DAGs), we generate a random permutation of the vertex labels from 1 to d, then orient the edges according to this ordering. For SF graphs, edges are directed as new nodes are added, ensuring that the resulting graph is a DAG. After generating the ground-truth DAG, we simulate the structural equation model (SEM) for linear cases, conducting experiments accordingly.

Metrics. The performance of each algorithm is assessed using the following four key metrics:

- **Structural Hamming Distance (SHD):** A widely used metric in structure learning that quantifies the number of edge modifications (additions, deletions, and reversals) required to transform the estimated graph into the true graph.
- **True Positive Rate (TPR):** This metric calculates the proportion of correctly identified edges relative to the total number of edges in the ground-truth DAG. It is also known as **recall**.
- **Precision:** is the proportion of all the model's positive classifications that are actually positive.
- **F1 Score:** is the harmonic mean of precision and recall.
- **False Positive Rate (FPR):** This measures the proportion of incorrectly identified edges relative to the total number of absent edges in the ground-truth DAG.
- **Runtime:** The time taken by each algorithm to complete its execution provides a direct measure of the algorithm's computational efficiency.
- **Stochastic gradient computations:** Number of gradient computed.

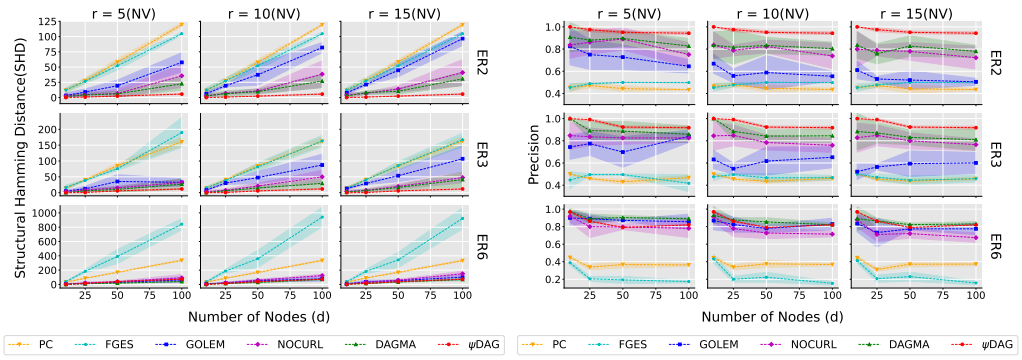
Linear SEM. In the linear case, the functions are directly parameterized by the weighted adjacency matrix W. Specifically, the system of equations is given by $X_i = \mathbf{X}\mathbf{W}_i + N_i$, where $\mathbf{W} = [\mathbf{W}_1|\cdots|\mathbf{W}_d] \in \mathbb{R}^{d\times d}$, and $N_i \in \mathbb{R}$ represents the noise. The matrix $\mathbf{W}$ encodes the graphical structure, meaning there is an edge $X_j \to X_i$ if and only if $W_{j,i} \neq 0$. Starting with a ground-truth DAG $B \in \{0,1\}^{d\times d}$ obtained from one of the two graph models, either ER or SF, edge weights were sampled independently from Unif$[-2,-0.5] \cup [0.5,2]$ to produce a weight matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$. Using this matrix $\mathbf{W}$, the data $X = X\mathbf{W} + N$ was sampled under the following three noise models:

- **Gaussian noise:** $N_i \sim N(0,1)$ for all $i \in [d]$,
- **Exponential noise:** $N_i \sim \text{Exp}(1)$ for all $i \in [d]$,
- **Gumbel noise:** $N_i \sim \text{Gumbel}(0,1)$ for all $i \in [d]$.

Using these noise models, random datasets $X \in \mathbb{R}^{n\times d}$ were generated by independently sampling the rows according to one of the models described above. Unless otherwise specified, we generate the same number of samples $n \in \{5000, 10000\}$ for training and validation datasets, respectively.

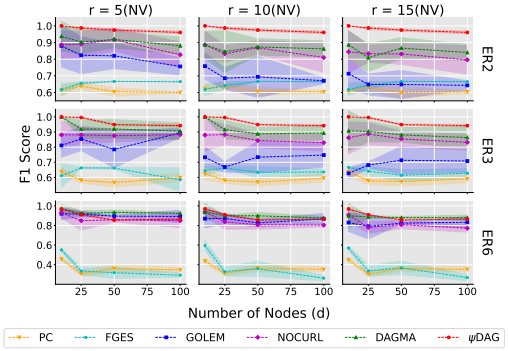
The implementation details of the baseline methods are as follows:

- FGES **(Meek, 1997; Chickering, 2002)** using the FGES algorithm found in the py-tetrad package Scheines et al. (1998) in `https://github.com/cmu-phil/py-tetrad`.
- PC **(Spirtes & Glymour, 1991; Ramsey et al., 2012)** using the PC algorithm from the py-tetrad package Scheines et al. (1998) in `https://github.com/cmu-phil/py-tetrad`.
- NOTEARS **(Zheng et al., 2018)** using the authors' publicly available Python code, which can be found at `https://github.com/xunzheng/notears`. This method employs a least squares score function, and we used their default set of hyperparameters without modification. We used the default choice of $\lambda = 0.1$ as in authors' code.
- GOLEM **(Ng et al., 2020)** using the authors' Python code, available at `https://github.com/ignavierng/golem`, along with their PyTorch version at `https://github.com/huawei-noah/trustworthyAI/blob/master/gcastle/castle/algorithms/gradient/notears/torch/golem_utils/golem_model.py`. We adopted the default hyperparameter settings, specifically $\lambda_1 = 0.02$ and $\lambda_2 = 5$. Additional details of GOLEM are listed in Ng et al. (2020)(Appendix F).
- NOCURL **(Yu et al., 2021)** using the authors' publicly available code at `https://github.com/fishmoon1234/DAG-NoCurl`. We used the default choice of hyperparameters.
- DAGMA **(Bello et al., 2022)** using the authors' Python code, which is available at `https://github.com/kevinsbello/dagma`. We used the default choice of hyperparameters.

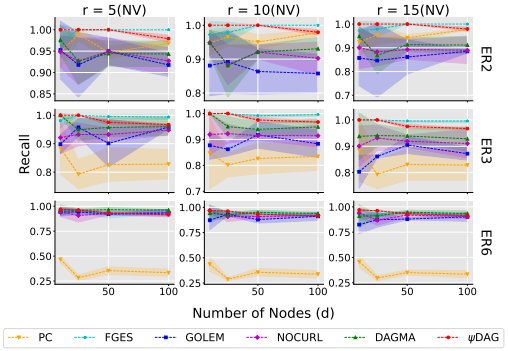


(a) SHD

(b) Precision



(c) F1 Score



(d) Recall (TPR)

Figure 6: Structure recovery performance under Gaussian noise with non-equal variances (NV) across varying noise ratios ($r \in \{5, 10, 15\}$). Rows correspond to different random graph types, and columns represent increasing noise ratios. Metrics reported include (from left to right): Structural Hamming Distance (SHD, lower is better), Precision, F1 score, and Recall (or TPR) (all higher is better). Each method's mean performance is shown, with standard error indicated by shaded regions around the curves.

Thresholding. Following the approach taken in previous studies, including the baseline methods (Zheng et al., 2018; Ng et al., 2020; Yu et al., 2021; Bello et al., 2022), for all the methods, we apply a final thresholding step of 0.3 to effectively reduce the number of false discoveries.

D Supplementary Experiments Results

D.1 Structure Recovery Performance

In addition to the results presented in the main paper (Section 5.2), we include extended evaluations on ER6 graphs under Gaussian noise with non-equal variances (NV). As illustrated in Figure 6, ψDAG consistently achieves the best performance in most metrics, including the Structural Hamming distance (SHD), precision, and F1 score, for all noise levels and graph sizes. In particular, while ψDAG slightly trails FGES in recall (or TPR) for graphs with $d > 50$, FGES exhibits a significantly worse SHD, precision, and F1 score in those regimes, suggesting that it produces denser graphs with more false positives. This reinforces that ψDAG not only maintains structural accuracy, but also avoids overfitting, particularly in complex, high-noise environments. These results highlight the robustness of our approach across graph densities and noise heterogeneity.

