Towards Privacy-Aware Causal Structure Learning in Federated Setting

Jianli Huang[†], Xianjie Guo[†], Kui Yu*, Fuyuan Cao, and Jiye Liang

Abstract—Causal structure learning has been extensively studied and widely used in machine learning and various applications. To achieve an ideal performance, existing causal structure learning algorithms often need to centralize a large amount of data from multiple data sources. However, in the privacy-preserving setting, it is impossible to centralize data from all sources and put them together as a single dataset. To preserve data privacy, federated learning as a new learning paradigm has attached much attention in machine learning in recent years. In this paper, we study a privacy-aware causal structure learning problem in the federated setting and propose a novel Federated PC (FedPC) algorithm with two new strategies for preserving data privacy without centralizing data. Specifically, we first propose a novel layer-wise aggregation strategy for a seamless adaptation of the PC algorithm into the federated learning paradigm for federated skeleton learning, then we design an effective strategy for learning consistent separation sets for federated edge orientation. The extensive experiments validate that FedPC is effective for causal structure learning in federated learning setting.

Index Terms—Causal structure learning, Federated learning, Privacy preserving

1 Introduction

A causal structure is often represented using a directed acyclic graph (DAG). In a DAG, a directed edge $X \rightarrow Y$ means that X is a direct cause of Y while Y is a direct effect of X [1]. Causal structure learning aims to learn a DAG from observational data for revealing causal relations and it plays a vital role in causal inference, machine learning, and many other scientific tasks [2], [3], [4].

Various methods have been proposed for learning DAGs from observational data in the past decades [5], [6], [7], [8], [9]. Most existing DAG learning algorithms are designed to work only on a single dataset. For achieving an ideal performance, users often need to collect data from multiple decentralized data resources and put them together as a large-scale dataset, and the large scale of data are often generated from different companies and parties. For example, a large amount of web log data of an individual user are distributed in different websites and healthcare data are owned by different hospitals.

With the increasing data privacy concerns from both governments and users, a series of data protection initiatives and laws have been proposed in recent years. Decentralized data (e.g. healthcare and log data) contains sensitive information and the centralizing data strategy leads to potential privacy leakage. Many data owners prefer not to share their raw data with others owning to data privacy risk.

This makes most of DAG learning algorithms impractical in the privacy-preserving setting. To preserve data privacy, federated learning as a new learning paradigm has attracted increasing attention in machine learning [10]. Federated learning learns a local model from each client and only sends the local model parameters to a central server for aggregating them into a global model without access to each client's data [11].

In causal discovery area, little research has been done in developing algorithms for privacy-aware causal structure learning with the consideration of data privacy, although there are many well-developed algorithms for causal structure learning.

To fill this gap, in this paper, we propose to make the Peter-Clark (PC) algorithm [5] fit into the federated learning setting for privacy-aware causal structure learning. The PC algorithm is a constraint-based and widely used algorithm for learning causal structures in various settings, and it is computationally feasible to tackle sparse DAGs with up to thousands of variables [12]. Furthermore, most of well-established constraint-based algorithms are derived from the PC algorithm, and thus the idea of the paper can be directly applied to the PC-derived algorithms for designing privacy-aware DAG learning algorithms.

To make the PC algorithm fit into the federated learning setting, we face two challenging issues. First, since federated learning is a distributed learning paradigm while the PC algorithm is a centralized type of methods, it is challenging to seamlessly adapt the PC algorithm to the federated learning paradigm. Second, in the PC algorithm, learning separation sets is the key to orient edges. If adapting the PC algorithm to federated learning, different clients may have different separations sets for two non-adjacent variables. It is challenging on how to identify consistent separation sets

- [†] *represents equal contribution.*

- *Jianli Huang, Xianjie Guo, and Kui Yu are with the Intelligent Interconnected Systems Laboratory of Anhui Province and the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China (e-mail: janelee@mail.hfut.edu.cn, xianjieguo@mail.hfut.edu.cn, yukui@hfut.edu.cn).*
*(*Corresponding author: Kui Yu).*
- *Fuyuan Cao and Jiye Liang are with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China (e-mail: {cfy, ljy}@sxu.edu.cn).*

in the federated learning setting.

To tackle the issues, our contributions are as follows.

- We propose a novel federated PC (FedPC) algorithm for privacy-aware causal structure learning in the federated learning setting. FedPC comprises two novel subroutines, FedSkele and FedOrien, to address the above two challenging issues.
- We design the FedSkele subroutine with a novel layer-wise aggregation strategy to seamlessly adapt the PC algorithm to the federated learning paradigm for skeleton learning. This layer-wise strategy enables each client to share and update its skeleton parameters learnt at each layer of the FedPC algorithm at the server without sharing their original data. Moreover, this strategy can guarantee that the subroutine naturally converges without requiring any special parameters.
- We design the FedOrien subroutine with an effective strategy to identify consistent separation sets across clients for accurate edge orientation without centralizing data from each client to the server.
- We have conducted extensive experiments using synthetic, benchmark, and real datasets, and have compared FedPC with the state-of-the-art algorithms to demonstrate the effectiveness of FedPC.

2 Related Work

In the past decades, many algorithms have been designed for learning DAGs from observed data for causal structure learning [13]. In general, existing DAG learning algorithms are categorized into three types, score-based, constraint-based, and hybrid approaches. Score-based algorithms, such as GES [14] and GGSL [15], use a scoring function and a greedy search method to learn a DAG with the highest score by searching over all possible DAGs in a dataset and the representative algorithms. Constraint-based methods, such as PC and FCI [5], employ conditional independence (CI) tests to first learn a skeleton of a DAG, then orient the edges in the skeleton. The hybrid methods, such as MMHC [16], BCSL [17] and ADL [18], are the combinations of score-based and constraint-based methods. The score-based and constraint-based methods often return a completed partially DAG (CPDAG), i.e., a Markov equivalence class.

Zheng et al. [19] recently proposed to learn DAGs using gradient-based methods and designed the NOTEARS algorithm. NOTEARS defines a DAG as a weighted adjacency matrix and formulates the acyclic constraint as an equality acyclicity constraint and find the DAG using gradient-based methods with least squares loss. NOTEARS is designed under the assumption of the linear relations between variables. Subsequent works have extended NOTEARS to handle nonlinear cases, such as GOLEM [20] and DAG-NoCurl [21]. Another kind of methods employ different types of deep neural networks and a series of algorithms for learning DAGs. For example, DAG-GNN [22], GraN-DAG [23], NOTEARS-MLP [24] are some of the methods proposed. Moreover, the MCSL (Masked gradient-based Causal Structure Learning) algorithm [25] is also a gradient-based causal structure learning method. By using the binary mask, the results are mostly near either zero or one, so that the edges are easily identified by the threshold. We refer readers to the recent two survey papers of DAG learning for more details [13], [26].

