DAGs with No Curl: Efficient DAG Structure Learning

Yue Yu
Department of Mathematics
Lehigh University
Bethlehem, PA
`yuy214@lehigh.edu`

Tian Gao
IBM Research
T. J. Watson Research Center
Yorktown Heights, NY
`tgao@us.ibm.com`

Abstract

In recent works, directed acyclic graph (DAG) structure learning was formulated as a constrained continuous optimization problem with continuous acyclicity constraints and was solved iteratively through subproblem optimization. We propose a novel learning framework to model and learn the weighted adjacency matrices of DAGs without iterations. Specifically, we first show that the set of weighted adjacency matrices of DAGs are equivalent to the set of weighted gradients of graph potential functions, and one may perform structure learning by searching in this equivalent set of DAGs directly. To instantiate this idea, we propose an approximation algorithm, DAG-NoCurl, which approximately solves the optimization problem efficiently with a two-phase procedure: 1) in the prediction phase we compute an approximate non-acyclic graph to the optimization problem, and 2) then the projection phase employs the Hodge decomposition of graphs and forms an acyclic approximation of the predicted graph by projecting it to the gradient of a potential function. Experimental studies on synthetic datasets demonstrate that this projection method provides comparable accuracy but better efficiency than baseline DAG structure learning methods, often by more than one order of magnitude.

1 Introduction

Bayesian Networks (BN) have been widely used in various machine learning applications [29, 37]. Efficient structure learning of BN is an active area of research. The structure takes the form of a directed acyclic graph (DAG) and plays a vital part in other machine learning sub-areas such as causal inference [30]. However, DAG learning is proven to be NP-hard [5] and scalability becomes a major issue. Conventional DAG learning methods usually perform score-and-search for discrete variables [35, 16, 7]. Learning DAG structures for continuous variables is often limited to Gaussian graphical models [44, 11, 33, 24, 23]. Recently, a fully continuous optimization formulation is proposed [46, 42] , which transforms the discrete DAG constraint into a continuous equality constraint. This approach enables a suite of continuous optimization techniques such as gradient descent to be used, and has been extend to a more general parameter class with various neural methods [19, 48, 47]. In this work, we take a step further and propose a continuous optimization framework for DAG structure learning *without any explicit constraint*. Independently, [41] and [25] have studied the usage of empirical correlation matrix and covariance matrix to formulate the continuous optimization for structure learning without an explicit acyclicity constraint. Different from their works, a new graph-exterior-calculus-based framework of DAG is proposed which implicitly enforces the acyclicity of the learned graph, and an efficient approximate algorithm, DAG-NoCurl, is developed based on the graph Hodge theory [17, 20] to approximately solve for the resultant unconstrained optimization problem.

Contributions. We make several major contributions in this work: 1) We propose a new model for DAG based on the graph combinatorial gradient operator. Specifically, we theoretically show that the weighted adjacency matrix of a DAG can be represented as the Hadamard product of a skew-symmetric matrix and the gradient of a potential function on graph vertices, and vice verse. 2) Based on the new model, we develop a new continuous optimization framework for DAG structure learning without any explicit constraint. 3) To approximately solve for the optimization problem efficiently, we develop a projection method for DAG structure learning, or DAG-NoCurl, that provides an approximate solution of the weighted adjacency matrix to the original problem without constraints or iterations.

DAG-NoCurl relies on the key step of projecting the adjacency weighted matrix of an estimated directed graph onto an acyclic one. The main advantages of the proposed method over conventional structure learning algorithms are: 1) the new model provides a equivalent representation of DAGs, which can be readily combined with many existing structural equation models [46, 42], 2) the new model naturally transfers the DAG structure learning problem as a continuous optimization framework, which avoids the combinatorial search and enables a suite of continuous optimization techniques; 3) comparing with other fully continuous optimization frameworks for DAG learning [19, 42, 47], our framework needs no explicit DAG constraints or iterations, hence the model achieves a substantially better efficiency. Results on synthetic datasets show that while obtaining the same accuracy, the proposed framework improves the computational efficiency by up to one or two orders of magnitude.

2 Problem Statement and Motivation

In this work we consider the DAG structure learning problem as follows: let V denote a set of d numbers of random variables, $X = (X_1, \cdots, X_d) \in \mathbb{R}^d$ be an observation on V, and $\mathbb{D}$ denotes the space of DAGs $\mathcal{G} = (V, E)$ on V, we aim to learn a DAG $\mathcal{G} \in \mathbb{D}$ given n i.i.d. observations of the random vector $X^i \in \mathbb{R}^d$, $i = 1, \cdots, n$. To model X, we consider a (generalized) structural equation model (SEM) defined by a weighted adjacency matrix $A = [a_1| \cdots |a_d] \in \mathbb{R}^{d\times d}$ such that $\mathbb{E}(X_j|X_{pa(X_j)}) = f(a_j^T X)$, where $pa(X_j)$ denote the parents of random variable j in V. Therefore $[A]_{ij} \neq 0$ indicates a directed edge from vertex i to vertex j in a directed graph $\mathcal{G}_A$ and zero otherwise. Although with a slight abuse of notation, we will treat the weighted adjacency matrix A as a (weighted) directed graph $\mathcal{G}_A$.

