
Meta-D²AG: Causal Graph Learning with Interventional Dynamic Data

Tian Gao*
IBM Research

Songtao Lu*
The Chinese University of Hong Kong

Junkyu Lee
IBM Research

Elliot Nelson
Independent

Debarun Bhattacharjya
IBM Research

Yue Yu
Lehigh University

Miao Liu
IBM Research

Abstract

Causal discovery in the form of a directed acyclic graph (DAG) for dynamic time series data has been widely studied in various applications. In this work, we propose a dynamic DAG discovery algorithm, Meta-D²AG, based on online meta-learning. Meta-D²AG is designed to learn dynamic DAG structures from potentially nonlinear and non-stationary time series datasets, accounting for changes in both parameters and graph structures. Unlike most of the existing work focusing on observational, offline, and/or stationary settings, Meta-D²AG explicitly treats data collected at different time points with distribution shifts as distinct domains, which is assumed to occur as a result of external interventions. Moreover, Meta-D²AG involves a new online meta-learning framework to take advantage of the temporal transition among existing domains such that it can quickly adapt to new domains with few measurements. A first-order optimization approach is utilized to efficiently solve the meta-learning framework, and theoretical analysis establishes the identifiability conditions and the convergence of the learning process. We demonstrate the promising performance of the proposed meta learning framework through better accuracy on benchmark datasets against state-of-the-art baselines.

1 Introduction

Probabilistic graphical models [50, 32] and causal graphical models [51] have been an active area of research in machine learning. Dynamic probabilistic graphical models [3] are particularly suitable to capture dynamics in temporal data like time series by explicitly modeling how variables change over time. These probabilistic models have been successfully applied to real-world problems, such as for neuroscience [55], molecular biology [35] and computer vision [42] tasks.

Dynamic Bayesian networks (DBNs) [12, 43] are among the most popular dynamic graphical models in the literature, initially proposed for discrete-time models with discrete variables. Later, plenty of subsequent work extended these models to consider continuous variables that are more suitable for time series data since they often involve continuous-valued measurements. For example, DBNs are used for structured vector auto-regressive (SVAR) models in the statistics and econometrics literature [56, 57, 13, 64, 34, 30, 66, 65]. Parameter learning and directed acyclic graph (DAG) structure learning from *observational time series data* are the typical learning tasks for DBNs. One can categorize standard structure learning methods into score-based [23, 8, 5], constraints-based [62, 67, 10, 40, 17, 60], or more recently, neural-based methods [41]. A more tractable continuous optimization framework has been proposed based on the algebraic characterization of the DAG [77] to address super-exponential computational complexity in exact structure learning. This

*Equal Contributions. Corresponds to: {tgao@us.ibm.com, stlu@cse.cuhk.edu.hk}.

continuous optimization formulation [46, 70, 74] has been successfully extended to nonlinear neural models [73, 33, 26, 78, 71, 72] and dynamic DAG structure learning [49, 47, 25].

Despite its success, typical DAG learning has one major limitation. Often, the data distribution may change over time and the model parameters or even graph structures may fluctuate. For instance, in applications such as monitoring sensor networks, different sensors may be active at different times due to operating costs or weather conditions. Changes in city policies, urban development, or even major events (such as festivals or construction projects) can lead to temporal shifts in the data distribution. In retail, the sales of certain products may exhibit seasonal variations, such as increased demand for winter clothing during colder months. Standard DAG learning algorithms assume datasets involve samples that are independent and identically distributed (i.i.d.), ignoring correlations across time steps. Moreover, structure learning in DBNs over time series data generally ignores the distribution shift and assumes stationarity, which may affect the accuracy of the learned graphs. Datasets collected over a long period of time could contain distribution changes, such as monthly data or even yearly data. When these changes are significant, one may have to re-learn the graph from scratch, which loses relevant historical information and reduces learning efficiency, especially if some parameters or graph edges remain constant over time.

As a step toward addressing such non-stationary problems, we propose a meta learning based dynamic DAG learning approach, Meta-D²AG, to model such distribution shifts over time. We consider the data collected over different periods as different domains under the distribution shifts, and utilize historical data to learn a set of shared parameters across the whole domain and private parameters that adapt to domain-specific changes per domain. In Meta-D²AG, we view distribution shifts as caused by external interventions, e.g., updating the public health policy would influence the health record. Accordingly, we utilize the explicit intervention distribution when intervention targets are known, or the observational distribution when those targets are unknown.

The proposed framework employs a two-level optimization process to separate the invariant and domain-specific subspaces, and it can quickly adapt to a new data distribution and achieve higher graph learning accuracy with a few samples. This procedure is particularly beneficial when the data collector may not be able to capture a sufficient amount of time series data due to non-stationarity, making a low sample requirement crucial. Moreover, the chosen meta-learning framework is especially suitable for those non-stationary environments where some aspects of the environment evolve slowly or change periodically. This characteristic can make learning stationary components easier, as data across different interventions or domains are observed.

There have been a few works using a transfer-learning approach to handle invariant causal structures across domains, but they are mainly for reinforcement learning or observational time series data [11, 45, 37, 29]. There is also a line of works for multi-task DAG learning, but they require additional order information to be provided [6] and do not consider new domains in an online setting [2]. [25] considers a potential heterogeneous distribution, but it assumes a fixed graph structure where only parameters are assumed to change.

The major **contributions** of our work are as follows:

- We propose a novel meta-learning framework via bilevel optimization for time series DAG learning that explicitly models intervention as the source of distribution shift. It can handle both observational and interventional data distributions with nonlinear relationships by formulating a non-stationary collection of dynamic datasets as different domains.
- We propose an efficient first-order optimization approach to solve the proposed bilevel optimization framework with constraints in the lower level.
- Our theoretical analysis establishes the identifiability conditions in the batch setting and provides convergence rates in the online setting.

2 Background and Related Work

Consider a set of realizations of a potentially non-stationary time series, with each individual realization of size T in the form of $\mathbf{X}_t := [x_{t,i}]_{i=1}^d \in \mathbb{R}^d$. Here $t \in \{0, \dots, T\}$ represents the time index, $\mathbf{X}_t$ represents the observed values of all d number of variables in an observational or interventional time series dataset, and $x_{t,i}$ denotes the i -th component of $\mathbf{X}_t$. For simplicity of notation, we consider one sequence $\mathbf{X}_t$ without loss of generality.

A **causal Bayesian network** [51] is defined by a distribution P_X over a set of random variables $X \in \mathbb{R}^d$ and a DAG $\mathcal{G} = (V, E)$ with nodes V and edges E . Each node $V_i \in V = \{V_1, \dots, V_d\}$ is associated with a random variable x_i and each edge $(i, j) \in E$ represents a direct causal relation from variable x_i to x_j . With a slight abuse of notation, we will use X and V interchangeably. We assume the distribution P_X is Markov with respect to graph $\mathcal{G}$, which enables the factorized joint distribution as $P(X) = \prod_{j=1}^d p_j(x_j | x_{\pi_j^G})$, where π_j^G is the set of parents of node j in the graph $\mathcal{G}$ and x_B denotes the instantiations of a subset of X whose indices are $B \subset V$. We also assume causal sufficiency, i.e., there are no hidden common causes between any pair of variables in X [52]. As typically assumed in such models, there may exist instantaneous or contemporaneous influences as well as a time-delayed impact among variables.

Among many possible ways to model P_X in a time series dataset, we follow the typical setting in [49], characterizing X through a standard structured vector autoregressive (SVAR) model [13, 64, 30]:

$$\mathbf{X}_t = \mathbf{X}_t \mathbf{W}^a + \mathbf{X}_{t-1} \mathbf{W}_1^b + \dots + \mathbf{X}_{t-p} \mathbf{W}_p^b + \mathbf{Z}_t, \quad (1)$$

where $t \in \{p, \dots, T\}$ with horizon T , p is the autoregressive order, and $\mathbf{Z}_t \in \mathbb{R}^d$ is a vector of noise variables drawn from any continuous distribution. We assume that $\mathbf{Z}_t$ is independent of $\mathbf{Z}_{t' \neq t}$ and of $\mathbf{X}_{t'}$ for all $t' < t$. The $d \times d$ matrices $\mathbf{W}^a$ and $\mathbf{W}_i^b$, $i \in \{1, \dots, p\}$, represent weighted adjacency matrices for the intra-slice and inter-slice edges in $\mathcal{G}$, respectively, and they model the contemporaneous and time-lagged causal relations. Eq. 1 can be written in the matrix form as $\mathbf{X} = \mathbf{X} \mathbf{W}^a + \mathbf{Y}_1 \mathbf{W}_1^b + \dots + \mathbf{Y}_p \mathbf{W}_p^b + \mathbf{Z}$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a matrix whose rows are $\mathbf{X}_t$, $\mathbf{Z} \in \mathbb{R}^{n \times d}$ is a matrix formed similarly by $\mathbf{Z}_t$, and $\mathbf{Y}_j$, $j \in \{1, \dots, p\}$, are time lagged versions of $\mathbf{X}$. The number n is the effective sample size, which is equal to $T - p + 1$.

The goal of typical **causal structure learning** tasks is to recover the DAG $\mathcal{G}$ using samples from P_X and/or from the interventional distributions. We follow a continuous constrained optimization re-formulation for DAG learning [75] that uses a continuous DAG constraint, $h(\mathbf{W}) = 0$ on the weighted adjacency matrix $\mathbf{W}$ to avoid the combinatorial search on the feasible solutions $\mathbf{W}$:

$$\min_{\theta, \mathbf{W}} \mathcal{L}_\theta(\mathbf{X}; \mathbf{W}) + \lambda \Omega(\theta, \mathbf{W}) \quad \text{s.t.} \quad h(\mathbf{W}) = 0, \quad (2)$$

where $\mathcal{L}_\theta$ is the loss function and θ indicates parameters other than $\mathbf{W}$, $\Omega(\theta, \mathbf{W})$ is a regularization term on model parameters and/or the edge complexity in $\mathbf{W}$ with a tunable regularization parameter λ . [75] proposed $h(\mathbf{W}) = \text{Tr}(e^{\mathbf{W}}) - d$, where Tr represents the matrix trace, and showed that the graph is acyclic if and only if the constraint $h(\mathbf{W}) = 0$. Typically, the loss f_θ can be the least square loss [75] in linear structured equation models (SEM), or evidence lower bound [73], among other losses [26]. The problem can be approximately solved using the classic augmented Lagrangian method.