We present a comparative analysis of structure recovery and runtime performance under Gaussian noise with equal variances across two random graph types: ER2 and ER3. Figure 7 illustrates Structural Hamming Distance (SHD) and runtime (in seconds) across varying node sizes $d \in \{25, 50, 100, 500\}$. Compared methods include constraint-based (PC), score-based (FGES), and gradient-based approaches (NOTEARS, GOLEM, NOCURL, DAGMA, and ψDAG). As the number

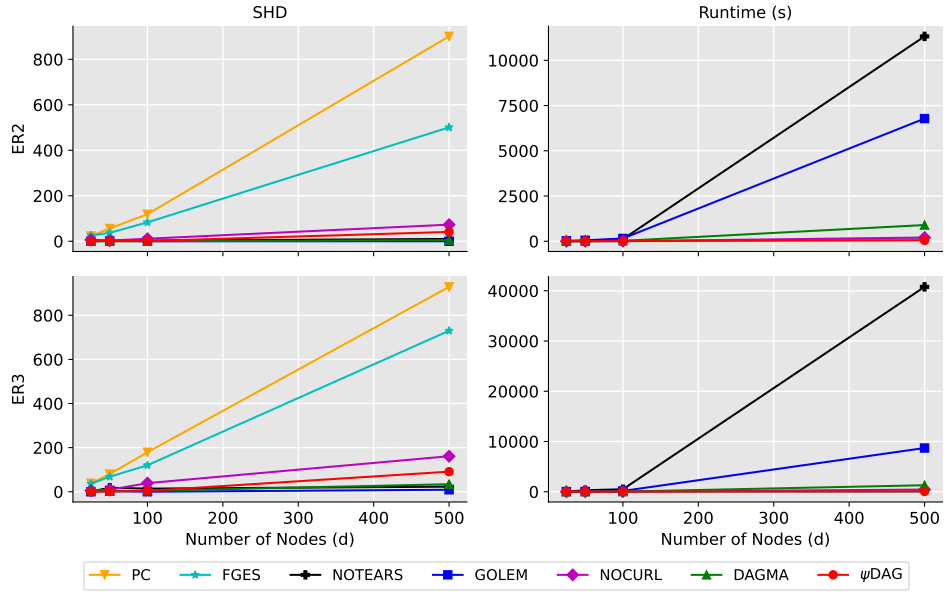
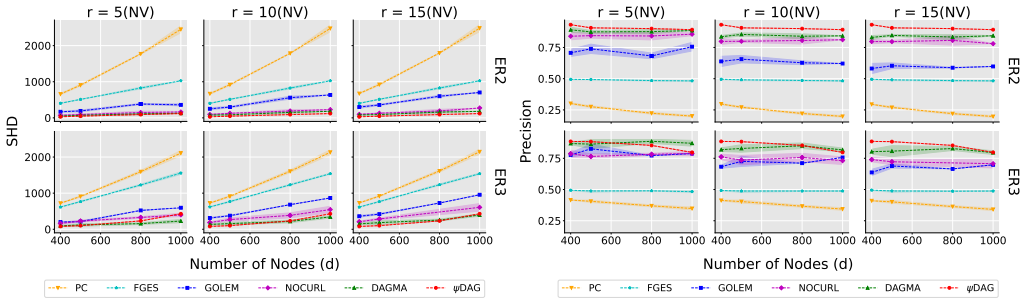
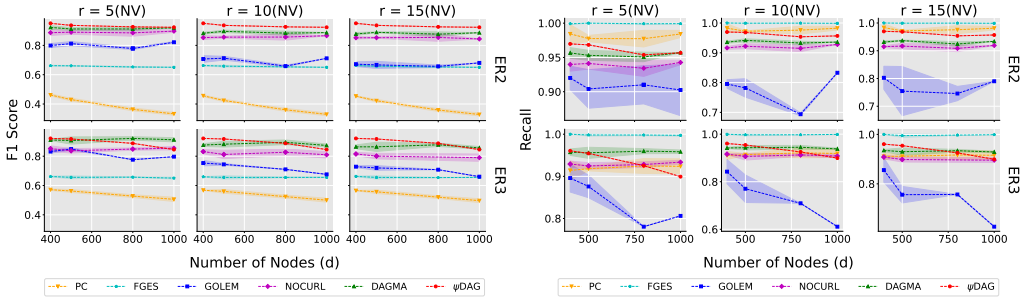


Figure 7: Comparison of Structural Hamming Distance (SHD) and Runtime (in seconds) across constraint-based (PC), score-based (FGES) and gradient-based approaches (NOTEARS, GOLEM, NOCURL, DAGMA and ψDAG) on ER2 and ER3 Gaussian-EV random graphs. Lower SHD indicates better structural accuracy; lower runtime indicates greater efficiency.



(a) SHD

(b) Precision



(c) F1 Score

(d) Recall (TPR)

Figure 8: Structure recovery performance under Gaussian noise with non-equal variances (NV) across varying noise ratios ($r \in \{5, 10, 15\}$). Rows correspond to ER2 and ER3 graphs with $d \in \{400, 500, 800, 1000\}$, and columns represent increasing noise ratios. Metrics reported include (from left to right): Structural Hamming Distance (SHD, lower is better), Precision, F1 score, and Recall (or TPR) (all higher is better). Each method's mean performance is shown, with standard error indicated by shaded regions around the curves.

of nodes increases, PC and FGES exhibit a marked rise in SHD, indicating limited scalability in structural accuracy. NOTEARS and GOLEM, while competitive on small scales, incur significantly higher runtime as d grows, making them less practical for large graphs. In contrast, ψDAG maintains a low SHD and a consistently competitive runtime, demonstrating both scalability and robustness in high-dimensional settings.

To evaluate structure recovery under more challenging conditions, we include additional results on large-scale ER2 and ER3 graphs ($d \in \{400, 500, 800, 1000\}$) with Gaussian noise exhibiting non-equal variances (NV) across varying noise ratios. As shown in Figure 8, ψDAG consistently achieves the best or near-best performance across most metrics, demonstrating robustness even at a large scale.

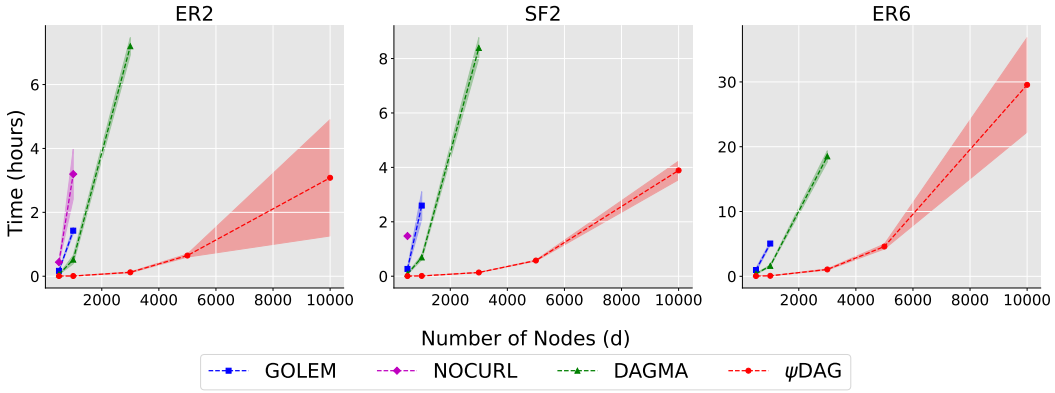


Figure 9: Runtime comparison of ψDAG, GOLEM, NOCURL, and DAGMA on ER2, SF2 and ER6 graphs with $d \in \{500, 1000, 3000, 5000, 10000\}$. The noise distribution is Gaussian with equal variances. ψDAG scales efficiently to 10,000 nodes, while the baselines exhibit sharp runtime increases or fail to complete within the 36-hour time budget. Notably, GOLEM, NOCURL, and DAGMA fail to converge or exceed the time limit in multiple settings, especially for ER6. All non-converging runs were excluded from the figures.

D.2 Scalability Comparison

We provide complementary scalability results to those reported in the main paper (Section 5.3). ψDAG scales efficiently to graphs with up to $d = 10{,}000$ nodes, as shown in Figure 9. In sparse settings such as ER2 and SF2, it converges within a few hours even for the largest graphs. In contrast, the runtime of GOLEM, NOCURL, and DAGMA increases sharply with graph size. On ER2 graphs, both GOLEM and NOCURL exceed the 36-hour runtime limit for $d \geq 3000$, while DAGMA fails to complete within the time budget for $d \geq 5000$.

We also observe frequent convergence failures among baselines. For instance, NOCURL fails to converge on SF2 graphs with Gaussian-EV noise when $d = 500$, and entirely fails on ER6. Both GOLEM and DAGMA exhibit non-convergence in at least one of the three random seeds on ER6. All such failed runs are excluded from reported statistics. In contrast, ψDAG converges in all runs and maintains competitive runtime performance even at large scales, highlighting both its robustness and practical efficiency.