In addition to the algorithms designed to work only on a single dataset, there is a line of work that is able to learn causal structures directly from multiple datasets. On the one hand, most of these methods require the multiple datasets with non-identical sets but sharing a small common set of variables [27], [28]. On the other hand, all of these algorithms do not consider data privacy problems.

Recently, Xiong et al. [29] presented a causal inference algorithm in the federated setting, but this algorithm does not learn causal structures. It aims to calculate causal effects. The closely related federated DAG learning algorithms using gradient-based optimization are FedDAG proposed by Gao et al. [30] and NOTEARS-ADMM proposed by Ng and Zhang [31]. NOTEARS-ADMM uses the alternating direction method of multipliers (ADMM), such that only the model parameters have to be exchanged during the optimization process and is capable of handling both linear and nonlinear cases. In contrast, FedDAG proposes a two-level structure consisting of a graph structure learning part and a mechanism approximating part, separately learns the mechanisms on local data and jointly learns the DAG structure to handle the data heterogeneity elegantly. Furthermore, the FedDAG method is based on MCSL [21] that uses binary masks, resulting in a binary adjacency matrix instead of a real-valued one. However, these methods often require employing complex neural network models or optimization techniques to achieve satisfactory performance, which can result in high computational overhead. In addition, although existing methods perform well on synthetic linear or nonlinear datasets, they may not be effective on discrete datasets.

In summary, many algorithms have been proposed for learning DAGs, but few methods have been designed for learning causal structures by considering data privacy problems. In this paper, we propose to develop new algorithms of causal structure learning by considering data privacy.

3 Methodology

In this section, we first briefly describe the original PC algorithm in Section 3.1, then present our proposed FedPC algorithm in Section 3.2, and finally discuss the privacy and costs of FedPC in Section 3.3. Table 1 provides a summary of the notations frequently used in this paper.

3.1 The PC algorithm

Let $\mathcal{X} = \{X_1, X_2, ..., X_m\}$ be the set of random variables in a dataset, $\mathcal{S}$ an undirected graph representing the skeleton of a DAG over $\mathcal{X}$, and $\boldsymbol{ne}(\mathcal{S}, X_i)$ the set of direct neighbors of X_i in $\mathcal{S}$, and $\mathcal{S}^c$ the complete (fully connected) undirected graph over $\mathcal{X}$. $\boldsymbol{SepSet}(X_i, X_j)$ is a separation set (conditioning set) that makes X_i and X_j conditionally independent, and $X_i \perp\!\!\!\perp X_j | \boldsymbol{Z}$ ($X_i \not\perp\!\!\!\perp X_j | \boldsymbol{Z}$) represents that given a conditioning set $\boldsymbol{Z}$, X_i and X_j are conditionally independent (dependent). The size of a conditioning set is denoted as ℓ. An unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}$ denotes the local skeleton $X_i - X_k - X_j$ where X_i and X_j

TABLE 1
Summary of Notations.

Notation	Meaning
$\mathcal{X}$	the set of variables in a dataset
X_i, X_j	a single variable in $\mathcal{X}(i, j = 1, 2, ..., m)$
N	number of clients
$X_i \perp\!\!\!\perp X_j \| \mathbf{Z}$	$X_i$and X_j are conditionally dependent given $\mathbf{Z}$
$X_i \not\perp\!\!\!\perp X_j \| \mathbf{Z}$	$X_i$and X_j are conditionally independent given $\mathbf{Z}$
$\boldsymbol{ne}(\mathcal{S}, X_i)$	the set of direct neighbors of X_i in $\mathcal{S}$
$\boldsymbol{SepSet}(X_i, X_j)$	a separation set of X_i from X_j
ℓ	the size of a separation set
$\mathbf{Z}$	a separation set with the length ℓ
$\mathcal{S}$	a direct acyclic graph over $\mathcal{X}$
$\mathcal{S}^c$	the complete undirected graph over $\mathcal{X}$
$\mathcal{S}^\ell$	the skeleton $\mathcal{S}^\ell$ obtained at the ℓ-th layer
$\mathcal{S}^*$	the final skeleton obtained in FedSkele subroutine
$< X_i, X_k, X_j >$	an unshield triple $X_i - X_k - X_j$ in the skeleton
α	the significance level of the statistical test

are not directly adjacent, but X_k is a direct neighbour of X_i and X_j. An unshielded triple $< X_i, X_k, X_j >$ forms a v-structure, i.e., $X_i \rightarrow X_k \leftarrow X_j$, if both $X_i \perp\!\!\!\perp X_j|\mathbf{Z}$ and $X_k \notin \mathbf{Z}$ hold.

The well-known PC algorithm [5] as a constraint-based method consists of the following three steps. Step 1 takes $\mathcal{S}^c$ as the initial skeleton, then updates the skeleton layer by layer by conducting conditional independence (CI) tests with an increasing value of ℓ.

The value of ℓ initially is set 0. At the first layer where $\ell = 0$ and $\mathcal{S}^c$ is taken as the initial skeleton, all pairs of adjacent variables in $\mathcal{S}^c$ ($\mathcal{S}^c$ is iteratively updated at this layer) are tested with an empty conditioning set $\mathbf{Z}$ (i.e. $\ell = |\mathbf{Z}| = 0$). If $X_i \perp\!\!\!\perp X_j$ holds, the edge between X_i and X_j is removed from $\mathcal{S}^c$ and the empty set $\mathbf{Z}$ is saved as a separation set in both $\boldsymbol{SepSet}(X_i, X_j)$ and $\boldsymbol{SepSet}(X_j, X_i)$. After all pairs of adjacent variables have been checked, the algorithm proceeds to the next layer with $\ell = 1$, and the skeleton obtained at the end of this layer is marked as $\mathcal{S}^0$. Then at the layer with $\ell = 1$ and $\mathcal{S}^0$ as the initial skeleton, the algorithm chooses a pair of adjacent variables (X_i, X_j) in $\mathcal{S}^0$, and checks whether there exsits a subset $\mathbf{Z} \subseteq \boldsymbol{ne}(\mathcal{S}^0, X_i) \setminus \{X_j\}$ with $|\mathbf{Z}| = 1$ which makes $X_i \perp\!\!\!\perp X_j|\mathbf{Z}$ hold. If so, the edge between X_i and X_j is deleted from $\mathcal{S}^0$ and the conditioning set $\mathbf{Z}$ is saved in the separation sets for both X_i and X_j. If all pairs of adjacent variables have been checked, the algorithm increases ℓ by one up to 2 and the skeleton obtained at the layer is marked as $\mathcal{S}^1$. This process continuous layer by layer until the sizes of direct neighbors of all variables in the skeleton of current layer are smaller than ℓ. We record the final skeleton as $\mathcal{S}^*$.