For time series datasets, DYNOTEARS [49] extends the continuous optimization framework to DBNs by explicitly modeling the intra-slice and inter-slice adjacency matrices separately with a linear SEM model:

$$\min_{\theta, \mathbf{W}^a, \mathbf{W}^b} \mathcal{L}_\theta(\mathbf{X}; \mathbf{W}^a, \mathbf{W}^b) + \lambda \Omega(\theta, \mathbf{W}^a) \quad \text{s.t.} \quad h(\mathbf{W}^a) = 0, \quad (3)$$

where $\mathbf{W}^a$ is the matrix for intra-slice edges and $\mathbf{W}^b$ is the matrix for inter-slice connections, and $\mathcal{L}_\theta(\mathbf{X}; \mathbf{W}^a, \mathbf{W}^b) = \frac{1}{n} \|\mathbf{X} - \mathbf{X} \mathbf{W}^a - \mathbf{Y} \mathbf{W}^b\|_F^2$. Here n is the total sample size, and $\|\cdot\|_F$ is the Frobenius norm. To distinguish with the Frobenius norm, we use $\|\cdot\|_2$ to denote the (vector) ℓ_2 -norm.

Beyond observational time series data, learning a causal graph structure in Markov Decision Process (MDP) requires additional machinery to model actions, and this can be seen as interventional dynamic DAG learning. Let a binary indicator matrix $\mathbf{R}^\mathcal{I} = [r_{kj}^\mathcal{I}] \in \{0, 1\}^{K \times d}$ encode the interventional family $\mathcal{I}$ such that $r_{kj}^\mathcal{I} = 1$ when x_j is an intervened target in I_k , and 0 otherwise. For each intervention family k , we denote the weighted intra-slice adjacency matrix by $\mathbf{W}_{(k)}^a \in \mathbb{R}^{d \times d}$ and the weighted p -order inter-slice adjacency matrix by $\mathbf{W}_{(k)}^b \in \mathbb{R}^{pd \times d}$. Let $\mathbf{W}_{(k)} = \{\mathbf{W}_{(k)}^a, \mathbf{W}_{(k)}^b\}$, and $\mathbf{W} = \{\mathbf{W}_{(1)}, \dots, \mathbf{W}_{(K)}\}$ represent the collection of all matrices over K interventional families. Then, IDYNO [18] formulates the continuous optimization as

$$\min_{\mathbf{W}, \mathcal{G}, \theta} \frac{1}{n} \sum_{k=1}^K \sum_{j=1}^d [\mathcal{L}_{\theta_j}(\mathbf{X}, \text{MLP}(\mathbf{X}; \theta_j, \mathbf{W}_{(1)}))^{1-r_{kj}^\mathcal{I}} \cdot \mathcal{L}_{\theta_j}(\mathbf{X}, \text{MLP}(\mathbf{X}; \theta_j, \mathbf{W}_{(k)}))^{r_{kj}^\mathcal{I}}] \quad \text{s.t.} \quad h(\mathbf{W}) = 0$$

where θ_j is the parameter of one MLP that predicts the value of each variable X_j .

The above learning procedure has several limitations: 1) it needs offline data, meaning that the data needs to be saved prior to the learning process and they cannot adapt to new data when distributions change, and 2) the interventional family is fixed and limited to a known fixed number K , while K changes over time in practice. Ultimately, the method neglects the sequential and non-stationary aspects of time series problems. This problem is also relevant in reinforcement learning, where the policies are learned and updated online, and hence the distributions and underlying MDP may change particularly around the action variables.

3 Meta-Learning for Dynamic DAG Learning

We focus on DAG learning for times series data, which can quickly adapt to non-stationary distributions and graphs. First, we propose a generic meta DAG learning framework via bilevel optimization for time series data when all the data are given under the batch or offline setting. Then, we extend the framework to the online setting, where new data from different domains are constantly arriving. Recall that existing work on online learning of DAG structures [9, 31] have largely focused on parameter learning settings or assumed i.i.d data. Finally, we present an efficient online Meta-D²AG an algorithm based on the single-level penalty-based bilevel gradient descent.

3.1 Problem Formulation with Meta DAG Learning

Consider a non-stationary time series setting where data can be divided into Q disjoint domains $\{1, \dots, Q\}$. In each domain q , time series $\mathbf{X}_q = [\mathbf{X}_{q,t}]_{t=1}^T$ is stationary; however, one or more variables may form different distributions across the domains. The goal is to recover a set of causal DAGs, represented with weighted adjacency matrices $\mathbf{W}_q = \{\mathbf{W}_q^a, \mathbf{W}_q^b\}_{q=1}^Q$, governing the underlying data generation process. Given the potential for variation across domains, a desired property of effective learning algorithms is the ability to use minimal samples at each domain. This can be achieved by leveraging historical data from previous domains that share similarities with the current domain. A meta-learning approach is well-suited to meet this requirement.

Meta Learning. A common view of meta-learning [24] is to learn a general purpose algorithm that can generalize across a set of tasks $\mathcal{M}$ with shared parameters, and learn a new task more efficiently given the previous tasks. The goal is to evaluate the expected loss of data over a distribution over tasks $\mathcal{M}$. In general, meta-learning algorithms require a set of meta-training and meta-testing tasks drawn from the distribution $p(\mathcal{T})$. This way we can learn new task-specific parameters quickly with few samples. Meta-learning has been applied to sequential data and in particular reinforcement tasks [16, 44, 68, 54, 79], and continuous learning [59, 20].

In a standard two-step meta-learning procedure, domains Q are divided into source M and target U datasets, $|Q| = |M| + |U|$, for meta-train and meta-test steps. The meta-training step uses a set of M tasks with data $\mathcal{D}_{\text{source}} = \{(\mathcal{D}_m^{\text{train}}, \mathcal{D}_m^{\text{test}})\}_{m=1}^M$. Let Φ_s and Φ_p be the shared parameters over all tasks and the private parameters specific to each particular task, respectively. The meta-training stage aims to optimize loss function $\mathcal{L}$ by

$$\Phi_s^* = \arg \min_{\Phi_s} \log \mathcal{L}(\Phi_s | \mathcal{D}_{\text{source}}).$$

The meta-test stage finds the optimal parameters Φ_p for each specific task with data $\mathcal{D}_{\text{target}} = \{(\mathcal{D}_u^{\text{train}}, \mathcal{D}_u^{\text{test}})\}_{u=1}^U$ by

$$\Phi_p^* = \arg \min_{\Phi_p} \log \mathcal{L}(\Phi_p, \mathcal{D}_{\text{target}} | \Phi_s^*, \mathcal{D}_{\text{source}}).$$

However, our proposed setting is slightly different. The goal of each task is to learn private parameters $\Phi_{p,m}$ and corresponding graph structure $\mathbf{W}_m$ with few samples $\mathcal{D}_{\text{source}}$ and shared parameters Φ_s in meta-training. One can indeed adapt to data from a new domain u and infer $\Phi_{p,u}$ in meta-testing, but since DAG learning is an unsupervised learning task, the meta-test stage is optional. Note that none of the existing meta-learning algorithms focuses on learning a causal graph that is applicable to general interventional time series datasets.

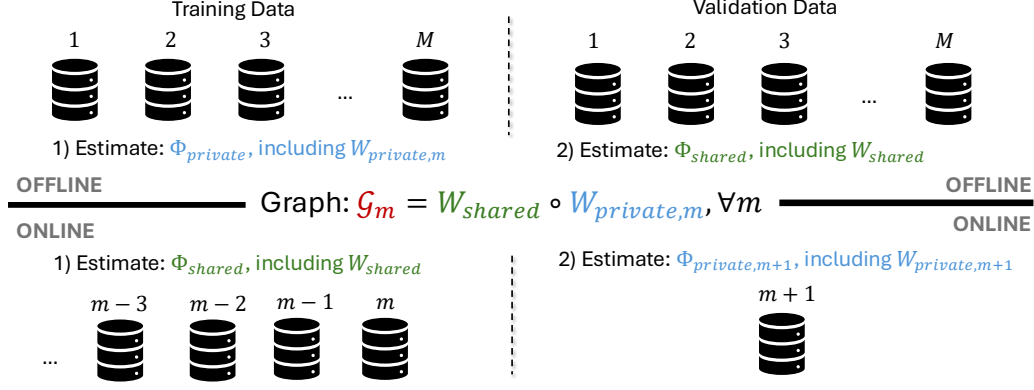


Figure 1: Overview of the proposed Meta-D2AG framework, with offline and online settings. Upper Half: In the offline setting, Meta-D²AG uses different sets of data to iteratively learns the private and shared parameters of each domain to infer the causal graph A_m for each domain; Lower Half: Meta-D²AG can also adapt to the online setting by using data of incoming domains to infer domain-specific private graphs, given the shared parameters learned from history.

3.2 Meta-Learning for Dynamic DAG

We propose a novel meta-learning framework [39] for dynamic causal DAG learning in the presence of the non-stationarity of parameters and graphs by treating each segment of the time series as a different domain. We define the private parameters $\Phi_p = \{W_p^a, W_p^b, \Theta_p\}$, and the shared parameters $\Phi_s = \{W_s^a, W_s^b, \Theta_s\}$, where Θ_p or Θ_s include other model parameters such as linear or nonlinear SEMs. Since DAG learning is an unsupervised task, we should modify the typical meta-learning setting accordingly. The source dataset contains a set of training data and potentially a set of validation data, i.e., $\mathcal{D}_{\text{source}} = \{(\mathbf{X}_m^{\text{train}}, \mathbf{X}_m^{\text{val}})\}_{m=1}^M$. $\mathbf{X}_m^{\text{train}}$ is used for training all M private task parameters, and $\mathcal{D}_m^{\text{val}}$ is to train shared parameters Φ_s . This batch or offline setting is illustrated by the upper half of Figure 1. When desired, the target dataset contains a new dataset to learn a DAG in new domains U , i.e., $\mathcal{D}_{\text{target}} = \{(\mathbf{X}_u^{\text{train}})\}_{u=1}^U$. $\mathbf{X}_u^{\text{train}}$ is then used to learn private parameters $\Phi_{p,u}$ in new task u , given shared parameters Φ_s .

