FedDAG: Federated DAG Structure Learning

Erdun Gao*
School of Mathematics and Statistics, The University of Melbourne
erdun.gao@student.unimelb.edu.au

Junjia Chen
Faculty of Electronic and Information Engineering, Xi'an Jiaotong University
cjj19970505@stu.xjtu.edu.cn

Li Shen
JD Explore Academy
shenli100@jd.com

Tongliang Liu
TML Lab, Sydney AI Centre, The University of Sydney
Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence
tongliang.liu@sydney.edu.au

Mingming Gong
School of Mathematics and Statistics, The University of Melbourne
mingming.gong@unimelb.edu.au

Howard Bondell
School of Mathematics and Statistics, The University of Melbourne
howard.bondell@unimelb.edu.au

Reviewed on OpenReview: *https://openreview.net/forum?id=MzWgBjZ6Le*

Abstract

To date, most directed acyclic graphs (DAGs) structure learning approaches require data to be stored in a central server. However, due to the consideration of privacy protection, data owners gradually refuse to share their personalized raw data to avoid private information leakage, making this task more troublesome by cutting off the first step. Thus, a puzzle arises: *how do we discover the underlying DAG structure from decentralized data?* In this paper, focusing on the additive noise models (ANMs) assumption of data generation, we take the first step in developing a gradient-based learning framework named FedDAG, which can learn the DAG structure without directly touching the local data and also can naturally handle the data heterogeneity. Our method benefits from a two-level structure of each local model. The first level structure learns the edges and directions of the graph and communicates with the server to get the model information from other clients during the learning procedure, while the second level structure approximates the mechanisms among variables and personally updates on its own data to accommodate the data heterogeneity. Moreover, FedDAG formulates the overall learning task as a continuous optimization problem by taking advantage of an equality acyclicity constraint, which can be solved by gradient descent methods to boost the searching efficiency. Extensive experiments on both synthetic and real-world datasets verify the efficacy of the proposed method.

1 Introduction

Bayesian Networks (BNs) have become prevalent over the last few decades by leveraging the graph theory and probability theory to model the probabilistic relationships among a set of random variables, which can potentially provide a mechanism for evidential reasoning (Pearl, 1985). The success of BNs has contributed to a furry of downstream real-world application problems in econometrics (Heckman, 2008), epidemiology (Greenland et al., 1999), biological sciences (Imbens & Rubin, 2015) and social sciences (Marini

*Work was done during an internship at JD Explore Academy.

& Singer, 1988). However, learning the graph structure of a BN from purely observational data remains a significant challenge due to the NP-hard property, and therefore, cutting-edge research that has drawn considerable attention in both academic and industrial fields (Koller & Friedman, 2009; Jensen & Nielsen, 2007; Glymour et al., 2019; Zheng et al., 2018; Zheng, 2020).

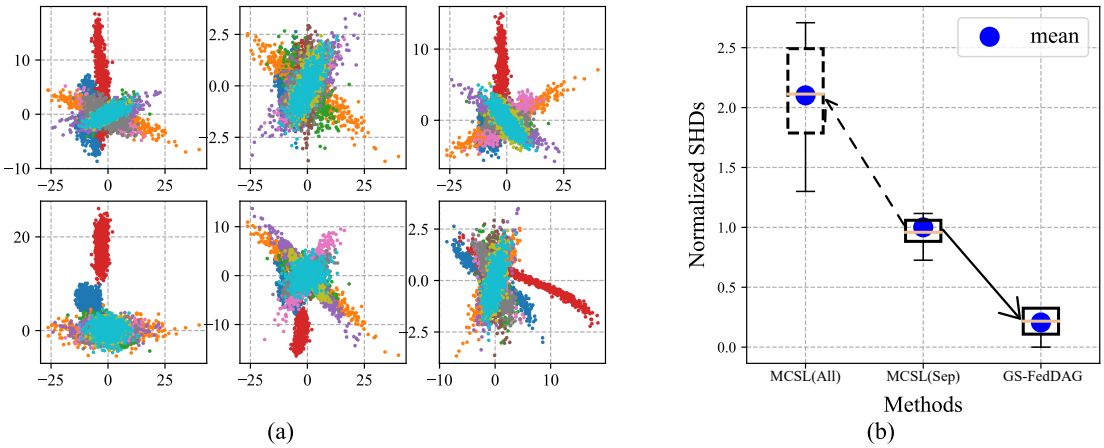


Figure 1: (a) Visualization of heterogeneous data. Different colors represent data from different sources, while each sub-figure includes the distribution of one fixed dimension of data from all clients. (b) Normalized structural hamming distances (SHDs) (↓) of three methods, where MCSL (Sep) (Ng et al., 2022b) separately trains one model on local data while MCSL (All) trains one model on all data, which is forbidden in FL.

Each BN is defined by a directed acyclic graph (DAG) and a set of parameters to depict the direction, strength, and shape of the mechanisms between variables (Kitson et al., 2021). Over the recent decades, a bunch of methods (Spirtes et al., 2001; Chickering, 2002; Zheng, 2020) for discovering the DAG structure encoded among concerned events from the observational data have been proposed. In practice, however, finite sample problem bears the brunt of the performance decrease of DAG structure learning methods. Regularly (1) collecting data from various sources and then (2) designing the structure learning algorithm on all collected data can serve as a direct and standard pipeline to alleviate this issue in this field. However, owing to the issue of data privacy, data owners gradually prefer not to share their personalized data[1] with others (Kairouz et al., 2021). Naturally, the new predicament, *how do we discover the underlying DAG structure from decentralized data?* has arisen. In statistical learning problems such as regression and classification, federated learning (FL) has been proposed to learn from locally stored data (McMahan et al., 2017). Inspired by the developments in FL, we aim to develop a federated DAG structure learning framework that enables learning the graph structure from the decentralized data. Compared to the traditional FL methods in statistical learning, federated DAG learning, a **structural learning** task, has the following two main **differences**:

- **Learning objective difference.** Most of the previous FL researches focus on learning *an estimator* to estimate the conditional distribution $P(Y|X)$ in supervised learning tasks, e.g., image classification (McMahan et al., 2017; Li et al., 2020), sequence tagging (Lin et al., 2022), and feature prediction (Kairouz et al., 2021). However, DAG structure learning, an unsupervised learning task (Glymour et al., 2019), tries to *find the underlying graph structure* among the concerned variables and the relationship mechanisms estimators to fit with the joint distribution of observations.

- **Data heterogeneity difference.** The setup of data heterogeneity in FL is mainly assumed to be caused by some specific distribution shifts such as label shift (the shift of $P(Y)$) Lipton et al. (2018) or covariate shift (the shift of $P(X)$) Reisizadeh et al. (2020), while federated DAG learning handles a generative model, which can allow the data heterogeneity resulted from the joint distribution shift of all variables (Figure 1(a)). This would bring more challenges than the FL paradigm model design.

[1]Notice that in this paper, we restrict our scope to define the privacy leakage by sharing the raw data of users.

In this paper, we present FedDAG, a gradient-based framework for learning the underlying graph structure from decentralized data, including the case of heterogeneous data caused by mechanisms change. (1) To alleviate the data leakage problem, FedDAG inherits the merits of FL, which proposes to deploy a local model to each client separately and collaboratively learn a joint model at the server end. Instead of sharing raw data, FedDAG exchanges model info among clients and the server to achieve collaboration. (2) Taking into consideration of the first main difference between FedDAG and FL, a two-level structure consisting of a graph structure learning (GSL) part and a mechanisms approximating (MA) part, respectively, is adopted as the local model. (3) Benefiting from this separated structure, the second difference between FL and FedDAG can naturally be handled by only sharing the GSL parts of clients during the learning procedure and locally updating the MA parts to get with data heterogeneity. Moreover, we provide the structure identifiability conditions for learning DAG structure from decentralized data in Appendix A. Our contributions are summarized as follows:

- We introduce the federated DAG structure learning task under the assumption that the underlying graph structure among different datasets remains invariant while mechanisms vary. We also show the structure identifiability conditions of DAG learning from the decentralized data.
- We propose FedDAG, which separately learns the mechanisms on local data and jointly learns the DAG structure to handle the data heterogeneity elegantly. Meanwhile, since 0 bits of raw data are shared, but only parameters of the GSL parts of models, the requirement of privacy protection is guaranteed, and the communication pressure is relatively low.
- We evaluate our proposed method on data following structure equation models of additive noise structure with a variety of experimental settings, including simulations and real datasets, against recent state-of-the-art algorithms to show its superior performance and the ability to use one model in all settings.

1.1 Potential applications of FedDAG

Compared to traditional DAG learning methods, the homogeneous data (no distribution shift) setup of our method brings one more assumption that local data cannot be directly collected owing to the consideration of *privacy leakage*. Then, we further extend our model to heterogeneous data, where mechanisms among variables may also vary among different local data. Therefore, our method could be directly applied to the applications of DAG structure learning, where privacy is also very important.

The first example can come from medical science. Exploring the underlying relations among concerned events from healthcare data can help to understand the disease mechanisms and causes (Yang et al., 2013). However, in real medical scenarios, the clinical data of patients are extremely sensitive and related to personal privacy, which faces stringent data protection regulations, such as HIPAA regulations (Annas, 2003). Each hospital may own finite clinical data for some rare diseases, which is not enough for DAG learning. How can hospitals cooperate to analyze the pathology while preventing sharing the private information (raw diagnostic data)? Naturally, this challenge can be addressed by our method. Depending on how each hospital collects data, e.g., medical devices and survey design, the data in each hospital may not share the same distribution. The second example can come from the recommendation system (RS) (Wang et al., 2020b; Yang et al., 2020). Leveraging BN model into RS is becoming prevail to perform robust recommendations by de-confounding some spurious relations. As users pay more attention to privacy and governments also exacerbate many strict regulations like the general data protection regulation (GDPR) (Voigt & Von dem Bussche, 2017), it increases difficulties in collecting raw personal data to the server. Accordingly, we think that RS can also benefit from FedDAG learning.

2 Preliminaries

Additive Noise Models (ANMs). We consider a specific structural equation model (SEM), which is defined as a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{F} \rangle$, where $\mathcal{X} = \{X_1, X_2, \cdots, X_d\}$ is a set of concerned variables. And $\mathcal{F} = \{f_1, f_2, \cdots, f_d\}$ is a set of functions. Each f_i, called the relationship mechanism of X_i, maps $\epsilon_i \cup \mathbf{PA}_i$ to X_i, i.e., $X_i = f_i(\mathbf{PA}_i, \epsilon_i)$, where the $\mathbf{PA}_i$ corresponds to the set including all direct parents of X_i and ϵ_i is

the random noise. $\mathcal{M}$ can be leveraged to describe how nature assigns values to variables of interest (Pearl et al., 2016). In this paper, we focus on a commonly used model named ANMs. They assume that

$$X_i = f_i(\mathbf{PA}_i) + \epsilon_i, \quad i = 1, 2, \cdots, d, \tag{1}$$

where ϵ_i is independent of variables in $\mathbf{PA}_i$ and mutually independent with any ϵ_j for $i \neq j$.

Bayesian Networks (BNs). Let $X = (X_1, X_2, \cdots, X_d)$ be a vector that includes all variables in $\mathcal{X}$ with the index set $\mathbb{V} := \{1, 2, \cdots, d\}$ and $P(X)$ with the probability density function $p(X)$ be a marginal distribution induced from $\mathcal{M}$. A DAG $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ consists of a nodes set $\mathbb{V}$ and an edge set $\mathbb{E} \subseteq \mathbb{V}^2$. Every SEM $\mathcal{M}$ can be associated with a DAG $\mathcal{G}_{\mathcal{M}}$, in which each node i corresponds to the variable X_i and directed edges point from $\mathbf{PA}_i$ to X_i[2] for $i \in [d]$[3]. A BN is defined as a pair $\langle P(X), \mathcal{G}_{\mathcal{M}} \rangle$. Then $\mathcal{G}_{\mathcal{M}}$ is called the graph structure associated with $\mathcal{M}$ and $P(X)$ is Markovian to $\mathcal{G}_{\mathcal{M}}$. Throughout the main text, we assume that there is no latent variable[4] (Spirtes et al., 2001) and then $p(X)$ can be factorized as

$$p(X) = \prod_{i=1}^{d} p(X_i | X_{pa_i}) \tag{2}$$

according to $\mathcal{G}_{\mathcal{M}}$ (Lauritzen, 1996). X_{pa_i} is the parental vector that includes all variables in $\mathbf{PA}_i$.

Characterizations of Acyclicity. A DAG $\mathcal{G}$ with d nodes can be represented by a binary adjacency matrix $\boldsymbol{B} = [\boldsymbol{B}_{:,1} | \boldsymbol{B}_{:,2} | \cdots | \boldsymbol{B}_{:,d}]$ with $\boldsymbol{B}_{:,i} \in \{0, 1\}^d$ for $\forall i \in [d]$. NOTEARS (Zheng et al., 2018) formulates a sufficient and necessary condition for $\boldsymbol{B}$ representing a DAG by an equation. The formulation is as follows:

$$\mathrm{Tr}[e^{\boldsymbol{B}}] - d = 0, \tag{3}$$

where $\mathrm{Tr}[\cdot]$ means the trace of a given matrix. $e^{(\cdot)}$, here, is the matrix exponential operation. However, NOTEARS is only designed to solve the linear Gaussian models, which assume that all relationship mechanisms are linear. Therefore, the DAG structure and relationship mechanisms can be modeled together by a weighted matrix. To extend NOTEARS to the non-linear cases, MCSL (Ng et al., 2022b) proposes to use a mask $\boldsymbol{M}$, parameterized by a continuous proxy matrix $\boldsymbol{U}$, to approximate the adjacency matrix $\boldsymbol{B}$. To enforce the entries of $\boldsymbol{M}$ to approximate the binary form, i.e., 0 or 1, a two-dimensional version of Gumbel-Softmax (Jang et al., 2017) approach named Gumbel-Sigmoid is designed to reparameterize $\boldsymbol{U}$ and to ensure the differentiability of the model. Then, $\boldsymbol{M}$ can be obtained element-wisely by

$$\boldsymbol{M}_{ij} = \frac{1}{1 + \exp(-\log(\boldsymbol{U}_{ij} + \mathrm{Gumb}_{ij})/\tau)}, \tag{4}$$

where τ is the temperature, $\mathrm{Gumb}_{ij} = g^1_{ij} - g^0_{ij}$, g^1_{ij} and g^0_{ij} are two independent samples from $\mathrm{Gumbel}(0, 1)$. For simplicity but equivalence, g^1_{ij} and g^0_{ij} also can be sampled from $-\log(\log(a))$ with $a \sim \mathrm{Uniform}(0, 1)$. See Appendix D in (Ng et al., 2022b). MCSL names Eq. (4) as Gumbel-Sigmoid w.r.t. $\boldsymbol{U}$ and temperature τ, which is written as $g_\tau(\boldsymbol{U})$. Then, the acyclicity constraint can be reformulated as

$$\mathrm{Tr}[e^{(g_\tau(\boldsymbol{U}))}] - d = 0. \tag{5}$$

3 Problem definition

Here, we first describe the property of decentralized data and the data distribution shift among different clients if there exists data heterogeneity (Huang et al., 2020b; Mooij et al., 2020; Zhang et al., 2020). Then, we define the problem, federated DAG structure learning, considered in this paper.

Decentralized data and probability distribution set. Let $\mathcal{C} = \{c_1, c_2, \cdots, c_m\}$ be the client set which includes m different clients and s be the only server. The data $\mathcal{D}^{c_k} \in \mathbb{R}^{n_{c_k} \times d}$, in which each observation

[2]In the intact graph structure of ANMs, we just fix directed edges from ϵ_i to X_i and assume the distribution of ϵ_i. Therefore, in this paper, $\mathcal{G}$ is only defined over the endogenous variables.

[3]For simplicity, we use $[d] = \{1, 2, \cdots, d\}$ to represent the set of all integers from 1 to d.

[4]This assumption can be relaxed to some restricted cases with latent variables. See Appendix C.3 for details.

$\mathcal{D}_i^{c_k}$ for $\forall i \in [n_{c_k}]$ independently sampled from its corresponding probability distribution $P^{c_k}(X)$, represent the personalized data owned by the client c_k. n_{c_k} is the number of observations in $\mathcal{D}^{c_k}$. The dataset $\mathcal{D} = \{\mathcal{D}^{c_1}, \mathcal{D}^{c_2}, \cdots, \mathcal{D}^{c_m}\}$ is called a decentralized dataset and $P^{\mathcal{C}}(X) = \{P^{c_1}(X), P^{c_2}(X), \cdots, P^{c_m}(X)\}$ is defined as the decentralized probability distribution set. If $P^{c_{k_1}}(X) = P^{c_{k_2}}(X)$ for $\forall\ k_1, k_2 \in [m]$, then $\mathcal{D}$ is defined as a homogeneous decentralized dataset throughout this paper. The heterogeneous decentralized dataset is defined by assuming that there exists at least two clients, e.g., c_{k_1} and c_{k_2}, on which the local data are sampled from different distributions, i.e., $P^{c_{k_1}}(X) \neq P^{c_{k_2}}(X)$.

Assumption 3.1. (Invariant DAG) For $\forall c_k$, $P^{c_k}(X) \in P^{\mathcal{C}}(X)$ admits the product factorization of Eq. (2) relative to the same DAG $\mathcal{G}$.

Remark 3.2. If $P^{\mathcal{C}}(X)$ satisfies Assumption 3.1, then, each $P^{c_k}(X) \in P^{\mathcal{C}}(X)$ is Markovian relative to $\mathcal{G}$.

According to the general definition of *mechanisms change* in (Tian & Pearl, 2001), interventions can be seen as a special case of distribution shifts, where the external influence involves fixing certain variables to some predetermined values. Actually, in general, the external influence may be milder to merely change the conditional probability of certain variables given its causes. In this paper, we restrict our scope by assuming that the distribution shifts across $P^{c_k}(X)$ come from the changes of mechanisms in $\mathcal{F}$ or distribution shifts of the exogenous variables in $\mathcal{E}$ (see Appendix F.1 for detailed discussion). More justifications on Assumption 3.1 are in Appendix F.2.

Assumption 3.3. For $\forall c_{k_1}, c_{k_2}$, if $P^{c_{k_1}}(X) \neq P^{c_{k_2}}(X)$, the distribution shifts are caused by (1) $\exists\ i \in [d]$, $P^{c_{k_1}}(X_i|X_{pa_i}) \neq P^{c_{k_2}}(X_i|X_{pa_i})$, i.e., $f_i^{c_{k_1}} \neq f_i^{c_{k_2}}$. (2) $\exists\ i \in [d]$, $P^{c_{k_1}}(\epsilon_i) \neq P^{c_{k_2}}(\epsilon_i)$.