We present results across combinations of the number of vertices $d \in \{10, 50, 100, 500, 1000, 3000, 5000, 10000\}$, graph types $\in \{\text{ER}, \text{SF}\}$, graph densities $k \in \{2, 4, 6\}$, and noise types (Gaussian, Exponential, and Gumbel). Figures are grouped by noise type and graph size, with subplots showing results for ψDAG, GOLEM, and DAGMA.

ER graph types: Figures 10 and 11 report results on ER2 graphs; Figures 12 and 13 on ER4 graphs; and Figure 14 on ER6 graphs.

SF graph types: Figures 15 and 16 report results on SF2 graphs; Figures 17 and 18 on SF4 graphs; and Figure 19 on SF6 graphs.

We report the decrease in functional value over both **(i)** elapsed time and **(ii)** number of gradient evaluations, the latter serving as a proxy for computational effort.

Figure 11b highlights that DAGMA requires substantially more gradient computations compared to both ψDAG and GOLEM, further emphasizing the efficiency of our approach.

D.3 Small to Moderate Number of Nodes

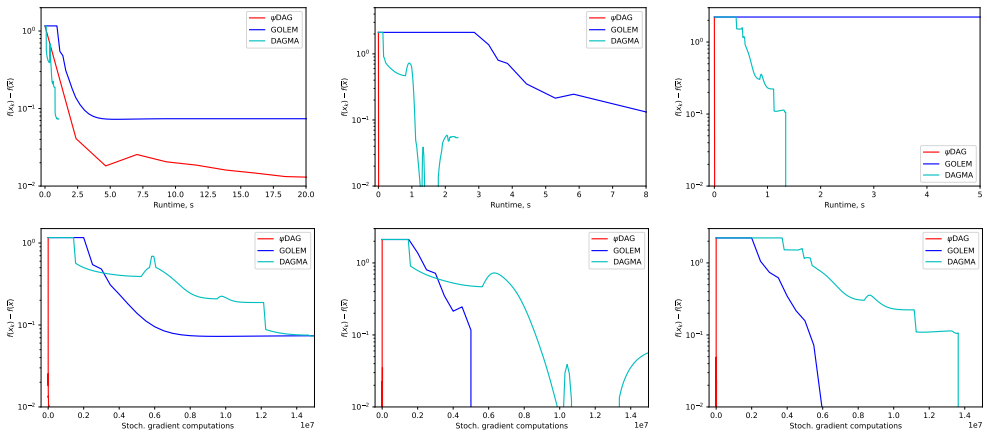
Our experiments demonstrate that while number of nodes is small, $d < 100$, GOLEM is more stable than DAGMA, and ψDAG method is the most stable. While DAGMA shows impressive speed for smaller node sets, the number of iterations required is still higher than both GOLEM and our method. Across all scenarios, ψDAG consistently demonstrates faster convergence compared to the other approaches, requiring fewer iterations to reach the desired solution.

D.4 Large Number of Nodes

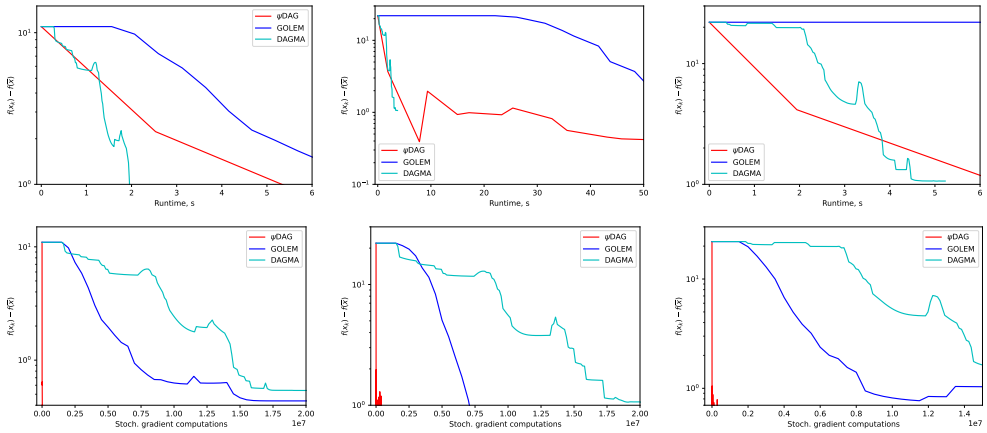
For graphs with a large number of nodes $d \in \{5000, 10000\}$, we were unable to run neither of the baselines, and consequently, Figure 20 includes only one algorithm. GOLEM was not feasible due to its computation time exceeding 350 hours. DAGMA was impossible as its runs led to kernel crashes. In all cases, we utilized a training set of 5,000 samples and a validation set of 10,000 samples.

D.5 Denser Graphs

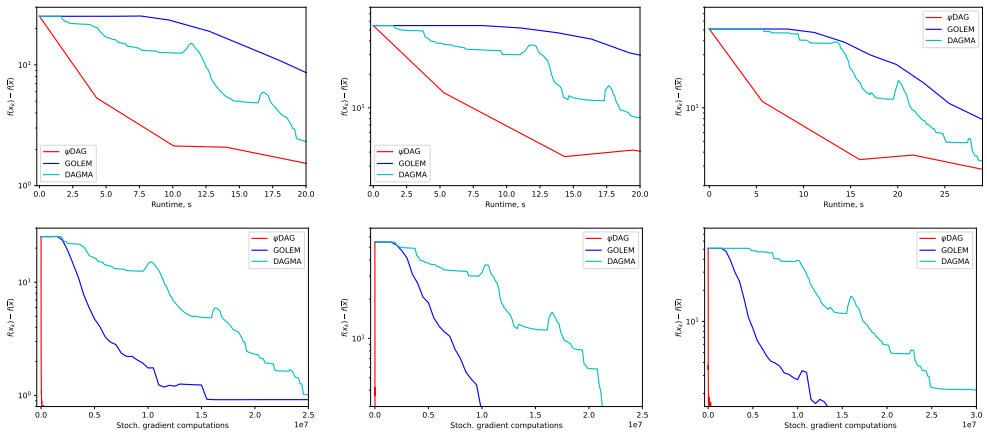
For a thorough comparison, in Figures 14 and 19, we compare graph structures ER6 and SF6 under the Gaussian noise type. Plots indicate that while DAGMA exhibits a fast runtime when the number of nodes is small, $d < 100$, it requires more iterations to achieve convergence. Algorithm ψDAG consistently outperforms GOLEM and DAGMA in both training time and a number of stochastic gradient computations, and the difference is more pronounced for a larger number of nodes and denser graphs.