As such, we formulate our *meta gradient-based dynamic DAG learning framework* (Meta-D²AG) as a **bilevel optimization** problem. In meta-training, the objective with the upper level loss ℓ and the lower level g can be written as:

$$\min_{\Phi_s} \frac{1}{M} \sum_{m=1}^M \ell_m(\Phi_s, \Phi_{p,m}^*; \mathbf{X}_m^{\text{val}}) \quad (4a)$$

$$\text{s.t. } \Phi_{p,m}^*(\Phi_s) = \arg \min_{\Phi_{p,m}} g_m(\Phi_s, \Phi_{p,m}; \mathbf{X}_m^{\text{train}}), \forall m, \quad \text{s.t. } h(\mathbf{W}_s, \mathbf{W}_{p,m}) = 0 \quad (4b)$$

where Φ_s indicates the shared parameters over different training domains and $\Phi_{p,m}$ indicates the private parameter evolving at each domain. n is the sample size of each domain. We use a ℓ_1 -regularized least square loss for ℓ_m and g_m in each domain, i.e., $\frac{1}{2n} \|\mathbf{X}_m - f_m(\Phi_s, \Phi_{p,m}^*, \mathbf{X}_m)\|_F^2$ with either $+\eta \|\Phi_s\|_1$ or $+\eta \|\Phi_{p,m}\|_1$, where f_m is the model function for each domain m , such as SEMs, and contains other trainable domain-specific parameters. $\mathbf{X}_m$ is either the train or validation data, and we will drop it to simplify notations as $\ell_m(\Phi_s, \Phi_{p,m}^*)$ and $g_m(\Phi_s, \Phi_{p,m}^*)$.

In meta-testing, one can solve the inner loop to obtain a new $\Phi_{p,m}^*$. Recall that this bilevel formulation is different from the standard meta-learning formulation in that the lower level (or the inner loop) optimization inside the constraint contains another constraint $h(\mathbf{W}_s, \mathbf{W}_{p,m}) = 0$. This poses a more complex meta-learning formulation with additional difficulties to satisfy constraints. Note that this formulation can also apply to DAG learning from i.i.d. data with multiple domains. Note also that we only enforce the acyclicity constraint h at the lower level, as the upper-level constraints would be redundant.

Remark 3.1. The meta formulation can reduce to the vanilla continuous optimization approach [49] with a single domain, when the outer-loop optimization is not needed. Hence, the formulation is more general and covers existing continuous DAG learning frameworks.

3.3 Online Meta-Learning for Dynamic DAG

In dynamic datasets, domains in time series datasets may not be independent of each other, given the innate dependency overtime, even if there exists non-stationarity. Hence, we assume that new domain parameters are a function of the new time sequences D_i and the parameters in the previous tasks, i.e., $\Phi_q = f(\Phi_1, \dots, \Phi_{q-1})$. Namely, all previous datasets with index $< q$ should be used to train the shared parameters for q dataset for the maximal utilization of information. The source dataset contains $\mathcal{D}_{\text{source}} = \{(\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{val}})\}_{i=1}^M$, with $\mathcal{D}_{<m}^{\text{train}}$ is used for training private domain-specific parameters for each domain m . Similarly, $\mathcal{D}_{<m}^{\text{val}}$ is used to train shared parameters. Since the online sequence can be arbitrarily long, we can determine the total number of the historical set M to train the shared parameters. On the other hand, the target dataset $\mathcal{D}_{\text{target}} = \{(\mathcal{D}_j^{\text{train}})\}_{j=1}^U$ contains $\mathcal{D}_m^{\text{train}}$ to learn private parameters in new domains given the shared parameters Φ_s , and U can be a set of sequential domains sampled directly after M domains. To further refine the model in an online fashion, we could consider some $\mathcal{D}_j^{\text{val}}$ for the additional meta-training procedure to update Φ_s . This online setting is illustrated by the bottom half of Figure 1.

As such, the modified online meta gradient-based dynamic DAG learning framework at the current domain m can be written as

$$\min_{\Phi_s} \frac{1}{m} \sum_{i=1}^m \ell_i(\Phi_s, \Phi_{p,i}^*) \quad (5a)$$

$$\text{s.t. } \Phi_{p,i}^*(\Phi_s) = \arg \min_{\Phi_{p,i}} g_i(\Phi_s, \Phi_{p,i}), \forall i, \quad \text{s.t. } h(\mathbf{W}_s, \mathbf{W}_{p,i}) = 0, \quad (5b)$$

Relation between $\mathcal{G}$ and $(\mathbf{W}_s, \mathbf{W}_p)$. To instantiate the bilevel optimization framework to learn graph $\mathcal{G}$, we can choose to use a Hadamard product as $\mathcal{G}_i = \mathbf{W}_s \circ \mathbf{W}_{p,i}$ with linear SEMs. In nonlinear SEMs with neural networks, we adopt an existing approach [78] by including $\mathbf{W}_s$ and $\mathbf{W}_p$ as the first layer parameters. Putting everything together with a two-layer feedforward neural network with domain-specific parameter $\theta_{p,i}$ as an example and using the least square loss, the proposed formulation has the following forms:

$$\min_{\mathbf{W}_s} \frac{1}{m} \sum_{i=1}^m \frac{\lambda_i}{2n} \|\mathbf{X}_i - \theta_{p,i}(\mathbf{W}_s) \sigma((\mathbf{W}_{p,i}(\mathbf{W}_s) \circ \mathbf{W}_s) \mathbf{X}_i)\|_2^2 + \eta \|\mathbf{W}_s\|_1 \quad (6a)$$

$$\text{s.t. } \mathbf{W}_{p,i}^*(\mathbf{W}_s), \theta_{p,i}^*(\mathbf{W}_s) = \arg \min_{\mathbf{W}_{p,i}} \frac{1}{2n} \|\mathbf{X}_i - \theta_{p,i} \sigma((\mathbf{W}_{p,i} \circ \mathbf{W}_s) \mathbf{X}_i)\|_2^2, \forall i, \quad (6b)$$

$$\text{s.t. } h(\mathbf{W}_s \circ \mathbf{W}_{p,i}) = 0 \quad (6c)$$

Interventional loss can be used when such targets are known, following equation 2.

Remark 3.2. The choice of multiplying shared and private parameters is inspired by existing meta-learning approaches and the masking approach in DAG learning [48]. As a result, in order to recover correct graphs in each domain, $\mathbf{W}_s$ has to include the edges of all $\mathbf{W}_{p,i}$, hence it is unlikely to be a DAG alone. An alternative formulation $\mathcal{G}_i = \mathbf{W}_s + \mathbf{W}_{p,i}$, which is also typical in many meta-learning literature [15], could be used, but it may be slightly more restrictive as it assumes all adjacency matrices share similar weights.

3.4 Penalty-based Bilevel Proximal Gradient Descent

Solving the above bilevel optimization is not straightforward, as there is an additional constraint in the inner loop. We propose a penalty-based bilevel gradient descent inspired by [61] to solve the problem. The problem can be written in a general form of

$$\min_{\Phi_s, \{\Phi_{p,i}\}} \frac{1}{m} \sum_{i=1}^m \ell_i(\Phi_s, \Phi_{p,i}) \quad (7a)$$

$$\text{s.t. } \Phi_{p,i} \in \mathcal{S}_i(\Phi_s) := \arg \min_{\Phi_{p,i}} g_i(\Phi_s, \Phi_{p,i}), \forall i, \quad \text{s.t. } h(\Phi_s, \Phi_{p,i}') = 0, \quad (7b)$$

where $\mathcal{S}_i(\Phi_s)$ denotes the optimal solution set of the i th lower-level problem.

Towards this end, by moving both the objectives and constraints in the inner loop as different penalty terms to the outer loop, we can write the penalized objective as:

$$\mathcal{L}(\Phi_s, \{\Phi_{p,i}\}; \gamma_i, \nu_i) = \frac{1}{m} \sum_{i=1}^m \ell_i(\Phi_s, \Phi_{p,i}) + \gamma_i (g_i(\Phi_s, \Phi_{p,i}) - g_i^*(\Phi_s)) + \nu_i h(\Phi_s, \Phi_{p,i}), \quad (8)$$

where $\{\gamma_i, \nu_i\}$ and $g_i^*(\Phi_s) = \min_{\Phi_{p,i}} g_i(\Phi_s, \Phi_{p,i})$ are penalty coefficients. The second term of equation 8 penalizes the constraints directly in the objective. One issue with equation 7a is that m can be large and computation would become expensive for long-history domains. As in typical online algorithms, it is prudent to use a sliding window to capture the historical influence as follows.

$$\min_{\Phi_s, \{\Phi_{p,i}\}} \frac{1}{w} \sum_{i=m-w+1}^m \ell_i(\Phi_s, \Phi_{p,i}) \quad (9a)$$

$$\text{s.t. } \Phi_{p,i} \in \mathcal{S}_i(\Phi_s) := \arg \min_{\Phi'_{p,i}} g_i(\Phi_s, \Phi'_{p,i}), \forall i, \quad \text{s.t. } h(\Phi_s, \Phi'_{p,i}) = 0, \quad (9b)$$

where w denotes the size of the sliding window.

Sliding Window Formulation. Due to the inherent nonlinearity of neural networks, the corresponding objective functions exhibit non-convex properties. [22] has shown that it is possible to measure the regret using a local regret metric, allowing us to establish a regret bound based on a smooth nonconvex objective function. Then, the penalty-based sliding average is:

$$S_{i,w}(\Phi_s, \{\Phi_{p,i}\}) = \frac{1}{w} \sum_{i=m-w+1}^m \ell_i(\Phi_s, \Phi_{p,i}) + \gamma_i (g_i(\Phi_s, \Phi_{p,i}) - g_i^*(\Phi_s)) + \nu_i h(\Phi_s, \Phi_{p,i}). \quad (10)$$

As such, the optimization becomes a single level and we can adopt a proximal gradient descent method with a penalty function to optimize $\Phi = (\Phi_s, \Phi_{p,i})$ together. For a more detailed discussion of the algorithm and proofs on the following convergence property, we refer to Appendix A.1.