Step 2 identifies the v-structures by considering all unshielded triples in $\mathcal{S}^*$, and orients an unshielded triple $< X_i, X_k, X_j >$ as a v-structure if and only if $X_k \notin \boldsymbol{SepSet}(X_i, X_j)$. Finally, Step 3 orients as many of the remaining undirected edges as possible using the Meek rules [32]. The Meek rules require that orienting a remaining undirected edge does not form a new v-structure or a directed cycle in the current structure.

3.2 The proposed FedPC algorithm

Since the PC algorithm works only on a single dataset, a simple strategy to adapt the PC algorithm into federated learning is to learn a skeleton at each client independently, then aggregate the learnt skeletons and orient edges at a central server. However, the strategy has two potential problems. First, the different qualities of data (e.g. noise or small-sized samples) at different clients may lead to learnt skeletons with highly varying qualities, then directly aggregating them may not get a satisfactory final skeleton. Second, during skeleton learning, when X_i and X_j are conditionally independent, they may have a different separation set at each client due to the data quality problems, while an inaccurate separation set will lead to errors in edge orientations.

To protect data privacy and tackle the above two problems, we have designed the FedPC algorithm to work in the federated setting and have two subroutines, FedSkele (Federated Skeleton construction) and FedOrien (Federated edge Orientation).

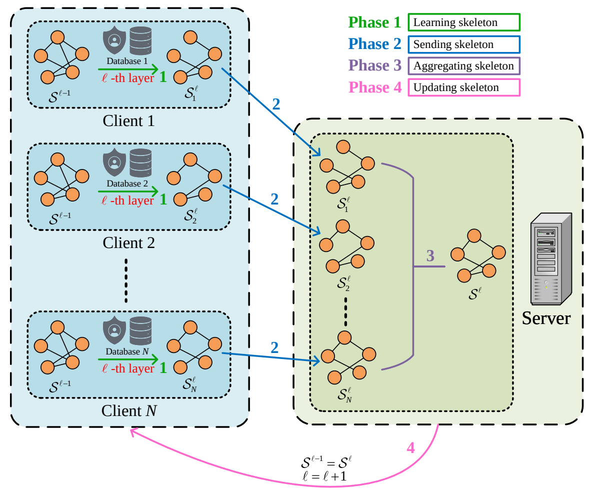


Fig. 1. The FedSkele subroutine of FedPC

3.2.1 *The FedSkele subroutine.*

Assuming that there are N clients (labeled as Client 1, Client 2,$\cdots$, Client N) and one central server. Inspired by the layer-by-layer skeleton learning idea of the PC algorithm, FedSkele is equipped with a novel layer-wise aggregation strategy to iteratively learn the skeleton with four phases in each layer as shown in Figure 1.

FedSkele starts from the layer of $\ell = 0$ with a complete undirected graph $\mathcal{S}^c$.

Phase 1: Each client learns the skeleton at the ℓ-th layer independently. At the ℓ-th layer, each client uses the FedSkele subroutine and its local dataset to update the initial skeleton ($\mathcal{S}^c$ when $\ell = 0$, or the skeleton $\mathcal{S}^{\ell-1}$ obtained at the $(\ell - 1)$-th layer) by first setting the size of the conditioning set for CI tests to ℓ. Then it traverses each adjacent variable pairs (X_i, X_j) in $\mathcal{S}^{\ell-1}$ (or $\mathcal{S}^c$ if $\ell = 0$), and checks if $\exists \mathbf{Z} \subseteq \boldsymbol{ne}(\mathcal{S}^{\ell-1}, X_i) \setminus \{X_j\}$ (or $\exists \mathbf{Z} \subseteq \boldsymbol{ne}(\mathcal{S}^c, X_i) \setminus \{X_j\}$ if $\ell = 0$) with the size of ℓ makes X_i and X_j independent. If so, the edge between X_i and X_j is removed from $\mathcal{S}^{\ell-1}$ (or $\mathcal{S}^c$). The skeleton learnt at the end of this phase at client n is $\mathcal{S}_n^\ell$ ($n \in \{1, 2, \ldots, N\}$).

Phase 2: Each client sends the skeleton learnt at the end of Phase 1, i.e., $\mathcal{S}_n^\ell$ ($n \in \{1, 2, \ldots, N\}$) simultaneously to the server at the ℓ-th layer.

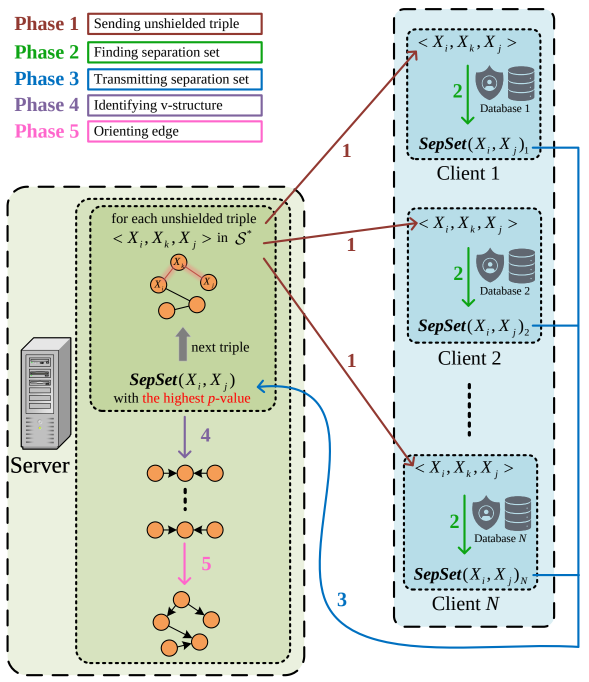

Phase 1 Sending unshielded triple
Phase 2 Finding separation set
Phase 3 Transmitting separation set
Phase 4 Identifying v-structure
Phase 5 Orienting edge
for each unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$
next triple
$SepSet(X_i, X_j)$ with the highest p-value
Server
Client 1
Client 2
Client N
Database 1
Database 2
Database N
$SepSet(X_i, X_j)_1$
$SepSet(X_i, X_j)_2$
$SepSet(X_i, X_j)_N$

Fig. 2. The FedOrien subroutine of FedPC

Phase 3: The server aggregates all skeletons learnt at the ℓ-th layer. The server receives the learnt skeletons from all clients, then aggregates these skeletons by merging $\mathcal{S}_1^\ell$, $\mathcal{S}_2^\ell, \cdots, \mathcal{S}_N^\ell$ into a global $\mathcal{S}^\ell$. Specifically, for each pair of variables X_i and X_j, if more than 30% (a default value) of clients compute that there is an edge between X_i and X_j at the ℓ-th layer, the edge is kept in $\mathcal{S}^\ell$. In Section 4.5, we will discuss why we use 30% as the default threshold.