Given the data matrix $\mathbf{X} = (X^1, \cdots, X^n) \in \mathbb{R}^{d\times n}$ and a loss function $F(A, \mathbf{X})$ that measures the goodness of fit of $\mathbf{X}$ for A, in this *structure learning* problem we aim to find the best DAG A^* that minimizes $F(A, \mathbf{X})$. Hence, the overall objective can be written as:

$$A^* = \underset{A}{\operatorname{argmin}} \quad F(A, \mathbf{X}), \qquad \text{subject to} \qquad \mathcal{G}_A \in \mathbb{D} \tag{1}$$

As one can see, the DAG space $\mathbb{D}$ for d variables is super-exponential. For structure learning over discrete variables, many exact and approximate algorithms from observational data have been proposed [38, 4, 18, 6, 8, 34] with different search strategies, such as dynamic programming [36, 35], A* [43], or integer programming [16, 7]. In large-scale problems, approximate methods are often needed, with additional assumptions such as bounded tree-width [26] or efficient approximating scores [32]. Sampling [22, 13, 10, 14, 28, 15] and topological order search [12, 39, 32] are also popular.

In this work we assume that all variables are continuous, and study the DAG structure learning problem with the focus of SEM and smooth loss function F defined over A. By using an alternative continuous DAG constraint constraint $h(A) = 0$ [46, 42], the constrained optimization problem (1) becomes fully continuous. Similar to [46, 42], we seek to learn a continuous representation for A. More importantly, we investigate whether the DAG constraint can be eliminated entirely. The key device in accomplishing this is the observation that DAG is associated with curl-free functions on edges, which motivates a new representation of DAGs $A = \gamma(W, p)$, $W \in \mathbb{R}^{d\times d}$ and $p \in \mathbb{R}^d$, as will be elaborated in the next section. The constraint optimization problem (1) can then by replaced by the following objective:

$$(W^*, p^*) = \underset{W,p}{\operatorname{argmin}} \, F(\gamma(W, p), \mathbf{X}), \tag{2}$$

with the optimal DAG $A^* = \gamma(W^*, p^*)$. To solve for the optimization problem in (2), we further propose an approximation algorithm in Section 4, based on the combinatorial Hodge theory on graphs

[20, 17], which decomposes a graph onto curl-free and divergence-free components. To the best of our knowledge, both the equivalence representation of DAGs and the application of Hodge theory in DAG structure learning have never been studied.

Notation-wise, we use A^0 to denote the ground truth DAG, A^* for the global optimal solution in (1), and $\tilde{A}$ for the approximated solution from numerical algorithms. The ultimate goal of our DAG structural learning problem is to obtain a DAG $\tilde{A}$ which recovers the structure of A^0 and obtains a comparable score $F(\tilde{A}, \mathbf{X})$ as $F(A^*, \mathbf{X})$.

3 An Equivalent Model for DAGs

To propose the new formulation for DAG structure learning, we first briefly introduce the basic graph exterior calculus operators [1, 17, 20], then formulate DAGs into an equivalent model. For a more thorough introduction of graph calculus, we refer the interested readers to [20].

Let $\widehat{\mathcal{G}} = (V, E)$ be a **complete undirected** graph where $V := \{1, \cdots, d\}$ is the set of vertices and E is the set of undirected edges. Note here since $\widehat{\mathcal{G}}$ is a complete graph, the set of k-th cliques of $\widehat{\mathcal{G}}$ is equivalent to $\left\{\binom{V}{k}\right\}$. The ordered and unordered pairs of vertices are delimited by (i, j) and $\{i, j\}$, respectively, where i, j denote the i-th and j-th vertices. Real-valued functions on graph can be defined on vertices, edges, triangles, and so on. On vertices, a real-valued function $f : V \to \mathbb{R}$ is called a **potential function**, and we denote the Hilbert space of all potential functions as $L^2(V)$. We may also define real-valued functions on edges $E = \{\{i, j\}, i, j \in V\}$ and triangles $T = \{\{i, j, k\}, i, j, k \in V\}$, with the requirement that these functions are **alternating**. In particular, for an alternating function on edges $Y : V \times V \to \mathbb{R}$, we require $Y(i, j) = -Y(j, i)$; for an alternating function on triangles $\Theta : V \times V \times V \to \mathbb{R}$, we require $\Theta(i, j, k) = -\Theta(j, i, k) = -\Theta(i, k, j)$. In the following we use $L^2_\wedge(E)$ and $L^2_\wedge(T)$ to denote the Hilbert spaces of real-valued alternating functions on edges and triangles, respectively. Moreover, we note that $p \in L^2(V)$ corresponds to a real vector $p = [p(1), \cdots, p(d)] \in \mathbb{R}^d$, and an alternating function $Y \in L^2_\wedge(E)$ corresponds to a skew-symmetric real matrix $Y \in \mathbb{R}^{d\times d}$ with $[Y]_{ij} = Y(i, j)$ and $Y = -Y^T$. Here we use the same letter to denote a vector/matrix and the corresponding function on vertices/edges.

Next, we introduce graph calculus operators grad, curl and their adjoint operators below.