Algorithm 1 Online Meta-D²AG Algorithm

Require: Data $\mathbf{X}_m, m \in \{1, \dots, M\}, w, \eta, \gamma$ 1: Output: Learned shared weighted adjacency matrix $\mathbf{W}_s$ and private $\mathbf{W}_{p,m}, \forall m$ 2: initialize $\mathbf{W}_s$ and $\mathbf{W}_{p,0}$ 3: for each domain m do 4: Initialize $\Phi_{p,m} := \Phi_{p,m-1}$ 5: while equation 11 holds do	6: $g_i^*(\Phi_s) = \min_{\Phi_{p,i}} g_i(\Phi_s, \Phi_{p,i})$ {Minimize inner loop} 7: Compute $S_{i,w}$ with equation 10 8: $\Phi \leftarrow \text{prox}_{\eta f}(\Phi - \eta \nabla_{\Phi} \widehat{S}_{i,w}(\Phi))$ 9: end while 10: end for
---	--

As there is a non-smooth term in the objective, we adopt the proximal residual, i.e., $\mathcal{P}_{\eta}^f(x, d) = \eta^{-1}(x - \text{prox}_{\eta f}(x - \eta d)), \forall \eta > 0$ [21] in the optimality gap to quantify the local regret, where $\text{prox}_{\eta f}(v) := \min_x \eta f(x) + \frac{1}{2} \|x - v\|_2^2$ denotes the proximal operator, and d is any vector, $f(x) = \|x\|_1$ and $\text{prox}_{\eta f}(v)$ is the soft-thresholding operator. To summarize, we show the online algorithm in Algorithm 1, where $\nabla_{\Phi} \widehat{S}_{i,w}(\Phi)$ denotes an approximate evaluation of $\nabla_{\Phi} S_{i,w}(\Phi)$ obtained by optimizing $\min_{\Phi_{p,i}} g_i(\Phi_s, \Phi_{p,i})$ using any optimization algorithm practically, rather than evaluating it exactly at $\Phi_{p,i} \in \mathcal{S}_i(\Phi_s)$.

4 Theoretical Results

Convergence Before presenting our theoretical convergence results, we first introduce a set of mathematical assumptions regarding the problem.

Assumptions 1. We assume that $g(\cdot), h(\cdot)$, and $\ell_i(\cdot)$ for all i are differentiable and Lipschitz continuous. Moreover, for every i , $g_i(\Phi_s, \cdot)$ is gradient Lipschitz continuous with constant L_g and satisfies the proximal Polyak-Łojasiewicz (PL) error bound.

Note that these assumptions are standard and quite moderate. For example, in the applications of DAG learning, the loss functions are typically score functions, which are smooth. In addition, the causal structure constraint $h(\cdot)$ for DAG is also smooth. Since the non-smooth component involves the ℓ_1 -norm, it effectively penalizes variable magnitudes, which in turn ensures the boundedness of the iterates. This observation consequently implies the Lipschitz continuity of the loss functions.

The assumption of the proximal PL error bound [27] is primarily employed here to make the optimization process of the lower-level (or inner loop) more tractable, a condition that holds for a broad spectrum of non-convex loss functions. Specifically, when the neural network size is large, i.e., overparameterized, the associated loss function satisfies the PL condition. In this context, due to the presence of the non-smooth term, we utilize the proximal PL error bound. Under this condition, a few steps of gradient descent can approximately find the global optimal solution, meaning that the last iterates of the gradient descent can be used to approximate g_i^* , allowing $S_{i,w}(\Phi)$ to be easily evaluated, denoted as $\hat{S}_{i,w}(\Phi)$, where Φ is the concatenation of all the model parameters. This approach is both practical and can be rigorously quantified theoretically.

The convergence criterion in Line 5 of Algorithm 1 is based on the sum of gradients in the sliding window:

$$\|\mathcal{P}_\eta^f(\Phi_{i+1}, \nabla \hat{S}_{i,w}(\Phi_{i+1}))\| > \delta/w, \quad (11)$$

where δ is a constant, which means that if the proximal residual is not sufficiently small, we continue optimizing the model parameters using the current domain data.

Theorem 4.1. Regret Bound. *Under these assumptions, suppose that $\Phi := \{\Phi_s, \Phi_{p,i} \mid i = m - w + 1, \dots, m\}$ is generated by the algorithm. Then, the following holds:*

$$\text{Reg}_w(m) = \sum_{i=1}^m \|\mathcal{P}_\eta^f(\Phi_i, \nabla S_{i,w}(\Phi_i))\|^2 = \mathcal{O}\left(\frac{m}{w^2}\right). \quad (12)$$

Remark 4.2. Since there are two levels in the optimization process, quantifying convergence requires solving the lower-level problems, unlike classic single-level problems. Thus, our proof must account for errors in solving both levels. Due to the nonconvexity of the loss function, we use the sliding window concept to demonstrate the stationarity of the solutions obtained.

The above regret indicates that the changes in the sliding-window task-averaged functions converge to a small value when the sliding window size is large, meaning that $\mathbb{E}_i \|\mathcal{P}_\eta^f(\Phi_i, \nabla S_{i,w}(\Phi_i))\|^2 \leq \text{Reg}_w(m)/m = \mathcal{O}(w^2)$. It is worth noting that this algorithm achieves the optimal regret bound, as equation 12 matches the optimal regret of the time-smoothed online proximal gradient method for solving classic single-level nonconvex minimization problems [22, 21].

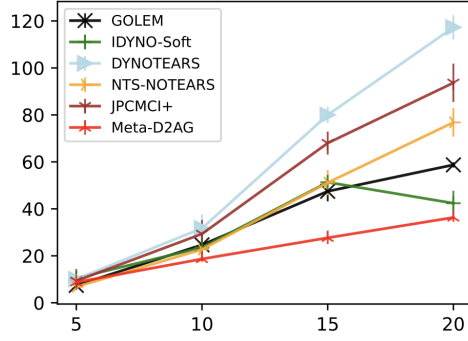
Other Assumptions We also make a few standard assumptions for dynamic DAG learning [49], such as causal sufficiency (no latent variables), following an SEM model (linear or nonlinear), and sampling frequency of the event sequence is at least as high as the fluctuations in the underlying causal process. However, we relax the typical assumption that the graph structure is fixed over time, and assume that it is only fixed within certain time steps (as fixed within a domain in the meta formulation).

Identifiability In Appendix A.2, we demonstrate that, under a mild condition on the shared parameters $\mathbf{W}_s$ estimated from previous domains, the earlier identifiability results from [4, 18] can be extended to our method for learning the graph with adjacency matrix $\mathbf{W}_s \circ \mathbf{W}_{p,m}$ within a given domain m .

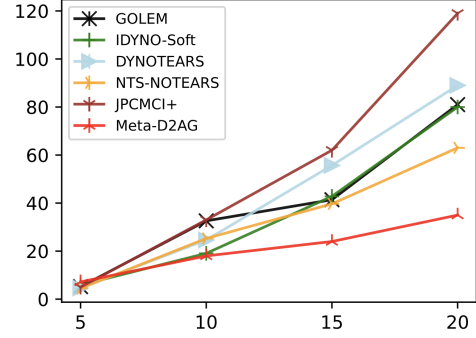
Complexity Our general meta-learning framework has the same computation complexity, with respect to the number of nodes, as existing continuous optimization approaches. The additional computation in our formulation mainly comes from the lower-level optimization, where we need to learn a DAG for each of M domains. The upper level can be solved faster without any constraints.

5 Empirical Evaluation

We evaluate our proposed meta dynamic DAG learning algorithm against some of the existing DAG learning algorithms, designed for dynamic time series datasets and/or with multi-domain capabilities.



(a) SHD results v.s. different variable size d of different algorithms on synthetic datasets (batch setting).



(b) SHD results v.s. different variable size d of different algorithms on synthetic datasets (online setting).

We test the algorithms on synthetic datasets as well as a simulated reinforcement learning (RL) environment, Sprites World, with intervention datasets. The experiments are run on a machine with a 3.2 GHz CPU and 64 GB of memory.

Baselines We compare our method with the following baselines with available code: continuous DAG learning methods GOLEM [47], dynamic DAG learning algorithm DYNOTEARS [49], NTS-NOTEARS [63], offline interventional dynamic DAG learning algorithms IDYNO [18], and multi-domain dynamic DAG algorithm J-PCMCI+ [19]. We repeat each experiment 5 times and compare the structural Hamming distance (SHD) between the estimated graph and the ground truth graph (the lower, the better). All baselines except J-PCMCI+ are designed to handle one domain, so we learn each domain separately and report the average SHD with standard deviation. J-PCMCI+ only finds one unique graph across all domains so we use it as the graph in each domain.

5.1 Synthetic Datasets

We first evaluate different approaches on synthetic time series datasets. We simulate the data according to the SEM from equation 1, mostly following the setup and code from [49] for ease of comparison. The generating process consists of the following three steps: 1) Generating weighted graphs in the form $\mathbf{W}^a$ and $\mathbf{W}^b$. Directed graphs are generated first as a lower triangular matrix, and then randomly permuted, 2) Generating data matrices $\mathbf{X}$ and corresponding $\mathbf{Y}$ per $\mathbf{W}^a$ and $\mathbf{W}^b$, and 3) Generating interventional data with random targets and different distributions. For a more detailed generation and training process, we refer readers to Appendix B.2.

We show results for the batch setting where all 10 domains are learned jointly per equation 4, and for the online setting where we use a sliding window size of 3 previous time domains to meta-train and the current domain as the meta-test dataset. For the batch setting, since all domains are used in meta-training and there is no new meta-test data, we report the SHD of training accuracy of each domain. For the online setting, we report SHD from the meta-test domain at each time step, given the previous 3 domains as meta-training (padding in the beginning with the data from the first domain). For baseline methods in the online setting, to distinguish from the batch setting where each domain are learned separately, we feed all domain data up to the current time stamp to each method and produce a graph.

As one can see from Figure 2a, in the batch setting, our method (colored in red) has the lowest SHD. Since each domain has different graphs, it hurts the performance of J-PCMCI+. For the online setting shown in Figure 2b, Meta-D²AGs again superior to baselines. In an appendix, we also conduct an ablation study with two variants of our method, one only optimizing the shared parameters and one with a linear loss function, showing the advantage of the proposed formulation. Moreover, in Meta-D²AG the DAG constraint follows the matrix exponential form as used previously by [49]. Other and better DAG constraints can also be used, such as [1, 76].

Ablation Study In the appendix, we have included further results of many ablation studies of our approach, including the necessity of the lower level optimization on private parameters and the impact

of sequence lengths and variable variance. Overall, the proposed Meta-D²AG shows promising performance in these settings.

5.2 Sprites World RL Environment

Sprites world [69] is a Python-based RL environment in which objects of various shapes interact with each other in 2-dimensional space. In the experiment, we created 5 and 10 objects, each associated with a continuous variable that assigns values following a linear structural equation model or non-linear structural equation model similar to causal discovery benchmarks [28, 53]. For further details about the data generation and training process, we refer readers to Appendix B.3.