(a) $d = 10$ vertices

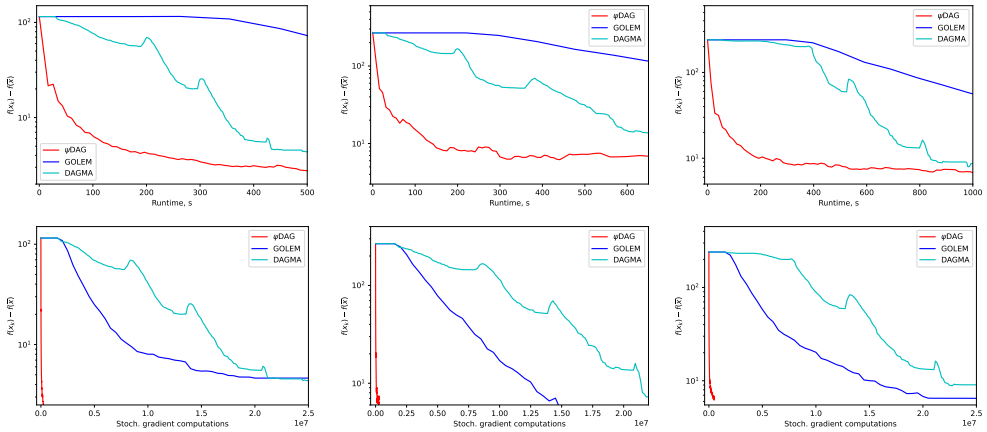


(b) $d = 50$ vertices

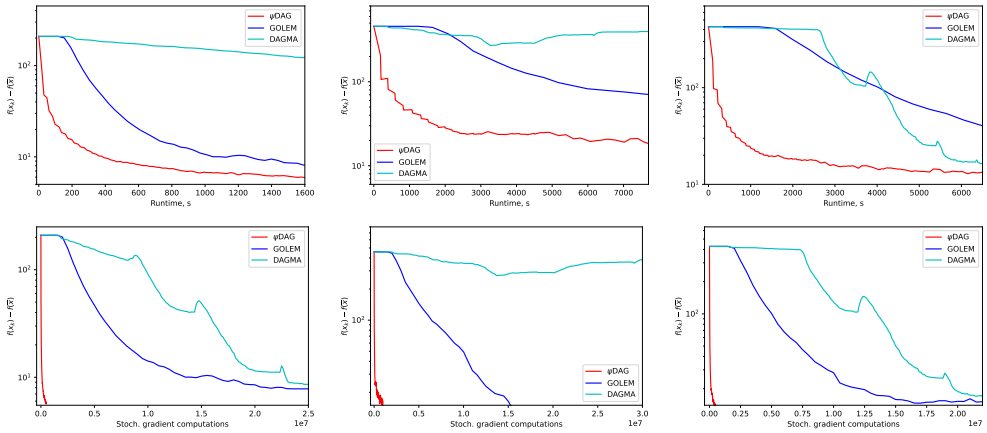


(c) $d = 100$ vertices

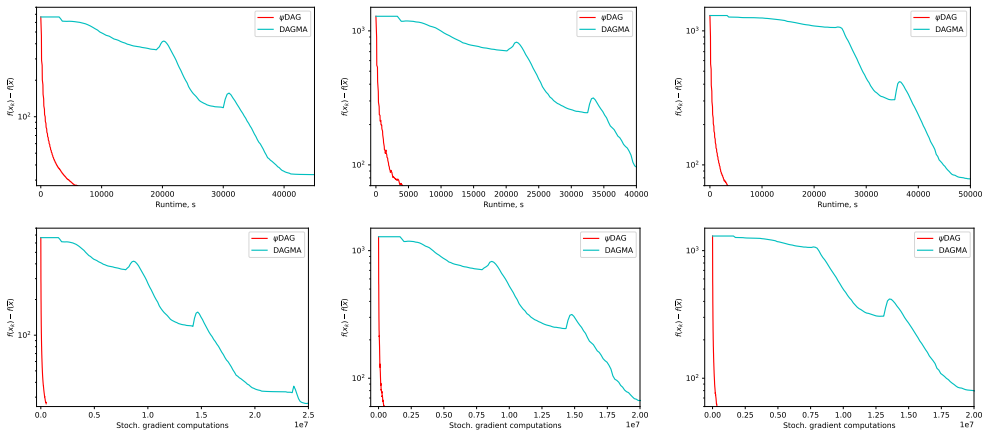
Figure 10: Linear SEM methods on ER2 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).



(a) $d = 500$ vertices

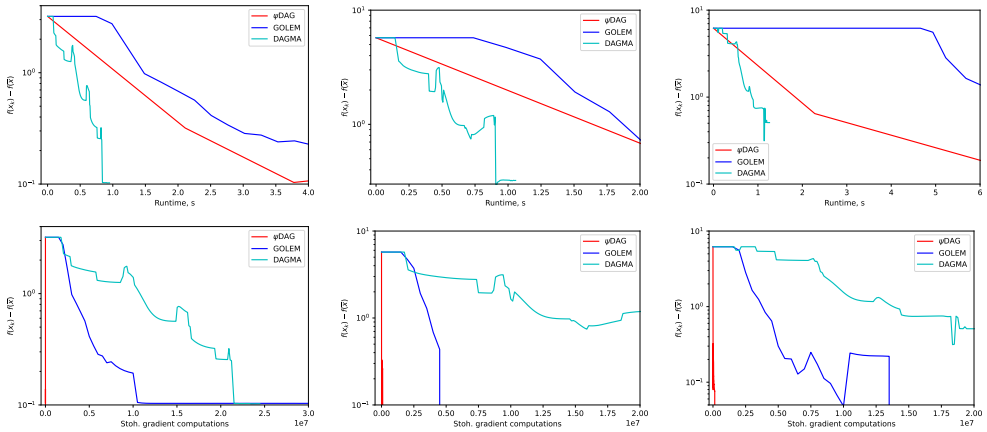


(b) $d = 1000$ vertices

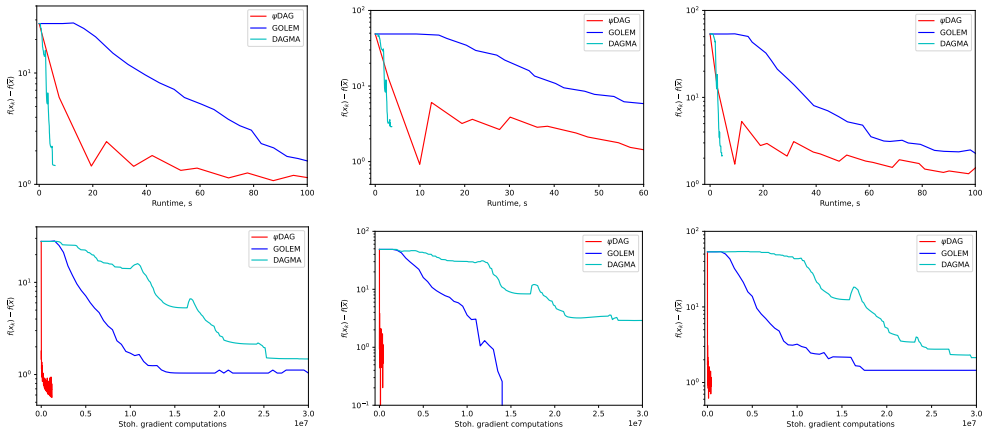


(c) $d = 3000$ vertices

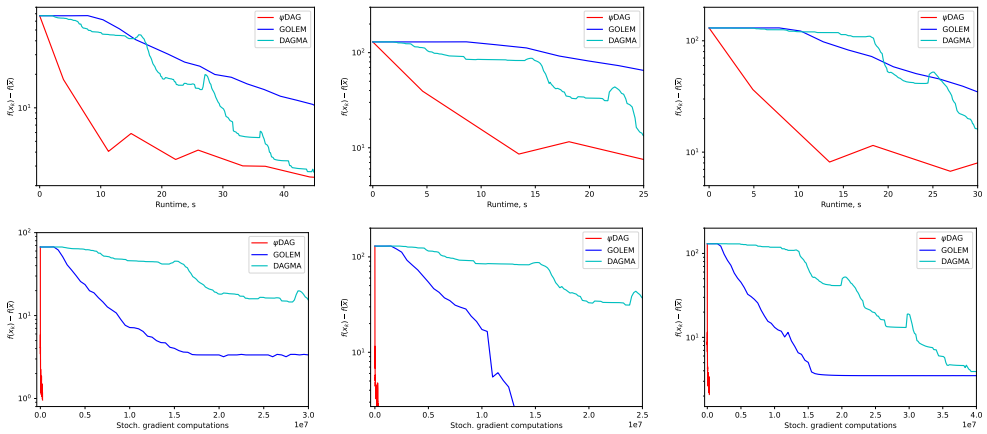
Figure 11: Linear SEM methods on ER2 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).