Federated DAG Structure Learning. Given the decentralized dataset $\mathcal{D}$ consisting of data from m clients while the corresponding $P^{\mathcal{C}}(X)$ satisfies Assumptions 3.1 and 3.3, federated DAG structure learning aims to identify the underlying DAG $\mathcal{G}$ from $\mathcal{D}$.

4 Methodology

To solve the federated DAG structure learning problem, we formulate a continuous score-based method named FedDAG to learn the DAG structure from decentralized data. Firstly, we define an objective function that guides all models from different clients to federally learn the underlying DAG structure $\mathcal{G}$ (or adjacency matrix $\boldsymbol{B}$), and at the same time also to learn personalized mechanisms for each client. As shown in Figure 2, for each client c_k, the local model consists of a graph learning part and a mechanisms approximation part. The GSL part is parameterized by a matrix $\boldsymbol{U}^{c_k} \in \mathbb{R}^{d \times d}$, which would be the same for all clients finally[5]. To make every entry of $\boldsymbol{U}^{c_k}$ to approximate the binary entry of the adjacency matrix, a Gumbel-Sigmoid method (Jang et al., 2017; Ng et al., 2022b), represented as $g_\tau(\boldsymbol{U}^{c_k})$, is further leveraged to transform $\boldsymbol{U}^{c_k}$ to a differentiable approximation of the adjacency matrix. The mechanisms approximation parts $f_1^{c_k}, f_2^{c_k}, \cdots, f_d^{c_k}$ are parameterized by d sub-networks, each of which has d inputs and one output. In the learning procedure, the GSL parts (specifically $\boldsymbol{U}^{c_k}$) of participating clients are shared with the server s. Then, the processed information is broadcast to each client for self-updating its matrix. The details of our method are demonstrated in the following subsections.

4.1 The overall learning objective

Now we present the overall learning objective of FedDAG as the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\Phi}, \boldsymbol{U}}{\arg\max} \quad & \sum_{k=1}^{m} \mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U}) \\ \text{subject to} \quad & g_\tau(\boldsymbol{U}) \in \mathbf{DAGs} \ \Leftrightarrow\ h(\boldsymbol{U}) = \mathrm{Tr}[e^{(g_\tau(\boldsymbol{U}))}] - d = 0, \end{aligned} \tag{6}$$

where $\boldsymbol{\Phi}^{c_k} := \{\boldsymbol{\Phi}_1^{c_k}, \boldsymbol{\Phi}_2^{c_k}, \cdots, \boldsymbol{\Phi}_d^{c_k}\}$ represents the MA part of the model on c_k. $\mathcal{S}^{c_k}(\cdot)$ is the scoring function for evaluating the fitness of the local model of client c_k and observations $\mathcal{D}^{c_k}$. For score-based DAG structure

[5]Please notice that GSL parts of different clients may not be the same during the training procedure. So we index them.

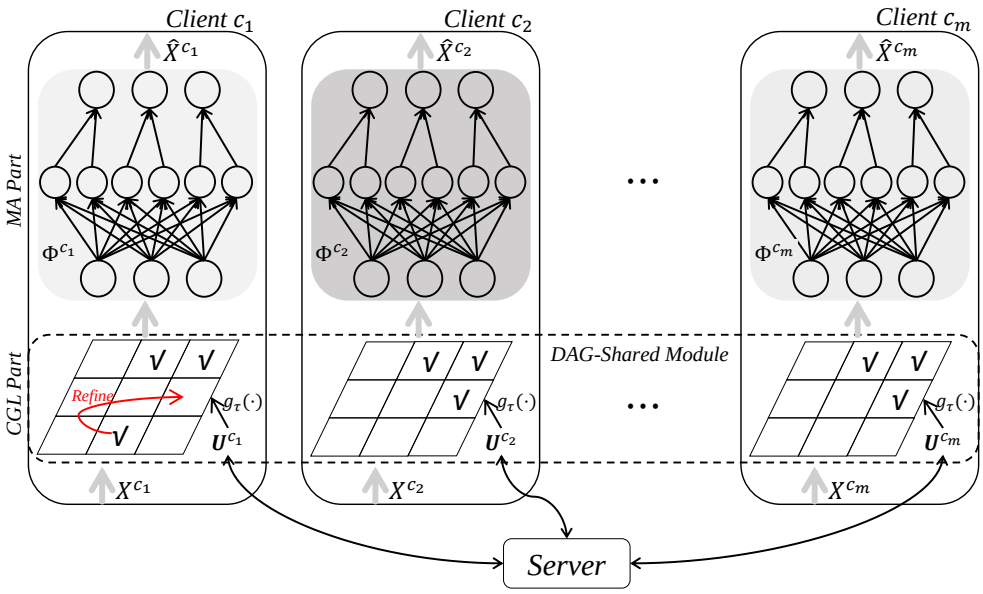


Figure 2: An overview of FedDAG. Each solid-line box includes the local model for each client. For client c_k, the GSL part includes a continuous proxy U^{c_k} and $g_\tau(\cdot)$, the Gumbel-Sigmoid function, which maps U^{c_k} to approximate the binary adjacency matrix. To approximate the mechanisms, the MA part uses Φ^{c_k}, including d neural networks. X^{c_k} represents observations on c_k and $\hat{X}^{c_k}$ is the predicted data. X^{c_k} firstly goes through the GSL part to select the parental variables and then the MA part to get $\hat{X}^{c_k}$. The server coordinates the FL procedures by leveraging $\boldsymbol{U}$ among clients.

learning, selecting a proper score function such as BIC score (Schwarz, 1978), generalized score function (Huang et al., 2018) or equivalently taking the likelihood of $P(X)$ with a penalty function on model parameters (Zheng et al., 2018; Ng et al., 2022b; Zheng et al., 2020; Lachapelle et al., 2020) can guarantee to identify up the underlying ground-truth graph structure $\mathcal{G}$ because $\mathcal{G}$ is supposed to have the maximal score over Eq. (6). Throughout all experiments in this paper, we assume the noise types are Gaussian with equal variance for each local distribution. And the overall score function utilized in this paper is as follows,

$$\mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U}^{c_k}) = -\frac{1}{2n_k}\sum_{i=1}^{n_k}\sum_{j}^{d}\|\mathcal{D}_{ij}^{c_k} - \boldsymbol{\Phi}_j^{c_k}(g_\tau(\boldsymbol{U}_{j,:}^{c_k}) \circ \mathcal{D}_i^{c_k})\|_2^2 - \lambda\|g_\tau(\boldsymbol{U})\|_1. \tag{7}$$

In our score function, we take the negative Least Squares loss and a sparsity term, which corresponds to the model complexity penalty in the BIC score (Schwarz, 1978).[6] However, the global minimum is hard to reach by using the gradient descent method due to the non-convexity of $h(\boldsymbol{U})$. More details on discussions of the optimization results can be found in Appendix. C.

In this paper, instead of directly taking the likelihood of $P(X)$, we leverage the well-known results on the density transformation to model the distribution of $P(\mathcal{E})$, i.e., maximizing the likelihood $P^{c_k}(\mathcal{E}|\mathcal{F}^{c_k}, \mathcal{G})$ for $\forall c_k \in \mathcal{C}$ (Mooij et al., 2011). According to Eq. (1), we have $\epsilon_i = X_i - f_i(\mathbf{PA}_i)$. That is to say, modeling $P(\mathcal{E})$ can be achieved by an auto-regressive model. To get ϵ_i, the first step is to select the parental set $\mathbf{PA}_i$ for X_i. This can be realized by $\boldsymbol{B}[:, i] \circ X$, where $\circ$ means the element-wise product. In our paper, for client c_k, we predict the noise by $\epsilon_i = X_i - \boldsymbol{\Phi}_i(g_\tau(\boldsymbol{U})[:, i] \circ X)$, where $g_\tau(\boldsymbol{U})$ is to approximate $\boldsymbol{B}$ and $\boldsymbol{\Phi}_i(\cdot)$ is parameterized by a neural network to approximate f_i. The specific formulation of $\mathcal{S}^{c_k}$ would depend on the assumption of noise distributions.

[6]The consistency of BIC score for learning graphs on ANMs is discussed in Appendix C.5.

Algorithm 1 FedDAG

1: **Input:** $\mathcal{D}$, $\mathcal{C}$, Parameter-list = $\{\alpha_{init}, \rho_{init}, h_{tol}, it_{max}, \rho_{max}, \beta, \gamma, r\}$
2: **Output:** $\mathbb{E}g_\tau(\boldsymbol{U}_t), \boldsymbol{\Phi}_t$
3: #Parameter Initializing
4: $t \leftarrow 1$, $\alpha_t \leftarrow \alpha_{init}$, $\rho_t \leftarrow \rho_{init}$
5: **while** $t \leq it_{max}$ and $h(\boldsymbol{U}_t) \geq h_{tol}$ and $\rho \leq \rho_{max}$ **do**
6: #Sub-problem Solving
7: $U_{t+1}, \Phi_{t+1} \leftarrow \mathrm{SPS}(\mathcal{D}, \mathcal{C}, \alpha_t, \rho_t, it_{in}, it_{fl}, r)$
8: #Coefficients Updating
9: $\alpha_{t+1} \leftarrow \alpha_t + \rho_t \mathbb{E}[h(\boldsymbol{U}_{t+1})], \quad t \leftarrow t+1$
10: **if** $\mathbb{E}[h(\boldsymbol{U}_{t+1})] > \gamma \mathbb{E}[h(\boldsymbol{U}_t)]$ **then**
11: $\rho_{t+1} = \beta \rho_t$
12: **else**
13: $\rho_{t+1} = \rho_t$
14: **end if**
15: **end while**

4.2 Federated DAG structure learning

As suggested in NOTEARS (Zheng et al., 2018), the hard-constraint optimization problem in Eq. (6) can be addressed by an Augmented Lagrangian Method (ALM) to get an approximate solution. Similar to the penalty methods, ALM transforms a constrained optimization problem by a series of unconstrained sub-problems and adds a penalty term to the objective function. ALM also introduces a Lagrangian multiplier term to avoid ill-conditioning by preventing the coefficient of the penalty term from going too large. To solve Eq. (6), the sub-problem can be written as

$$\arg\max_{\boldsymbol{\Phi}, \boldsymbol{U}} \sum_{k=1}^{m} \mathcal{S}^{c_k}\left(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, g_\tau(\boldsymbol{U})\right) - \alpha_t h(\boldsymbol{U}) - \frac{\rho_t}{2} h(\boldsymbol{U})^2, \tag{8}$$

where α_t and ρ_t are the Lagrangian multiplier and penalty parameter of the t-th sub-problem, respectively. These parameters are updated after the sub-problem is solved. Since neural networks are adopted to fit the mechanisms in our work, there is no closed-form solution for Eq. (8). Therefore, we solve it approximately via ADAM (Kingma & Ba, 2015). The method is described in Algorithms 1 and 2. And in Algorithm 1, we share the same coefficients updating strategy as in (Ng et al., 2022b).

Each sub-problem as Eq. (8) is solved mainly by distributing the computation across all local clients. Since data is prevented from sharing among clients and the server, each client owns its personalized model, which is only trained on its personalized data. The server communicates with clients by exchanging the parameters information of models and coordinates the joint learning task. To achieve so, our method alternately updates the server and clients in each communication round.

Client Update. For each model of client c_k, there are two main parts, named GSL part parameterized by $\boldsymbol{U}^{c_k}$ and MA part parameterized by $\boldsymbol{\Phi}^{c_k}$, respectively. Essentially, the joint objective in Eq. (8) guides the learning process. In the self-updating procedure as described in Algorithm 2, the clients firstly receive the updated penalty coefficients α_t and ρ_t and the averaged parameter $\boldsymbol{U}_{new}$. Then, the renewed learning personalized score of client c_k is defined as $\mathrm{SP}^{c_k} = \mathcal{S}^{c_k} - \alpha_t h(\boldsymbol{U}^{c_k}) - \frac{\rho_t}{2} h(\boldsymbol{U}^{c_k})^2$. it_{fl} times of local gradient-based parameter updates are operated to maximize its personalized score.

Server Update. After it_{fl} local updates, the server randomly chooses r clients to collect their $\boldsymbol{U}$s to the set $\mathbb{U}$. Then, $\boldsymbol{U}$s in $\mathbb{U}$ are averaged to get $\boldsymbol{U}_{new}$. The other operation on the server is updating the α_t, ρ_t to α_{t+1}, ρ_{t+1}. The detailed calculating rules are described at lines $8-14$ in Algorithm 1. Then, the new penalty coefficients and parameters are broadcast to all clients. Notice that assuming that data distribution across clients is homogeneous (no distribution shift), $\boldsymbol{\Phi}^{c_k}$ of the chosen r clients can also be collected and averaged to update the local models of clients in the same way, which is named as All-Shared FedDAG (AS-FedDAG) in this paper. For clarity, we name our general method as Graph-Shared (GS-FedDAG) to distinguish it from

Algorithm 2 Sub-Problem Solver (SPS) for FedDAG

Input: $\mathcal{D}$, $\mathcal{C}$, Parameter-list $= \{\alpha_t, \rho_t, it_{in}, it_{fl}, r\}$
Output: $\boldsymbol{U}_{new}$, $\boldsymbol{\Phi}^{it_{in}}$
Define $\mathrm{SP}^{c_k} = \mathcal{S}^{c_k} - \alpha_t h(\boldsymbol{U}^{c_k}) - \frac{\rho_t}{2} h(\boldsymbol{U}^{c_k})^2$
for i in $(1, 2, \cdots, it_{in})$ **do**
  for each client c_k **do**
    #Self-updating
    $\boldsymbol{U}^{i,c_k}, \boldsymbol{\Phi}^{i,c_k} \leftarrow \arg\max_{\boldsymbol{\Phi}^{c_k}, \boldsymbol{U}^{c_k}} \mathrm{SP}^{c_k}$
  end for
  if i (% it_{fl}) $= 0$ or $i = it_{in}$ **then**
    #Aggregating: randomly select r clients and collect their $\boldsymbol{U}$s into $\mathbb{U}$, then, send $\mathbb{U}$ to the server
    $\mathbb{U} \leftarrow \mathrm{Agg}(r, \mathcal{C})$
    #Server Updating: average $\boldsymbol{U} \in \mathbb{U}$
    $\boldsymbol{U}_{new} \leftarrow \mathrm{Avg}(\mathbb{U})$
    #Broadcasting: distribute U_{new} to all clients
    $\mathcal{C} \leftarrow \mathrm{BD}(\boldsymbol{U}_{new})$
    for each client c_k **do**
      #Clients Updating
      $\boldsymbol{U}^{i,c_k} \leftarrow \boldsymbol{U}_{new}$
    end for
  end if
end for

AS-FedDAG. It is worth noting that AS-FedDAG can further enhance the performance in the homogeneous case but introduce some additional communication costs.

4.3 Thresholding

For continuous optimization, as illustrated in the previous work (Ng et al., 2022b), we leverage Gumbel-Sigmoid to approximate the binary mask. That is to say, the exact 0 or 1 is hard to get. The other issue is raised by ALM since the solution of ALM only satisfies the numerical precision of the constraint. This is because we set h_{tol} and it_{max} maximally but not infinite coefficients of penalty terms to formulate the last sub-problem. Therefore, some entries of the output $\boldsymbol{M} = \mathbb{E}g_\tau(\boldsymbol{U})$ will be near but not exactly 0 or 1. To alleviate this issue, ℓ_1 sparsity is added to the objective function. In our method, since all mask values are in $[0, 1]$, we take the median value 0.5 as the threshold to prune the edges, which follows the same way in our baseline method MCSL (Ng et al., 2022b). The iterative thresholding method is also taken to deal with the case that the learned graph is cyclic. This may happen if the number of variables is large (40 variables in our paper). Because, in numerical optimization, the constraint penalty exponentially decreases with the number of variables. To deal with the cyclic graph, we one-by-one cut the edge with the minimum value until the graph is acyclic. Until now, all continuous search methods for DAG learning suffer from these two problems. It is an interesting future direction to be investigated.

4.4 Convergence analysis

Let us quickly review our method. For each client c_k, the model parameters include $\boldsymbol{\Phi}^{c_k}$ and $\boldsymbol{U}^{c_k}$. Each client optimizes its parameters on its own data $\mathcal{D}^{c_k}$. Like NOTEARS and its following works, our method can reach a stationary point instead of the global maximum (the ground-truth DAG). Then, we separate our discussion into homogeneous and heterogeneous data.

4.4.1 Homogeneous data

For the no distribution shift case, we have $\boldsymbol{\Phi}^{c_1} = \boldsymbol{\Phi}^{c_2} = \cdots = \boldsymbol{\Phi}^{c_m}$ and $\boldsymbol{U}^{c_1} = \boldsymbol{U}^{c_2} = \cdots = \boldsymbol{U}^{c_m}$. Our method named AS-FedDAG (All-Shared FedDAG) sets a central server, which regularly (1) receives all parameters

(or $\boldsymbol{U}^{c_k}$ for GS-FedDAG), (2) averages these parameters to get $\boldsymbol{\Phi}^{new}$ and $\boldsymbol{U}^{new}$ and (3) broadcasts $\boldsymbol{\Phi}^{new}$ and $\boldsymbol{U}^{new}$ to all clients during the learning procedures. AS-FedDAG benefits from an advanced technique named FedAvg (McMahan et al., 2017) for solving the FL problem in the homogeneous data case. FedAvg solves a similar problem by averaging all parameters learned from each client in the learning process.

4.4.2 Heterogeneous data

To solve the overall constraint-based problem, we take ALM to convert the hard constraint to a soft constraint with a series of increasing penalty co-efficiencies. The convergence of ALM for the non-convex problem has been well studied (Nemirovski, 1999) and presented in NOTEARS (Zheng et al., 2018). Thus, we only consider the convergence analysis of our method directly from the inner optimization, i.e., the t-th sub-problem, as follows.

$$\arg\max_{\boldsymbol{\Phi},\boldsymbol{U}} \sum_{k=1}^{m} \mathcal{S}^{c_k}\left(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, g_\tau(\boldsymbol{U})\right) - \alpha_t h(\boldsymbol{U}) - \frac{\rho_t}{2} h(\boldsymbol{U})^2. \tag{9}$$

Here, for simplification, we just define that $\hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}^{c_k}, \boldsymbol{U}) = -\mathcal{S}^{c_k}\left(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, g_\tau(\boldsymbol{U})\right) + \alpha_t h(\boldsymbol{U}) + \frac{\rho_t}{2} h(\boldsymbol{U})^2$. Then, the overall optimization problem can be reformulated as follows.

$$\arg\min_{\boldsymbol{\Phi},\boldsymbol{U}} \hat{\mathcal{S}}(\boldsymbol{U}, \boldsymbol{\Phi}) := \sum_{k=1}^{m} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k}). \tag{10}$$

Through the following part, we use $\nabla_{\boldsymbol{U}}$ and $\nabla_{\boldsymbol{\Phi}}$ to represent the gradients of $\hat{\mathcal{S}}(\boldsymbol{U}, \boldsymbol{\Phi})$ w.r.t $\boldsymbol{U}$ and $\boldsymbol{\Phi}^{c_k}$, respectively. And, we use $\tilde{\nabla}_{\boldsymbol{U}}$ and $\tilde{\nabla}_{\boldsymbol{\Phi}}$ to represent the stochastic gradients calculated by a mini-batch of observations w.r.t $\boldsymbol{U}$ and $\boldsymbol{\Phi}^{c_k}$, respectively.