Phase 4: The server sends the aggregated skeleton $\mathcal{S}^\ell$ to each client as the initial skeleton for skeleton learning at the $(\ell+1)$-th layer, if the value of ℓ is smaller than the maximum number of direct neighbors that a variable has in the ℓ-th layer skeletons learnt by all clients. Otherwise, the FedSkele subroutine is finished and the FedOrien subroutine starts.

This layer-wise aggregation strategy enables each client only to send and update its skeleton learnt at each layer at the server while protects the data privacy of each client. Then at each layer, by Phases 3 and 4, each client can achieve a high quality of an initial skeleton from the server for skeleton learning especially for the client owning low-quality data. For example, due to a small-sized sample problem, at the ℓ-th layer, if a true edge is wrongly deleted and not in $\mathcal{S}_n^\ell$ at Client n, by the aggregation strategy, the edge could be added to the skeleton $\mathcal{S}^\ell$ for the $(\ell+1)$-th layer skeleton learning of Client n. Thus this layer-wise strategy alleviates the problem of data quality. Additionally, as we discussed above, through the number of direct neighbors of a variable in the skeleton to control the layer size ℓ, the strategy guarantees that the FedSkele subroutine can converge to a high-quality skeleton without requiring any special parameters.

3.2.2 *The FedOrien subroutine.*

Let $\mathcal{S}^*$ denote the final skeleton obtained in the FedSkele subroutine. The FedOrien subroutine first identifies v-structures in $\mathcal{S}^*$, then orients the remaining undirected edges. Note that Our proposed FedPC algorithm uses the PC algorithm to learn the skeleton and orient the undirected edges, hence the output of FedPC is a CPDAG instead of a DAG. After five phases in the FedSkele subroutine, we employ the acyclic constraint technology proposed in reference [5] to transform CPDAG returned by FedPC into DAG.

For an unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$, to check whether the triple is a v-structure, as discussed above, we need to have the separation set of X_i and X_j which makes X_i and X_j conditionally independent. In fact, we can attain such a separation set in the FedSkele subroutine. However, due to potential data quality problems at clients, different clients may have different separation sets for an unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$. In addition, $\mathcal{S}^*$ is an aggregated skeleton, then the separation set for an unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$ may be different from those separation sets attained at the clients in the FedSkele subroutine. Therefore FedPC does not save separation sets in the FedSkele subroutine. But this brings challenges for finding a separation set for the unshielded triple $< X_i, X_k, X_j >$ that is consistent with $\mathcal{S}^*$ for v-structure learning in the FedOrien subroutine.

Since each client does not share its raw data with the server, the FedOrien subroutine cannot directly compute the separation set for any non-adjacent variables in $\mathcal{S}^*$ at the server. Thus we have designed an effective strategy for the FedOrien subroutine to identify consistent separation sets for learning v-structures in $\mathcal{S}^*$ across clients. To validate the effectiveness of this proposed strategy, in experiments, we also present a FedPC-simple-II algorithm. It saves separation sets in the FedSkele subroutine and takes the intersection of the separation sets of two non-adjacent variables learnt from each client as their separation set in $\mathcal{S}^*$.

As outlined in Figure 2, the FedOrien subroutine consists of the following five phases.

Phase 1 (Sending unshielded triples): The server first identifies all the unshielded triples in $\mathcal{S}^*$, then sends each of them and the set of their direct neighbors to each client for learning separation sets. Here we use an unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$ as an example. During this phase, the server sends the triple $< X_i, X_k, X_j >$ and $\boldsymbol{ne}(\mathcal{S}^*, X_i)$ to each client. Then the FedOrien subroutine asks each client to independently find a subset $\mathbf{Z} \subseteq \boldsymbol{ne}(\mathcal{S}^*, X_i) \setminus \{X_j\}$ that makes $X_i \perp\!\!\!\perp X_j | \mathbf{Z}$ hold using its local data.

Phase 2 (Finding separation sets): Assume the null hypothesis of "$H_0 : X_i \perp\!\!\!\perp X_j | \mathbf{Z}$", for a CI test of X_i and X_j given a subset $\mathbf{Z}$, $X_i \perp\!\!\!\perp X_j | \mathbf{Z}$ holds, if and only if the p-value is greater than α (α is the significance level of the statistical test). At Phase 2, if a client finds that $X_i \perp\!\!\!\perp X_j | \mathbf{Z}$ holds using its local data, the client saves the separation set $\mathbf{Z}$ and the p-value of the CI test . When finding all separation sets, the client selects the separation set with the highest p-value as the consistent separation set and send it to the server.

Phase 3 (Transmitting separation set): To achieve a consistent separation set for identifying whether the unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$ is a v-structure, all clients send separation sets and p-values about the unshielded triple $< X_i, X_k, X_j >$ to the server. Then at the sever the FedOrien subroutine selects the separation set with the highest p-value from those separation sets as the consistent separation set for v-structure learning. The rationale behind the idea is described as follows. For an unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$, X_i and X_j are assumed that they are independent in the skeleton S^*. Thus, if a CI test of X_i and X_j given a separation set with the highest p-value, the separation set has a high probability to make X_i and X_j really independent and to be a true separation set in the underlying DAG. Then this kind of separation sets can help us identify accurate v structures.

Phase 4 (Identifying v-structure): Based on the selected consistent separation sets for all unshielded triples in $\mathcal{S}^*$, at the server, the FedOrien algorithm identifies the v-structures from all unshielded triples without access to each client's data. For example, for the unshielded triple $< X_i, X_k, X_j >$ in $\mathcal{S}^*$, if X_k is not in the separation set of X_i and X_j found in Phase 3, the FedOrien subroutine considers $< X_i, X_k, X_j >$ as a v-structure and orients it as $X_i \rightarrow X_k \leftarrow X_j$.