(a) $d = 10$ vertices

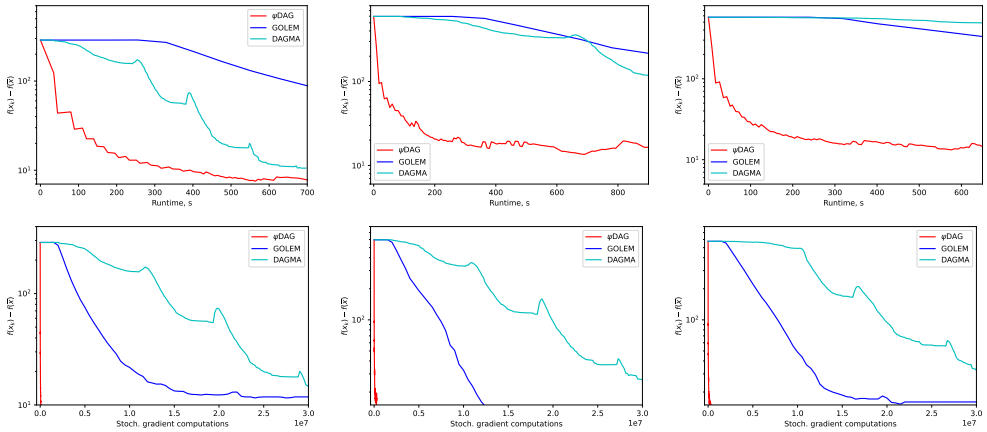


(b) $d = 50$ vertices

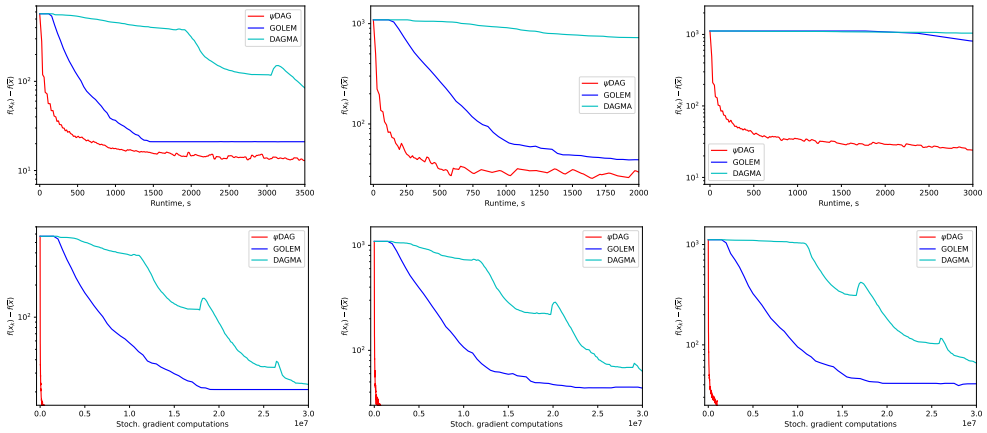


(c) $d = 100$ vertices

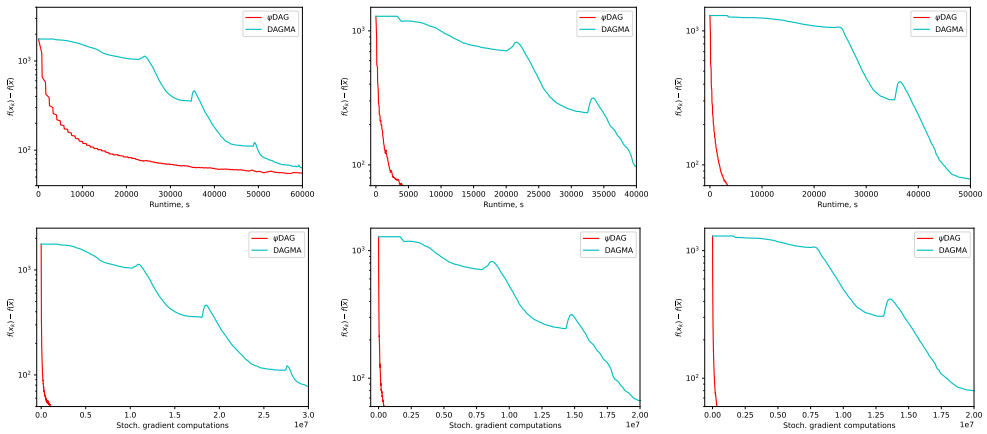
Figure 12: Linear SEM methods on ER4 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).



(a) $d = 500$ vertices

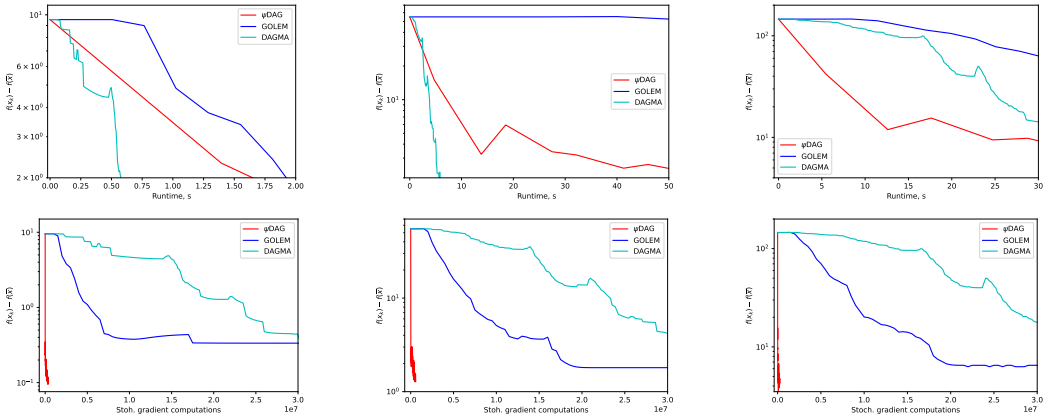


(b) $d = 1000$ vertices

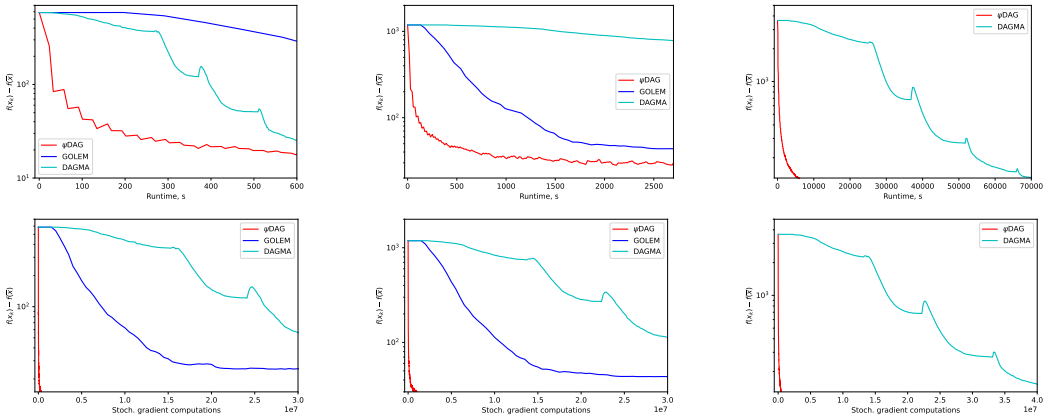


(c) $d = 3000$ vertices

Figure 13: Linear SEM methods on ER4 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).