Definition 3.1. *The gradient (*$\operatorname{grad} : L^2(V) \to L^2_\wedge(E)$*) is an operator on any function* p *on vertices:*

$$(\operatorname{grad} p)(i, j) = p(j) - p(i), \quad \forall \{i, j\} \in E,$$

and $\operatorname{grad}(p)$ *is called a gradient flow. Its adjoint operator (*$\operatorname{grad}^* : L^2_\wedge(E) \to L^2(V)$*, or sometimes called the negative divergence operator) is defined on any alternating function* Y *on edges:*

$$(\operatorname{grad}^* Y)(i) = -(\operatorname{div} Y)(i) = -\sum_{j=1}^{d} Y(i, j), \quad \forall i \in V.$$

The curl ($\operatorname{curl} : L^2_\wedge(E) \to L^2_\wedge(T)$*) is an operator for any alternating function* Y *on edges:*

$$(\operatorname{curl} Y)(i, j, k) = Y(i, j) + Y(j, k) + Y(k, i), \quad \forall \{i, j, k\} \in T,$$

and its adjoint operator $\operatorname{curl}^* : L^2_\wedge(T) \to L^2_\wedge(E)$ *is for any alternating function* Θ *on triangles:*

$$(\operatorname{curl}^* \Theta)(i, j) = \sum_{k=1}^{d} \Theta(i, j, k), \quad \forall \{i, j\} \in E.$$

The graph Laplacian ($\triangle_0 : L^2(V) \to L^2(V)$*) is an operator on any function* p *on vertices:*

$$(\triangle_0 p)(i) = -(\operatorname{div} \operatorname{grad} p)(i) = dp(i) - \sum_{j=1}^{d} p(j), \quad \forall i \in V.$$

The graph Helmholtzian ($\triangle_1 : L^2_\wedge(E) \to L^2_\wedge(E)$*) is defined on any alternating function* Y *on edges:*

$$(\triangle_1 Y)(i, j) = (\operatorname{grad} \operatorname{grad}^* Y + \operatorname{curl}^* \operatorname{curl} Y)(i, j), \forall \{i, j\} \in E.$$

Lemma 3.2. *[17] Let* $p \in L^2(V)$ *and* $\Theta \in L^2_\wedge(E)$*, denote* $D = \operatorname{grad}(p)$ *and* $R = \operatorname{curl}^*(\Theta)$*, then* D *and* R *are curl-free and divergence-free, respectively:*

$$\operatorname{curl}(D)(i, j, k) = 0, \quad \forall \{i, j, k\} \in T; \qquad \operatorname{div}(R)(i) = -\operatorname{grad}^*(R)(i) = 0, \quad \forall i \in V.$$

Algorithm 1 DAG-NoCurl algorithm for DAG Structure Learning

```
1. Prediction phase: Solve for an initial prediction $A^{pre} \in \mathbb{R}^{d\times d}$ with (4) and threshold $A^{pre}$.
2. Projection phase: Based on $A^{pre}$, obtain an approximate solution of $p^*$ as $\tilde{p}$ with (9), then solve
   for $\tilde{W}$ with fixed $\tilde{p}$ in (11).
3. Obtain the final approximation solution $\tilde{A} = \tilde{W} \circ \mathrm{ReLU}(\mathrm{grad}(\tilde{p}))$ and threshold $\tilde{A}$.
```

Given a complete undirected graph $\widehat{\mathcal{G}}(V, E)$ and a function $Y \in L^2_\wedge(E)$, with ReLU denoting the rectified linear unit function, we can define a weighted adjacency matrix $A \in \mathbb{R}^{d\times d}$ as:

$$A(i,j) = \mathrm{ReLU}(Y)(i,j) = \begin{cases} Y(i,j), & \text{if } Y(i,j) > 0, \\ 0, & \text{else,} \end{cases}$$

and further define a **weighted directed graph** $\mathcal{G}_{\mathrm{ReLU}(Y)}$ from $A = \mathrm{ReLU}(Y)$ in the following:

Definition 3.3. *Consider a complete undirected graph $\widehat{\mathcal{G}}(V, E)$ and $Y \in L^2_\wedge(E)$, a directed graph $\mathcal{G}_{\mathrm{ReLU}(Y)}(V, E_{\mathrm{ReLU}(Y)})$ is defined such that there is a directed edge from vertex i to vertex j in $\mathcal{G}_{\mathrm{ReLU}(Y)}$ if and only if $Y(i,j) > 0$, i.e., the set of directed edges $E_{\mathrm{ReLU}(Y)} = \{(i,j)|Y(i,j) > 0\}$. Moreover, note that $\mathrm{ReLU}(Y)$ is a weighted adjacency matrix of $\mathcal{G}_{\mathrm{ReLU}(Y)}$.*

Based on the above definition, we show that curl-free functions are associated with DAGs:

Lemma 3.4. *Consider a complete undirected graph $\widehat{\mathcal{G}}(V, E)$ and a* ***curl-free*** *function $Y \in L^2_\wedge(E)$, then $\mathrm{ReLU}(Y) \in \mathbb{R}^{d\times d}$ is the weighted adjacency matrix of a DAG. Moreover, given any skew-symmetric matrix $W \in \mathbb{R}^{d\times d}$, $W \circ \mathrm{ReLU}(Y)$ is also a DAG, where $\circ$ is the Hadamard product.*

All proofs can be found in Appendix A. Since the gradient of any potential function (gradient flow) is curl-free, with Lemma 3.4 the first part of our main theoretical results is obtained:

Theorem 3.5. *Consider a complete undirected graph $\widehat{\mathcal{G}}(V, E)$, given any function $p \in L^2(V)$ and any skew-symmetric matrix $W \in \mathbb{R}^{d\times d}$, $W \circ \mathrm{ReLU}(\mathrm{grad}(p))$ is the weighted adjacency matrix of a DAG, i.e., $\{\mathcal{G}_{W\circ\mathrm{ReLU}(\mathrm{grad}(p))}\} \subset \mathbb{D}$.*

Remark 3.6. *The skew-symmetry assumption of W was made based on the fact that at least one or both of $\mathrm{ReLU}(\mathrm{grad}(p))(i,j) = 0$ and $\mathrm{ReLU}(\mathrm{grad}(p))(j,i) = 0$ must hold true for any $i, j \in V$. This assumption reduces the number of unknown parameters associated with W from d^2 to $\frac{d(d-1)}{2}$.*

We will now show that the other direction also holds true:

Theorem 3.7. *Let $A \in \mathbb{R}^{d\times d}$ be the weighted adjacency matrix of a DAG with d nodes, denote $\widehat{\mathcal{G}}(V, E)$ as the complete undirected graph on these d nodes, then there exists a skew-symmetric matrix $W \in \mathbb{R}^{d\times d}$ and a potential function $p \in L^2(V)$ such that $A = W \circ \mathrm{ReLU}(\mathrm{grad}(p))$, i.e., $\mathbb{D} \subset \{\mathcal{G}_{W\circ\mathrm{ReLU}(\mathrm{grad}(p))}\}$.*

The key step of the proof is to define a function $p \in L^2(V)$ based on the topological order of the DAG, such that $p(j) > p(i)$ if there is a directed path from vertex i to j. Proof is in Appendix A.

Hence, we have shown that the set of weighted adjacency matrices for DAGs is equivalent to the set of $W \circ \mathrm{ReLU}(\mathrm{grad}(p))$. Denoting $S := \{W | W \in \mathbb{R}^{d\times d}, W = -W^T\}$ as the space of all $d \times d$ skew-symmetric matrices, $\{W \circ \mathrm{ReLU}(\mathrm{grad}(p)) | W \in S, p \in \mathbb{R}^d\}$ provides an admissible solution set for DAG structural learning problems. With a given loss function $F(A, \mathbf{X})$, the structural learning problem can be written as the following optimization problem without an explicit constraint:

$$(W^*, p^*) = \underset{W\in S, p\in\mathbb{R}^d}{\mathrm{argmin}} \; F(W \circ \mathrm{ReLU}(\mathrm{grad}(p)), \mathbf{X}), \tag{3}$$

and the optimal solution $A^* = W^* \circ \mathrm{ReLU}(\mathrm{grad}(p^*))$.

4 A Prediction-Projection Method for DAG Structure Learning

As one can see, the loss function in (3) is generally nonconvex. Therefore, we inherit the difficulties associated with nonconvex optimization. As will be discussed in the ablation study in Section 5, numerically solving (3) with a random initialization of W and p may result in a stationary point which is far from the global optimum. Therefore, instead of solving for (3) directly we propose a novel two-phase approximation algorithm, DAG-NoCurl. In the prediction phase, we compute an initial estimate solution matrix $A^{pre} \in \mathbb{R}^{d\times d}$ whose associated graph $\mathcal{G}_{A^{pre}}$ is not necessarily acyclic. In the projection phase, we refine the predicted solution by projecting A^{pre} into the set of DAGs $\{W \circ \mathrm{ReLU}(\mathrm{grad}(p))\}$ and obtaining an approximated solution $\tilde{A} = \tilde{W} \circ \mathrm{ReLU}(\mathrm{grad}(\tilde{p}))$. The full algorithm is outlined in Algorithm 1, and we introduce each component of the algorithm in more details as follows.

Prediction Phase: In order to compute an initial prediction A^{pre}, we solve for the following unconstrained problem

$$A^{pre} = \operatorname*{argmin}_{A} F(A, \mathbf{X}) + \lambda h(A) \tag{4}$$

where $\lambda > 0$ is a constant penalty coefficient, $F(A, \mathbf{X})$ is the loss function and $h(A)$ is a proper continuous constraint for acyclicity [46, 42]. For example, in NOTEARS [46] the authors proposed $h(A) = \mathrm{tr}[\exp(A \circ A)] - d$ and in DAG-GNN [42] $h(A) = \mathrm{tr}[(I + \alpha A \circ A)^d] - d$, $\alpha > 0$ is a constant, was employed. Note that when increasing the penalty parameter λ to infinity and solving (4) iteratively, an acyclic graph A will be obtained and the procedure will be equivalent to the classical penalty method for constraint optimization problems. However, here we solve for (4) with fixed λ, $h(A^{pre})$ is generally nonzero and A^{pre} is likely not a DAG, and therefore a projection step in the next phase is required to obtain a DAG. In practice, we find that solving for (4) at most twice, once with a fixed λ (as denoted by the NoCurl $\lambda = \cdot$ cases) or twice with two fixed λs (as denoted by the NoCurl $\lambda = (\lambda_1, \lambda_2)$ cases), gives satisfactory initializations. In the NoCurl $\lambda = (\lambda_1, \lambda_2)$ cases, we first solve for (4) with a $\lambda = \lambda_1$, and then solve for (4) with a larger fixed penalty coefficient $\lambda = \lambda_2$. To achieve a good balance for the computational time and solution accuracy in structural discovery, in Section 5 we perform a hyperparameter study for both one λ and two λ cases.