Phase 5 (Orienting remaining edges): Based on the learnt v-structures, at the server, the FedOrien subroutine orients the remaining undirected edges as many as possible using the Meek rules without requiring clients' data.

3.3 Privacy preservation and communication cost

Data privacy when using FedPC. In a federated learning setting, when the clients and the sever exchange causal skeletons, it would reveal the independence/dependence relationships between variables. To protect the semantic information of variables while avoiding direct communication between clients, we design an easily implementable privacy protection strategy in the FedPC algorithm. Specifically, we require the remote server to send instructions to each client, requesting them to sort and assign unique identifiers (e.g., "1", "2", "3", ...) to the semantic information of all variables based on their alphabetical order. If variables share the same first letter, they are further sorted based on the second letter of their semantic information, and so on. Each client then sends only the assigned identifiers to the remote server for aggregation (we provide an illustrative example of this strategy in Section S-2 of the Supplementary Material.). Given our assumption that the feature space dimensions are the same across different clients, this strategy ensures variable alignment without the need for communication between clients, while maintaining the confidentiality of variable semantics from the remote server. Meanwhile, compared with existing federated causal structure learning algorithms, FedPC exchanges the skeleton without including edge directions, which further reduces the possibility of leaking sensitive information indirectly.

Communication costs of FedPC. The communication costs of FedPC are analyzed as follows. In the FedSkele subroutine, at each layer, each client needs to send to and receive from the server an adjacency matrix respectively. Since FedPC uses the layer-wise aggregation strategy to learn skeletons, it seems that there might have many communication rounds between the server and clients. In fact, the communication costs can be managed and regulated by the maximum size of direct neighbors of a variable in the currently learnt skeleton. As we discussed in Section 4.4, as the value of ℓ increases, falsely direct neighbors will be removed from the currently learnt skeleton and the sizes of direct neighbors of the variables become small. The FedSkele subroutine ends if the value of ℓ is bigger than the sizes of direct neighbors of all variables in the currently learnt skeleton. Thus, the convergence of this subroutine is determined by the maximum size of direct neighbors of a variable in the learnt skeleton. The subroutine often converges with a small value of ℓ, especially with a spare underlying DAG. On average, the parameter converges when the value of ℓ lies between 3 and 5. This makes the communication costs of FedPC predictable and adjustable, and the system can be scaled up or down depending on the communication resources available.

TABLE 2
Details of the five benchmark Bayesian networks

Network	Number of variables	Number of edges	Maximum in/out-degree
alarm	37	46	4/5
insurance	27	52	3/7
win95pts	76	112	7/10
andes	223	338	6/12
pigs	441	592	2/39

4 Experiments

In this section, we conduct experiments to verify the effectiveness of FedPC against its rivals. We first describe the experimental settings in Section 4.1, then discuss the experimental results on synthetic and real datasets in Sections 4.2 and 4.3, respectively. Finally, we analyze the parameters related to FedPC in Section 4.4 and 4.5.

4.1 Experiment settings

4.1.1 Datasets.

The datasets used in the experiments include the following synthetic and real datasets.

We assume that there are K data samples in a dataset and N clients exists, and N lies in $\{3, 5, 10, 15\}$. To introduce unevenness, we randomly assigned the data samples to each client while ensuring that each client contains at least $\frac{K}{N*2}$ data samples. This approach aims to ensure that the data distribution is not heavily skewed towards any specific client. The dataset in each client has the same set of variables and we assume that the dataset in each client is generated from the same ground-truth DAG.

- **Benchmark Bayesian network (BN) datasets.** We use five benchmark BNs, alarm, insurance, win95pts, andes and pigs, to generate five discrete datasets, respectively. Each dataset contained 5000 samples. The details of the five benchmark BNs are presented in Table 2.
- **Synthetic datasets.** We conduct FedPC on synthetic linear and nonlinear datasets. We present experimental results of FedPC on linear datasets in Section 4.2, and the results on nonlinear datasets in Section S-1-3 in the Supplementary Material. For the linear synthetic datasets, we generate five continuous datasets

using an open-source toolkit [33] with the number of variables to 10, 20, 50, 100 and 300 respectively. Each synthetic dataset contains 5000 continuous samples. The generative process employs a linear causal mechanism represented as follows:

$$y = \mathbf{X}W + \times E, \quad (1)$$

where $+\times$ denotes either addition or multiplication, $\mathbf{X}$ denotes the vector of causes, and E represents the noise variable accounting for all unobserved variables. For the nonlinear synthetic datasets, the causal mechanism used in the generative process is Gaussian Process (GP), and the mechanisms are represented as:

$$y = GP(\mathbf{X}) + \times E. \quad (2)$$

The proportion of noise in the mechanisms is set to 0.4. Gaussian noise was used in the generative process. On nonlinear datasets, we utilize the Kernel-based Conditional Independence test (KCI-test) [34] instead of Fisher's Z Conditional Independence test for detecting nonlinear dependencies.

- **Real dataset.** We also compare FedPC with its rivals on a real biological dataset with 853 samples, Sachs [35]. Sachs is a protein signaling network expressing the level of different proteins and phospholipids in human cells. It is commonly considered as a benchmark graphical model with 11 nodes (cell types) and 17 edges.

4.1.2 Evaluation metrics.

To evaluate the performance of FedPC with its rivals, we use the following frequently used metrics in causal structure learning (More evaluation metrics and the corresponding results please see the Supplementary Material).

- SHD (Structural Hamming Distance). The value of SHD is calculated by comparing the learnt causal structure with the true causal structure. Specifically, the value of SHD is the sum of undirected edges, reverse edges, missing edges and extra edges. In Section S-1 of the Supplementary Material, we present reverse, extra, miss metrics of our method and its rivals to further validate the effectiveness of our method.
- F1. The F1 measure is a comprehensive evaluation metric, and it is calculated as $F1 = \frac{2*Recall*Precision}{Recall+Precision}$. Precision is equal to the number of correctly predicted arrowheads in the output divided by the number of edges in the output of an algorithm, while Recall is the number of correctly predicted arrowheads in the output divided by the number of true arrowheads in the true causal structure. We present more experimental results in Section S-1 of the Supplementary Material, where Precision and Recall are shown to compare the performance of FedPC with that of its rivals. SHD and F1 are used to measure structure error and structure correctness, respectively.
- Time. We report running time (in seconds) as the efficiency measure of different algorithms.