As shown in Table 1, comparing with the baselines, our method achieves the lowest SHD performance for 5 and 10 objects, again demonstrating promising graph discovery performance in an RL environment.

Table 1: SHD performance of different methods in Sprites World.

Method	$O = 5$	$O = 10$
GOLEM	6.0 ± 1.3	14 ± 2.4
IDYNO	5.8 ± 1.7	12 ± 3.0
DYNOTEAR	11 ± 2.3	39 ± 1.3
JPCMCI+	12 ± 1.7	31 ± 0.6
Meta-D ² AG	5.2 ± 0.6	9.1 ± 1.5

6 Conclusion

We propose a new Meta-D²AG method for causal graph discovery in dynamic datasets. Leveraging the meta-learning framework to capture temporal transitions between different time domains is well-suited for causal graph discovery in dynamic datasets. We make three major contributions. First, we formulate a meta-learning framework tailored for dynamic DAG learning, accommodating slightly varied distributions in different domains over time. We further extend the framework to an online setting, where a windowed formulation is proposed to improve learning efficiency in scenarios with extensive historical data. Second, recognizing the challenges posed by acyclicity constraints in inner-level optimization, we propose an efficient first-order method to solve the bilevel framework. Finally, theoretical analysis and experimental evaluations are provided to show the convergence of the learning process and the superiority in empirical performances. Future research could explore methods for the automatic detection of potential distribution shifts without the need for such a priori information. For example, distribution shift detection is an active area of research, and we can adopt various strategies, such as measuring prediction error, using distribution distance, and/or hypothesis testing to detect shifts [38] and form a new domain. It would also be interesting to combine the proposed approach with irregular [7] or sub-sampled time series [36].

Limitation and Impact Statement One limitation of our approach is its reliance on predefined domains in time series datasets, which may be restrictive when distribution shifts occur irregularly. Obtaining such partitions of temporal data when no domain labels are provided would be a useful future direction. This paper presents work whose goal is to advance the field of causal discovery algorithms. Although there are no substantial potential societal consequences of our work, misuse could result in wrong conclusions about the causal relationships, which need to be carefully empirically validated in a controlled environment.

Acknowledgments and Disclosure of Funding

The authors thank anonymous reviewers for their constructive feedback. The work of Songtao Lu is supported in part by the Chinese University of Hong Kong (CUHK) Direct Research Grant (Project No. 4055259). Yue Yu is partially supported by the National Science Foundation (NSF) under award DMS-2436624 and the Defense Advanced Research Projects Agency (DARPA) under award HR00112590032. This research was, in part, funded by the U.S. Government by an agreement with Cornell University. The views and conclusions contained in this document are those of the authors and do not necessarily reflect the views of Cornell University, and should not be interpreted as representing the official policies, either expressed or implied of the U.S. Government.

References

- [1] K. Bello, B. Aragam, and P. Ravikumar. Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization. *Advances in Neural Information Processing Systems*, 35:8226–8239, 2022.
- [2] J. Berrevoets, N. Seedat, F. Imrie, and M. van der Schaar. Differentiable and transportable structure learning. *arXiv preprint arXiv:2206.06354*, 2022.
- [3] J. Bilmes. Dynamic graphical models. *IEEE Signal Processing Magazine*, 27(6):29–42, 2010.
- [4] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, pages 21865–21877, 2020.
- [5] R. Chen, S. Dash, and T. Gao. Integer programming for causal structure learning in the presence of latent variables. In *International Conference on Machine Learning*, pages 1550–1560. PMLR, 2021.
- [6] X. Chen, H. Sun, C. Ellington, E. Xing, and L. Song. Multi-task learning of order-consistent causal graphs. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 11083–11095, 2021.
- [7] Y. Cheng, L. Li, T. Xiao, Z. Li, J. Suo, K. He, and Q. Dai. Cuts+: High-dimensional causal discovery from irregular time-series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11525–11533, 2024.
- [8] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.
- [9] I. Cohen, A. Bronstein, and F. G. Cozman. Online learning of bayesian network parameters. *Hewlett Packard Laboratories Technical Report, HPL-2001-55 (R. 1)*, 2001.
- [10] D. Colombo, M. H. Maathuis, M. Kalisch, and T. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294–321, 2012.
- [11] I. Dasgupta, J. Wang, S. Chiappa, J. Mitrovic, P. Ortega, D. Raposo, E. Hughes, P. Battaglia, M. Botvinick, and Z. Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019.
- [12] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.
- [13] S. Demiralp and K. D. Hoover. Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, 65:745–767, 2003.
- [14] D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018.
- [15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
- [16] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.
- [17] S. Gao, R. Addanki, T. Yu, R. Rossi, and M. Kocaoglu. Causal discovery in semi-stationary time series. *Advances in Neural Information Processing Systems*, 36, 2024.
- [18] T. Gao, D. Bhattacharjya, E. Nelson, M. Liu, and Y. Yu. Idyno: Learning nonparametric dags from interventional dynamic data. In *International Conference on Machine Learning*, pages 6988–7001. PMLR, 2022.
- [19] W. Günther, U. Ninad, and J. Runge. Causal discovery for time series from multiple datasets with latent contexts. *arXiv preprint arXiv:2306.12896*, 2023.

- [20] G. Gupta, K. Yadav, and L. Paull. La-maml: Look-ahead meta learning for continual learning. *arXiv preprint arXiv:2007.13904*, 2020.
- [21] N. Hallak, P. Mertikopoulos, and V. Cevher. Regret minimization in stochastic non-convex learning via a proximal-gradient approach. In *Proceedings of International Conference on Machine Learning*, pages 4008–4017. PMLR, 2021.
- [22] E. Hazan, K. Singh, and C. Zhang. Efficient regret minimization in non-convex games. In *Proceedings of International Conference on Machine Learning*, pages 1433–1441. PMLR, 2017.
- [23] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
- [24] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.
- [25] B. Huang, K. Zhang, J. Zhang, J. Ramsey, R. Sanchez-Romero, C. Glymour, and B. Schölkopf. Causal discovery from heterogeneous/nonstationary data. *The Journal of Machine Learning Research*, 21(1):3482–3534, 2020.
- [26] D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, and M. Sebag. SAM: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*, 2018.
- [27] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.
- [28] N. R. Ke, A. Didolkar, S. Mittal, A. Goyal, G. Lajoie, S. Bauer, D. Rezende, Y. Bengio, C. Pal, and M. Mozer. Systematic evaluation of causal discovery in visual model based reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.
- [29] N. R. Ke, J. Wang, J. Mitrovic, M. Szummer, D. J. Rezende, et al. Amortized learning of neural causal representations. *arXiv preprint arXiv:2008.09301*, 2020.
- [30] L. Kilian. Structural vector autoregressions. In *Handbook of Research Methods and Applications in Empirical Macroeconomics*. Edward Elgar Publishing, 2013.
- [31] D. Kocacoban and J. Cussens. Online causal structure learning in the presence of latent variables. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 392–395. IEEE, 2019.
- [32] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [33] S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien. Gradient-based neural DAG learning. In *International Conference on Learning Representations*, 2020.
- [34] M. Lanne, M. Meitz, and P. Saikkonen. Identification and estimation of non-Gaussian structural vector autoregressions. *Journal of Econometrics*, 196(2):288–304, 2017.
- [35] D. Linzner, M. Schmidt, and H. Koepl. Scalable structure learning of continuous-time Bayesian networks from incomplete data. *Advances in Neural Information Processing Systems*, 32:3746–3756, 2019.
- [36] M. Liu, X. Sun, L. Hu, and Y. Wang. Causal discovery from subsampled time series with proxy variables. *Advances in neural information processing systems*, 36, 2023.
- [37] S. Löwe, D. Madras, R. Zemel, and M. Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Proceedings of Conference on Causal Learning and Reasoning (CLeaR)*, pages 509–525, 2022.

- [38] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.
- [39] S. Lu and T. Gao. Meta-dag: Meta causal discovery via bilevel optimization. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [40] D. Malinsky and P. Spirtes. Learning the structure of a nonstationary vector autoregression. In *International Conference on Artificial Intelligence and Statistics*, pages 2986–2994, 2019.
- [41] R. Marcinkevičs and J. E. Vogt. Interpretable models for granger causality using self-explaining neural networks. *arXiv preprint arXiv:2101.07600*, 2021.
- [42] Z. Meng, S. Han, P. Liu, and Y. Tong. Improving speech related facial action unit recognition by audiovisual information fusion. *IEEE Transactions on Cybernetics*, 49(9):3293–3306, 2018.
- [43] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California Berkeley, USA, 2002.
- [44] A. Nagabandi, C. Finn, and S. Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018.
- [45] S. Nair, Y. Zhu, S. Savarese, and L. Fei-Fei. Causal induction from visual observations for goal directed tasks. *arXiv preprint arXiv:1910.01751*, 2019.
- [46] I. Ng, Z. Fang, S. Zhu, Z. Chen, and J. Wang. Masked gradient-based causal structure learning. *arXiv preprint arXiv:1910.08527*, 2019.
- [47] I. Ng, A. Ghassami, and K. Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.
- [48] I. Ng, S. Zhu, Z. Fang, H. Li, Z. Chen, and J. Wang. Masked gradient-based causal structure learning. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 424–432. SIAM, 2022.
- [49] R. Pamfil, N. Sriwattanaworachai, S. Desai, P. Pilgerstorfer, K. Georgatzis, P. Beaumont, and B. Aragam. DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605, 2020.
- [50] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [51] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.
- [52] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.
- [53] S. Pitis, E. Creager, A. Mandlekar, and A. Garg. Mocoda: Model-based counterfactual data augmentation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [54] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. A. Eslami, and M. Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.
- [55] J. C. Rajapakse and J. Zhou. Learning effective brain connectivity with dynamic Bayesian networks. *Neuroimage*, 37(3):749–760, 2007.
- [56] M. Reale and G. T. Wilson. Identification of vector ar models with recursive structural errors using conditional independence graphs. *Statistical Methods and Applications*, 10(1-3):49–65, 2001.
- [57] M. Reale and G. T. Wilson. The sampling properties of conditional independence graphs for structural vector autoregressions. *Biometrika*, 89(2):457–461, 2002.