Definition 4.1. (Partial Gradient Diversity). The gradient diversity among all local learning objectives as:

$$\sum_{i=1}^{m} \left\| \nabla_{\boldsymbol{U}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k}) - \nabla_{\boldsymbol{U}} \hat{\mathcal{S}}(\boldsymbol{U}, \boldsymbol{\Phi}) \right\|^2 \leq \delta^2. \tag{11}$$

Note that the notation of gradient diversity is introduced (Yin et al., 2018; Haddadpour & Mahdavi, 2019) as a measurement to compute the similarity among gradients updated on different clients.

Assumption 4.2. (Smoothness and Lower Bound). The local objective function $\hat{\mathcal{S}}^{c_k}(\cdot)$ of the k-th client is differentiable for $\forall k \in [m]$. Also, $\nabla_{\boldsymbol{U}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k})$ is $L_{\boldsymbol{U}}$-Lipschitz w.r.t $\boldsymbol{U}$ and $L_{\boldsymbol{U}\boldsymbol{\Phi}}$ w.r.t $\boldsymbol{\Phi}^{c_k}$, and $\nabla_{\boldsymbol{\Phi}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k})$ is $L_{\boldsymbol{\Phi}}$-Lipschitz w.r.t $\boldsymbol{\Phi}^{c_k}$ and $L_{\boldsymbol{\Phi}\boldsymbol{U}}$ w.r.t $\boldsymbol{U}$. We also assume the overall objective function can be bounded by a constant $\hat{\mathcal{S}}^*$ and denote $\Delta\hat{\mathcal{S}}_0 = \hat{\mathcal{S}}(\boldsymbol{U}_0, \boldsymbol{\Phi}_0) - \hat{\mathcal{S}}^*$.

The relative cross-sensitivity of $\nabla_{\boldsymbol{U}} \hat{\mathcal{S}}^{c_k}$ w.r.t $\boldsymbol{\Phi}^{c_k}$ and $\nabla_{\boldsymbol{\Phi}} \hat{\mathcal{S}}^{c_k}$ w.r.t $\boldsymbol{U}$ with the scalar

$$\chi := \max\left\{L_{\boldsymbol{U}\boldsymbol{\Phi}}, L_{\boldsymbol{\Phi}\boldsymbol{U}}\right\} / \sqrt{L_{\boldsymbol{U}} L_{\boldsymbol{\Phi}}}. \tag{12}$$

Assumption 4.3. (Bounded Local Variance) For each local data $\mathcal{D}^{c_k}, k \in [m]$, we can independently sample a batch of data denoted as $\xi \subset \mathcal{D}^{c_k}$. Then, there exist constant $\delta_{\boldsymbol{U}}$ and $\delta_{\boldsymbol{\Phi}}$ such that

$$\mathbf{E}\left[\left\|\tilde{\nabla}_{\boldsymbol{U}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}^{c_k}, \boldsymbol{U}) - \nabla_{\boldsymbol{U}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k})\right\|^2\right] \leq \sigma_{\boldsymbol{U}}^2,$$
$$\mathbf{E}\left[\left\|\tilde{\nabla}_{\boldsymbol{\Phi}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}^{c_k}, \boldsymbol{U}) - \nabla_{\boldsymbol{\Phi}} \hat{\mathcal{S}}^{c_k}(\boldsymbol{U}, \boldsymbol{\Phi}^{c_k})\right\|^2\right] \leq \sigma_{\boldsymbol{\Phi}}^2,$$

The bounded variance assumption is a standard assumption on the stochastic gradients (Haddadpour & Mahdavi, 2019; Pillutla et al., 2022).

Theorem 4.4. *(Convergence of GS-FedDAG). For GS-FedDAG with all clients involved in the aggregation, for $\forall 0 \leq it \leq T-1$, under Assumptions 4.2, 4.3 and 4.1, and the learning rate for the $\boldsymbol{U}$ part is set as $\eta/(L_{\boldsymbol{U}} it_{in})$ and the learning rate for the $\boldsymbol{\phi}$ part is set as $\eta/(L_{\boldsymbol{\phi}} it_{in})$. Then, for η, depending on the problem parameters, we have*

$$\frac{1}{T}\sum_{it=0}^{T-1}\left(\frac{1}{L_{\boldsymbol{U}}}\mathbb{E}\left[\left\|\nabla_{\boldsymbol{U}}\hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}_{it}^{c_k},\boldsymbol{U}_{it})\right\|^2\right]+\frac{1}{L_{\boldsymbol{\Phi}}}\mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\left\|\nabla_{\boldsymbol{U}}\hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}_{it}^{c_k},\boldsymbol{U}_{it})\right\|^2\right]\right)\leq \frac{(\Delta\hat{\mathcal{S}}_0\sigma^2_{\text{FedDAG},1})^{1/2}}{\sqrt{T}}+\frac{(\Delta\hat{\mathcal{S}}_0^2\sigma^2_{\text{FedDAG},2})^{1/3}}{T^{2/3}}+\mathcal{O}(\frac{1}{T}). \tag{13}$$

where we define the effective variance terms

$$\begin{aligned}\sigma^2_{\text{FedDAG},1}&=\left(1+\chi^2\right)\left(\frac{\sigma^2_{\boldsymbol{U}}}{L_{\boldsymbol{U}}}+\frac{\sigma^2_{\boldsymbol{\Phi}}}{L_{\boldsymbol{\Phi}}}\right),\\ \sigma^2_{\text{FedDAG},2}&=\left(1+\chi^2\right)\left(\frac{\delta^2}{L_{\boldsymbol{U}}}+\frac{\sigma^2_{\boldsymbol{U}}}{L_{\boldsymbol{U}}}+\frac{\sigma^2_{\boldsymbol{\Phi}}}{L_{\boldsymbol{\Phi}}}\right)\left(1-\frac{1}{it_{in}}\right),\end{aligned} \tag{14}$$

where it_{in} is the total step of one inner loop used in lines $4-21$ in Algorithm 2.

From Theorem 4.4, we can see that the gradients $\nabla_{\boldsymbol{U}}\hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}_{it}^{c_k},\boldsymbol{U}_{it})$ w.r.t $\boldsymbol{U}$ and $\nabla_{\boldsymbol{U}}\hat{\mathcal{S}}^{c_k}(\boldsymbol{\Phi}_{it}^{c_k},\boldsymbol{U}_{it})$ w.r.t $\boldsymbol{\Phi}$ at the t-th step can be bounded if we choose a proper η, which affects the learning rates of the model.

The proof of Theorem 4.4 can be borrowed from the proof of Theorem 2 in (Pillutla et al., 2022). Notice that, in our theorem, we have assumed that all clients participate in the aggregation for simplification and the conclusion can be easily extended to the general partial participation case.

4.5 Privacy and costs discussion

Privacy issues of FedDAG. The strongest motivation of FL is to avoid *personalized raw data leakage*. To achieve this, FedDAG proposes to exchange the parameters for modeling the graph. Here, we argue that the information leakage of local data is rather limited. The server, receiving parameters with client index, may infer some data property. However, according to the data generation model (1), the distribution of local data is decided by (1) DAG structure, (2) noise types/strengths, and (3) mechanisms. The gradient information of the shared matrix is decided by (1) the learning objective and (2) model architecture, which are agnostic to the server. Especially for the network part, clients may choose different networks to make the inference more complex. Moreover, suppose the graph structure is also private information for clients. In that case, this problem can be easily solved by selecting a client to serve as the proxy server[7]. The proxy server needs to play two roles, including training its own model and taking on the server's duties. Then, other clients communicate with the proxy server instead of a real server in the communication round. Moreover, the aim of our work, and federated learning in general, is not to provide a full solution to privacy protection. Instead, it is the first step towards this goal, i.e., no data sharing between clients. To further protect privacy, more constraints need to be added to the federated learning framework, such as the prevention of information leakage from gradient sharing, which are studied under the privacy umbrella. To further enhance privacy protection, our method can also include more advanced privacy protection techniques (Wei et al., 2020b), which would be an interesting work to be investigated.

Communication cost. Since FedDAG requires exchanging parameters between the server and clients. Additional communication costs are raised. In our method, however, we argue that GS-FedDAG only brings rather small additional communication pressures. For the case of d variables, a single communication only exchanges a $d\times d$ matrix twice (sending and receiving). For homogeneous data, which assumes that local data are sampled from the same distribution, one can also transmit the neural network together to further improve the performance since mechanisms are also shared among clients. The trade-off between performances and communication costs can also be controlled by r in Algorithm 2, i.e., enlarging or reducing r. Surprisingly, we find that reducing r does not harm the performance severely (see Table 17 in Appendix D for detailed results). Moreover, the partial communication method, which only chooses some clients to exchange training information, is also leveraged to address the issue that not all clients are always online at the same time.

[7]Notice that the DAG structure encoded in the data is not a secret for the data owners (clients).

5 Experimental Results

In this section, we study the empirical performances of FedDAG on both synthetic and real-world data. More detailed ablation experiments can also be found in Appendix D.

Baselines We compare our method with various baselines including some continuous search methods, named NOTEARS (Zheng et al., 2018), NOTEARS-MLP (N-S-MLP, for short) (Zheng et al., 2020), DAG-GNN (Yu et al., 2019) and MCSL (Ng et al., 2022b), and also two traditional combinatorial search methods named PC (Spirtes et al., 2001) and GES (Chickering, 2002). The comparison results with another method named causal additive models (CAM) (Bühlmann et al., 2014) are put in Appendix D.5. Furthermore, we also include a concurrent work named NOTEARS-ADMM (Ng & Zhang, 2022), which also considers learning the Bayesian network in the federated setup. Since NOTEARS-ADMM focuses more on the homogeneous case and linear settings and pays less attention to the nonlinear cases, we only include the results on linear cases of NOTEARS-ADMM in the main paper for fair comparisons. More detailed comparisons are shown in Appendix D.6. Moreover, we also compare our FedDAG with a voting method (Na & Yang, 2010) in Appendix D.4, which also tries to learn DAG from decentralized data. We provide two training ways for these compared methods. The first way named "All data" is using all data to train only one model, which, however, is not permitted in FedDAG since the ban of data sharing in our setting. For the homogeneous data case, the results on this setting can be an *approximate upper bound* of our method but unobtainable. The second one named "Separated data" is separately training each siloed model over its personalized data, of which the performances reported are the average results of all clients.

Metrics. We report two metrics named Structural Hamming Distance (SHD) and True Positive Rate (TPR) averaged over 10 random repetitions to evaluate the discrepancies between estimated DAG and the ground-truth graph $\mathcal{G}$. See more details about SHD, and TPR in Appendix B.3. Notice that PC and GES can only reach the completed partially DAG (CPDAG, or MEC) at most, which shares the same Skeleton with the ground-truth DAG $\mathcal{G}$. When we evaluate SHD, we just ignore the direction of undirected edges learned by PC and GES. That is to say, these two methods can get SHD 0 if they can identify the CPDAG. The implementation details of all methods are given in Appendix B.

5.1 Synthetic data

The synthetic data we consider here is generated from Gaussian ANMs (Model (1)). Two random graph models named Erdős-Rényi (ER) and Scale-Free (SF) (detailed definitions are shown in Appendix B.1.) are adopted to generate the graph structure $\mathcal{G}$. And then, for each node V_i corresponding to X_i in $\mathcal{G}$, we sample a function from the given function sets to simulate f_i. Finally, data are generated according to a specific sampling method. In the following experiments, we take 10 clients and each with 600 observations (unless otherwise specified in some ablation studies.) throughout this paper. According to Assumption 3.1, data across all clients share the same DAG structure for both homogeneous and heterogeneous data settings. Due to the space limit, more ablation experiments, such as *unevenly distributed observations*, *varying clients*, *dense graph*, *different non-linear functions*, and *different number of observations*, etc., are put in Appendix D. All detailed discussions on the experimental results are in Appendix E.

5.1.1 Homogeneous data setting

Results on linear models. For a fair comparison, here, we also provide the linear version of our method. Since linear data are parameterized with an adjacency matrix, we can directly take the adjacency matrix as our model instead of a GSL part and a MA part. During training, the matrix is communicated and averaged by the server to coordinate the joint learning procedures.

NOTEARS-ADMM is also a DAG structure learning method from decentralized data. Different from our averaging strategy to exchange training information among clients, the optimization problem is solved by the alternating direction method of multipliers (ADMM). From Table 1, we find that our method can consistently show its advantage in the linear case. In the *ER2 with 10 nodes* setting, our AS-FedDAG is even better than NOTEARS with all training data. While it is possible and the detailed explanation can be found in Appendix E.

Table 1: Results on the linear model (Homogeneous data).

	ER2 with 10 nodes		SF2 with 10 nodes		ER2 with 20 nodes		SF2 with 20 nodes	
	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
PC-All	9.0 ± 3.9	0.58 ± 0.14	4.4 ± 1.3	0.76 ± 0.07	18.2 ± 5.9	0.59 ± 0.12	22.3 ± 4.8	0.48 ± 0.08
GES-All	7.5 ± 10.1	0.82 ± 0.25	4.1 ± 5.6	0.89 ± 0.14	25.2 ± 22.1	0.81 ± 0.16	22.1 ± 11.8	0.74 ± 0.15
NOTEARS-All	1.6 ± 1.6	0.93 ± 0.06	*1.4 ± 1.1*	*0.92 ± 0.05*	*3.0 ± 2.7*	*0.94 ± 0.06*	*6.9 ± 7.0*	*0.86 ± 0.12*
NOTEARS-Sep	3.0 ± 2.2	0.85 ± 0.08	3.6 ± 2.1	0.83 ± 0.10	4.1 ± 2.4	0.91 ± 0.05	10.2 ± 5.9	0.82 ± 0.10
NOTEARS-ADMM	4.7 ± 3.9	0.89 ± 0.12	4.4 ± 3.0	0.86 ± 0.09	7.9 ± 5.9	0.89 ± 0.07	10.7 ± 5.3	0.82 ± 0.08
AS-FedDAG	$\mathbf{1.3 \pm 1.5}$	$\mathbf{0.94 \pm 0.07}$	$\mathbf{1.6 \pm 1.0}$	$\mathbf{0.91 \pm 0.06}$	$\mathbf{3.9 \pm 3.1}$	0.91 ± 0.06	$\mathbf{9.4 \pm 6.7}$	0.82 ± 0.12

Table 2: Results on the nonlinear ANM with GP (Homogeneous data).

		ER2 with 10 nodes		SF2 with 10 nodes		ER2 with 40 nodes		SF2 with 40 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	15.3 ± 2.6	0.37 ± 0.10	14.1 ± 4.3	0.44 ± 0.20	84.9 ± 13.4	0.40 ± 0.08	95.0 ± 10.4	0.36 ± 0.07
	GES	13.0 ± 3.9	0.50 ± 0.18	9.6 ± 4.4	0.71 ± 0.17	59.0 ± 9.8	0.53 ± 0.08	73.8 ± 11.9	0.47 ± 0.10
	NOTEARS	16.5 ± 2.0	0.05 ± 0.04	14.5 ± 1.1	0.09 ± 0.07	71.2 ± 7.2	0.08 ± 0.03	70.8 ± 2.3	0.07 ± 0.03
	N-S-MLP	8.1 ± 3.8	0.56 ± 0.17	8.3 ± 2.8	0.51 ± 0.16	45.3 ± 6.8	0.43 ± 0.08	49.2 ± 7.7	0.39 ± 0.09
	DAG-GNN	16.2 ± 2.1	0.07 ± 0.06	15.2 ± 0.8	0.05 ± 0.05	73.0 ± 7.7	0.06 ± 0.03	72.4 ± 1.6	0.05 ± 0.02
	MCSL	1.9 ± 1.5	*0.90 ± 0.08*	*1.6 ± 1.2*	*0.91 ± 0.07*	*25.4 ± 13.1*	*0.68 ± 0.14*	31.6 ± 10.0	*0.59 ± 0.13*
Sep data	PC	14.1 ± 2.4	0.31 ± 0.06	13.6 ± 2.7	0.30 ± 0.10	83.8 ± 7.4	0.24 ± 0.03	86.1 ± 4.6	0.23 ± 0.04
	GES	12.7 ± 2.7	0.37 ± 0.09	12.7 ± 2.4	0.33 ± 0.11	71.0 ± 6.7	0.29 ± 0.03	73.2 ± 4.4	0.29 ± 0.05
	NOTEARS	16.5 ± 2.0	0.06 ± 0.04	14.6 ± 1.0	0.09 ± 0.06	71.1 ± 7.3	0.08 ± 0.03	70.7 ± 2.0	0.07 ± 0.03
	N-S-MLP	8.5 ± 2.9	0.56 ± 0.13	8.7 ± 2.9	0.53 ± 0.16	51.0 ± 6.9	0.41 ± 0.06	53.6 ± 5.5	0.39 ± 0.08
	DAG-GNN	15.7 ± 2.3	0.11 ± 0.05	14.5 ± 1.0	0.10 ± 0.06	71.5 ± 7.5	0.08 ± 0.02	70.8 ± 1.8	0.07 ± 0.02
	MCSL	7.1 ± 3.2	0.83 ± 0.08	6.9 ± 2.8	0.84 ± 0.08	77.3 ± 19.8	0.64 ± 0.11	72.9 ± 16.4	$\mathbf{0.58 \pm 0.13}$
	GS-FedDAG	$\mathbf{2.4 \pm 2.0}$	$\mathbf{0.86 \pm 0.13}$	$\mathbf{2.7 \pm 2.2}$	$\mathbf{0.86 \pm 0.13}$	$\mathbf{36.5 \pm 12.1}$	$\mathbf{0.65 \pm 0.15}$	$\mathbf{46.4 \pm 10.4}$	0.57 ± 0.13
	AS-FedDAG	$\mathbf{1.8 \pm 2.0}$	$\mathbf{0.89 \pm 0.12}$	$\mathbf{2.5 \pm 2.7}$	$\mathbf{0.85 \pm 0.15}$	$\mathbf{30.0 \pm 12.3}$	$\mathbf{0.74 \pm 0.15}$	$\mathbf{31.5 \pm 10.0}$	$\mathbf{0.59 \pm 0.13}$

Results on the nonlinear model. For the nonlinear setting, all data are generated by an ANM and divided into 10 pieces. Each f_i is sampled from a Gaussian Process (GP) with RBF kernel of bandwidth one (See Table 14 and Table 15 in Appendix. D for results of other functions.) and noises are sampled from one zero-mean Gaussian distribution with fixed variance. We consider graphs of d nodes and $2d$ expected edges.