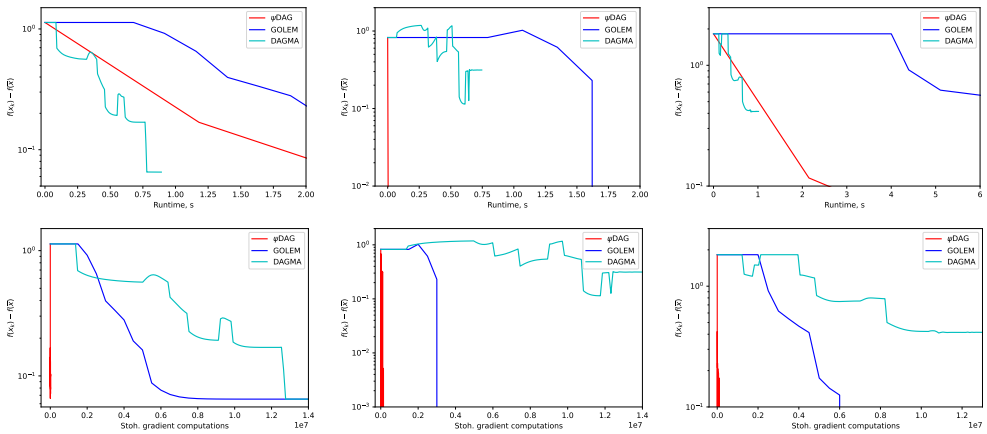


(a) $d = 10$ vertices (b) $d = 50$ vertices (c) $d = 100$ vertices

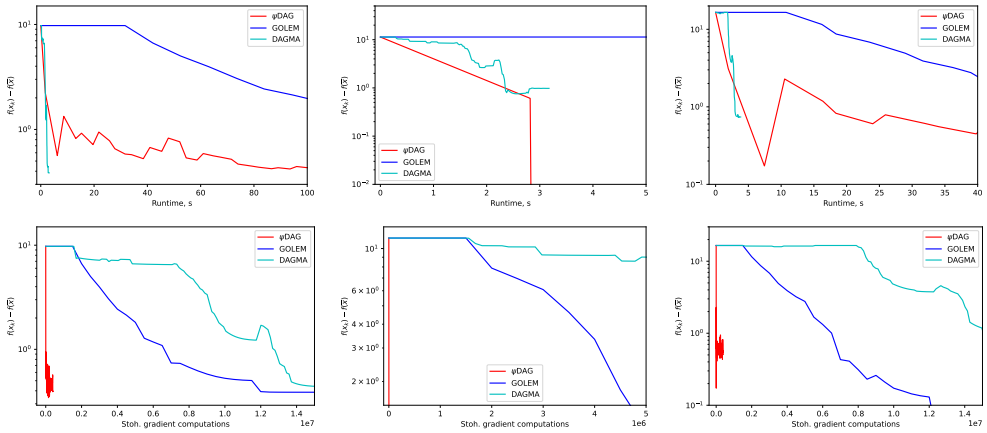


(d) $d = 500$ vertices (e) $d = 1000$ vertices (f) $d = 3000$ vertices

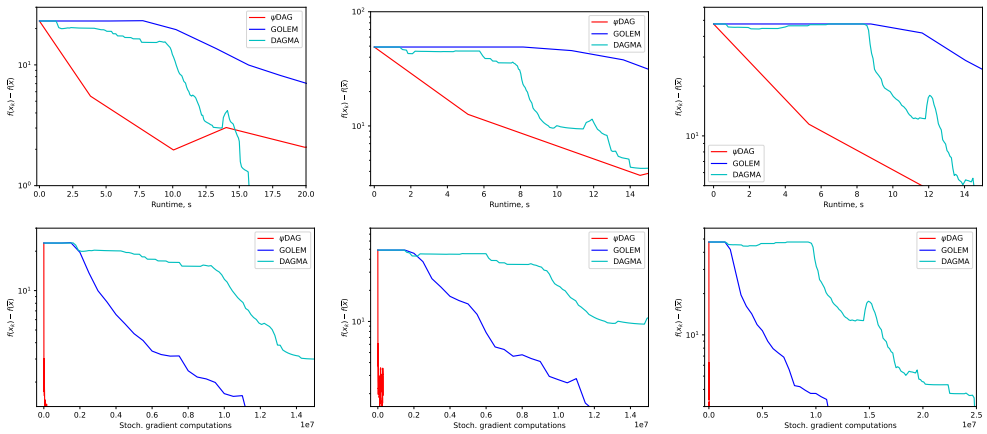
Figure 14: Linear SEM methods on graphs of type ER6 with the Gaussian noise distribution.



(a) $d = 10$ vertices

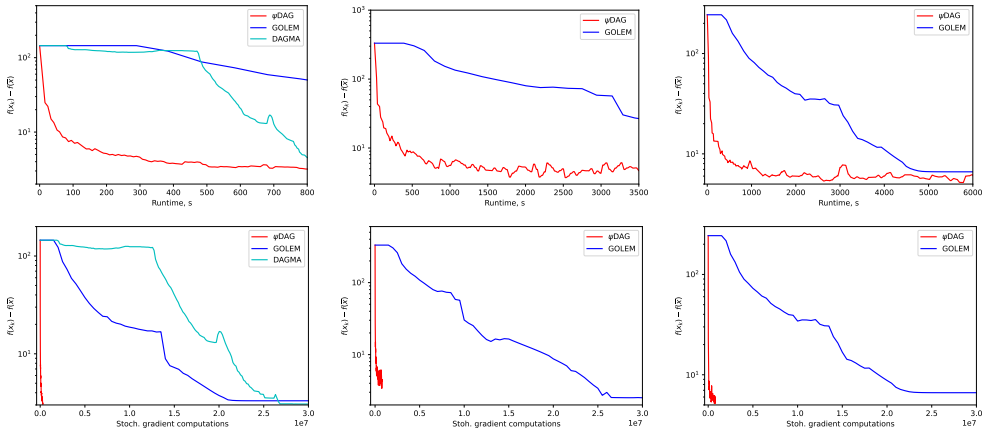


(b) $d = 50$ vertices

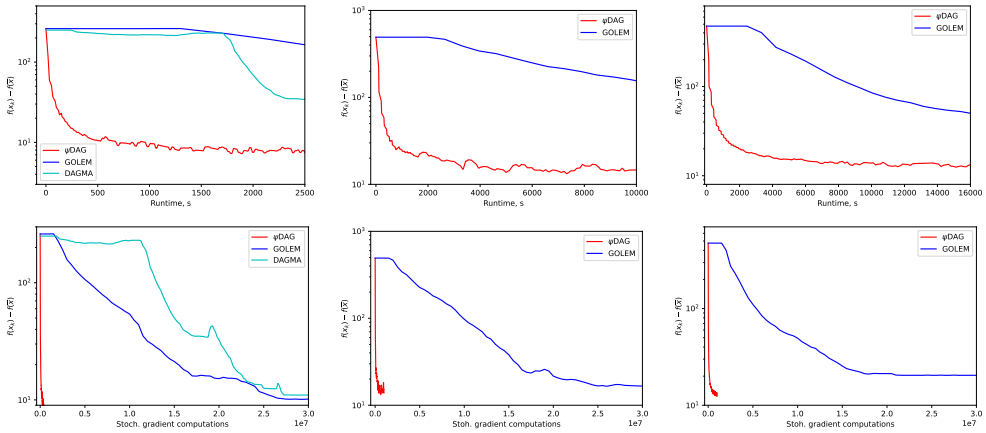


(c) $d = 100$ vertices

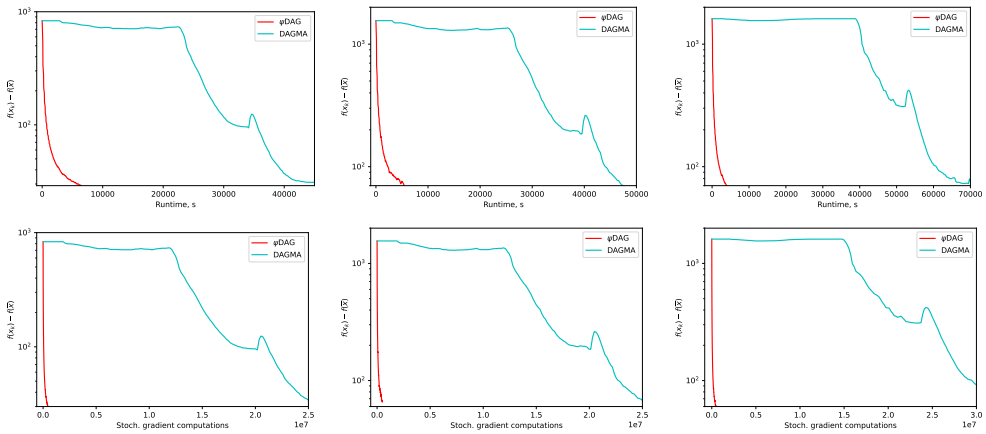
Figure 15: Linear SEM methods on SF2 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).