- [58] A. Reisach, C. Seiler, and S. Weichwald. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.
- [59] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.
- [60] E. Saggioro, J. de Wiljes, M. Kretschmer, and J. Runge. Reconstructing regime-dependent causal relationships from observational time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(11), 2020.
- [61] H. Shen, Q. Xiao, and T. Chen. On penalty-based bilevel gradient descent method, 2023.
- [62] P. Spirtes, C. Glymour, and R. Scheines. *Causality, Prediction, and Search*. MIT Press, Cambridge, MA, USA, 2nd edition, 2001.
- [63] X. Sun, O. Schulte, G. Liu, and P. Poupart. Nts-notears: Learning nonparametric dbns with prior knowledge. *arXiv preprint arXiv:2109.04286*, 2021.
- [64] N. R. Swanson and C. W. Granger. Impulse response functions based on a causal approach to residual orthogonalization in vector autoregressions. *Journal of the American Statistical Association*, 92(437):357–367, 1997.
- [65] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. B. Fox. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.
- [66] A. Tank, E. B. Fox, and A. Shojaie. Identifiability and estimation of structural vector autoregressive models for subsampled and mixed-frequency time series. *Biometrika*, 106(2):433–452, 2019.
- [67] I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.
- [68] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.
- [69] N. Watters, L. Matthey, M. Bosnjak, C. P. Burgess, and A. Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.
- [70] D. Wei, T. Gao, and Y. Yu. DAGs with No Fears: A closer look at continuous optimization for learning bayesian networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 3895–3906, 2020.
- [71] N. Yin, T. Gao, Y. Yu, and Q. Ji. Effective causal discovery under identifiable heteroscedastic noise model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16486–16494, 2024.
- [72] N. Yin, Y. Yu, T. Gao, and Q. Ji. Efficient nonlinear dag learning under projection framework. In *International Conference on Pattern Recognition*, pages 445–460. Springer, 2025.
- [73] Y. Yu, J. Chen, T. Gao, and M. Yu. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163, 2019.
- [74] Y. Yu, T. Gao, N. Yin, and Q. Ji. DAGs with no curl: An efficient DAG structure learning approach. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 12156–12166, 2021.
- [75] A. Zhang, Z. C. Lipton, L. Pineda, K. Azizzadenesheli, A. Anandkumar, L. Itti, J. Pineau, and T. Furlanello. Learning causal state representations of partially observable environments. *arXiv preprint arXiv:1906.10437*, 2019.

- [76] Z. Zhang, I. Ng, D. Gong, Y. Liu, E. Abbasnejad, M. Gong, K. Zhang, and J. Q. Shi. Truncated matrix power iteration for differentiable dag learning. *Advances in Neural Information Processing Systems*, 35:18390–18402, 2022.
- [77] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pages 9472–9483, 2018.
- [78] X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. Xing. Learning sparse nonparametric DAGs. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3414–3425, 2020.
- [79] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

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:

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: At conclusion.

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:

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:

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: Will release code upon acceptance of the paper.

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:

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:

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:

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:

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: [Yes]

Justification:

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:

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:

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: [NA]

Justification:

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:

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:

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: [No]

Justification:

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.

A Technical Appendix

A.1 Algorithm and Convergence

As the optimization algorithm is general, for ease of reading, we simplify the notation here. Instead of using Φ_s and Φ_p , we use (x, y) .

Let define

$$\ell_i(x, y_i) = f_i(\mathbf{X}_i^{\text{val}}, \mathbf{W}_s, \mathbf{W}_{p,i}^*(\mathbf{W}_s), \theta_{p,i}^*(\mathbf{W}_s)) + \eta \|\mathbf{W}_s\|_1 \quad (13)$$

and

$$g_i(x, y_i) = \frac{1}{2n} \|\mathbf{X}_i^{\text{train}} - \theta_{p,i} \sigma((\mathbf{W}_{p,i} \circ \mathbf{W}_s) \mathbf{X}_i^{\text{train}})\|_2^2 + \eta \|\mathbf{W}_{p,i}\|_1. \quad (14)$$

Then, the problem can be concisely written as

$$\min_{x, y_i} \frac{1}{m} \sum_{i=1}^m \ell_i(x, y_i) + \eta f(x) \quad (15a)$$

$$\text{s.t. } y_i \in \mathcal{S}(x) := \arg \min_{y_i} g_i(x, y_i) + \eta f(y_i) \quad \text{s.t. } h(x, y_i) = 0 \quad (15b)$$

where in this case $f(\cdot) := \|\cdot\|_1$, but it can generally represent any projection-friendly non-smooth term.

To solve this problem, we can apply the penalized method. Towards this end, we first write the penalized objective as follows

$$\mathcal{L}_i(x, \{y_i\}; \gamma_i, \nu_i) = \frac{1}{m} \left(\sum_{i=1}^m \ell_i(x, y_i) \right) + \gamma (g_i(x, y_i) - g_i^*(x)) + \nu h(x, y_i) + \eta f(x) + \eta f(y_i) \quad (16)$$

where $\{\gamma, \nu\}$ are penalizers, and $g_i^*(x) = \min_{y_i} g_i(x, y_i)$.

Recall the definition of the proximal operator:

$$\begin{aligned} \text{prox}_{\eta f}(x - \eta d) \\ = \arg \min_z \eta f(z) + \frac{1}{2} \|x - \eta d - z\|^2 \end{aligned} \quad (17)$$

$$= \arg \min_z \eta f(z) + \eta \langle d, z - x \rangle + \frac{1}{2} \|z - x\|^2. \quad (18)$$

Then, we can have the proximal residual defined as

$$\mathcal{P}_\eta^f(x, d) = \frac{1}{\eta} (x - \text{prox}_{\eta f}(x - \eta d)), \quad (19)$$

which will serve as the optimality gap for measuring the convergence of the algorithm.

For simplicity of the notations, define $z = (x, \{y_j, j = i - w + 1, \dots, i\})$. Then, we can construct the window average as follows

$$S_{i,w}(z) = \frac{1}{w} \sum_{j=i-w+1}^i \ell_j(x, y_j) + \gamma (g_j(x, y_j) - g_j^*(x)) + \nu h_j(x, y_j). \quad (20)$$

Due to the non-convexity and non-smoothness of the loss function, we follow the existing notation of the local regret of a policy z_i up to time m with window length w , which is defined

$$\text{Reg}_w(m) = \sum_{i=1}^m \|\mathcal{P}_\eta^f(z_i, \nabla S_{i,w}(z_i))\|^2. \quad (21)$$

From the definition of $S_{i,w}(z)$, we can have

$$\begin{aligned} S_{i,w}(z) &= S_{i-1,w}(z) + \frac{1}{w} (\ell_i(x, y_i) - \ell_{i-w}(x, y_{i-w})) \\ &\quad + \frac{\gamma}{w} (g_i(x, y_i) - g_i^*(x) - (g_{i-w}(x, y_{i-w}) - g_{i-w}^*(x))) \\ &\quad + \frac{\nu}{w} (h_i(x, y_i) - h_{i-w}(x, y_{i-w})). \end{aligned} \quad (22)$$

However, due to the bilevel structure of the problem, we can only achieve an estimate of $g_i^*(x)$ through the inner loop update $\hat{y}$, which is denoted as $\hat{g}_i(x) = g(x, \hat{y})$. Then, we can use $\|\mathcal{P}_\eta^f(z_{i+1}, \nabla S_{i,w}(z_{i+1}))\|$ to determine whether we need to update the model.

Recall our algorithm: when $\|\mathcal{P}_\eta^f(z_{i+1}, \hat{\nabla} S_{i,w}(z_{i+1}))\| > \delta/w$, we implement the proximal gradient descent. Otherwise, we update our model, where

$$\begin{aligned} \hat{S}_{i,w}(z) &= S_{i-1,w}(z) + \frac{1}{w}(\ell_i(x, y_i) - \ell_{i-w}(x, y_{i-w})) \\ &\quad + \frac{\gamma}{w}(g_i(x, y_i) - \hat{g}_i(x) - (g_{i-w}(x, y_{i-w}) - \hat{g}_{i-w}(x))) \\ &\quad + \frac{\nu}{w}(h_i(x, y_i) - h_{i-w}(x, y_{i-w})). \end{aligned} \quad (23)$$

Next, we will find an upper bound of $\|\mathcal{P}_\eta^f(z_{i+1}, \nabla S_{i,w}(z_{i+1}))\|$. First, we can have

$$\begin{aligned} &\|\mathcal{P}_\eta^f(z_i, \nabla S_i(z_i))\| \\ &\stackrel{(a)}{\leq} \|\mathcal{P}_\eta^f(z_i, \nabla S_i(z_i))\| + \|\mathcal{P}_\eta^f(z_i, \nabla S_i(z_i)) - \mathcal{P}_\eta^f(z_i, \hat{\nabla} S_i(z_i))\| \\ &\leq \|\mathcal{P}_\eta^f(z_i, \nabla S_{i-1}(z_i))\| \end{aligned} \quad (24)$$

$$\begin{aligned} &\quad + \frac{\gamma}{w} \|\nabla g_i(x_i, y_i) - \nabla g_i^*(x_i) - (\nabla g_{i-w}(x_i, y_{i-w}) - \nabla g_{i-w}^*(x_i))\| \\ &\quad + \frac{\nu}{w} \|\nabla h_i(x_i, y_i) - \nabla h_{i-w}(x_i, y_{i-w})\| + \frac{1}{w} \|\nabla \ell_i(x_i, y_i) - \nabla \ell_{i-w}(x_i, y_{i-w})\| \\ &\quad + \frac{\gamma}{w} \|\nabla g_i^*(x_i) - \nabla \hat{g}_i(x_i)\| + \frac{\gamma}{w} \|\nabla g_{i-w}^*(x_i) - \nabla \hat{g}_{i-w}(x_i)\| \end{aligned} \quad (25)$$

$$\stackrel{(b)}{\leq} \frac{\delta}{w} + \frac{\gamma}{w} \|\nabla g_i^*(x_i) - \nabla \hat{g}_i(x_i)\| + \frac{\gamma}{w} \|\nabla g_{i-w}^*(x_i) - \nabla \hat{g}_{i-w}(x_i)\| + V_{i,w} \quad (26)$$

$$\stackrel{(c)}{\leq} \frac{\delta}{w} + \frac{\gamma}{w} L_g d_{\mathcal{S}(x_i)}(\hat{y}_i) + \frac{\gamma}{w} L_g d_{\mathcal{S}(x_i)}(\hat{y}_{i-w}) + V_{i,w} \quad (27)$$

where (a) is true due to the triangle inequality, (b) holds by following the condition of updating the model and $V_{i,w}$ is defined as

$$\begin{aligned} V_{i,w} &= \frac{\gamma}{w} \|\nabla g_i(x_i, y_i) - \nabla g_i^*(x_i) - (\nabla g_{i-w}(x_i, y_{i-w}) - \nabla g_{i-w}^*(x_i))\| \\ &\quad + \frac{\nu}{w} \|\nabla h_i(x_i, y_i) - \nabla h_{i-w}(x_i, y_{i-w})\| + \frac{1}{w} \|\nabla \ell_i(x_i, y_i) - \nabla \ell_{i-w}(x_i, y_{i-w})\| \end{aligned} \quad (28)$$

and in (c) we apply the gradient Lipschitz continuity of $g_i(x, \cdot)$, $\forall i$ and $d_{\mathcal{S}}(y) = \arg \min_{y' \in \mathcal{S}} \|y' - y\|$.