FedPC utilizes the PC algorithm to learn the skeleton and orient the undirected edges in the causal graph. Therefore, the output of FedPC is a CPDAG, which contains both directed and undirected edges. We employ the acyclic constraint technology proposed in reference [5] to transform CPDAGs returned by FedPC into DAGs, and then calculated the Structural Hamming Distance (SHD) and F1 scores of these DAGs.

4.1.3 Comparison methods.

FedPC is compared with 8 rivals. The comparison methods are as follows:

(1) NOTEARS-Avg. We run the NOTEARS [19] algorithm at each client independently, then calculate the averaging results of SHD and F1 of all learnt DAGs as the final results.

(2) NOTEARS-ADMM. We run the NOTEARS-ADMM algorithm [31], and then calculate the SHD and F1 of the learnt DAG as the final results.

(3) FedDAG. We run the FedDAG [30] algorithm and then calculate the SHD and F1 of the learnt DAG as the final results.

(4) PC-Avg. We first run the PC algorithm at each client independently for obtaining N DAGs (N is the number of clients), and then calculate the averaging SHD values and F1 values of all learnt DAGs as the final results of PC-Avg.

(5) PC-Best. We first run the PC algorithm at each client independently to get N DAGs, and then select the DAG with the lowest SHD value as the final output.

(6) PC-All. We centralize all clients' data to a single dataset and run the PC algorithm on it.

(7) FedPC-Simple-I. We run the PC algorithm at each client independently to learn the DAGs, then aggregate all learnt DAGs at the server by the strategy that if more than 30% (the same ratio as our method) of the learnt DAGs contain a directed edge between two variables, this edge is kept in the final DAG.

(8) FedPC-Simple-II. We run the PC algorithm to learn the skeletons independently at each client, then aggregate all learnt skeletons at the server by the strategy that if an undirected edge between two variables exists on more than 30% of the skeletons, this edge will be kept in the final skeleton. Then we take the intersection of the separation sets between two variables learnt from each client to learn v-structures. This is an ablation study of our proposed algorithm, by removing the layer-wise strategy of the FedOrien subroutine.

4.1.4 Implementation Details.

All experiments were conducted on a computer with Intel Core i9-10900F 2.80-GHz CPU and 32-GB memory. The significance level for CI tests is set to 0.01. For PC, NOTEARS, and NOTEARS-ADMM, we used the source codes provided by their authors. NOTEARS-Avg and NOTEARS-ADMM use 0.3 as the threshold to prune edges in a DAG and FedDAG uses 0.5 as the threshold, those are the same as the original paper. The source codes of FedPC are provided in the Supplementary Materials.

4.2 Results on synthetic and benchmark data

In the section, we report SHD, F1 and running time of FedPC and its rivals, respectively.

4.2.1 The SHD metric.

Figures 3 to 4 (the top row in each figure) show the SHD values of FedPC and its 8 rivals using five discrete BN datasets and five continuous synthetic datasets, respectively. The smaller value of SHD denotes better performance of an algorithm. We can see that FedPC outperforms its six rivals.

- The value of SHD of FedPC is much smaller than that of NOTEARS-Avg, NOTEARS-ADMM and FedDAG on all datasets. Since it is hard for the two rivals to select a suitable threshold to prune false directed edges, the DAGs learnt by these two rivals often contain a larger number of false edges, leading to inaccurate DAGs.
- FedPC is superior to FedPC-Simple-I and FedPC-Simple-II on the SHD metric. FedPC-Simple-I directly applies the PC algorithm to each client and directly aggregates the learnt DAGs to get the final DAG. FedPC-Simple-II simply aggregates skeletons and takes the intersection of the separation sets to orient edges. However, FedPC uses a layer-wise aggregation strategy for accurate skeleton learning and a consistent separation set identification strategy for accurate edge orientations.
- FedPC is better than PC-Avg and PC-Best, since PC-Avg and PC-Best do not exchange information between a client and the server. This further verifies the effectiveness of the two strategies of FedPC. The results obtained by FedPC may not be as accurate as those have a sufficient sample size like PC-All because PC-All centralizes all independent clients datasets. However, on synthetic datasets, FedPC performs better than PC-All. The explanation is that synthetic datasets is ideal while benchmark networks and the real dataset are not ideal datasets. So data samples with uncertainty cannot be generated in synthetic dataset.
- We can observe that as the number of clients increases, the Structural Hamming distance (SHD) metric increases while the F1 metric decreases. This is due to the reduced number of samples per client when the total number of data samples among all clients. The decrease in sample size interferes with the accuracy of the conditional independence test and reduces the confidence in accepting the null hypothesis of conditional independence between two variables. Consequently, a larger number of erroneous separation sets are identified, leading to incorrect identification of v-structures and a decline in the algorithm's performance. However, to mitigate the impact of the reduced sample size, we choose the separation set with the highest p-value, indicating the highest likelihood of conditional independence between the two variables. This approach minimizes the influence of the smaller sample size on the final determination of independence or dependence.

4.2.2 The F1 metric.

Figures 3 to 4 (the second row in each figure) show the F1 values of FedPC and its rivals using five benchmark BN datasets and five synthetic datasets, respectively. The higher value of F1 denotes better performance of an algorithm. We can see that FedPC achieves a higher F1 than its rivals. The explanations are as follows.

FedDAG identifies most of correct directed edges while many wrong edges are included. NOTEARS-Avg and NOTEARS-ADMM identify only a small number of correct directed edges, while PC-Avg and PC-Best wrongly remove many correct edges. FedPC-Simple-I and FedPC-Simple-II just aggregate the DAGs and skeletons learnt by each client and lack an effective strategy to identify consistent separation sets, thus there are many false edges in the final result.

In addition, NOTEARS-Avg and NOTEARS-ADMM are sensitive to the user-defined threshold. Different clients may have different thresholds for edge pruning. An unsuitable threshold may prune either correct edges or retain false edges.

As the number of clients increases, using the five BN datasets, the F1 and SHD values of FedPC and its six rivals decrease, while they do not change much, even increase a little. The possible explanation is that PC-derived algorithms employ the chi-squared test for discrete data while using the Fisher Z test for continuous data for CI tests. In general, PC-derived algorithms need more data samples for discrete BN data sets than continuous synthetic data for reliable CI tests. Then as the number of clients increases, the number of data samples of each client becomes insufficient, accordingly the performance of FedPC and its six rivals degrades a little using discrete data. However, with FedPC a client owns insufficient data examples leading to incorrect CI tests, the layer-wise aggregation strategy makes FedPC exchange enough information between clients and the server for accurate DAG learning.