Experimental results are reported in Table 2 with nodes 10 and 40. Since all local data are homogeneous, here, we also provide another effective training method named AS-FedDAG, in which the MA parts are also shared among clients. In all settings, AS-FedDAG shows a better performance than GS-FedDAG because more model information is shared during training. While GS-FedDAG can also show a consistent advantage over other methods. When separately training local models, all models suffer from data scarcity. Therefore, we can observe that both GS-FedDAG and AS-FedDAG perform better than the other methods in the fashion of separate training. NOTEARS and DAG-GNN, as continuous search methods, obtain unsatisfactory results due to the weak model capacity and improper model assumption. In contrast, the BIC score of GES gets a linear-Gaussian likelihood, which is incapable to deal with non-linear data[8]. With the number of nodes increasing, GS-FedDAG still shows better results than the closely-related baseline method MCSL. However, NOTEAES-MLP can show a comparable result with GS-FedDAG owing to the advantage over MCSL.

Here, we give a more detailed explanation of why our FedDAG method performs better than the baseline methods. For PC and GES, they can only reach the CPDAG (or MEC) at most, which shares the same skeleton with the ground-truth DAG. When we evaluate the SHD, we just ignore the direction of undirected edges learned by PC and GES. That is to say, these two methods can get SHD 0 if they can identify the true CPDAG. Therefore, the final results are not caused by unfair comparison. For PC, the independence test is leveraged to decode the (conditional) independence from the data distribution. Therefore, the accuracy would be affected by (1) the number of observations and (2) the effectiveness of *the non-parametric kernel independence test* method. GES leverages greedy search with BIC score. However, the likelihood part of BIC in GES is Linear Gaussian, which is unsuitable for data generated by the Non-linear model. NOTEARS

[8]Please find the ablation experiment with linear data and more discussions of the experimental results in Appendix D.

is a linear model but the mechanisms are non-linear. The reason will be the unfitness between data and model. Therefore, the comparisons with GES and NOTEARS on linear homogeneous data are implemented in the Table 1. DAG-GNN is also a non-linear model. However, the non-linear assumption of DAG-GNN is not the same as the data generation model ANMs assumed in our paper. The second reason comes from its *mechanisms approximation* modules are compulsory to share some parameters. Both NOTEARS-MLP and MCSL have their advantages. Please refer to Tables 14 and 15, you will find that NOTEARS-MLP performs better when the non-linear functions are MIM and MLP while MCSL works better on GP and GP-add models.

Visualization of the learned DAG of FedDAG. We take an example of the AS-FedDAG optimization process on linear Gaussian model with NOTEARS as the baseline method and plot the change of estimated parameters in Fig. 3 and Fig. 4. In this example, the number of nodes is set as 10 and the edges are 10. The data is simulated by ER graph and evenly assign 200 observations on two different clients. In Fig. 3, we can see that the learned graph is asymptotically approximating the ground-truth DAG B_G, including the existence of edges and their weights. From Fig. 4, we can find that with the increase of the penalty coefficients, h_{loss} decreases quickly. For learned graphs on the different clients, we can see that the SHD distance is smaller during the optimization procedures.

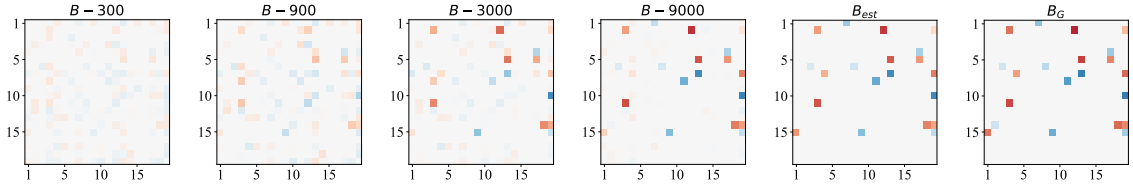


Figure 3: Visualization of the learned graph during the optimization process. $B - n$ means the learned graph in the n steps. B_{est} is the final estimated DAG. B_G is the ground-truth DAG.

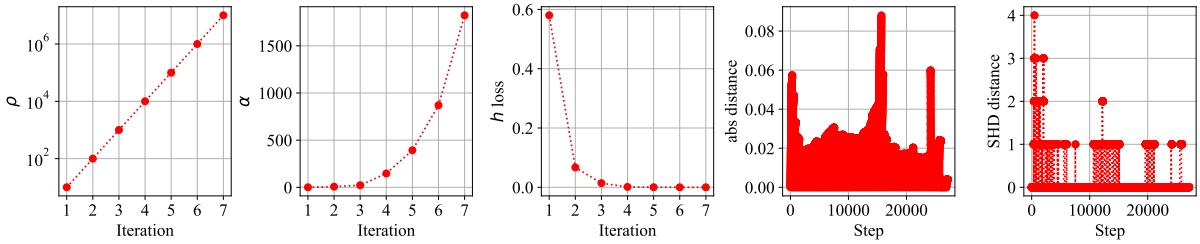


Figure 4: Parameters changing during the optimization process. The first three sub-figures include the changes of penalty coefficients ρ, α, and the DAG constraint loss h_{loss}. The fourth sub-figure records the ℓ_1 distance between two learned graphs on the different clients. The fifth sub-figure records the SHD distance between two learned graphs on the different clients.

5.1.2 Heterogeneous data setting

As defined in Section 3, the heterogeneous data property of data across clients come from the changes in mechanisms or the shift of noise distributions. To simulate the heterogeneous data, we first generate a graph structure shared by all clients and then decide the types of mechanisms $f_i^{c_k}$ and noises ϵ_i for $i \in [d]$ for each client c_k. In our experiments, we allow that f^{c_k} can be linear or non-linear for each client. If being linear, f^{c_k} here is a weighted adjacency matrix with coefficients sampled from Uniform $([-2.0, -0.5] \cup [0.5, 2.0])$, with equal probability. If being non-linear, $f_i^{c_k}$ is independently sampled from GP, GP-add, MLP, or MIM functions (Yuan, 2011), randomly. Then, a fixed zero-mean Gaussian noise is set to each client with a randomly sampled variance from $\{0.8, 1\}$.

We can see that the conclusion of experimental results on the heterogeneous data setting is rather similar to that of the homogeneous data. As can be read from Table 3, GS-FedDAG always shows the best performances

Table 3: Results on ANMs with heterogeneous data.

		ER2 with 10 nodes		SF2 with 10 nodes		ER2 with 40 nodes		SF2 with 40 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	22.3 ± 4.2	0.41 ± 0.11	21.0 ± 3.6	0.41 ± 0.12	151.9 ± 14.2	0.27 ± 0.08	152.5 ± 5.4	0.26 ± 0.04
	GES	26.4 ± 6.2	0.53 ± 0.14	25.4 ± 4.6	0.54 ± 0.13	NaN	NaN	NaN	NaN
	NOTEARS	20.4 ± 4.1	0.49 ± 0.14	18.7 ± 3.3	0.45 ± 0.11	164.8 ± 47.4	0.39 ± 0.07	178.1 ± 33.0	0.40 ± 0.10
	N-S-MLP	22.8 ± 5.0	0.87 ± 0.07	24.7 ± 3.3	0.88 ± 0.07	344.4 ± 71.9	*0.92 ± 0.08*	325.0 ± 50.2	*0.85 ± 0.08*
	DAG-GNN	21.2 ± 6.0	0.39 ± 0.11	16.6 ± 3.0	0.48 ± 0.18	146.6 ± 41.6	0.29 ± 0.08	168.2 ± 34.2	0.31 ± 0.09
	MCSL	19.4 ± 4.4	0.75 ± 0.19	19.0 ± 4.0	0.81 ± 0.14	118.6 ± 18.1	0.68 ± 0.11	126.9 ± 16.5	0.59 ± 0.12
Sep data	PC	12.5 ± 2.7	0.45 ± 0.07	11.0 ± 2.1	0.49 ± 0.07	65.7 ± 11.0	0.43 ± 0.06	73.7 ± 5.5	0.36 ± 0.05
	GES	12.9 ± 2.6	0.58 ± 0.07	10.3 ± 2.8	0.60 ± 0.09	68.2 ± 20.8	0.65 ± 0.09	77.2 ± 13.8	0.60 ± 0.07
	NOTEARS	7.6 ± 2.6	0.60 ± 0.11	7.6 ± 1.8	0.58 ± 0.09	34.9 ± 12.7	0.63 ± 0.11	43.4 ± 8.4	0.53 ± 0.10
	N-S-MLP	5.2 ± 1.4	0.80 ± 0.05	6.1 ± 1.6	0.76 ± 0.05	46.0 ± 10.2	0.73 ± 0.08	56.0 ± 9.5	0.66 ± 0.09
	DAG-GNN	8.2 ± 2.9	0.67 ± 0.12	8.4 ± 2.1	0.67 ± 0.09	45.7 ± 13.5	0.64 ± 0.11	52.7 ± 8.4	0.60 ± 0.11
	MCSL	9.2 ± 1.8	0.72 ± 0.06	8.9 ± 2.0	0.71 ± 0.08	76.1 ± 13.7	0.53 ± 0.09	78.1 ± 6.3	0.47 ± 0.07
	AS-FedDAG	3.4 ± 1.7	0.97 ± 0.04	2.7 ± 1.6	0.90 ± 0.07	35.9 ± 17.0	0.84 ± 0.09	41.8 ± 12.6	$\mathbf{0.73 \pm 0.07}$
	GS-FedDAG	$\mathbf{1.9 \pm 1.6}$	$\mathbf{0.99 \pm 0.02}$	$\mathbf{2.6 \pm 1.3}$	$\mathbf{0.93 \pm 0.07}$	$\mathbf{24.3 \pm 10.2}$	$\mathbf{0.86 \pm 0.09}$	$\mathbf{33.9 \pm 10.9}$	0.73 ± 0.09

across all settings. If taking all data together to train one model using other methods, we can see that data heterogeneity would put great trouble to all compared methods while GS-FedDAG plays pretty well. Here, we also provide the experimental results of AS-FedDAG on this setting. We can find that the model misspecification problem would lead to unsatisfactory results, which motivate us to design the GS-FedDAG. Moreover, GS-FedDAG shows consistently good results with different numbers of observations on each client (see Table 16). NOTEARS takes second place at the setting of 40 nodes because there are some linear data among clients, which is also the reason that GS-FedDAG shows lower SHDs on heterogeneous data in Table 3 than Table 2. Compared with non-linear models, NOTEARS easily fits well with even fewer linear data.

5.2 Real data

We consider a real public dataset named **fMRI Hippocampus** (Poldrack et al., 2015) to discover the underlying relationships among six brain regions. This dataset records signals from six separate brain regions in the resting state of one person in 84 successive days and the anatomical structure provide 7 edges as the ground truth graph (see Figure 10 in (Appendix D). Herein, we separately select 500 records in each of the 10 days, which can be regarded as different local data. It is worth noting that though this data does not have a real data privacy problem, we can use this dataset to evaluate the learning accuracy of our method. Here, in Table 4 we show part of the experimental results while others lie in Table 18). AS-FedDAG shows the best performance on all criteria while GS-FedDAG also performs better than most of the other methods.

Table 4: Empirical results on **fMRI Hippocampus** dataset (Part 1).

	All data			Separate data			GS-FedDAG	AS-FedDAG
	PC	NOTEARS	MCSL	PC	NOTEARS	MCSL		
SHD ↓	9.0 ± 0.0	*5.0 ± 0.0*	9.0 ± 0.6	8.7 ± 1.3	8.0 ± 1.9	8.3 ± 1.7	$\mathbf{6.4 \pm 0.9}$	$\mathbf{5.0 \pm 0.0}$
NNZ	11.0 ± 0.0	4.0 ± 0.0	12.0 ± 0.6	7.6 ± 1.3	5.4 ± 1.5	9.0 ± 1.7	6.8 ± 0.6	5.0 ± 0.0
TPR ↑	0.43 ± 0.00	0.29 ± 0.00	*0.44 ± 0.04*	0.26 ± 0.11	0.19 ± 0.18	$\mathbf{0.35 \pm 0.15}$	0.27 ± 0.12	$\mathbf{0.29 \pm 0.00}$
FDR ↓	0.73 ± 0.00	*0.50 ± 0.00*	0.74 ± 0.03	0.76 ± 0.10	0.78 ± 0.19	0.73 ± 0.11	$\mathbf{0.72 \pm 0.11}$	$\mathbf{0.60 \pm 0.00}$

6 Related work

Two mainstreams, named constraint-based and score-based methods, push the development of DAG structure learning. Constraint-based methods, including SGS and PC (Spirtes et al., 2001), take conditional independence constraints induced from the observed distribution to decide the graph skeleton and part of the directions. Another branch of methods (Chickering, 2002) define a score function, which evaluates the fitness between the distribution and graph, and identifies the graph $\mathcal{G}$ with the highest score after searching

the DAG space. To avoid solving the combinatorial optimization problem, NOTEARS (Zheng et al., 2018) introduces an equivalent acyclicity constraint and formulates a fully continuous optimization for searching the graph. Following this work, many works leverages this constraint to non-linear case (Ng et al., 2019; Zheng et al., 2020; Lachapelle et al., 2020; Zhu et al., 2020; Wang et al., 2021; Gao et al., 2021; Ng et al., 2022b), low-rank graph (Fang et al., 2020), interventional data (Brouillard et al., 2020; Ke et al., 2019; Scherrer et al., 2021), time-series data (Pamfil et al., 2020), incomplete data (Gao et al., 2022; Geffner et al., 2022) and unmeasured confounding (Bhattacharya et al., 2021). GOLEM (Ng et al., 2020) leverages the full likelihood and soft constraint to solve the optimization problem. Ng et al. (2022a), DAG-NoCurl (Yu et al., 2021) and NOFEARS (Wei et al., 2020a) focus on the optimization aspect.

The second line of related work is on the Overlapping Datasets (OD) (Danks et al., 2009; Tillman & Spirtes, 2011; Triantafillou & Tsamardinos, 2015; Huang et al., 2020a) problem in DAG structure learning. However, OD assumes that each dataset owns observations of partial variables and targets learning the integrated DAG from multiple datasets. In these works, data from different sites need to be collected on a central server.

The last line is on federated learning (Yang et al., 2019; Kairouz et al., 2021), which provides the joint training paradigm to learn from decentralized data while avoiding sharing raw data during the learning process. FedAvg (McMahan et al., 2017) first formulates and names federated learning. FedProx (Li et al., 2020) studies the heterogeneous case and provides the convergence analysis results. SCAFFOLD leverages variance reduction by correcting client-shift to enhance training efficiency. Besides these fundamental problems in FL itself, this novel learning way has been widely co-operated with or applied to many real-world tasks such as healthcare (Sheller et al., 2020), recommendation system (Yang et al., 2020), and smart transport (Samarakoon et al., 2019).

6.1 Concurrent work (NOTEARS-ADMM)

In NOTEARS-ADMM (Ng & Zhang, 2022), the authors also consider the same setting for discovering the underlying relations from distributed data owing to privacy and security concerns. The main advantage of our FedDAG over NOTEARS-ADMM is to handle heterogeneous data, which is very common in real applications. Then, NOTEARS-ADMM mainly considers the linear case, which shares the same learning object with our method. Instead of taking an average to share training information, ADMM is taken to make the adjacency matrix close. More detailed experimental comparisons can be found in Appendix D.6, from which we can see that our FedDAG performs better in most settings.

7 Conclusion and Discussions

Learning the underlying DAG structure from decentralized data brings considerable challenges to traditional DAG learning methods. In this context, we have introduced one of the first federated DAG structure learning methods called FedDAG, which uses a two-level structure for each local model. During the learning procedure, each client tries to learn an adjacency matrix to approximate the graph structure and neural networks to approximate the mechanisms. The matrix parts of some participating clients are aggregated and processed by the server and then broadcast to each client for updating its personalized matrix. The overall problem is formulated as a continuous optimization problem and solved by gradient descent. Structural identifiability conditions are provided, and extensive experiments on various data sets are to show the effectiveness of our FedDAG.

The first limitation of our framework is with the *no latent variable* assumption, which is seldom right in real scenarios. While, as a general framework, the advanced methods (Bhattacharya et al., 2021), which can handle the no observed confounder case, can be well incorporated with our method to deal with the federated setup (More details can be seen in Appendix C.3). Another limitation relies on privacy protection. As we said, we focus on the statistical and optimization perspectives of federated DAG structure learning and leave the problem of combining the advanced privacy protection methods (Wei et al., 2020b) into our framework as a future work. The last limitation is to loose the invariant DAG assumption and allow the causal graph change among different clients, which is more common in the real world.

References

George J Annas. Hipaa regulations: a new era of medical-record privacy? *New England Journal of Medicine*, 348:1486, 2003.

Bryon Aragam, Arash Amini, and Qing Zhou. Globally optimal score-based learning of directed acyclic graphs in high-dimensions. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Rohit Bhattacharya, Tushar Nagarajan, Daniel Malinsky, and Ilya Shpitser. Differentiable causal discovery under unmeasured confounding. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 2020.

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

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

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. *International Conference on Machine Learning*, 2021.

David Danks, Clark Glymour, and Robert Tillman. Integrating locally learned causal structures with overlapping variables. *Advances in Neural Information Processing Systems*, 2009.

Zhuangyan Fang, Shengyu Zhu, Jiji Zhang, Yue Liu, Zhitang Chen, and Yangbo He. Low rank directed acyclic graphs and causal structure learning. *arXiv preprint arXiv:2006.05691*, 2020.

Erdun Gao, Ignavier Ng, Mingming Gong, Li Shen, Wei Huang, Tongliang Liu, Kun Zhang, and Howard Bondell. MissDAG: Causal discovery in the presence of missing data with continuous additive noise models. *Advances in Neural Information Processing Systems*, 2022.

Yinghua Gao, Li Shen, and Shu-Tao Xia. Dag-gan: Causal structure learning with generative adversarial nets. In *International Conference on Acoustics, Speech and Signal Processing*, pp. 3320–3324, 2021.

Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, et al. Deep end-to-end causal inference. *arXiv preprint arXiv:2202.02195*, 2022.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10:524, 2019.

Sander Greenland, Judea Pearl, and James M Robins. Causal diagrams for epidemiologic research. *Epidemiology*, pp. 37–48, 1999.

Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

James J Heckman. Econometric causality. *International Statistical Review*, 76(1):1–27, 2008.

Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. *Advances in Neural Information Processing Systems*, 2008.

Biwei Huang, Kun Zhang, Yizhu Lin, Bernhard Schölkopf, and Clark Glymour. Generalized score functions for causal discovery. In *International Conference on Knowledge Discovery & Data Mining*, 2018.

Biwei Huang, Kun Zhang, Mingming Gong, and Clark Glymour. Causal discovery from multiple data sets with non-identical variable sets. *Association for the Advancement of Artificial Intelligence*, 34(06):10153–10161, 2020a.

Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *Journal of Machine Learning Research*, 21:1–53, 2020b.

Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*, 2017.

Finn V Jensen and Thomas Dyhre Nielsen. *Bayesian networks and decision graphs*, volume 2. Springer, 2007.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Now Foundations and Trends, 2021.

Marcus Kaiser and Maksim Sipos. Unsuitability of notears for causal graph discovery. *arXiv preprint arXiv:2104.05441*, 2021.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Bernhard Schölkopf, Michael C Mozer, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Neville K Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of bayesian network structure learning. *arXiv preprint arXiv:2109.11415*, 2021.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. In *International Conference on Learning Representations*, 2020.

Steffen Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Conference on Machine Learning and Systems*, 2020.

Bill Yuchen Lin, Chaoyang He, Zihang Ze, Hulin Wang, Yufen Hua, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. In *Findings of the Association for Computational Linguistics*, pp. 157–175, 2022.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning*, pp. 3122–3130. PMLR, 2018.

Rui Liu and Han Yu. Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256*, 2022.

Margaret Mooney Marini and Burton Singer. Causality in the social sciences. *Sociological methodology*, 18: 347–409, 1988.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial intelligence and statistics*. PMLR, 2017.

Joris M Mooij, Dominik Janzing, Tom Heskes, and Bernhard Schölkopf. On causal discovery with cyclic additive noise models. *Advances in neural information processing systems*, 24, 2011.

Joris M Mooij, Sara Magliacane, and Tom Claassen. Joint causal inference from multiple contexts. *Journal of Machine Learning Research*, 21:1–108, 2020.

Yongchan Na and Jihoon Yang. Distributed bayesian network structure learning. *IEEE International Symposium on Industrial Electronics*, pp. 1607–1611, 2010.

Arkadi Nemirovski. Optimization ii: Standard numerical methods for nonlinear continuous optimization. *Lecture Note*, 1999.

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

Ignavier Ng, Shengyu Zhu, Zhitang Chen, and Zhuangyan Fang. A graph autoencoder approach to causal structure learning. *arXiv preprint arXiv:1911.07420*, 2019.

Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

Ignavier Ng, Sébastien Lachapelle, Nan Rosemary Ke, Simon Lacoste-Julien, and Kun Zhang. On the convergence of continuous constrained optimization for structure learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 8176–8198. PMLR, 2022a.

Ignavier Ng, Shengyu Zhu, Zhuangyan Fang, Haoyang Li, Zhitang Chen, and Jun Wang. Masked gradient-based causal structure learning. In *SIAM International Conference on Data Mining*, pp. 424–432. SIAM, 2022b.

Nooshin Omranian, Jeanne MO Eloundou-Mbebi, Bernd Mueller-Roeber, and Zoran Nikoloski. Gene regulatory network inference using fused lasso on multiple data sets. *Scientific reports*, 6(1):1–14, 2016.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Judea Pearl. Bayesian netwcrks: A model cf self-activated memory for evidential reasoning. In *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pp. 15–17, 1985.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

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

Jonas Peters, Joris M Mooij, Dominik Janzing, and Bernhard Schölkopf. Identifiability of causal graphs using functional models. *Conference on Uncertainty in Artificial Intelligence*, 2011.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 947–1012, 2016.

Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pp. 17716–17758. PMLR, 2022.

Russell A Poldrack, Timothy O Laumann, Oluwasanmi Koyejo, Brenda Gregory, Ashleigh Hover, Mei-Yen Chen, Krzysztof J Gorgolewski, Jeffrey Luci, Sung Jun Joo, Ryan L Boyd, et al. Long-term neural and physiological phenotyping of a single human. *Nature Communications*, 6(1):1–15, 2015.

Amirhossein Reisizadeh, Farzan Farnia, Ramtin Pedarsani, and Ali Jadbabaie. Robust federated learning: The case of affine distribution shifts. *arXiv preprint arXiv:2006.08907*, 2020.

Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2): 1146–1159, 2019.

Nino Scherrer, Olexa Bilaniuk, Yashas Annadani, Anirudh Goyal, Patrick Schwab, Bernhard Schölkopf, Michael C Mozer, Yoshua Bengio, Stefan Bauer, and Nan Rosemary Ke. Learning neural causal models with active interventions. *arXiv preprint arXiv:2109.02429*, 2021.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, pp. 461–464, 1978.

Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1):1–12, 2020.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

Peter Spirtes, Clark Glymour, Richard Scheines, et al. *Causation, Prediction, and Search*, volume 1. The MIT Press, 2001.

Jin Tian and Judea Pearl. Causal discovery from changes. In *Uncertainty in Artificial Intelligence*, pp. 512–521, 2001.

Robert Tillman and Peter Spirtes. Learning equivalence classes of acyclic models with latent and selection variables from multiple datasets with overlapping variables. *International Conference on Artificial Intelligence and Statistics*, 2011.

Sofia Triantafillou and Ioannis Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *Journal of Machine Learning Research*, 16(1):2147–2205, 2015.

Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020a.

Xiaoqiang Wang, Yali Du, Shengyu Zhu, Liangjun Ke, Zhitang Chen, Jianye Hao, and Jun Wang. Ordering-based causal discovery with reinforcement learning. *International Joint Conference on Artificial Intelligence*, 2021.

Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. Causal inference for recommender systems. In *ACM Conference on Recommender Systems*, pp. 426–431, 2020b.

Dennis Wei, Tian Gao, and Yue Yu. Dags with no fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 2020a.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020b.

Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems*, 34:18839–18852, 2021.

Jing Yang, Ning An, Gil Alterovitz, Lian Li, and Aiguo Wang. Causal discovery based on healthcare information. In *International Conference on Bioinformatics and Biomedicine*, pp. 71–73. IEEE, 2013.

Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. Federated recommendation systems. *Federated Learning*, pp. 225–239, 2020.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007. PMLR, 2018.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. *International Conference on Machine Learning*, 2019.

Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. Dags with no curl: An efficient DAG structure learning approach. *International Conference on Machine Learning*, 2021.

Ming Yuan. On the identifiability of additive index models. *Statistica Sinica*, pp. 1901–1911, 2011.

Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: A python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021.

Kun Zhang and Aapo Hyvarinen. On the identifiability of the post-nonlinear causal model. *Conference on Uncertainty in Artificial Intelligence*, 2009.

Kun Zhang, Mingming Gong, Petar Stojanov, Biwei Huang, Qingsong Liu, and Clark Glymour. Domain adaptation as a problem of inference on graphical models. *Advances in Neural Information Processing Systems*, 2020.

Xun Zheng. *Learning DAGs with Continuous Optimization*. PhD thesis, Carnegie Mellon University, 2020.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*, 2020.

Appendix

Table of Contents

A Structure identifiability **22**

B Implementations **23**
B.1 Graph generation . . . 23
B.2 SEM simulation . . . 23
B.3 Detailed metrics . . . 23
B.4 Hyper-parameters setting . . . 24
B.5 Hyper-parameters in real-data setting . . . 24
B.6 Model parameters . . . 25
B.7 Training parameters . . . 25
B.8 Sensitivity analysis of hyper-parameters . . . 26

C Discussions on our method **26**
C.1 Novelty and contributions . . . 26
C.2 Difference with graph neural network (GNN) learning . . . 27
C.3 FedDAG as a framework . . . 27
C.4 Broader impact statement . . . 28
C.5 The consistency results by BIC score . . . 28
C.6 Does the global maximum of Eq. (6) correspond to the ground-truth DAG? . . . 28
C.7 Can Algorithms 1 and 2 solve Eq. (6)? . . . 28
C.8 Can U finally satisfy the acyclicity constraint in Eq. (3)? . . . 28

D Supplementary experimental details **29**
D.1 Uneven distributions . . . 29
D.2 Varying clients . . . 29
D.3 Dense graphs . . . 30
D.4 Comparisons with voting method . . . 30
D.5 Comparisons with CAM . . . 30
D.6 Comparisons with NOTEARS-ADMM . . . 31

E More discussions on the experimental results **31**

F Discussions on Assumptions **32**
F.1 Data heterogeneity . . . 32
F.2 Invariant DAG assumption . . . 32

A Structure identifiability

Besides exploring effective DAG structure learning methods, identifiability conditions of graph structure (Spirtes et al., 2001) are also important. In general, unique identification of the ground truth DAG is impossible from purely observational data without some specific assumptions. However, accompanying some specific data generation assumptions, the graph can be identified (Peters et al., 2011; Peters & Bühlmann, 2014; Zhang & Hyvarinen, 2009; Shimizu et al., 2006; Hoyer et al., 2008). We first give the definition of identifiability in the decentralized setting.

Definition A.1. Consider a decentralized distribution set $P^{\mathcal{C}}(X)$ satisfying Assumption 3.1. Then, $\mathcal{G}$ is said to be identifiable if $P^{\mathcal{C}}(X)$ cannot be induced from any other DAG.

Condition A.2. (Minimality condition) Given the joint distribution $P(X)$, $P(X)$ is Markovian to a DAG $\mathcal{G}$ but not Markovian to any sub-graph of $\mathcal{G}$.

Condition A.3. (Cond. 19 in (Peters & Bühlmann, 2014)) The triple $(f_j, P(X_i), P(\epsilon_j))$ does not solve the following differential equation $\forall x_i$, x_j with $v''(x_j - f(x_i))f'(x_i) \neq 0$:

$$\xi''' = \xi''\left(-\frac{\nu'''f'}{\nu''} + \frac{f''}{f'}\right) - 2\nu''f''f' + \nu'f''' + \frac{\nu'\nu'''f''f'}{\nu''} - \frac{\nu'\left(f''\right)^2}{f'}.$$

Here, $f := f_j$ and $\xi := \log P(X_i)$, and $v := \log P(\epsilon_j)$ are the logarithms of the strictly positive densities.

Definition A.4. (Restricted ANM. Def. 27 in (Peters & Bühlmann, 2014)) Consider an ANM with d variables. This SEM is called restricted ANM if $\forall j \in \mathbb{V}, i \in \mathbf{PA}_j$ and all sets $\mathbb{S} \subseteq \mathbb{V}$ with $\mathbf{PA}_j \backslash \{i\} \subseteq \mathbb{S} \subseteq \mathbf{PA}_j \backslash \{i, j\}$, there is an $x_{\mathbb{S}}$ with $P(x_{\mathbb{S}}) > 0$, s.t. the tripe

$$\left(f_j(x_{\mathbf{PA}_j \backslash \{i\}}, \underbrace{\cdot}_{X_i}), P\left(X_i \mid X_{\mathbb{S}} = x_{\mathbb{S}}\right), P\left(\epsilon_j\right)\right)$$

satisfies ConditionA.3. Here, the under-brace indicates the input component of f_j for variable X_i. In particular, we require the noise variables to have non-vanishing densities and the functions f_j to be continuous and three times continuously differentiable.

Assumption A.5. (Faithfulness) Let $P^{\mathcal{C}}(X)$ satisfy Assumption 3.1. At least one distribution $P^{c_k}(X) \in P^{\mathcal{C}}(X)$ meets Assumption A.6 and the other distributions are faithful to $\mathcal{G}$.

Assumption A.6. Let a distribution $P(X)$ with $X = (X_1, X_2, \cdots, X_d)$ be induced from a restricted ANM A.3 with graph $\mathcal{G}$, and $P(X)$ satisfies *Minimality condition* w.r.t $\mathcal{G}$.

Proposition A.7. *Given $P^{\mathcal{C}}(X)$ satisfying Assumption A.5, and then, $\mathcal{G}$ can be identified up from $P^{\mathcal{C}}(X)$.*

Proof. From Remark 3.2, we have $P^{c_k}(X) \in P^{\mathcal{C}}(X)$ for $\forall c_k$, is Markov with $\mathcal{G}$. For each $c_k \in \mathcal{C}$ with $P^{c_k}(X)$ does not satisfy Assumption A.6, the Completed Partially DAG (CPDAG) $\hat{\mathcal{G}}$ (Pearl, 2009), which represents the CPDAG induced by $\mathcal{G}$, can be identified (Spirtes et al., 2001). (1) That also says that these distributions can be induced from any DAG induced from $\mathcal{M}(\mathcal{G})$, including $\mathcal{G}$ definitely. Notice that skeleton$(\hat{\mathcal{G}})$ = Skeleton$(\mathcal{G})$ and any $X_i \leftarrow X_j$ in $\hat{\mathcal{G}}$ is also existed in $\mathcal{G}$. Then, for those c_k with with $P^{c_k}(X)$ satisfying Assumption A.6, $\mathcal{G}$ can be identified. (2) That is to say, distributions satisfying Assumption A.6 can only be induced from $\mathcal{G}$. Then, two kinds of graph, $\hat{\mathcal{G}}$ and $\mathcal{G}$, are obtained. Therefore, $\mathcal{G}$ can be easily identified. With (1) and (2), $P^{c_k}(X) \in P^{\mathcal{C}}(X)$ for $\forall c_k$ can only be induced by $\mathcal{G}$. Then, $\mathcal{G}$ is said to be identifiable ■

Future work is to relax our invariant DAG assumption to the invariant CPDAG assumption, which means that a group of DAGs across different clients share the same conditional independence. For this case, the generalized score functions (Huang et al., 2018) can be adopted to search for the Markov Equivalence Class. However, it is not straightforward to incorporate this method into our FedDAG framework since the score of this method is motivated by a kernel-based (conditional) independence test rather than penalized likelihood. Moreover, this method does not support a continuous search strategy. It would be interesting to explore the penalized likelihood-based method for this case and incorporate it into our framework.

B Implementations

The comparing DAG structure learning methods used in this paper all have available implementations, listed below:

- MCSL: Codes are available at `gCastle` https://github.com/huawei-noah/trustworthyAI/tree/master/gcastle. The first author of MCSL added the implementation in this package.
- NOTEARS and NOTEARS-MLP: Codes are available at the first author's GitHub repository https://github.com/xunzheng/notears
- NOTEARS-ADMM: Codes are available at the first author's GitHub repository https://github.com/ignavierng/notears-admm
- DAG-GNN: Codes are available at the author's GitHub repository https://github.com/fishmoon1234/DAG-GNN
- PC and GES: the implementations of PC and GES are available at `causal-learn` package repository https://causal-learn.readthedocs.io/en/latest/index.html
- CAM: the codes are available at `CRAN R` package repository https://cran.r-project.org/src/contrib/Archive/CAM/

Our implementation is highly based on the existing Tool-chain named `gCastle` (Zhang et al., 2021), which includes many gradient-based DAG structure learning methods.

B.1 Graph generation

To simulate DAG for generating observations, we introduce two kinds of graph generation methods named Erdős-Rényi (ER) and Scale-Free (SF) graphs. To simulate the ER graph generation, we firstly randomly sample a topological order and by adding directed edges where it is allowed independently with probability $p = \frac{2s}{d^2-d}$ where s is the number of edges in the resulting DAG. To generate Scale-free (SF) graphs, we first take the Barabasi-Albert model and then add all nodes one by one. From the above descriptions, we can find that the degree distribution of ER graphs follows a Poisson distribution, and the degree of SF graphs follows a power law: few nodes, often called hubs, have a high degree (Lachapelle et al., 2020).

B.2 SEM simulation

In our experimental setup, for each client, we randomly choose a nonlinear type from the given four functions with equal probability in the heterogeneous data setting. The nonlinear function choice is totally the same as used in NOTEARS-MLP (Zheng et al., 2020). The details are as follows.

We simulate the SEM $X_j = f_j(X_{pa_j}) + Z_j$ for all $j \in [d]$ in the topological order induce by G.

GP: f_j is drawn from the Gaussian process with RBF kernel with length-scale one.

GP-add: $f_j(X_{pa_j}) = \sum_{k \in pa_j} f_{jk}(X_k)$, where each f_{jk} is from GP.

MLP: f_j is randomly initialized MLP with one hidden layer of size 100 and Sigmoid activation.

MiM: also named as index model. $f_j(X_{pa_j}) = \sum_{m=1}^{3} h_m(\sum_{k \in pa_j} \theta_{jmk} X_k)$, where $h_1 = \tanh$, $h_2 = \cos$, $h_3 = \sin$, and each θ_{jmk} is uniformly drawn from range $[-2, -0.5] \cup [0.5, 2]$.

B.3 Detailed metrics

SHD is a kind of measurement which is defined to calculate the Hamming distance between two partially directed acyclic graphs (PDAG) by counting the number of edges for which the edge type differs in both PDAGs. In PDAG, there exist four kinds of edges between two nodes: $i \to j$, $i \leftarrow j$, $i - j$, and $i\ j$. SHD just

counts the different edges between the two graphs. SHD is defined over the space of PDAGs, so we can, of course, use it to calculate distances in DAG and CPDAG spaces.

True Positive Rate (TPR) and False Discovery Rate (FDR) are two common metrics in the machine learning community. True positive rate, also referred to as sensitivity or recall, is used to measure the percentage of actual positives which are correctly identified. The FDR is defined as the expected proportion of errors committed by falsely rejecting the null hypothesis. Let TP be true positives (samples correctly classified as positive), FN be false negatives (samples incorrectly classified as negative), FP be false positives (samples incorrectly classified as positive), and TN be true negatives (samples correctly classified as negative). Then, $TPR = \frac{TP}{TP+FN}$ and $FDR = \frac{FP}{FP+TP}$.