Projection Phase: To further obtain an acyclic graph solution $\tilde{A}$, we use the prediction A^{pre} as an initialization and project it into the admissible solution set $\{W \circ \mathrm{ReLU}(\mathrm{grad}(p))\}$. Specifically, we first design a projection method to obtain $\tilde{p}$ based on the following theorem:

Theorem 4.1 (Hodge Decomposition Theorem [20, 2, 17])**.** *Consider an undirected graph $\widehat{\mathcal{G}}(V, E)$, any function $Y \in L^2_\wedge(E)$ can be decomposed into three orthogonal components:*

$$Y = \mathrm{grad}(\phi) + \mathrm{curl}^*(\Psi) + H \tag{5}$$

where $\phi \in L^2(V)$, $\Psi \in L^2_\wedge(T)$, and $H \in L^2_\wedge(E)$. Moreover, H is a harmonic function satisfying $\triangle_1 H = 0$, $\mathrm{curl}\, H = 0$ and $\mathrm{div}\, H = 0$.

The Hodge decomposition shows that every alternating function on edges $Y \in L^2_\wedge(E)$ can be decomposed into two orthogonal components: a gradient flow $\mathrm{grad}(\phi)$, which represents the L^2-optimal ordering of the vertices, and a divergence-free component $\mathrm{curl}^*(\Psi) + H$, which measures the inconsistency of the vertices ordering. Therefore, we define a L^2 **projection operator** $\mathrm{Proj} : L^2_\wedge(E) \to \mathrm{grad}(L^2(V))$ as:

$$\mathrm{Proj}(Y) = \mathrm{grad}(\phi), \tag{6}$$

such that $\langle \mathrm{Proj}(Y), Y - \mathrm{Proj}(Y)\rangle = 0$, where $\langle \cdot, \cdot \rangle$ is the standard L^2 inner product on $L^2_\wedge(E)$: $\langle Y, Z\rangle := \sum_{i,j} Y(i,j)Z(i,j)$.

We note that ϕ is only unique up to a constant potential function, i.e., if $\mathrm{Proj}(Y) = \mathrm{grad}(\phi)$, then $\mathrm{Proj}(Y) = \mathrm{grad}(\phi + C)$ also holds. Adding a constant to ϕ will yield the same ordering of vertices and the same gradient flow. For the sake of well-posedness, in practice we determine a unique solution ϕ by fixing its value on the last vertex such that $\phi(d) = 0$. Taking the divergence of (5) yields:

$$\mathrm{div}(Y) = \mathrm{div}\ \mathrm{grad}(\phi) = -\triangle_0 \phi. \tag{7}$$

We obtain the following theorem for the solution of ϕ:

Theorem 4.2. *Consider an undirected complete graph $\widehat{\mathcal{G}}(V, E)$ and $Y \in L^2_\wedge(E)$, a solution of* (6) *is given by*

$$\phi = -\triangle_0^\dagger \,\mathrm{div}(Y) = -\triangle_0^\dagger \sum_j Y(i,j) \tag{8}$$

where $\dagger$ indicates the Moore-Penrose pseudo-inverse. Fixing $\phi(d) = 0$, the matrix representing the graph Laplacian $\triangle_0$ is given by

$$[\triangle_0]_{ij} = \begin{cases} d-1, & \text{if } i = j \text{ and } i, j \neq d, \\ -1, & \text{if } i \neq j \text{ and } i, j \neq d, \\ 0, & \text{otherwise.} \end{cases}$$

Note that a key requirement in the above projection operator is that Y has to be an alternating function on edges, which corresponds to a $d \times d$ skew-symmetric matrix. However, A^{pre} is generally not skew-symmetric. Fortunately, any square matrix M can be written uniquely as a symmetric and a skew-symmetric components: $M = M_{sym} + M_{skew}$ with $M_{sym} = \frac{1}{2}(M + M^T)$ and $M_{skew} = \frac{1}{2}(M - M^T)$. Let $C(M)$ denote the connectivity matrix [27] of a directed graph M such that $[C(M)]_{ij} = 1$ only if a directed path exists from vertex i to vertex j. We apply the projection to the skew-symmetric component of the connectivity matrix of A^{pre}, and we will show that this operation preserves the topological order of vertices in a DAG:

Theorem 4.3. *Let $A \in \mathbb{R}^{d \times d}$ be the weighted adjacency matrix of a DAG with d nodes, then*

$$p = -\triangle_0^{\dagger} \operatorname{div}\left(\frac{1}{2}(C(A) - C(A)^T)\right), \tag{9}$$

preserves the topological order in A such that $p(j) > p(i)$ if there is a directed path from vertex i to j. Moreover, we have $A = W \circ \operatorname{ReLU}(\operatorname{grad}(p))$ with the skew-symmetric matrix W defined as

$$[W]_{ij} = \begin{cases} 0, & \textit{if } p(i) = p(j) \textit{ or } A(i,j) = A(j,i) = 0; \\ \frac{A(i,j)}{p(j)-p(i)}, & \textit{if } A(i,j) \neq 0 \textit{ and } A(j,i) = 0; \\ \frac{A(j,i)}{p(j)-p(i)}, & \textit{if } A(i,j) = 0 \textit{ and } A(j,i) \neq 0. \end{cases} \tag{10}$$

A detailed proof is provided in Appendix A.