(a) $d = 500$ vertices

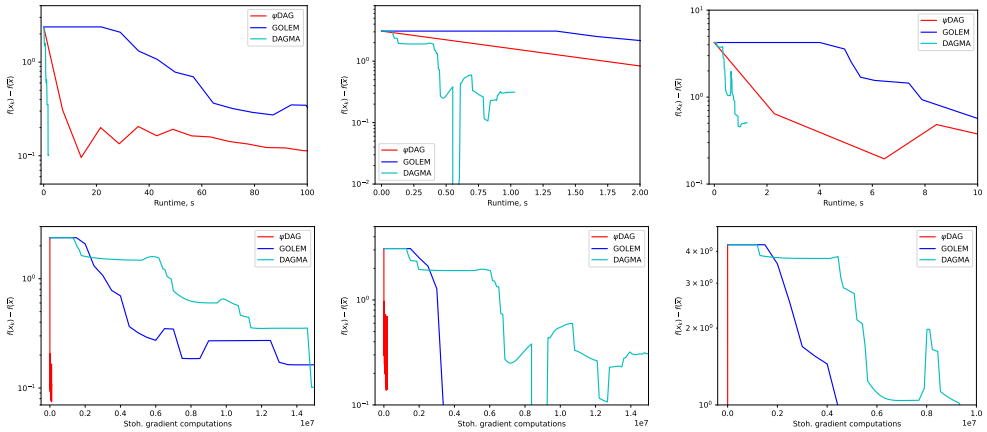


(b) $d = 1000$ vertices

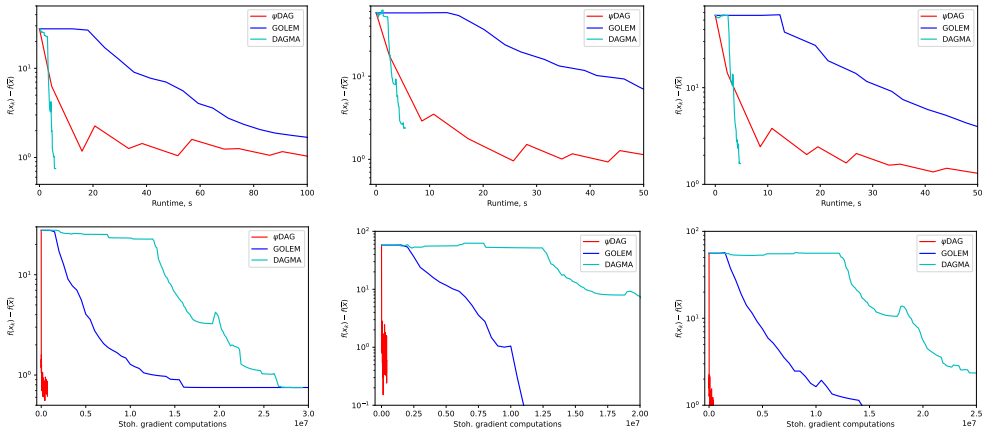


(c) $d = 3000$ vertices

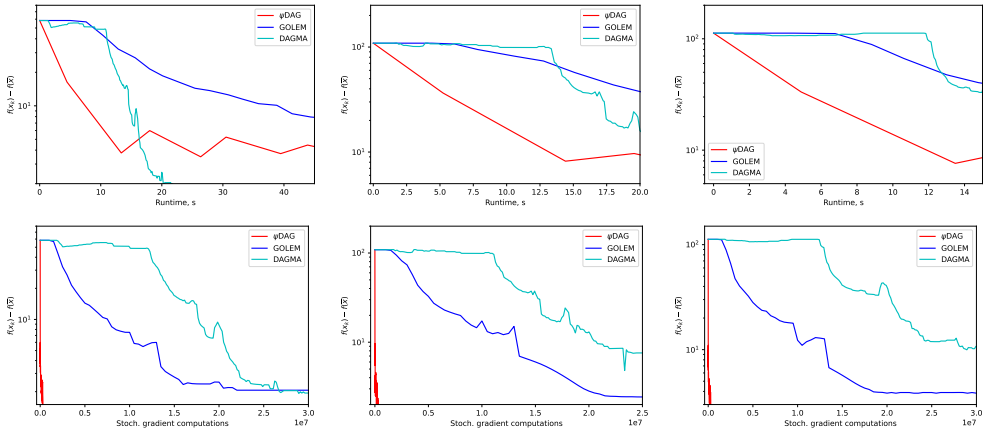
Figure 16: Linear SEM methods on SF2 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).



(a) $d = 10$ vertices

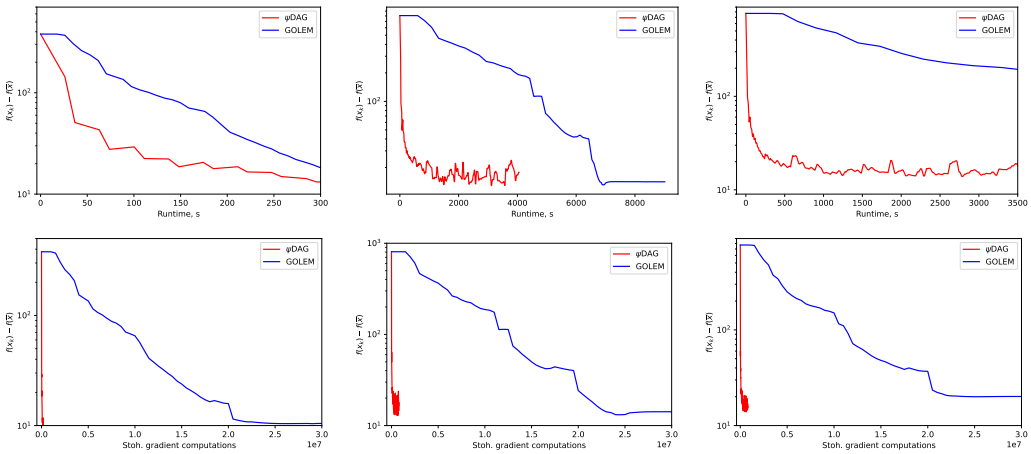


(b) $d = 50$ vertices

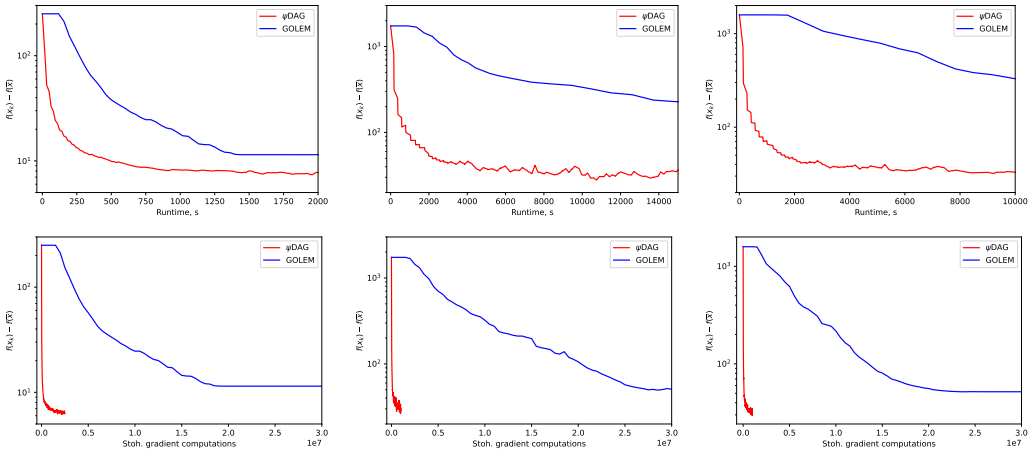


(c) $d = 100$ vertices

Figure 17: Linear SEM methods on SF4 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).

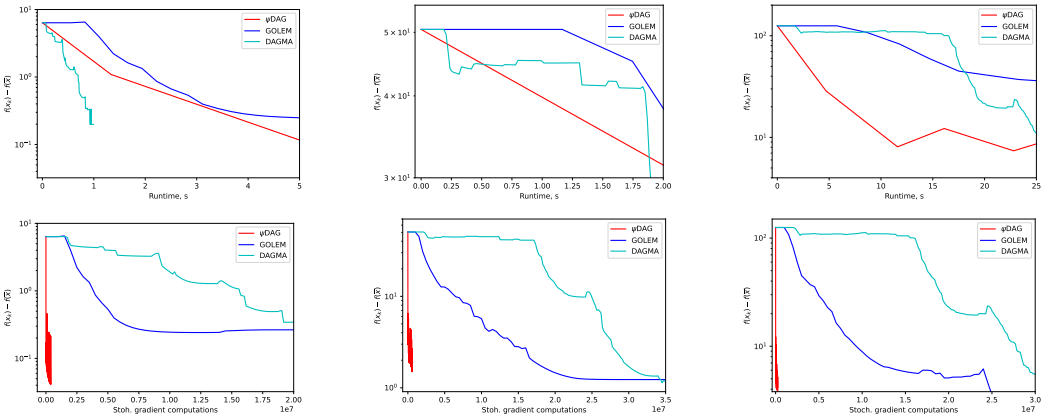


(a) $d = 500$ vertices

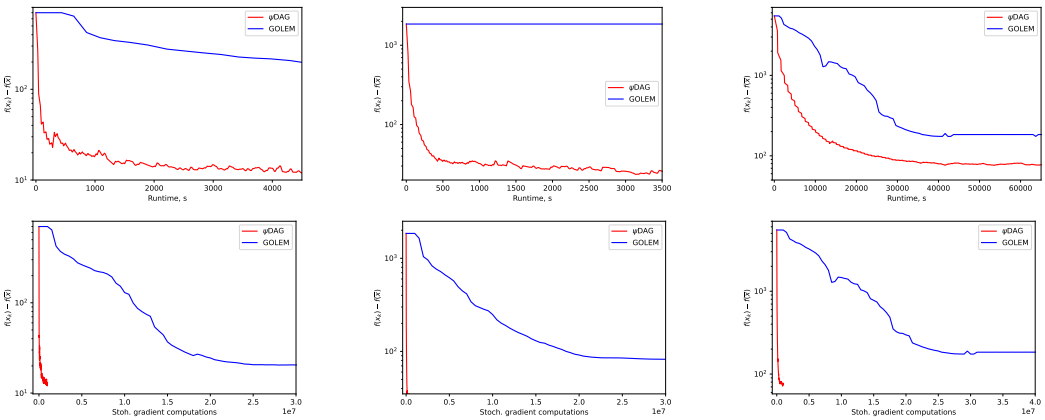


(b) $d = 1000$ vertices

Figure 18: Linear SEM methods on SF4 graphs with different noise distributions: Gaussian (first), exponential (second), Gumbel (third).