B.4 Hyper-parameters setting

In all experiments, there is no extra hyper-parameter to adjust for PC (with Fisher-z test and p-value 0.01) and GES (BIC score). For NOTEARS, NOTEARS-MLP, and DAG-GNN, we use the default hyper-parameters provided in their papers/codes. For MCSL, the hyper-parameters that need to be modified are ρ_{init} and β. Specifically, if experimental settings (10 variables and 20 variables) are the same as those in their paper, we just take all the recommended hyper-parameters. For settings not implemented in their paper (40 variables exactly), we have two kinds of implementations. The first one is taking a linear interpolation for choosing the hyper-parameters. The second one is taking the same parameters as ours. We find that the second choice always works better. In our experiment, we report the experimental results in a second way. Notice that CAM pruning is also introduced to improve the performance of MCSL, which however can not guarantee a better result in our settings. For simplicity and fairness, we just take the direct outputs of MCSL.

Similar to MCSL (Ng et al., 2022b) and GraN-DAG (Lachapelle et al., 2020), we implement several experiments on simulated data with known graph structure to search for the hyper-parameters and then use these hyper-parameters for all the simulated experiments. Specifically, we use seeds from 1 to 10 to generate the simulated data to search for the best combination of hyper-parameters while all our experimental results reported in this paper are all conducted using seeds from 2021 to 2030.

B.5 Hyper-parameters in real-data setting

Most DAG learning methods have hyper-parameters, more or less, which need to be decided prior to learning. Moreover, NN-based methods are especially sensitive to the selection of hyper-parameters. For instance, Gran-DAG (Lachapelle et al., 2020) defines a really large hyper-parameters space for searching the optimal combination, which even uses different learning rates for the first sub-problem and the other sub-problems. MCSL and GS-FedDAG are sensitive to the selection of ρ_{init} and β when constructing and solving the sub-problem. As pointed out in (Kairouz et al., 2021), NOTEARS focuses more on optimizing the scoring term in the early stage and pays more attention to approximate DAG in the late stage. If NOTEARS cannot find a graph near $\mathcal{G}$ in the early stage, then, it would lead to a worse result.

To alleviate this problem, one may choose to (1) enlarge the learning rate or take more steps when solving the first few sub-problems as Gran-DAG; (2) reduce the value of coefficient ρ_{init} to let the optimizer pay more attention to the scoring term in the early stages as MCSL. The other trick we find when dealing with real data is increasing ℓ_1. This mostly results from that real data may not fit well with the data generation assumptions in most papers. Therefore, we choose to conduct a grid search to find the best combination of $\rho_{init}, \beta, \ell_1$ for DAG structure learning on real data.

In the practice of DAG structure learning, it is impossible to have $\mathcal{G}$ to select the hyper-parameters. One common approach is trying multiple hyper-parameter combinations and keeping the one yielding the best score evaluated on a validation set (Koller & Friedman, 2009; Ng et al., 2022b; Lachapelle et al., 2020). However, the direct use of this method may not work for some algorithms, such as MCSL, NOTEARS-MLP, and GS-FedDAG. This mainly lies in the similar explanations of the property of the traditional solution of FL. In the late stage of optimization, the optimizer focuses heavily on finding *a DAG* by enlarging the penalty coefficient ρ. Then, the learning of relationship mechanisms would be nearly ignored. To address this problem, we first report the DAG directly learned by a combination of hyper-parameters. And then,

we replace the parameters part for describing the graph with the learned DAG. Afterward, we just take the score without DAG constraint to optimize the relationship mechanisms approximating part (which may not be the same name in the other algorithms). Finally, the validation set is taken to evaluate the learned model. The final hyper-parameters used on the real dataset in our paper are as follows:

Table 5: The hyper-parameters used on real data.

Parameters	ρ_{init}	β	λ_{ℓ_1}
Values	0.008	2	0.3

B.6 Model parameters

The GSL part in each local model is parameterized by a $d \times d$ matrix named $\boldsymbol{U}$ and the Gumbel-Sigmoid approach is leveraged for approximating the binary form. Each entry in $\boldsymbol{U}$ is initialized as 0. The temperature τ is set to 0.2 for all settings. Then, for the relationship mechanism approximating part, we use 4 dense layers with 16 variables in each hidden layer. All weights in the Network are initialized using the Xavier uniform initialization. The number of parameters used in each method is shown in Table 6.

Table 6: Model parameters for each client with different nodes.

	10 nodes	20 nodes	40 nodes
NOTEARS	100	400	1600
Mask of MCSL	100	400	1600
NN of MCSL	9930	19860	39720

B.7 Training parameters

Our GS-FedDAG and AS-FedDAG reach this point and are implemented with the following hyper-parameters. We take ADAM (Kingma & Ba, 2015) with learning rate 3×10^{-2} and all the observational data $\mathcal{D}^{c_k}$ on each client are used for computing the gradient. And the detailed parameters used in Algorithms 1 and 2 are listed in Table 7.

Table 7: The hyper-parameters used on simulated data in this paper.

Parameters	α_{init}	h_{tol}	it_{max}	it_{inner}	it_{fl}	γ	ρ_{max}	λ_{ℓ_1}
Values	0	1×10^{-10}	25	1000	200	0.25	1×10^{14}	0.01

Notice that as illustrated in MCSL (Ng et al., 2022b), the performance of the algorithm is affected by the initial value of ρ_{init} and the choice of β. Since a small initial of ρ_{init} and β would result in a rather long training time. As said in (Kaiser & Sipos, 2021), MLE plays an important role in the early stage of training and highly affects the final results. Therefore, carefully picking a proper combination of ρ_{init} and β will lead to a better result. In our method, we tune these two parameters via the same scale of experiment with seeds $1 \sim 10$. For each variable scale and training type, the parameters are adjusted once and are applied to all other experiments with the same variable scale. We find the combinations of the following parameters in Table 8 work well in our method. Our method also adopts a ℓ_1 sparsity term on $g_\tau(\boldsymbol{U})$, where the sparsity coefficient λ_{ℓ_1} is chosen as 0.01 for all settings.

Table 8: The combinations of ρ_{init} and β on simulated data in our method.

	10 nodes		20 nodes		40 nodes	
	ρ_{init}	β	ρ_{init}	β	ρ_{init}	β
AS-FedDAG	6×10^{-3}	10	1×10^{-5}	20	1×10^{-11}	120
GS-FedDAG	6×10^{-3}	10	6×10^{-5}	20	1×10^{-11}	120

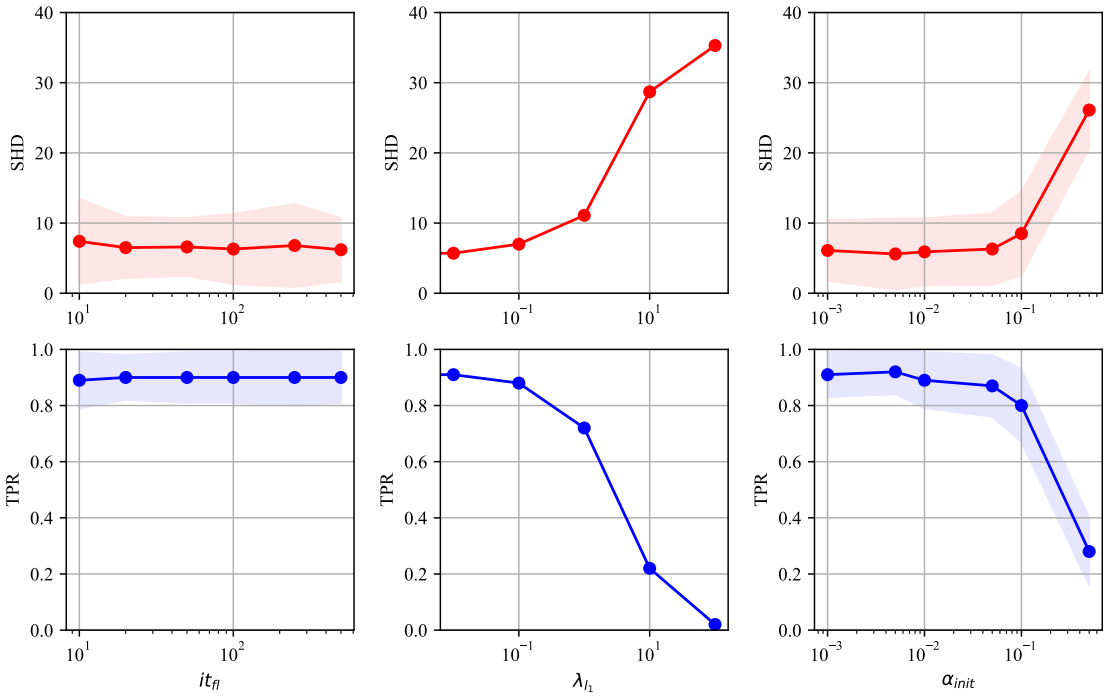


Figure 5: The sensitivity analysis of hyper-parameters

B.8 Sensitivity analysis of hyper-parameters

Here, we show the sensitivity analysis of it_{fl}, α_{init}, and λ_{l_1}. From the experimental results in Figure 5, we find that our method is relatively robust to it_{fl}. That is to say, the it_{fl} can be reduced to alleviate the pressure of communication costs while the performance can be well kept. λ_{l_1} is the coefficient of l_1 sparsity, which will affect the final results. Because we have no sparsity information of the underlying graph, we set $\lambda_{l_1} = 0.01$ in all settings. When dealing with real data, we recommend the audiences adjust this parameter by using our parameter-tuning method provided in Section B.5. The results of α_{init} are exactly as expected. As discussed before, our method tries to maximize the likelihood term of the total loss in the early stages, which is important to find the final ground-truth DAG. If setting a relatively large α_{init}, the early learning stages would be affected. Therefore, we recommend directly taking α_{init} as 0 in all settings.

C Discussions on our method

C.1 Novelty and contributions

Firstly, we acknowledge the contribution of our baseline method MCSL (Ng et al., 2022b), which performs well in many settings and helps to guarantee the performance of our proposed method. We also appreciate the excellent baseline method FedAvg (McMahan et al., 2017), which provides an efficient federated learning way. Our FedDAG is highly inspired and benefits from these two works. The main contributions, which can

be taken by our proposed method, are (1) **one of the first works** that investigate the practical problem of DAG structure learning in a federated setup and (2) further providing the FedDAG approach that can guarantee the **privacy protection** by avoiding the raw data leakage and allow the **data heterogeneity** across the clients. Another concurrent work NOTEARS-ADMM (Ng & Zhang, 2022) also considers the same problem while our GS-FedDAG can (1) gain better performances in most of the settings, (2) well handle the nonlinear cases, (3) allow heterogeneous data, and (4) provide a quite flexible federated DAG structure learning framework.

Discussions on the simple averaging Even though averaging is the simplest way to aggregating and exchanging information, we find it is **quite an effective** way to solve the federated DAG structure learning problem, which is an advantage of our method. Our simple averaging for homogeneous cases can nearly approach the same performance as using all data. For the heterogeneous cases, GS-FedDAG can still obtain satisfactory results. While, as future work, more advanced information aggregation methods (Wang et al., 2020a) can be well incorporated into our framework to boost the performances further.

C.2 Difference with graph neural network (GNN) learning

Four main reasons make DAG structure learning and GNN two different research lines. (1) Nodes in DAG represent variables, and directed edges describe the single-direction relation between different variables. In GNN, graph talks more about graph-type data, such as social networks, protein networks, and traffic networks. (2) Networks in DAG structure learning are leveraged to learn the relationship mechanisms, while networks in GNN are taken to achieve node embedding and feature extraction. (3) Learned DAG can be taken for interventional and counterfactual reasoning (Kitson et al., 2021). (4) DAG structure learning cares more about identifiability. It is essential to identify the true underlying relationship of the observations precisely. In the federated setup, most existing federated GNN (Xie et al., 2021; Liu & Yu, 2022) methods assume that the underlying graphs are known and localized. What is being learned in the federation is the weight aggregation of the GNN but not its graph. This also leads to the main difference between our federated DAG learning and federated GNN learning.

C.3 FedDAG as a framework

In this paper, we restrict our attention to the case that all concerned variables can be well observed. We also only take MCSL (Ng et al., 2022b) as the baseline method. However, all gradient-based methods can be incorporated into our AS-FedDAG framework to deal with homogeneous data. To deal with the heterogeneous data, we prefer that the baseline methods can separately learn the DAG structure and relationship mechanisms. The other baseline methods that can be easily combined into our framework are NOTEARS-MLP (Zheng et al., 2020) and DAG-GNN (Yu et al., 2019). Unfortunately, many works are not in this fashion, such as GraN-DAG (Lachapelle et al., 2020), CD-RL (Zhu et al., 2020), and their following works.

Latent variables. In this paper, we carry with no unobserved common confounder assumption. Handling latent confounders is a fundamentally important but challenging problem in the traditional DAG structure learning, not to mention the federated setup. Until now, the theoretical results on the structure identifiability of DAG learning with latent confounder are always too weak to be used in practice since too strict assumptions are taken. In the recent progress of the latent variables research, Bhattacharya et al. (2021) takes the acyclic directed mixed graphs (ADMGs) to describe the graphs with latent confounders. With different types of restrictions, three classes of proprieties, named Ancestral graph, Acid graph, and Bow-free graph, are given. According to different proprieties, different graph constraints are given. For example, trace $\left(e^{D}\right)-d+\operatorname{sum}(D \circ B)=0$ is set for the Bow-free graph[9], where D is the adjacency matrix recording the directed edges and B records the double-directed edges. We can directly replace the constraint to incorporate this method in our framework. However, this method can only deal with the linear Gaussian case, which is somewhat limited.

[9]See more details at Section 4 in (Bhattacharya et al., 2021)

C.4 Broader impact statement

In federated learning, the server and some clients participate in this process. While as we talked about above, the DAG is shared among all clients. FedDAG is motivated by the "data on each client is not enough for identifying up the ground-truth DAG." The graph information is not private for clients. For the server, it depends. In our previous motivations, we only cared about the "raw data leakage" problem but did not consider the privacy of the graph. Some relations can be public in real-world scenarios, such as disease research. For these cases, our method can still work. However, graph structure may sometimes also be private information. This problem can be easily solved by picking one client as the proxy server.

C.5 The consistency results by BIC score

Actually, for linear additive noise models with Gaussian noises, the consistency results for maximizing the BIC score to identify the DAG (Markov Equivalence Class or DAG) have been well established (Tian & Pearl, 2001; Huang et al., 2020a). For this case, with the DAG space constraint, the unique maximum of score function $\mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U}^{c_k})$ with BIC score corresponds to the ground-truth DAG. Even for the high-dimensional consistency for linear Gaussian SEM when the model is identifiable (Aragam et al., 2019). Since the ground-truth $\mathcal{G}$ corresponds to each $\mathcal{S}^{c_k}$, the global maximum $\arg\max_{\boldsymbol{\Phi},\boldsymbol{U}} \sum_{k=1}^{m} \mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U})$ with DAG constraint can lead to the ground-truth DAG graph. For nonlinear ANMs, however, even many practical methods, e.g., MCSL (Ng et al., 2022b), NOTEARS-MLP (Zheng et al., 2020), and CD-RL (Zhu et al., 2020), have been proposed to solve this problem by maximizing the BIC score, the theoretical results of consistency are still lacking and would be an interesting future work to be investigated. Therefore, our framework based on these methods inherits the theoretical limit for the nonlinear case. From our paper, however, empirical results can still show the method's effectiveness.

C.6 Does the global maximum of Eq. (6) correspond to the ground-truth DAG?

Firstly, for observations of identifiable ANMs on each client, the unique maximum of score function $\mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U}^{c_k})$ with BIC score corresponds to the ground-truth DAG (Zheng et al., 2018; Ng et al., 2022b). Even for the high-dimensional consistency of linear Gaussian SEM in the case when the model is identifiable. Since the ground-truth $\mathcal{G}$ corresponds to each $\mathcal{S}^{c_k}$, the global maximum $\arg\max_{\boldsymbol{\Phi},\boldsymbol{U}} \sum_{k=1}^{m} \mathcal{S}^{c_k}(\mathcal{D}^{c_k}, \boldsymbol{\Phi}^{c_k}, \boldsymbol{U}^{c_k})$ with DAG constraint can lead to the ground-truth DAG.

C.7 Can Algorithms 1 and 2 solve Eq. (6)?

Unfortunately, the global maximum of Eq. (6) can not be well reached by the gradient-based optimization methods, which is mainly caused by the non-convex property of the acyclicity constraint. Firstly, discovering the ground-truth DAG is an NP-hard problem. Traditional methods like PC and GES search the discrete DAG space to solve this problem, which is relatively time-consuming. Then, NOTEARS introduces an equality constraint (3) to formulate the DAG search problem as a continuous optimization problem, which can be easily solved by the gradient descent methods. However, the trade-off is that this equality constraint is non-convex, which pushes us away from finding the ground-truth DAG (the global minima of (6)). That is to say, using gradient descent to solve (6) only can reach the local minima of (6). This similar conclusion stands for recent continuous optimization-based CD methods such as GraNDAG (Lachapelle et al., 2020), DAG-GNN (Yu et al., 2019), and NOTEARS-MLP (Zheng et al., 2020).

C.8 Can U finally satisfy the acyclicity constraint in Eq. (3)?

For simplicity, please do not mind if we explain our method by setting some parameters with specific values. Firstly, following NOTEARS (Zheng et al., 2018) and MCSL (Ng et al., 2022b), we take the Augmented Lagrangian Method (ALM) to covert the constrained optimization problem into a series of sub-problems without the hard constraint but with two penalty terms. For the t-th sub-problem, the specific formulation of Eq. (8) is related to α_t and ρ_t. α_t and ρ_t will be updated to α_{t+1} and ρ_{t+1} after solving the t-th sub-problem for 1000 steps (gradient descent step). When dealing with each sub-problem, each client locally updates its personalized model *with acyclicity penalty terms*, which is indeed for the acyclicity constraint. During

the 1000 steps, $\boldsymbol{U}$s are averaged every 200 steps (Yes, the simple average is nothing with acyclicity). When finishing 1000 steps (also *the* 5*-th* 200 *steps* is just finished), a new $\boldsymbol{U}^{new}$ is obtained. Then, α_t and ρ_t are updated to α_{t+1} and ρ_{t+1} to formulate the next sub-problem of ALM, which are described in steps $5 \sim 9$ in Algorithm 1. Then, a new circulation begins. Therefore, we argue that (1) *the acyclicity constraint is guaranteed by taking the acyclicity penalty when solving each sub-problem.* (2) *the convergence of* $\boldsymbol{U}$ *is supported by the convergence analysis of Personalized FedAvg of heterogeneous data.*

D Supplementary experimental details

D.1 Uneven distributions

For federated learning problems in the real world, different clients may own different amounts of observations. To verify the stability of our method, we simulate the setting of uneven distributions in different clients. For each client, the number of observations is randomly chosen from a list $[20\%, 40\%, 60\%, 80\%] \times n$, where n is the maximal observation. The experimental results are shown in Fig. 6, from which we can find that our method show relatively stable performance in this setting.