Intuitively, from Theorem 4.3 we can see that $\tilde{p} = -\triangle_0^{\dagger} \operatorname{div}\left(\frac{1}{2}(C(A^{pre}) - C(A^{pre})^T)\right)$ provides the topological order for an acyclic approximation of A^{pre}. For an approximation solution of W, when A^{pre} is not a DAG and there exists a directed cycle $(c_1, c_2, \cdots, c_k, c_1)$, we note that the formulations in (9) and (10) will eliminate all the edges between vertices c_i and c_j, $\forall \{i, j\} \subset \{1, \cdots, k\}$, which then results in a DAG solution.

To further improve the quality of the approximation solution of W, instead of employing (10) directly we compute $\tilde{W}$ via:

$$\tilde{W} = \operatorname*{argmin}_{W \in S} F(W \circ \operatorname{ReLU}(\operatorname{grad}(\tilde{p})), \mathbf{X}). \tag{11}$$

Then, we obtain an approximate DAG solution $\tilde{A} = \tilde{W} \circ \operatorname{ReLU}(\operatorname{grad}(\tilde{p}))$. To ensure the skew-symmetric property of W, one can optimize parameters only from the upper triangular matrix of W and set the lower triangular matrix to be the negative of the upper triangular matrix.

By the standard practice of thresholding in DAG learning problems (see, e.g., [46]), we employ the same procedure to post-process A^{pre} after the prediction phase and $\tilde{A}$ after the projection phase. Specifically, we threshold the edge weights of a learned A as follows: given a fixed threshold $\epsilon > 0$, set $A(i,j) = 0$ if $|A(i,j)| < \epsilon$. In all the numerical tests we use a default threshold value $\epsilon = 0.3$ as suggested in NOTEARS [46]. Note that The thresholding step in NOTEARS is motivated by rounding the numerical solution into exact DAG and removing false discoveries. Since the our characterization returns an exact DAG, so the purpose of the thresholding step is to remove false discoveries, which is slightly different from NOTEARS.

As shown in Algorithm 1, the full procedure consists of only two (if using only one λ) or three (if using two λs) unconstrained optimization problems, since this prediction-projection approach doesn't require any iterative procedure to increase the penalty coefficient λ. We note that $(\tilde{W}, \tilde{p})$ is only an approximate solution of (3). However, as will be discussed in the experiments of Section 5, the approximate solution found by our algorithm are typically close to global minima in practice, by comparing our results to the global minimizer.

5 Experiments

We present empirical evaluations to demonstrate the effectiveness of the proposed DAG continuous representation and the approximation algorithm. Specifically, we conduct experiments on synthetic datasets, and compare the proposed method DAG-NoCurl against competitive baselines. Here we outline the empirical set-up, with more details including all parameter choices provided in Appendix B.

For the experiments with synthetic datasets, we employ similar experimental setups as existing work [46]. In each experiment a random graph $\mathcal{G}$ is generated by using the Erdős–Rényi (ER) model or the scale-free (SF) model with kd expected edges (denoted as ERk or SFk cases) and uniformly random edge weights to obtain a ground-truth weighted matrix A^0. Given A^0, we take $n = 1000$ i.i.d. samples X from the linear SEM $X = (A^0)^T X + Z$ where the noise $Z \in \mathbb{R}^d$ is generated from two different noise models: Gaussian and Gumbel. To apply the NoCurl algorithm in the linear SEM we use the least-square loss $F_{SEM}(A, \mathbf{X}) = \frac{1}{2n}||\mathbf{X} - A^T\mathbf{X}||_F^2$ and numerically solve unconstrained optimization problems with L-BFGS [21], although we note that other standard smooth optimization schemes can also be employed.

We use Structure Hamming Distance (SHD) [46] to show the structure learning accuracy. To assess the ability of methods in solving the original problem (1), we also report its score difference from the ground truth: $\Delta F = F(\tilde{A}, \mathbf{X}) - F(A^0, \mathbf{X})$. For each graph type-noise type combination, 100 trials are performed, and the mean accuracy, mean time, mean score difference along with their perspective standard error of the mean are reported in Appendix C-F. We compare our method with fast greedy equivalent search (FGS) [31], Causal Additive Models [3], MMPC [40], and NOTEARS [46]. In the appendix, we also compare with equal variance methods, Eq-BU and Eq-TD [9].

Hyperparameter Study: We first perform a quantitative study on the hyperparameter λ choices as well as an albation study on different components of the proposed algorithm. We focus on the ER3-Gaussian and ER6-Gaussian cases. Numberical results are listed in Table 3 for ER3-Gaussian and Table 4 for ER6-Gaussian cases in Appendix C. It is observed that as long as $\lambda \geq 10$, the accuracy results are all satisfactory. Among which, $\lambda = 10^2$ and $\lambda = (10, 10^3)$ are the best values in term of both accuracy and computational efficiency. Hence, we select them as the default values. For more discussion, please see Appendix C.