To give a comprehensive performance comparison between FedPC with its rivals, in Section 4.2.4, we conducted statistical tests to show that FedPC is significantly better than other methods.

4.2.3 Time Efficiency.

Figures 3 to 4 (the last row in each figure) show the execution time of FedPC and its rivals on the five discrete BN datasets and the five continuous synthetic datasets. With the five continuous datasets and most BN datasets, FedPC is faster than NOTEARS-Avg and NOTEARS-ADMM. FedPC is a little slower than PC-Avg, PC-Best, FedPC-Simple-I and FedPC-Simple-II since FedPC needs more time than those baselines for communications between clients and the server during skeleton learning and it also needs to find separation sets at each clients for getting consistent separation sets. As the number of clients increases, the running time of most algorithms increases accordingly. FedPC spends more time on the pigs dataset since the size of direct neighbors of variables in the pigs dataset is much larger than that in the other datasets, leading to more time to learn skeletons and separation sets. In summary, at most times, FedPC is competitive with PC-Avg, PC-Best, FedPC-Simple-I and FedPC-Simple-II on running time and faster than NOTEARS-Avg and NOTEARS-ADMM.

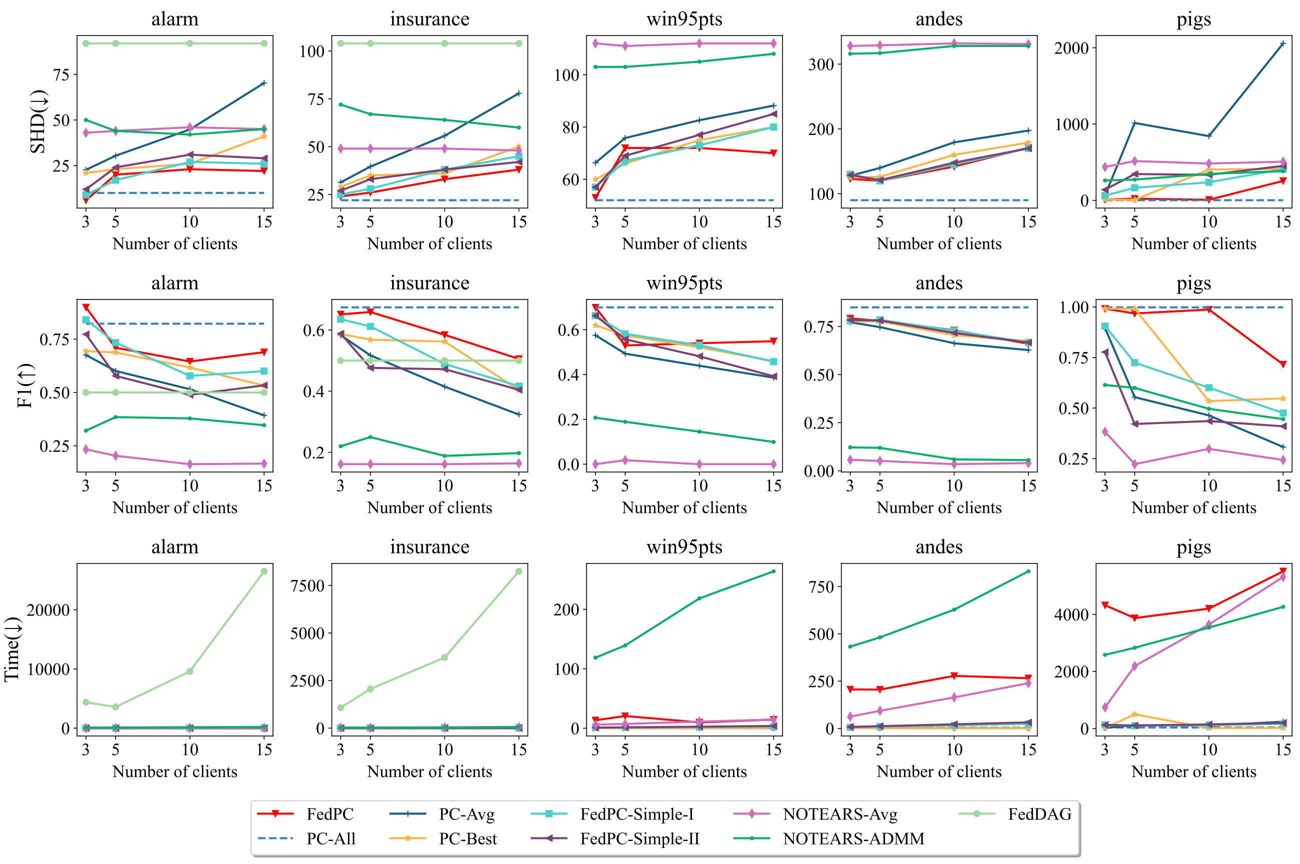


Fig. 3. Results on 5 benchmark BN datasets

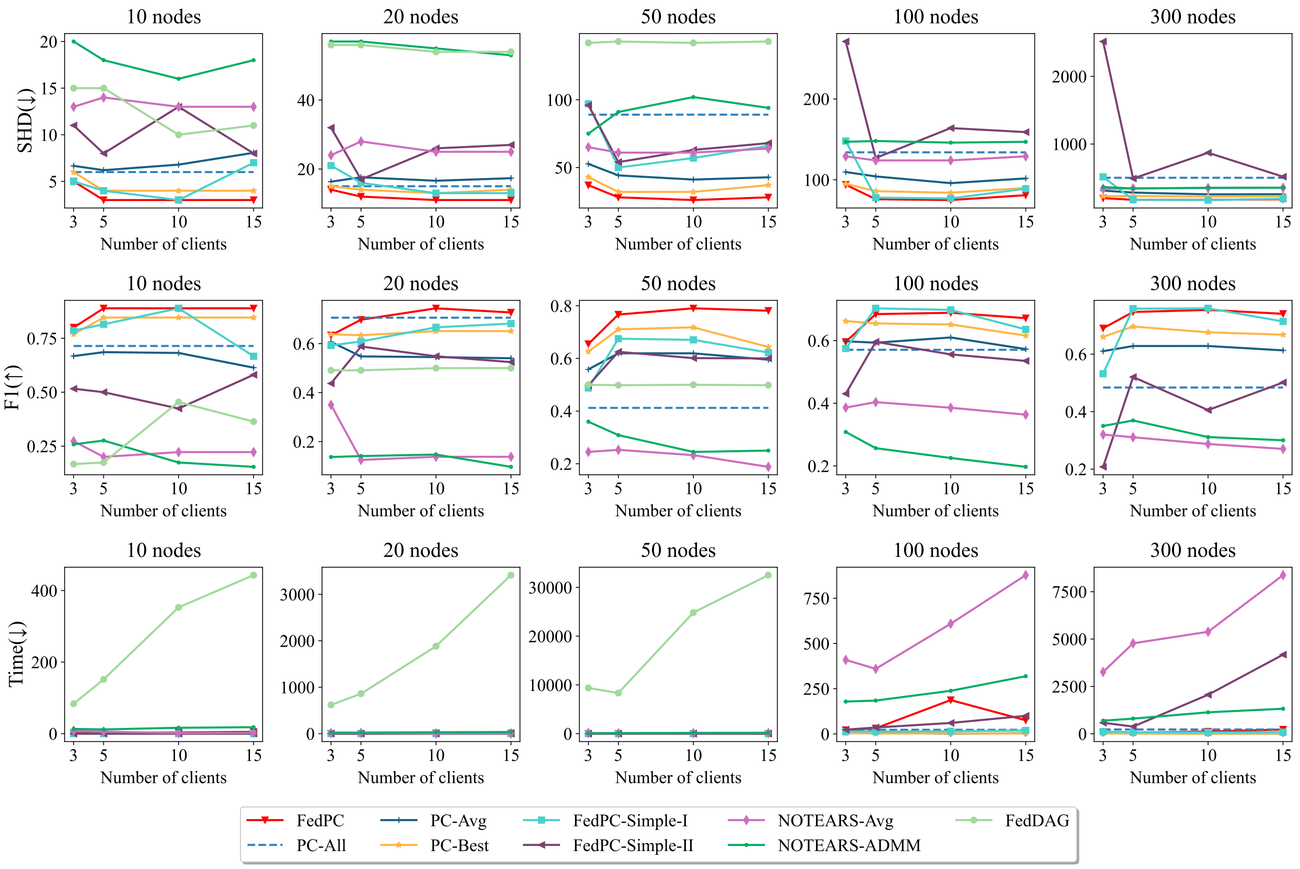


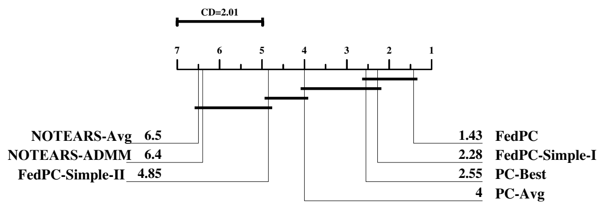
Fig. 4. Results of 5 synthetic datasets

4.2.4 Statistical Tests.

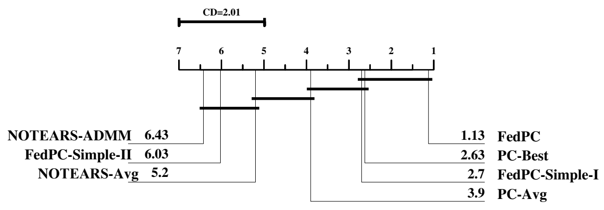
To give a comprehensive performance comparison between FedPC with its rivals, the Friedman test and Nemenyi test [36] are performed.

We first perform the Friedman test at the 0.05 significance level under the null-hypothesis, which states that the performance of all algorithms is the same on all datasets (i.e., the average ranks of all algorithms are equivalent). Then, we perform the Nemenyi test, which states that the performance of two algorithms is significantly different if the corresponding average ranks differ by at least one critical difference (CD). Note that we only perform these tests on the datasets where all algorithms can run the results.