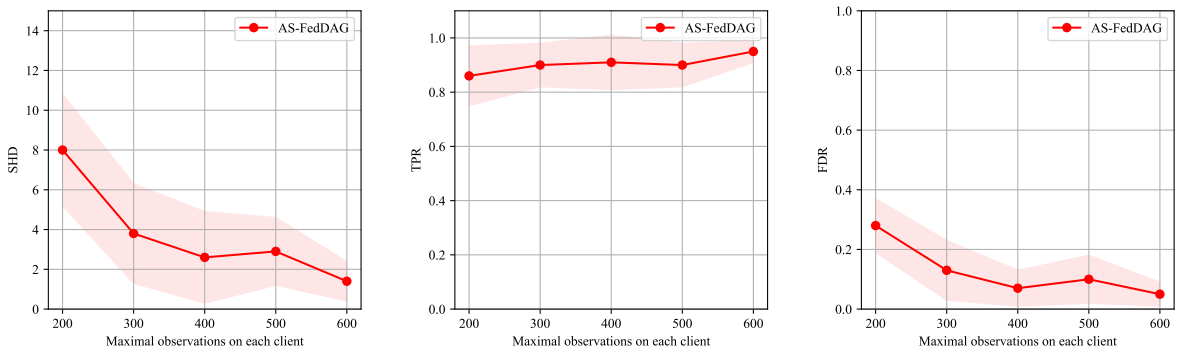


Figure 6: Results of uneven distributions on different clients.

D.2 Varying clients

In this setting, we now consider a fixed number of samples distributed across different clients. We conduct experiments for $(2, 4, 6, 8)$ clients and show the results in Fig. 7. With the increase in clients number, our method can show better performance.

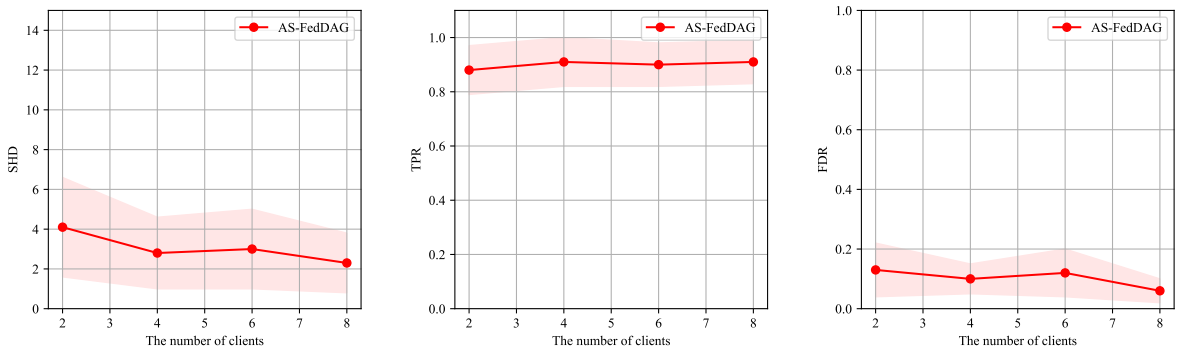


Figure 7: Results of performances with varying clients.

D.3 Dense graphs

Our method is also implemented on some denser graphs. Experimental results in Table 9 and Table 10. From these experimental results, we can see that our method shows consistently better performance over other methods on the denser graph setting. For the homogeneous case, both AS-FedDAG and GS-FedDAG obtain the nearly low SHD as MCSL trained on all data and far better than all methods trained on separated data. For the heterogeneous case, our GS-FedDAG still shows the best performance. Compared to NOTEARS in 20 variables case, GS-FedDAG shows similar SHD results but a much better TPR result. Therefore, how to reduce the false discovery rate of GS-FedDAG would be an exciting thing.

Table 9: Results on nonlinear ANM with dense graphs (Homogeneous data).

		ER4 with 10 nodes		SF4 with 10 nodes		ER4 with 20 nodes		SF4 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	27.3 ± 3.2	0.29 ± 0.07	18.9 ± 4.9	0.37 ± 0.16	68.2 ± 9.5	0.23 ± 0.06	60.2 ± 9.3	0.30 ± 0.08
	NOTEARS	34.3 ± 1.7	0.03 ± 0.02	22.7 ± 1.3	0.05 ± 0.05	71.8 ± 7.2	0.03 ± 0.01	62.8 ± 0.9	0.02 ± 0.01
	MCSL	*15.5 ± 5.9*	0.57 ± 0.15	*4.5 ± 3.1*	*0.83 ± 0.11*	*33.8 ± 10.4*	0.55 ± 0.11	*19.8 ± 7.5*	0.69 ± 0.11
Sep data	PC	31.5 ± 2.1	0.14 ± 0.03	20.4 ± 0.58	0.21 ± 0.03	68.7 ± 8.1	0.13 ± 0.03	60.9 ± 2.8	0.15 ± 0.02
	NOTEARS	34.3 ± 1.8	0.03 ± 0.01	22.7 ± 1.0	0.06 ± 0.04	70.1 ± 6.9	0.03 ± 0.01	62.3 ± 0.56	0.03 ± 0.01
	MCSL	$\mathbf{15.8 \pm 3.3}$	$\mathbf{0.61 \pm 0.09}$	8.3 ± 4.3	$\mathbf{0.78 \pm 0.11}$	49.3 ± 11.8	$\mathbf{0.63 \pm 0.10}$	39.7 ± 5.6	$\mathbf{0.73 \pm 0.07}$
	GS-FedDAG	$\mathbf{16.9 \pm 4.9}$	$\mathbf{0.53 \pm 0.12}$	$\mathbf{5.4 \pm 3.0}$	0.78 ± 0.12	$\mathbf{35.4 \pm 10.9}$	0.53 ± 0.11	$\mathbf{20.7 \pm 5.1}$	0.69 ± 0.08
	AS-FedDAG	17.4 ± 4.8	$\mathbf{0.53 \pm 0.12}$	$\mathbf{5.5 \pm 2.8}$	$\mathbf{0.79 \pm 0.11}$	$\mathbf{40.7 \pm 4.8}$	$\mathbf{0.57 \pm 0.10}$	$\mathbf{24.1 \pm 5.8}$	$\mathbf{0.71 \pm 0.09}$

Table 10: Results on nonlinear ANM with dense graphs (Heterogeneous data).

		ER4 with 10 nodes		SF4 with 10 nodes		ER4 with 20 nodes		SF4 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
Sep data	PC	29.3 ± 1.3	0.23 ± 0.03	20.3 ± 2.1	0.31 ± 0.06	71.9 ± 8.1	0.19 ± 0.03	62.7 ± 2.8	0.22 ± 0.03
	NOTEARS	20.5 ± 2.6	0.45 ± 0.08	12.2 ± 2.9	0.54 ± 0.11	43.2 ± 7.0	0.49 ± 0.08	$\mathbf{39.4 \pm 6.8}$	0.47 ± 0.10
	MCSL	20.0 ± 3.2	0.52 ± 0.07	13.7 ± 2.2	0.65 ± 0.07	65.1 ± 7.7	0.33 ± 0.05	59.4 ± 5.3	0.31 ± 0.05
	GS-FedDAG	$\mathbf{8.5 \pm 3.7}$	$\mathbf{0.84 \pm 0.09}$	4.5 ± 2.0	$\mathbf{0.93 \pm 0.07}$	$\mathbf{40.7 \pm 14.5}$	$\mathbf{0.74 \pm 0.07}$	39.9 ± 10.8	$\mathbf{0.68 \pm 0.07}$

D.4 Comparisons with voting method

There is another interesting research line (Na & Yang, 2010), which also tries to learn DAG from decentralized data. We add a DAG combination method proposed in (Na & Yang, 2010), which proposes to vote for each entry of the adjacency matrix to get the final DAG. From the experimental results in Table 11, we can find that For PC and NOTEARS, the combining method seems to contribute little improvement. This is because the reported DAGs local clients are too bad to get a good result. For MCSL, this combing method works well for improving performance. The reason is easy to be inferred from the results. For MCSL, DAGs reported by local clients are of bad SHDs but good TPR, which means that the False Discovery Rates (FDRs) are high. In contrast, the combing method can further reduce the FDRs and keep the TPRs still good. Then, SHD can be further reduced. Luckily, our GS-FedDAG still shows the best performances in all settings.

D.5 Comparisons with CAM

Here, we add one more identifiable baseline named causal additive model (CAM) (Bühlmann et al., 2014), which also serves as a baseline in MCSL (Ng et al., 2022b), GraNDAG (Lachapelle et al., 2020), and DAG-GAN (Yu et al., 2019). From results in Table 12 and 13, we can see that our methods always show an advantage over CAM. CAM also assumes a non-linear ANM for data generation. However, CAM limits the non-linear function to be additive. In normal ANM, $X_i = f_i(X_{pa_i}) + \epsilon_i$ while CAM assumes $X_i = \sum_{j \in X(pa_i)} f_{i \leftarrow j}(X_j) + \epsilon_i$, which limits the capacity of its model. From the above experimental results, we can see that our methods show consistent advantages over CAM.

Table 11: Comparison with the voting method.

		Homogeneous data (GP)				Heterogeneous data			
		ER2 with 10 nodes		ER2 with 20 nodes		ER2 with 10 nodes		ER2 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
Sep data	PC	14.1 ± 2.4	0.31 ± 0.06	32.7 ± 6.5	0.28 ± 0.07	12.5 ± 2.7	0.45 ± 0.07	28.5 ± 6.3	0.44 ± 0.07
	NOTEARS	16.5 ± 2.0	0.06 ± 0.04	31.7 ± 6.0	0.11 ± 0.04	7.6 ± 2.6	0.60 ± 0.11	15.0 ± 3.1	0.62 ± 0.09
	MCSL	7.1 ± 3.2	0.83 ± 0.08	24.8 ± 5.5	*0.88 ± 0.07*	9.2 ± 1.8	0.72 ± 0.06	23.3 ± 5.8	0.56 ± 0.08
Voting	PC	13.3 ± 3.0	0.27 ± 0.11	29.7 ± 5.9	0.22 ± 0.05	11.4 ± 3.4	0.36 ± 0.13	25.5 ± 6.8	0.29 ± 0.13
	NOTEARS	15.6 ± 2.2	0.11 ± 0.06	32.6 ± 6.2	0.09 ± 0.05	7.8 ± 4.0	0.56 ± 0.20	18.4 ± 11.6	0.49 ± 0.30
	MCSL	8.0 ± 3.1	0.85 ± 0.16	18.1 ± 7.8	$\mathbf{0.88 \pm 0.06}$	6.9 ± 2.2	0.71 ± 0.13	10.1 ± 4.6	0.79 ± 0.09
	GS-FedDAG	$\mathbf{2.4 \pm 2.0}$	$\mathbf{0.86 \pm 0.12}$	$\mathbf{6.2 \pm 4.0}$	0.85 ± 0.10	$\mathbf{1.9 \pm 1.6}$	$\mathbf{0.99 \pm 0.02}$	$\mathbf{6.2 \pm 4.7}$	$\mathbf{0.89 \pm 0.09}$
	AS-FedDAG	$\mathbf{1.8 \pm 2.0}$	$\mathbf{0.89 \pm 0.12}$	$\mathbf{5.0 \pm 4.2}$	$\mathbf{0.88 \pm 0.11}$	NaN	NaN	NaN	NaN

Table 12: Comparisons with CAM on nonlinear ANM (Homogeneous data-GP).

		ER2 with 10 nodes		SF2 with 10 nodes		ER2 with 20 nodes		SF2 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	CAM	9.5 ± 2.9	0.87 ± 0.09	9.1 ± 3.1	0.84 ± 0.10	21.4 ± 4.7	$0.77, \pm 0.08$	26.6 ± 6.1	0.75 ± 0.07
Sep data	CAM	11.8 ± 2.6	0.40 ± 0.10	11.1 ± 1.5	0.38 ± 0.11	24.3 ± 5.8	0.40 ± 0.07	26.8 ± 2.0	0.36 ± 0.06
	GS-FedDAG	2.4 ± 2.0	0.86 ± 0.12	2.7 ± 2.2	$\mathbf{0.86 \pm 0.13}$	6.2 ± 4.0	0.85 ± 0.10	14.7 ± 7.0	$\mathbf{0.80 \pm 0.11}$
	AS-FedDAG	$\mathbf{1.8 \pm 2.0}$	$\mathbf{0.89 \pm 0.12}$	$\mathbf{2.5 \pm 2.7}$	0.85 ± 0.15	$\mathbf{5.0 \pm 4.2}$	$\mathbf{0.88 \pm 0.11}$	$\mathbf{7.8 \pm 5.5}$	0.80 ± 0.14

D.6 Comparisons with NOTEARS-ADMM

In this subsection, we give the experimental comparisons with NOTEARS-ADMM in detail to verify the advantage of our averaging strategy is simple but effective. Firstly, we conduct the results on linear models, which are the main part in Ng & Zhang (2022). As shown in Fig. 8, even on linear models, our AS-FedDAG can consistently show its advantage over NOTEARS-ADMM. Then, for the nonlinear models, we consider two different functions named MLP and Gaussian process (GP). The results are presented in Fig. 9, from which we can see that FedDAG always show better performance over all settings. Since NOTEARS-ADMM can not handle heterogeneous data, we do not give the results on heterogeneous data for fair comparison.

E More discussions on the experimental results

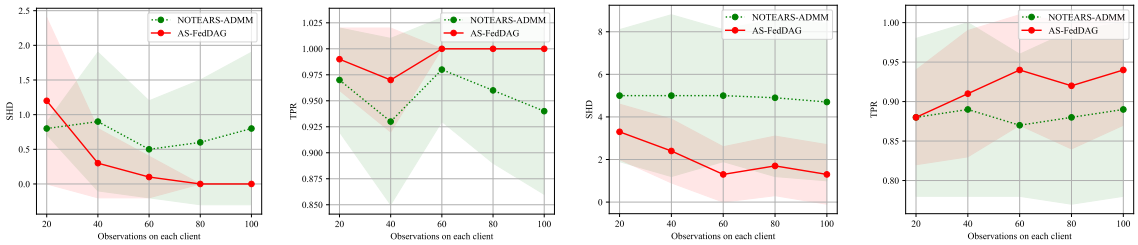
Why does our method outperform other methods even some baseline methods using all data for training?

Let us first discuss the AS-FedDAG (All-Shared FedDAG), which shares all model parameters (both $\boldsymbol{\Phi}$ and $\boldsymbol{U}$) among all clients. If we set it_{fl} as 1 in AS-FedDAG, AS-FedDAG is totally the same as MCSL using all data for training. For simplicity, we mark all parameters (actually $\boldsymbol{\Phi}$ and $\boldsymbol{U}$) of client c_k together as θ^{c_k}. Let us consider the t-th iteration when all clients receive the average parameters θ_t from the server and update their parameters by θ_t.

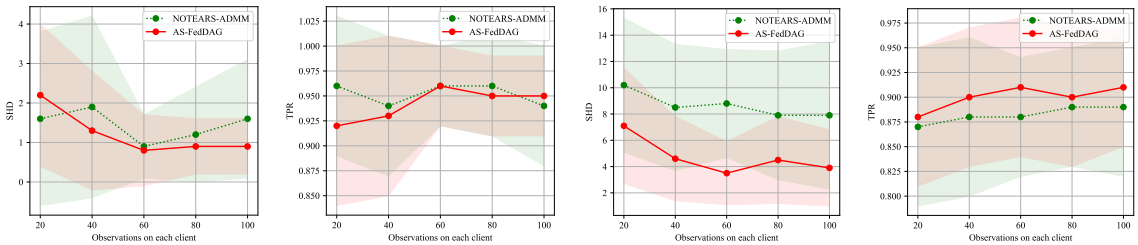
For AS-FedDAG, firstly, we mark the gradients obtained by using the local data of client c^k for $k \in [m]$ as $g_t^{c_k}$. Then each client c_k updates its parameters for one step by $\theta_t^{c_k} = \theta_t - lr \times g_t^{c_k}$, where lr is the learning rate. Afterward, the server collects all parameters and averages them to get $\theta_{t+1} = \frac{\sum_{k=1}^{m} \theta_t^{c_k}}{m} = \frac{\sum_{k=1}^{m} (\theta_t - lr \times g_t^{c_k})}{m} = \theta_t - lr \times \frac{\sum_{k=1}^{m} g_t^{c_k}}{m}$. For MCSL, there is only one θ. If MCSL uses full gradient information, then $\theta_{t+1} = \theta_t - lr \times \frac{\sum_{k=1}^{m} g_t^{c_k}}{m}$ (the full gradient is just the average of gradients from all samples). We can find that the updated parameters are totally the same. Then if $it_{fl} > 1$, we average all parameters every it_{fl} iterations. Even though the exact updating procedures are not the same, the expectations of updated parameters are the same. This is why we say that *MCSL trained on all data can serve as an approximate upper bound of our method but unobtainable* in our paper.

Table 13: Comparisons with CAM on nonlinear ANM (Heterogeneous data).

		ER2 with 10 nodes		SF2 with 10 nodes		ER2 with 20 nodes		SF2 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	CAM	31.9 ± 4.8	0.39 ± 0.15	31.8 ± 4.4	0.31 ± 0.17	104.6 ± 15.4	0.46 ± 0.15	116.9 ± 13.8	0.35 ± 0.07
Sep data	CAM	18.0 ± 1.7	0.52 ± 0.04	17.8 ± 2.1	0.51 ± 0.3	47.5 ± 9.2	0.52 ± 0.04	53.0 ± 6.1	0.50 ± 0.03
	GS-FedDAG	$\mathbf{1.9 \pm 1.6}$	$\mathbf{0.99 \pm 0.02}$	$\mathbf{2.6 \pm 1.3}$	$\mathbf{0.93 \pm 0.07}$	$\mathbf{6.2 \pm 4.7}$	$\mathbf{0.89 \pm 0.09}$	$\mathbf{11.5 \pm 6.7}$	$\mathbf{0.81 \pm 0.14}$



(a) ER1 with 10 nodes. (b) ER2 with 10 nodes.



(c) ER1 with 20 nodes. (d) ER1 with 20 nodes.

Figure 8: Comparisons with NOTEARS-ADMM on the linear model (IID).

In GS-FedDAG (Graph-Shared FedDAG) method, only all graphs are averaged. However, this partial information-sharing mechanism also helps on benefiting information from other clients to find a better solution (Collins et al., 2021).