Ablation Study we conduct an ablation study on Algorithm 1, testing how each step would affect the final result, with three settings: **1**. solving the optimization problem (3) directly with random initialization of (W, p) (denoted as "rand init"); **2**. NoCurl without the prediction phase, by solving for W with a random p then performing one additional step to jointly optimize (W, p) with (3) (denoted as "rand p"); **3**. NoCurl with one additional step to jointly optimize (W, p) by solving (3) after the projection step (denoted as $\lambda = \cdot +$). As one can see from Table 3 and 4 in the Appendix C, NoCurl with random initializations performs subpar, indicating the importance of the prediction step in Algorithm 1. Moreover, adding extra optimization steps after does not result much improvements on accuracy or ΔF. This result indicates that although NoCurl doens't guarantee to provide a stationary point of the problem (3), it reaches an approximate solution near a stationary point in many cases.

Table 1: Comparison on score differences from the ground truth, $\triangle F = F(\tilde{A}, \mathbf{X}) - F(A^0, \mathbf{X})$.

Method	d=10 ER3,Gauss	d=50 ER3,Gauss	d=100 ER3,Gauss	d=10 ER6,Gauss	d=50 ER6,Gauss	d=100 ER6,Gauss
GOBNILP	-0.03 ± 0.00	–	–	-0.03 ± 0.00	–	–
NOTEARS	0.03 ± 0.01	-0.25 ± 0.04	-1.65 ± 0.08	0.22 ± 0.40	1.97 ± 0.26	2.49 ± 0.37
$\lambda = 10^2$	0.09 ± 0.02	0.05 ± 0.05	-0.82 ± 0.16	0.54 ± 0.22	2.31 ± 0.41	4.30 ± 0.99
$\lambda = (10, 10^3)$	0.06 ± 0.02	-0.10 ± 0.05	-1.44 ± 0.09	0.36 ± 0.07	1.77 ± 0.38	2.61 ± 0.93

Optimization Objective Results: We now study the performance of NoCurl in approximately solving the optimization problem given by (1), by comparing the scores from NoCurl solution $\tilde{A}$ with those from NOTEARS and the exact global minimizer from the GOBNILP algorithm [7]. Since GOBNILP involves enumerating all possible parent sets, its experiments are limited to small DAGs with $d = 10$ cases. In Table 1, we show the relative score ΔF with respect to the ground truth graph A^0. Surprisingly, although NoCurl doesn't guarantee to return a minimizer, we can see that the score from NoCurl $\lambda = (10, 10^3)$ case is very close to the objective values of NOTEARS and GOBNILP, and even outperforms NOTEARS in the ER6 $d = 50$ case. When d increases and the optimization problem becomes more difficult, the fast projection of NoCurl remains competitive and does not degrade, which is an encouraging evidence that NoCurl can find good approximated solutions.

Structure Recovery Results: We now present the comparison with other baselines methods by comparing structure recovery accuracy and computational efficiency of NoCurl with NOTEARS, FGS, CAM, and MMPC. In Figure 1, the top row shows the SHD of different methods while the buttom row shows the CPU time, in seconds, of different methods, both in log scales. For the full numerical results in all graph types, please refer to Table 5-8 in Appendix F.

Consistent with existing observations [46, 3], FGS, MMPC, and CAM's performances deteriorates when the number of edges gets larger. While NOTEARS is significantly more accurate than other

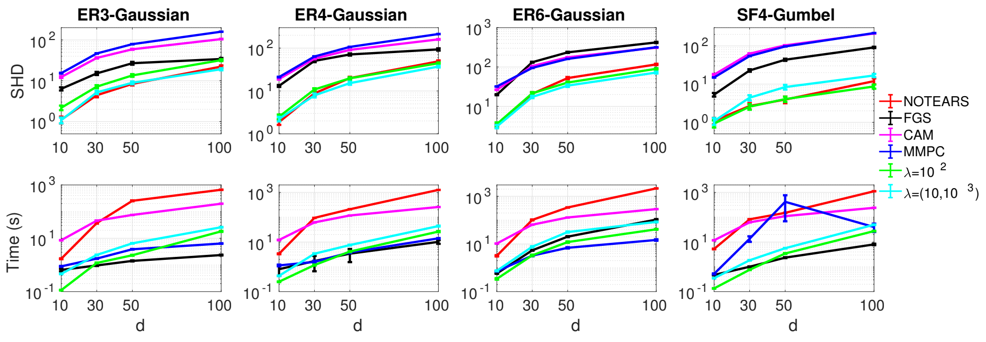

Figure 1: Structural discovery in terms of SHD (lower is better) and computational time in seconds on linear SEM datasets. Error bars represent standard errors over 100 simulations.

baselines, NoCurl achieves a similar accuracy as NOTEARS, and can sometimes beat NOTEARS, especially on dense and large graphs. For instance, when $d = 100$, in all ERk-Gaussian cases the NoCurl with $\lambda = (10, 10^3)$ achieves the lowest SHD, while in SF4-Gumbel cases the NoCurl with $\lambda = 10^2$ achieves the lowest SHD among all methods. When comparing the efficiency, NoCurl requires a similar runtime as FGS and MMPC, which is faster than NOTEARS by more than one or two orders of magnitude. Overall, NoCurl substantially improves the efficiency comparing with NOTEARS, while still sustaining the comparable structure discovery accuracy.