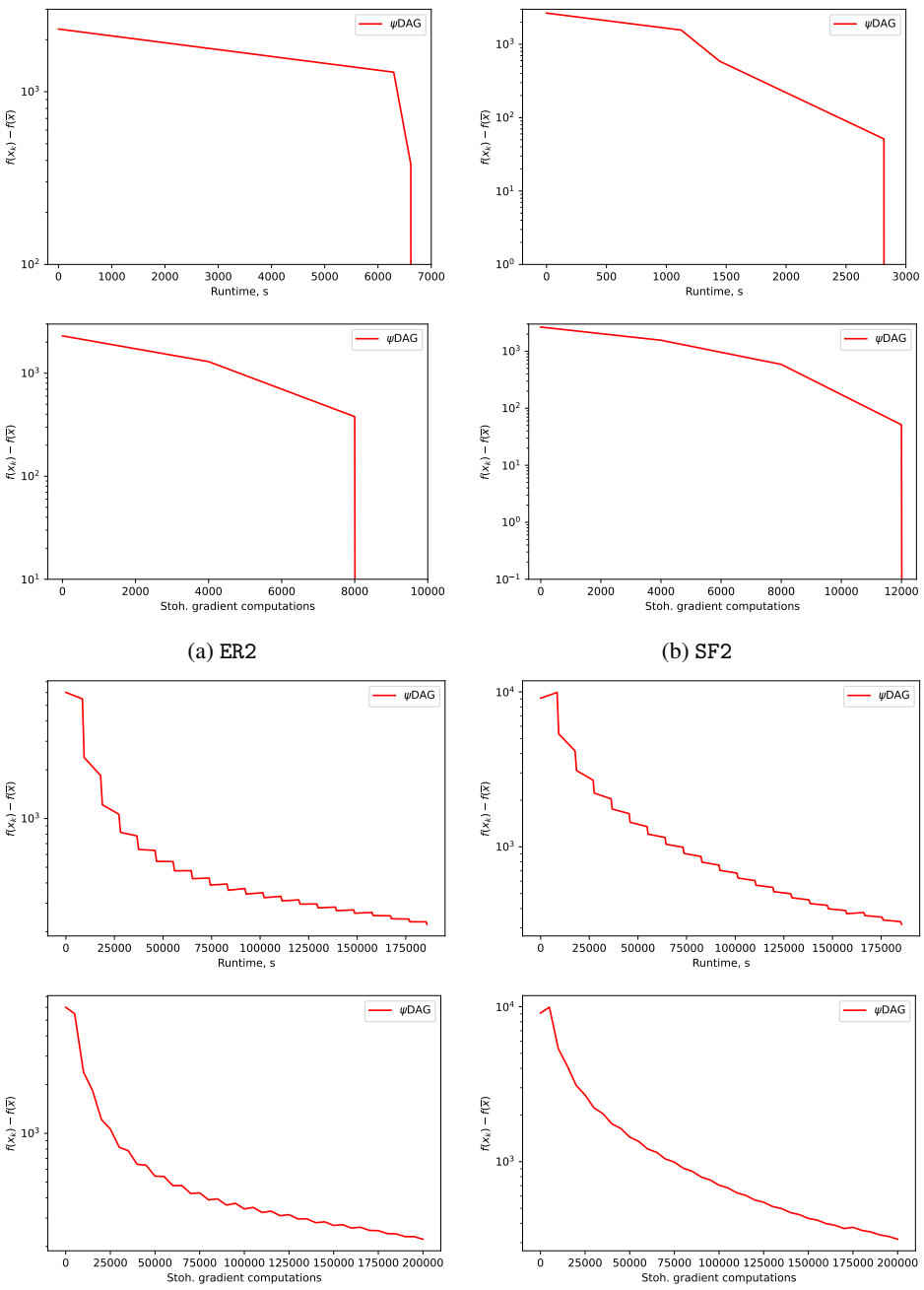


(a) $d = 10$ vertices (b) $d = 50$ vertices (c) $d = 100$ vertices



(d) $d = 500$ vertices (e) $d = 1000$ vertices (f) $d = 3000$ vertices

Figure 19: Linear SEM methods on graphs of type SF6 with the Gaussian noise distribution.



(a) ER2 (b) SF2

(c) ER4 (d) SF4

Figure 20: ψDAG method for graph types ER2, ER4, SF2 and SF4 graphs with $d = 10000$ and Gaussian noise. Other linear SEM methods do not converge in less than 350 hours.

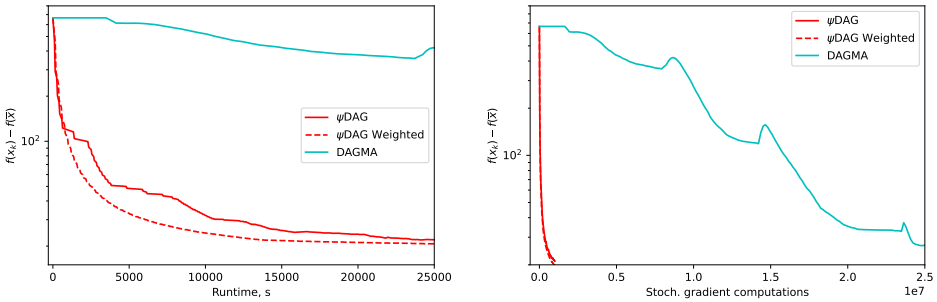


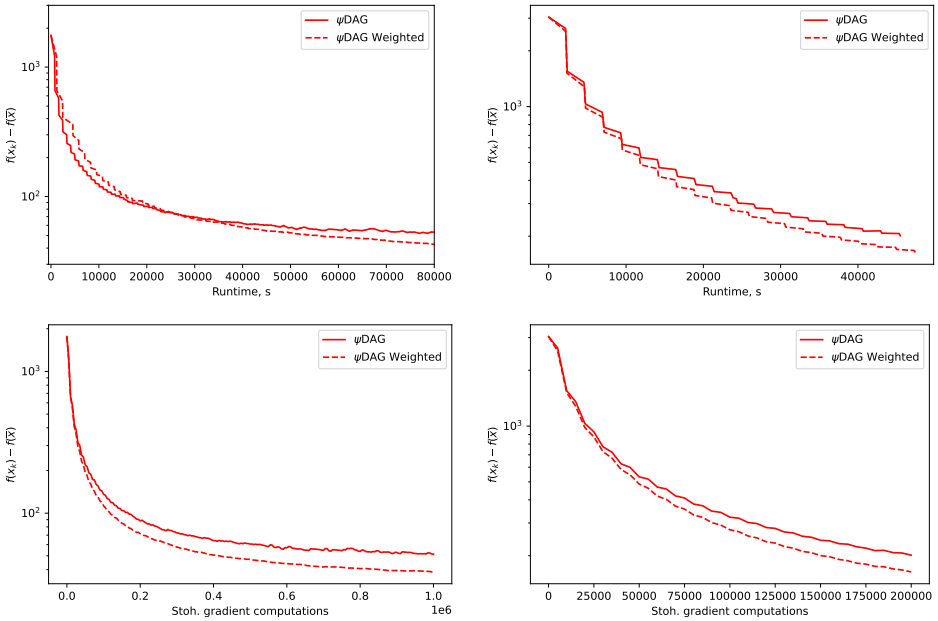
Figure 21: Comparison of ψDAG, ψDAG weighted and DAGMA for ER2 graph with $d = 3000$ nodes and Gaussian noise.

E Weighted Projection

Inspired by the importance sampling, we considered adjustment of the projection method by weights. Specifically, we considered the elements of the $\mathbf{W}$ to be weighted element-wisely by the second directional derivatives of the objective function, $\mathbf{L}[i][j] \stackrel{def}{=} \left(\frac{d}{d\mathbf{W}[i][j]}\right)^2 \mathbb{E}_{X\sim\mathcal{D}}\left[l(\mathbf{W}; X)\right]$. As we do not have access to the whole distribution $\mathcal{D}$, we approximate it by the mean of already seen samples,

$$\mathbf{L}_k[i][j] \stackrel{def}{=} \left(\frac{d}{d\mathbf{W}[i][j]}\right)^2 \frac{1}{k}\sum_{k=0}^{k-1} l\left(\mathbf{W}; X_k\right) = \frac{1}{k}\sum_{t=0}^{k-1}\left(X_k[j]\right)^2. \tag{14}$$

Weights (14) are identical for whole columns; hence, they impose storing only one vector. Updating them requires a few element-wise vector operations.



(a) $d = 3000$ vertices (b) $d = 5000$ vertices

Figure 22: ψDAG method with weighted projection for graph types ER4 and Gaussian noise.

Figures 21 and 22 show that this weighting can lead to an improved convergence (slightly faster convergence to a slightly lower functional value) without imposing any extra gradient computation. However, we noticed that the improvement over runtime is not consistent across different experiments; hence, for simplicity, we deferred this to the appendix.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Sections 4, 5, A and D.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Limiation paragraph in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Section A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 5 and D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See footnote at Section 5. (`https://anonymous.4open.science/r/psiDAG-8F42`)

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Section 5 and D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: See Section 5 and D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and respected the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper focuses on the efficient solving of a causal inference problem. We do not anticipate any direct societal impacts that require specific consideration.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section Appendix C.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See Section 5 and C.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.