When the lower level problem satisfies the proximal PL error bound, i.e.,

$$\frac{1}{\eta} \|y - \text{prox}_{\eta f}(y - \eta \nabla_y g(x, y))\| \geq c d_{\mathcal{S}(x)}(y) \quad (29)$$

where c is a constant and y here is generic, representing any y_i , then according to [27, Appendix G] we have the function also satisfies the proximal PL condition, which is

$$\frac{1}{2} \mathcal{D}_f(y; x) \geq \mu (F(x, y) - F^*(x)) \quad (30)$$

where $F(x, y) := g(x, y) + \eta f(x)$, $F^*(x) := g^*(x) + \eta f(x)$ and

$$\mathcal{D}_f(y; x) = -2L_g \min_{y'} \langle \nabla g(x, y), y' - y \rangle + \frac{L_g}{2} \|y' - y\|^2 + \eta f(y') - \eta f(y). \quad (31)$$

By the Lipschitz gradient of $g(x, y)$, we have

$$\begin{aligned} &g(x, y'_{t+1}) + \eta f(y'_{t+1}) \\ &\leq g(x, y'_t) + \eta f(y'_t) + \langle \nabla_y g(x, y'_t), y'_{t+1} - y'_t \rangle + \frac{L_g}{2} \|y'_{t+1} - y'_t\|^2 + \eta f(y'_{t+1}) - \eta f(y'_t) \end{aligned} \quad (32)$$

$$= g(x, y'_t) + f(y'_t) - 2L_g \mathcal{D}_f(y'_t; x) \quad (33)$$

where y'_t denotes the iterates in the inner loop.

Substituting equation 30 to equation 33 gives

$$F(x, y'_{t+1}) - F^*(x) \leq \left(1 - \frac{L_g}{\mu}\right) (F(x, y'_t) - F^*(x)) \leq \left(1 - \frac{L_g}{\mu}\right)^T (F(x, y'_1) - F^*(x)) \quad (34)$$

where T denotes the number of iterations in the inner loop.

From [14, Corollary 3.6], we know that the proximal PL error bound implies the quadratic growth property, meaning that there exists a constant $\bar{\mu}$ such that

$$d_{\mathcal{S}(x)}^2(\hat{y}_i) \leq \frac{\bar{\mu}}{2} \left(1 - \frac{L_g}{\mu}\right)^T (F(x, y'_1) - F^*(x)) \quad (35)$$

where $\hat{y}_i := y'_{T+1}$. As x is bounded, the term $d_{\mathcal{S}(x_i)}(\hat{y}_i)$ shrinks to 0 at a linear rate.

Summing over $i = 1, \dots, m$ gives

$$\begin{aligned} & \text{Reg}_w(m) \\ &= \sum_{i=1}^m \|\mathcal{P}_\eta^f(z_i, \nabla S_{i,w}(z_i))\|^2 \end{aligned} \quad (36)$$

$$\leq \frac{4m\delta^2}{w^2} + 4L_g^2\gamma^2 \sum_{i=1}^m d_{\mathcal{S}(x_i)}^2(\hat{y}_i) + 4L_g^2\gamma^2 \sum_{i=1}^m d_{\mathcal{S}(x_i)}^2(\hat{y}_{i-w}) + \frac{4}{w^2} \sum_{i=1}^m V_{i,w}. \quad (37)$$

When we choose $T \geq -\log_{(1-L_g/\mu)} \bar{\mu}(F(x, y'_1) - F^*(x))w^2$, we have $\sum_{i=1}^m d_{\mathcal{S}(x_i)}^2(\hat{y}_i) \sim m/w^2$. The last term $\sum_{i=1}^m V_{i,w}$ is also referred to as the sliding window variation. When the gradients of the loss function are bounded, we automatically have that this term is $\mathcal{O}(m)$. Therefore, we can conclude that our algorithm still can achieve the optimal regret bound as $\mathcal{O}(m/w^2)$.

A.2 Identifiability

In this section, we demonstrate that Theorem 3.2 from [18] (an extension of Theorem 1 from [4]) can be applied in our online meta-learning setting. We denote the current domain with index m , and recall that a model estimate $\mathbf{W}_s$ of the shared, cross-domain part of the graph adjacency matrix can be formed from the data from the previous $m-1$ domains. When encountering domain m , our objective is to optimize the part of the adjacency matrix specific to the domain, $\mathbf{W}_{p,m,k=1}$ (where $k=1$ indicates the observational graph, and $k>1$ indexes interventions on this graph, as in [18]).

We denote with $\mathcal{G}_m$ the DAG whose edges are given by the nonzero elements of the adjacency matrix $\mathbf{W}^{\mathcal{G}_m} := \mathbf{W}_s \circ \mathbf{W}_{p,m,k=1}$. Our score function for $\mathcal{G}_m$ can be written (following the notation of [4])

$$\mathcal{S}_{\mathcal{I}^*}(\mathcal{G}_m) := \sup_{\theta_m, \mathbf{W}|\mathcal{G}_m} \sum_{k=1}^K \mathbb{E}_{\mathbf{X} \sim p_m^{(k)}} \log f^{(k)}(\mathbf{X}; \mathbf{W}^{\mathcal{G}_m}, R^{\mathcal{I}^*}, \theta_{m,k}) - \eta |\mathcal{G}_m| \quad (38)$$

where the regularization term $|\mathcal{G}_m| = \|\mathbf{W}_s \circ \mathbf{W}_{p,m,1}\|_0$ is the L0 norm (which counts the number of edges in $\mathcal{G}_m$), $\theta_m = \{\theta_{m,k}\}_{k=1}^K$ with $\theta_{m,k} = \{\theta_{m,k,j}\}_{j=1}^d$ are neural network parameters, $p_m^{(k)}$ is the ground truth generating distribution of the data for domain m with intervention $I_k \in \mathcal{I}$, and $R^{\mathcal{I}} = \{r_{kj}^{\mathcal{I}}\}_{k,j=1}^{K,d}$ specifies the set of interventions, which influence the log-likelihood function as follows,

$$\log f^{(k)}(\mathbf{X}; \mathbf{W}, R^{\mathcal{I}}, \theta_{m,k}) = \sum_{j=1}^d r_{kj}^{\mathcal{I}} \cdot \log \tilde{f}(\mathbf{X}, \text{NN}(\mathbf{X}, \theta_{m,k,j}, \mathbf{W}_k)) + (1 - r_{kj}^{\mathcal{I}}) \cdot \log \tilde{f}(\mathbf{X}, \text{NN}(\mathbf{X}, \theta_{m,1,j}, \mathbf{W}_k))$$

where we again denote the observational distributions with $k=1$, and sum over the model likelihood function $\tilde{f}$ (for intervention k) for each node j in the graph. Here, “NN” denotes a neural network function, and $\mathbf{W}_k^{\mathcal{G}_m} := \mathbf{W}_s \circ \mathbf{W}_{p,m,k}$ is the full adjacency matrix for domain m and intervention k (we assume $\mathbf{W}_s$ does not depend on the intervention k). Lastly, when taking the supremum in equation 38, we use $\mathbf{W}|\mathcal{G}_m$ to indicate any values of the continuous matrix $\mathbf{W}$, subject to the constraint that edges absent in $\mathcal{G}_m$ correspond to zero entries in $\mathbf{W}$.

The only difference between this setting and the setting of [18] is that instead of learning an adjacency matrix, we are learning a factor $\mathbf{W}_{p,m,k}$ of the full adjacency matrix $\mathbf{W}_s \circ \mathbf{W}_{p,m,k}$. Thus, in the case where $\mathbf{W}_s$ is set to the all-ones matrix, we recover the setting of [18] exactly. Theorem 3.2 from [18] (as well as Theorem 1 from [4]) involve the set of DAGs maximizing a score function $\mathcal{S}_{\mathcal{I}^*}(\mathcal{G})$, where $\mathcal{I}^*$ is the ground truth set of interventions. Our score function, equation 38, is equivalent when $\mathbf{W}_s$ is set to the all-ones matrix.

For any given adjacency matrix $\mathbf{W}^*$ maximizing equation 38 (for a fixed graph $\mathcal{G}_m$), as long as the elements of $\mathbf{W}_s$ are nonzero wherever $\mathbf{W}^*$ is nonzero (wherever the graph $\mathcal{G}_m$ has an edge), then there exists a matrix $\mathbf{W}_p$ satisfying $\mathbf{W}^* = \mathbf{W}_s \circ \mathbf{W}_p^*$, for which each element is determined uniquely as $(\mathbf{W}_p^*)_{ij} = (\mathbf{W}^*)_{ij} / (\mathbf{W}_s)_{ij}$. Thus, the set of $\mathbf{W}^*$ maximizing equation 38 is in one-to-one correspondence with the set of $\mathbf{W}_p^*$ maximizing the same quantity (given a fixed $\mathbf{W}_s$). Conditional on a given graph $\mathcal{G}_m$, and given any matrix $\mathbf{W}^*$ picked out by the supremum over $\mathbf{W}|\mathcal{G}_m$, there exists a matrix $\mathbf{W}_p$ for which our score function, equation 38, is maximized with $\mathbf{W}^* = \mathbf{W}_s^* \circ \mathbf{W}_p^*$.