6 Conclusion

We proposed a novel theoretically-justified continuous representation of DAG structures based on graph exterior calculus operators, and proved that this new formulation can represent the adjacency matrices of DAGs without explicit acyclicity constraints, which is often the most intricate part of the optimization. We also proposed a prediction-projection approach, which we coin DAG-NoCurl, to approximately solve for the unconstrained optimization problem efficiently. The key step in this approach is based on the Hodge decomposition theorem, which projects a non-acyclic graph to the gradient of a potential function and obtains a DAG approximation of the graph. Empirically the proposed DAG-NoCurl achieves comparable accuracy but with substantially better computational efficiency than their counter parts with constraints in all the datasets tested. Future works could investigate different choices of hyperparameters and applications to more model classes. A full theoretical understanding of the proposed approximate solution to the global minimizer of Equation 1 and the local minimizer of the original NOTEARS framework is an interesting next step to better understand the optimization behaviors.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers who provided helpful comments on many drafts of the manuscript. Y. Yu is supported by the National Science Foundation under award DMS 1753031.

References

[1] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[2] Harsh Bhatia, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer. The helmholtz-hodge decomposition—a survey. *IEEE Transactions on visualization and computer graphics*, 19(8): 1386–1404, 2012.

[3] Peter Bühlmann, Jonas Peters, Jan Ernest, et al. Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

[4] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 2002.

[5] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[6] James Cussens. Bayesian network learning with cutting planes. In *UAI*, 2011.

[7] James Cussens, David Haws, and Milan Studenỳ. Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*, pages 1–40, 2016.

[8] Cassio P. de Campos, Zhi Zeng, and Qiang Ji. Structure learning of Bayesian networks using constraints. In *ICML*, pages 113–120, New York, NY, USA, 2009. ACM.

[9] Mathias Drton, Wenyu Chen, and Y Samuel Wang. On causal discovery with equal variance assumption. *arXiv preprint arXiv:1807.03419*, 2018.

[10] Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

[11] Rina Foygel and Mathias Drton. Extended bayesian information criteria for gaussian graphical models. In *Advances in neural information processing systems*, pages 604–612, 2010.

[12] Nir Friedman and Daphne Koller. Being bayesian about network structure. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 201–210. Morgan Kaufmann Publishers Inc., 2000.

[13] Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine learning*, 50(1-2):95–125, 2003.

[14] Marco Grzegorczyk and Dirk Husmeier. Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2-3):265, 2008.

[15] Ru He, Jin Tian, and Huaiqing Wu. Structure learning in Bayesian networks of a moderate size by efficient sampling. *Journal of Machine Learning Research*, 17(1):3483–3536, 2016.

[16] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning bayesian network structure using lp relaxations. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 358–365, 2010.

[17] Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

[18] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[19] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. In *The International Conference on Learning Representations (ICLR)*, 2019.

[20] Lek-Heng Lim. Hodge laplacians on graphs. *Siam Review*, 62(3):685–715, 2020.

[21] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[22] David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232, 1995.

[23] Abdolreza Mohammadi, Ernst C Wit, et al. Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.

[24] Karthik Mohan, Mike Chung, Seungyeop Han, Daniela Witten, Su-In Lee, and Maryam Fazel. Structured learning of Gaussian graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[25] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] Siqi Nie, Denis D Mauá, Cassio P De Campos, and Qiang Ji. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[27] Jurg Nievergelt and Klaus H. Hinrichs. *Algorithms and Data Structures: With Applications to Graphics and Geometry*. Prentice-Hall, Inc., USA, 1993. ISBN 0134894286.

[28] Teppo Niinimaki, Pekka Parviainen, and Mikko Koivisto. Partial order MCMC for structure discovery in Bayesian networks. *arXiv preprint arXiv:1202.3753*, 2012.

[29] Sascha Ott, Seiya Imoto, and Satoru Miyano. Finding optimal models for small gene networks. In *Pacific symposium on biocomputing*, 2004.

[30] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, Inc., 2 edition, 1988.

[31] Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3(2):121–129, 2017.

[32] Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning bayesian networks with thousands of variables. In *Advances in neural information processing systems*, pages 1864–1872, 2015.

[33] Mark Schmidt, Alexandru Niculescu-Mizil, Kevin Murphy, et al. Learning graphical model structure using l1-regularization paths. In *AAAI*, volume 7, pages 1278–1283, 2007.

[34] Shohei Shimizu. LiNGAM: Non-Gaussian methods for estimating causal structures. *Behaviormetrika*, 41(1):65–98, 2014.

[35] Tomi Silander and Petri Myllymaki. A simple approach for finding the globally optimal Bayesian network structure. In *UAI*, 2006.

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

[37] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Computation, Causation, and Discovery*. AAAI Press, 1999.

[38] Peter Spirtes, Clark Glymour, Richard Scheines, Stuart Kauffman, Valerio Aimale, and Frank Wimberly. Constructing Bayesian network models of gene expression networks from microarray data, 2000.

[39] Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*, 2012.

[40] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[41] Gherardo Varando. Learning dags without imposing acyclicity. *arXiv preprint arXiv:2006.03005*, 2020.

[42] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[43] Changhe Yuan and Brandon Malone. Learning optimal bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

[44] Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[45] Xun Zheng. *Learning DAGs with Continuous Optimization*. PhD thesis, University of Pittsburgh Medical Center, 2020.

[46] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 9472–9483, 2018.

[47] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[48] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.