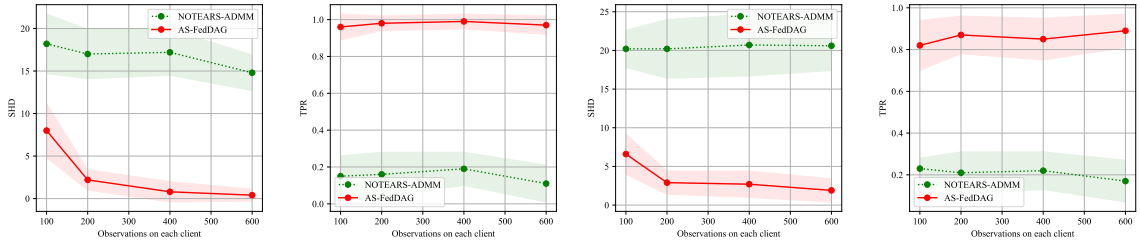
F Discussions on Assumptions

F.1 Data heterogeneity

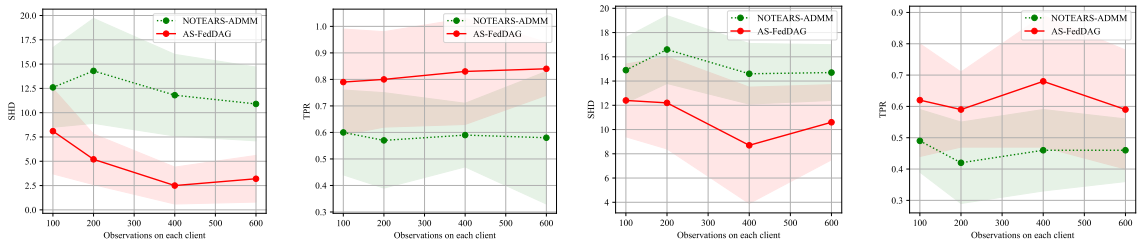
The general heterogeneous data setup should include the distribution shift caused by interventions since interventions on certain variables would also lead to heterogeneous distribution. Previous work (Huang et al., 2020b) has investigated this case and proposes the CD-NOD algorithm, which enhances the PC method, to learn from heterogeneous data. However, CD-NOD needs to identify some edge directions by capturing the changing information among distributions. That is to say, this method which needs to gather all data and cause the raw data leakage, of course. In our paper, we restrict our attention to the ANMs, which care more about the mechanisms and noise shift among different clients. Moreover, finding the identifiability conditions for learning graphs from the general heterogeneous data (both mechanisms shift and interventional data) in the federated setup is a challenging but important problem, which is left for future work.

F.2 Invariant DAG assumption

Firstly, let us skip the homogeneous data setting of FedDAG, which only assumes all SEMs are totally the same but data are generated at different local clients. Then, we mainly talk about the heterogeneous setting that assumes SEMs vary, but DAG is shared among different clients.



(a) ER1 with 10 nodes (GP). (b) ER2 with 10 nodes (GP).



(c) ER1 with 10 nodes (MLP). (d) ER2 with 10 nodes (MLP).

Figure 9: Comparisons with NOTEARS-ADMM on nonlinear models (Homogeneous data).

Table 14: Results on nonlinear ANM with different functions (Homogeneous data, 10 nodes, ER2).

		GP		MIM		MLP		GP-add	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	15.3 ± 2.6	0.37 ± 0.10	11.0 ± 4.9	0.60 ± 0.16	11.8 ± 4.3	0.61 ± 0.14	14.0 ± 4.7	0.49 ± 0.16
	GES	13.0 ± 3.9	0.50 ± 0.18	9.6 ± 4.4	0.71 ± 0.17	15.8 ± 6.0	0.63 ± 0.14	14.4 ± 4.9	0.57 ± 0.17
	DAG-GNN	16.2 ± 2.1	0.07 ± 0.06	13.7 ± 2.4	0.26 ± 0.10	18.2 ± 3.3	0.36 ± 0.12	13.3 ± 2.3	0.24 ± 0.10
	NOTEARS	16.5 ± 2.0	0.05 ± 0.04	12.1 ± 3.2	0.34 ± 0.13	13.3 ± 3.4	0.35 ± 0.15	13.4 ± 2.2	0.23 ± 0.09
	N-S-MLP	8.1 ± 3.8	0.56 ± 0.17	1.6 ± 1.3	0.95 ± 0.06	*5.6± 1.3*	*0.81± 0.11*	6.8 ± 4.0	0.65 ± 0.16
	MCSL	*1.9± 1.5*	*0.90± 0.08*	*0.7± 1.2*	*0.97± 0.06*	12.7 ± 3.6	0.58 ± 0.24	*1.9± 1.7*	*0.91± 0.07*
Sep data	PC	14.1 ± 2.4	0.31 ± 0.06	11.1 ± 3.6	0.48 ± 0.14	13.2 ± 3.6	0.42 ± 0.09	13.5 ± 3.2	0.37 ± 0.12
	GES	12.7 ± 2.7	0.37 ± 0.09	10.6 ± 3.3	0.54 ± 0.12	14.6 ± 4.6	0.50 ± 0.13	12.0 ± 2.6	0.48 ± 0.08
	DAG-GNN	15.7 ± 2.3	0.11 ± 0.05	11.7 ± 3.3	0.37 ± 0.12	17.7 ± 3.6	0.39 ± 0.11	13.0 ± 2.0	0.26 ± 0.10
	NOTEARS	16.5 ± 2.0	0.06 ± 0.04	12.3 ± 3.0	0.33 ± 0.12	13.4 ± 3.4	0.35 ± 0.14	13.3 ± 2.3	0.24 ± 0.09
	N-S-MLP	8.5 ± 2.9	0.56 ± 0.13	2.8 ± 1.5	$\mathbf{0.93 \pm 0.06}$	$\mathbf{6.4 \pm 1.3}$	$\mathbf{0.81 \pm 0.11}$	7.4 ± 2.9	0.67 ± 0.13
	MCSL	7.1 ± 3.2	0.83 ± 0.08	4.4 ± 2.1	$\mathbf{0.91 \pm 0.06}$	13.4 ± 3.9	0.57 ± 0.21	6.5 ± 3.5	0.84 ± 0.07
	GS-FedDAG	$\mathbf{2.4 \pm 2.0}$	$\mathbf{0.86 \pm 0.12}$	$\mathbf{2.1 \pm 1.4}$	0.91 ± 0.07	11.1 ± 3.1	0.57 ± 0.20	$\mathbf{2.6 \pm 1.6}$	$\mathbf{0.87 \pm 0.09}$
	AS-FedDAG	$\mathbf{1.8 \pm 2.0}$	$\mathbf{0.89 \pm 0.12}$	$\mathbf{1.7 \pm 1.6}$	0.91 ± 0.08	$\mathbf{10.5 \pm 3.5}$	$\mathbf{0.59 \pm 0.22}$	$\mathbf{2.4 \pm 1.6}$	$\mathbf{0.87 \pm 0.08}$

Essentially, an SEM models the physical processes of a system and the generation process behind observations. Intuitively, different SEMs usually describe different systems. Then, naturally, the DAGs may be different. In the case that the deployed systems on different clients are not the same, our method will break down because of the model misspecification. Unfortunately, it is not straightforward to extend our current framework to deal with this case, and we leave it for future work.

In this paper, we leave the variant causal graphs case aside and focus on the invariant graph case. This can be explained by the fact that a system can have various SEMs at different statuses (Huang et al., 2020b). In the real world, some cases can be supported by our assumptions. The first example can be fMRI recordings. As pointed in (Huang et al., 2020b), fMRI recordings are usually non-stationary because information flows in the brain may change with stimuli, tasks, and attention of the subject. Our federated setting only has one more assumption that fMRI recordings among different clients cannot be shared. The second example can be causal gene regulatory network inference (Omranian et al., 2016). The causal direction among genes, i.e., which gene regulates which gene, is believed to be the same. However, the SEM mechanism could vary in

Table 15: Results on nonlinear ANM with different functions (Homogeneous data, 20 nodes, ER2).

		GP		MIM		MLP		GP-add	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	32.7 ± 9.4	0.48 ± 0.13	22.8 ± 5.8	0.60 ± 0.15	33.7 ± 12.3	0.50 ± 0.13	35.2 ± 8.0	0.50 ± 0.09
	GES	27.1 ± 8.5	0.56 ± 0.11	21.5 ± 6.1	0.78 ± 0.09	44.9 ± 12.5	0.65 ± 0.11	41.7 ± 11.6	0.66 ± 0.08
	DAG-GNN	32.5 ± 6.8	0.10 ± 0.08	26.7 ± 7.4	0.26 ± 0.13	32.1 ± 10.4	0.38 ± 0.08	27.2 ± 2.4	0.24 ± 0.08
	NOTEARS	31.8 ± 6.0	0.11 ± 0.04	25.6 ± 6.1	0.29 ± 0.08	25.3 ± 8.0	0.40 ± 0.09	25.6 ± 3.9	0.28 ± 0.06
	N-S-MLP	18.2 ± 4.5	0.52 ± 0.10	4.1 ± 2.0	0.95 ± 0.04	*8.0 ± 3.9*	*0.86 ± 0.07*	12.6 ± 2.2	0.70 ± 0.06
	MCSL	*4.6 ± 4.6*	*0.90 ± 0.13*	*1.7 ± 1.6*	*0.97 ± 0.04*	18.1 ± 6.6	0.72 ± 0.14	*3.1 ± 1.9*	*0.92 ± 0.05*
Sep data	PC	32.7 ± 6.5	0.28 ± 0.07	24.4 ± 5.6	0.46 ± 0.11	30.6 ± 8.0	0.41 ± 0.09	29.5 ± 5.6	0.42 ± 0.10
	GES	28.6 ± 5.5	0.34 ± 0.06	20.5 ± 3.7	0.61 ± 0.06	34.4 ± 11.3	0.52 ± 0.09	29.3 ± 5.5	0.51 ± 0.07
	DAG-GNN	31.7 ± 6.1	0.12 ± 0.04	26.8 ± 5.8	0.26 ± 0.06	34.1 ± 9.7	0.46 ± 0.07	26.5 ± 4.0	0.27 ± 0.05
	NOTEARS	31.7 ± 6.0	0.11 ± 0.04	25.7 ± 5.9	0.29 ± 0.07	25.4 ± 7.4	0.42 ± 0.07	25.6 ± 3.8	0.29 ± 0.06
	N-S-MLP	19.5 ± 4.7	0.52 ± 0.07	$\mathbf{6.5 \pm 1.9}$	$\mathbf{0.92 \pm 0.03}$	$\mathbf{16.1 \pm 8.6}$	$\mathbf{0.86 \pm 0.07}$	16.2 ± 3.3	0.70 ± 0.07
	MCSL	24.8 ± 5.5	$\mathbf{0.88 \pm 0.07}$	20.4 ± 3.8	0.91 ± 0.05	30.2 ± 5.1	0.67 ± 0.12	16.2 ± 5.3	$\mathbf{0.87 \pm 0.05}$
	GS-FedDAG	$\mathbf{6.2 \pm 4.0}$	0.85 ± 0.10	8.5 ± 2.8	$\mathbf{0.93 \pm 0.05}$	21.4 ± 7.9	0.71 ± 0.14	$\mathbf{8.1 \pm 3.2}$	0.85 ± 0.05
	AS-FedDAG	$\mathbf{5.0 \pm 4.2}$	$\mathbf{0.88 \pm 0.11}$	$\mathbf{3.3 \pm 2.5}$	0.92 ± 0.07	$\mathbf{20.1 \pm 8.3}$	$\mathbf{0.72 \pm 0.14}$	$\mathbf{5.6 \pm 2.8}$	$\mathbf{0.86 \pm 0.06}$

Table 16: Results on heterogeneous setting with the different number of observations, (20 nodes, ER2).

		n =100		n =300		n =600		n =900	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
All data	PC	55.5 ± 8.5	0.21 ± 0.06	57.3 ± 5.7	0.29 ± 0.07	60.4 ± 9.8	0.32 ± 0.11	62.4 ± 6.6	0.29 ± 0.10
	GES	82.8 ± 13.7	0.38 ± 0.12	96.4 ± 14.9	0.48 ± 0.08	102.9 ± 13.6	0.51 ± 0.08	106.3 ± 14.3	0.50 ± 0.11
	DAG-GNN	61.8 ± 14.7	0.39 ± 0.07	56.8 ± 9.7	0.37 ± 0.08	57.7 ± 12.0	0.38 ± 0.08	57.9 ± 12.1	0.32 ± 0.08
	NOTEARS	58.7 ± 12.8	0.41 ± 0.12	57.6 ± 10.2	0.44 ± 0.06	57.3 ± 12.9	0.43 ± 0.08	59.4 ± 10.3	0.39 ± 0.10
	N-S-MLP	111.2 ± 14.4	0.92 ± 0.10	101.0 ± 16.8	0.92 ± 0.05	100.8 ± 14.7	0.90 ± 0.10	97.6 ± 14.8	0.90 ± 0.07
	MCSL	49.0 ± 8.1	0.62 ± 0.06	54.0 ± 10.0	0.70 ± 0.10	53.8 ± 9.6	0.73 ± 0.10	57.6 ± 11.6	0.73 ± 0.08
Sep data	PC	31.2 ± 5.7	0.30 ± 0.05	29.0 ± 5.9	0.39 ± 0.06	28.5 ± 6.3	0.44 ± 0.07	27.9 ± 6.6	0.47 ± 0.08
	GES	35.1 ± 8.3	0.48 ± 0.10	31.6 ± 9.8	0.57 ± 0.08	30.0 ± 8.0	0.62 ± 0.06	30.5 ± 10.7	0.64 ± 0.07
	DAG-GNN	29.9 ± 7.2	0.66 ± 0.09	20.3 ± 5.0	0.67 ± 0.09	18.5 ± 4.9	0.67 ± 0.09	18.0 ± 5.2	0.66 ± 0.11
	NOTEARS	$\mathbf{16.3 \pm 3.4}$	0.61 ± 0.08	$\mathbf{15.5 \pm 3.2}$	0.60 ± 0.08	15.0 ± 3.1	0.62 ± 0.09	15.2 ± 2.9	0.61 ± 0.09
	N-S-MLP	68.0 ± 5.4	$\mathbf{0.80 \pm 0.04}$	22.6 ± 3.3	$\mathbf{0.79 \pm 0.06}$	$\mathbf{12.7 \pm 2.6}$	$\mathbf{0.80 \pm 0.05}$	$\mathbf{11.8 \pm 2.8}$	$\mathbf{0.80 \pm 0.05}$
	MCSL	$32.8, \pm 5.4$	0.49 ± 0.08	26.4 ± 5.5	0.53 ± 0.09	23.3 ± 5.8	0.56 ± 0.08	23.1 ± 6.5	0.56 ± 0.07
	GS-FedDAG	$\mathbf{11.6 \pm 5.6}$	$\mathbf{0.83 \pm 0.11}$	$\mathbf{7.1 \pm 6.1}$	$\mathbf{0.90 \pm 0.12}$	$\mathbf{6.2 \pm 4.7}$	$\mathbf{0.89 \pm 0.09}$	$\mathbf{6.0 \pm 5.5}$	$\mathbf{0.91 \pm 0.11}$

each individual due to individual properties, such as age, gender, etc. Also, the assumption that domain shifts can also come from the distribution shifts of the exogenous variables (noise terms in our paper) has been widely accepted in the machine learning field, such as invariant causal prediction (Peters et al., 2016), IRM (Arjovsky et al., 2019).

Table 17: Results on randomly selecting models-info of partial clients (heterogeneous data, 20 nodes, ER2).

		Homogeneous data				Heterogeneous data			
		ER2 with 10 nodes		ER2 with 20 nodes		ER2 with 10 nodes		ER2 with 20 nodes	
		SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑	SHD ↓	TPR ↑
$\frac{r}{m}$	10%	3.8 ± 2.4	0.78 ± 0.14	8.6 ± 4.8	0.77 ± 0.13	3.8 ± 1.4	0.93 ± 0.05	8.5 ± 5.4	0.89 ± 0.07
	20%	3.2 ± 2.0	0.81 ± 0.12	6.7 ± 4.8	0.82 ± 0.13	2.5 ± 2.1	0.97 ± 0.04	8.2 ± 5.4	0.87 ± 0.09
	50%	2.9 ± 1.8	0.83 ± 0.11	5.8 ± 4.4	0.85 ± 0.12	1.8 ± 1.4	0.99 ± 0.02	6.3 ± 5.1	0.89 ± 0.10
	80%	2.7 ± 1.9	0.84 ± 0.12	6.0 ± 3.9	0.86 ± 0.10	1.8 ± 1.3	0.99 ± 0.02	5.9 ± 4.1	0.90 ± 0.08
	100%	2.4 ± 2.0	0.86 ± 0.12	6.2 ± 4.0	0.85 ± 0.10	1.9 ± 1.6	0.99 ± 0.02	6.2 ± 4.7	0.89 ± 0.09

Table 18: Empirical results on **fMRI Hippocampus** dataset (Part 2).

	All data			Separate data			GS-FedDAG	AS-FedDAG
	GES	N-S-MLP	DAG-GNN	GES	N-S-MLP	DAG-GNN		
SHD ↓	8.0 ± 0.0	9.0 ± 0.0	5.4 ± 0.5	8.3 ± 1.2	11.3 ± 1.0	8.2 ± 1.9	$\mathbf{6.4 \pm 0.9}$	$\mathbf{5.0 \pm 0.0}$
NNZ	11.0 ± 0.0	12.0 ± 0.0	3.3 ± 0.8	8.5 ± 1.1	14.4 ± 0.8	5.7 ± 1.4	6.8 ± 0.6	5.0 ± 0.0
TPR ↑	0.43 ± 0.00	0.43 ± 0.00	0.23 ± 0.07	0.31 ± 0.17	$\mathbf{0.44 \pm 0.10}$	0.17 ± 0.18	0.27 ± 0.12	$\mathbf{0.29 \pm 0.00}$
FDR ↓	0.73 ± 0.00	0.75 ± 0.00	0.52 ± 0.09	0.75 ± 0.12	0.78 ± 0.05	0.80 ± 0.18	$\mathbf{0.72 \pm 0.11}$	$\mathbf{0.60 \pm 0.00}$

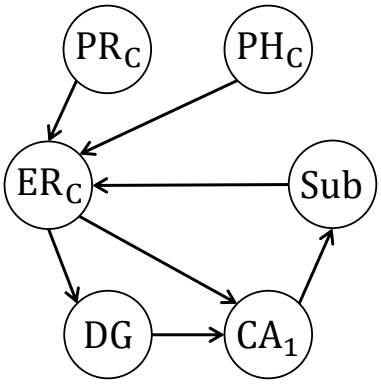


Figure 10: Anatomical causal-effect relationships of **fMRI Hippocampus** dataset