This allows us to extend Theorem 3.2 from [18], with an additional condition on $\mathbf{W}_s$:

Theorem A.1. *For a graph $\hat{\mathcal{G}} \in \mathcal{D}$, where $\mathcal{D} \subset \text{DAG}$, if $\hat{\mathcal{G}} \in \arg\max_{\mathcal{G} \in \text{DAG}} \mathcal{S}_{\mathcal{I}}(\mathcal{G})$ with $\mathcal{S}_{\mathcal{I}}$ defined in equation 38, and furthermore if (i) the density model has sufficient capacity to exactly represent the ground truth distributions, (ii) a given set of interventions $\mathcal{I}$ satisfies $\mathcal{I}$ -faithfulness for the true graph and distributions $(\mathcal{G}^*, P_X)$, (iii) the density models are strictly positive, (v) the ground truth densities $p^{(k)}(\mathbf{X})$ have finite differential entropy, and (vi) the elements of $\mathbf{W}_s$ are nonzero wherever the elements of the adjacency matrix of the true graph $\mathcal{G}^*$ are nonzero, then $\hat{\mathcal{G}}$ is $(\mathcal{I}, \mathcal{D})$ -Markov equivalent to $\mathcal{G}^*$, for small enough η .*

In practice, condition (vi) can be supported or enforced by adding an additional regularization term or hard constraint to prevent $\mathbf{W}_s$ from becoming too sparse.

Remark A.2. . Since learning $\mathbf{W}_s$ in the upper learning is conducted with any constraints (in particular without an acyclicity constraint), the upper-level optimization is the same in any $\mathcal{L}_1$ -regularized regression problem. Hence it is easy for the assumption on the nonzero elements of $\mathbf{W}_s$ to be satisfied.

B Data Appendix

B.1 Dynamic DAG Learning and Data Normalization

Recently [58] have found there may be biases existing in the data generating process, and data normalization is suggested. Here we would like to point out that time series DAG learning differs from the i.i.d. setting, and it is not obvious how the conclusion from [58] is directly applicable to time series data and what sort of standardization even makes sense to generate data in our setting. For example, standardization in i.i.d. data is across samples, which is different from typical time series standardization across time. Moreover, time series data like Brownian motion naturally increases variance over time, so real-world datasets should indeed show higher variance for variables at later epochs. Yet another complexity is that the data in our setting is non-stationary. It would be an interesting research question to investigate how standardization of data might affect time series DAG learning in general.

B.2 Implementation Details for Synthetic Datasets

We use the following steps and hyper-parameters to generate the data.

- For step 1), we use the Erdos-Renyi model to generate shared intra-slice graph $\mathbf{W}_s^a$ with degree of 3 for a total d nodes, from 5 to 20, with its weights sampled uniformly at random from $\mathcal{U}([-2.0, -0.5] \cup [0.5, 2.0])$. We use the same Erdos-Renyi model to generate inter-slice graph $\mathbf{W}_s^b$ with degree of 3, with its weights sampled uniformly at random from $\mathcal{U}([-0.5, -0.3] \cup [0.3, 0.5])$. Then for each domain m , we randomly remove or add 2 edges from the previous domain (randomly generated in the first domain), to ensure temporal smoothness. Then we generate weights of $\mathbf{W}_{p,m}$ similarly as $\mathbf{W}_s$, but with edge weight ranges of 5 times smaller, i.e., $\mathcal{U}([-0.4, -0.1] \cup [0.1, 0.4])$ for $\mathbf{W}_p^b$ and $\mathcal{U}([-0.1, -0.06] \cup [0.1, 0.06])$ for $\mathbf{W}_p^a$.
- For step 2), we generate data with the first autoregressive order, where data only depends on the previous time slice. The underlying function between a node and its parents becomes a two-layer MLP with a sigmoid activation function. In the MLP setting, the weights of the MLPs are sampled uniformly from $\mathcal{U}([-2.0, -0.5] \cup [0.5, 2.0])$, and then weights in the first layer are updated by the parental weights from $\mathbf{W}^a$ and $\mathbf{W}^b$ structures. The number of hidden neuron sizes is set to be 10. We generate 10 sequences with 100 time slices each with standard Gaussian noises. We estimate one graph from each sequence and report the average SHD along with its standard deviation.
- For step 3), to generate interventional data, we flip a fair coin to decide whether to intervene at any time slice, in which case we sample one node uniformly to be the intervened node. We sample intervened nodes' values from another distribution. We assume every node has a probability of 0.1 to be intervened upon at each time slice, and if a node is chosen, the soft intervention comes from a different 2-layer MLP, depending on the values of its parent nodes per its graph and similarly generated as the first MLP. This soft intervention distribution can be very different from the observation distribution.

Hence, overall we generate 10 sequences for 10 domains, each sequence with 100 time steps. For the hyperparameter values, we use a separate validation dataset to choose the best performing hyperparameters for each method per SHD². We search for the best value of each of 5 parameters sequentially, including two $\mathcal{L}_1$ penalty coefficients for $\mathbf{W}^a$ and $\mathbf{W}^b$, the threshold to obtain final $\mathbf{W}^a$ and $\mathbf{W}^b$, and the hidden neuron size. For λ_a and λ_w , we search over a value range of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Graph threshold search range is set to be $\{0.001, 0.01, 0.05, 0.1, 0.2, 0.3\}$, and neuron size range is searched over $\{8, 16, 32, 64\}$. We use a two-layer MLP architecture in the experiments: $\tilde{\mathbf{X}}_i = \mathbf{W}_1 \sigma((\mathbf{W}_{p,i}(\mathbf{W}_s) \circ \mathbf{W}_s) \mathbf{X}_i)$, where $\mathbf{W}_1$ is the MLP layer weight and σ is the ReLU activation function. For other baseline methods, we use the default parameter settings.

B.3 Implementation Details for Sprite World Datasets

To generate a meta-learning dataset, we first generated a random Erdos-Reny graph, replicated it to the next step, and added a single edge for each variable at two consecutive time steps to introduce

²Alternatively, if datasets do not contain ground truth graphs, we can use the reconstruction loss on data as an alternative to measuring the quality of the learned graph

correlation across time steps. When we switch to a different domain, we randomly add or remove an edge in the instantaneous graph and time-delayed graph. Since we are interested in structure learning, we used a random policy agent that randomly selects at most two objects and intervene the value with a truncated normal distribution at each time step. The details of hyper-parameters are given as follows. For non-linear SEM, we utilized a two-layer MLP with the weight in the first layer ranges between $(0.5, 2.0)$ or $(-2.0, -0.5)$ and the second layer between $(0.01, 0.5)$ or $(-0.5, -0.01)$ and the bias follows the Gaussian distribution with scale 1. We again generate 10 domains with 10 different sequences with 100 time steps, and report the SHD performances across sequences.

B.4 Ablation Study: Linearity and Private Parameters

We perform an ablation study with two new baselines, 1) a multi-task DAG learning where only shared parameters are learned (named MT-DAG), and 2) a linear meta-learning formulation (named Meta-D2AG-Linear), on synthetic datasets with scale-free (SF) graphs.

The data generation process is the same as ER graphs as discussed, except for the graph types. Our Meta-D2AG shows the best performance against these two variants of the proposed approaches, as well as the best performing IDYNO-Soft From Section 4.1, in both batch and online settings.

Var. Size	IDYNO	MT-DAG	Meta-D2AG-Linear	Meta-D2AG
$d = 5$	11 $\pm$ 3.9	9.3 $\pm$ 0.9	21 $\pm$ 0.5	7.2$\pm$1.4
$d = 10$	23 $\pm$ 3.1	25 $\pm$ 4.4	37 $\pm$ 1.7	18$\pm$0.8
$d = 15$	51 $\pm$ 3.2	58 $\pm$ 15	50 $\pm$ 1.8	24$\pm$2.6
$d = 20$	42 $\pm$ 5.3	46 $\pm$ 9	67 $\pm$ 4.1	35$\pm$1.9

Table 2: SHD Results for Scale-Free Graph (SF) in the Batch Setting

Var. Size	IDYNO	MT-DAG	Meta-D2AG-Linear	Meta-D2AG
$d = 5$	8.8 $\pm$ 0.3	24 $\pm$ 1.6	21 $\pm$ 1.3	8.7$\pm$0.8
$d = 10$	29 $\pm$ 1.9	37 $\pm$ 1.6	46 $\pm$ 2.7	18$\pm$0.7
$d = 15$	54 $\pm$ 2.9	55 $\pm$ 4.9	53 $\pm$ 2.4	27$\pm$0.4
$d = 20$	47 $\pm$ 5.5	77 $\pm$ 6.1	66 $\pm$ 5.8	36$\pm$1.2

Table 3: SHD Results for Scale-Free Graph (SF) in the Online Setting

B.5 Ablation Study: Sequence Length

We conduct an experiment on the effect of sample sizes vs SHD. We use variable dimension $d = 20$ and 10 total sequences, and the rest setting are the same as in Section 5.1. When the sequence lengths are lower, all methods perform worse, but our method remain competitive.

Sequence Length	10	50	100
GOLEM	115 $\pm$ 20	102 $\pm$ 10	81 $\pm$ 10
DYNOTEAR	187 $\pm$ 23	150 $\pm$ 20	89 $\pm$ 16
IDYNO	129 $\pm$ 16	97 $\pm$ 14	80 $\pm$ 21
JPCMCI+	132 $\pm$ 21	130 $\pm$ 20	119 $\pm$ 9
NTS-NOTEARS	119 $\pm$ 15	95 $\pm$ 5	63 $\pm$ 13
Meta-D2AG	101 $\pm$ 10	80 $\pm$ 7	46 $\pm$ 6

Table 4: Performance across different sequence lengths in Online Setting, $d = 20$.

B.6 Ablation Study: Random Variance

To study the impact of variable variance, we repeat the synthetic dataset in Experiment 5.1 with random noises between 0.1 and 2 for each variable noise term, with 4 different variable sizes. Below are the results against baselines. The relative errors are higher than of fixed variance of 1, but the Meta-D2AG method remains the best among methods tested.

d	5	10	15	20
GOLEM	5.9 ± 0.9	26.6 ± 1.8	43.7 ± 7.8	93.5 ± 7.9
DYNOTEAR	6.7 ± 1.1	22.3 ± 3.0	60.4 ± 4.6	95.6 ± 5.3
IDYNO	6.5 ± 1.1	26.5 ± 1.1	57.5 ± 5.7	90 ± 9.2
JPCMCi+	6.6 ± 0.1	23.6 ± 1.1	85 ± 10.2	105 ± 19.8
NTS-NOTEARS	12.3 ± 0.5	33.3 ± 2.1	55.7 ± 5.3	104 ± 10.2
Meta-D2AG	6.6 ± 0.1	22.9 ± 2.2	46.7 ± 4.6	82.7 ± 8.9

Table 5: SHD with random variable variance across varying dimensions d in the online setting.