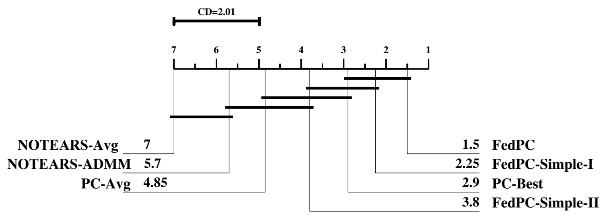
Figs. 5(a), (b), (c) and (d) provide the CD diagrams, where the average rank of each algorithm is marked along the axis (lower ranks to the right). FedPC is the only algorithm that achieves the lowest rank value on both synthetic and benchmark datasets. On synthetic datasets, whether F1 metric or SHD metric, we observe that FedPC achieves a comparable performance against PC-Best and FedPC-Simple-I, and it is significantly better than the other algorithms. In terms of the SHD metric of each algorithm on BN datasets, we note that FedPC significantly outperforms NOTEARS-Avg, NOTEARS-ADMM and PC-Avg, and it achieves a comparable performance against the other algorithms.



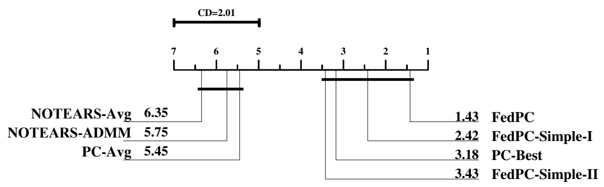
(a) F1 metric of each algorithm on synthetic dataset.



(b) SHD metric of each algorithm on synthetic dataset.



(c) F1 metric of each algorithm on benchmark datasets.



(d) SHD metric of each algorithm on benchmark datasets.

Fig. 5. Comparison of FedPC against its rivals with the Nemenyi test. (the lower the rank value, the better the performance.)

4.3 Results on real data

In this section, we compare FedPC with its rivals on a real dataset, Sachs [35], and the experimental results are shown in Figure 6. From Figure 6, we can see that FedPC achieves the lowest value of SHD regardless of the number of clients. For the F1 metric, when the number of clients is 5 or 10, the F1 value of FedPC is significantly higher than that of other algorithms. Due to the small scale of the Sachs dataset, the execution time of each algorithm is trival and thus we do not report the running time in Figure 6.

4.4 Parameter analysis

As discussed above, the value of ℓ is determined by the maximum size of direct neighbors of a variable in the skeleton. Figure 7 shows the relation of the value of ℓ and the convergence of FedPC. We can see that the number of edges in a skeleton gradually decreases when ℓ is from 0 to 5. On average, the FedSkele subroutine converges when the value of ℓ lies between 3 and 5. The possible explanation is that the underlying DAG of a dataset is often a sparse one.

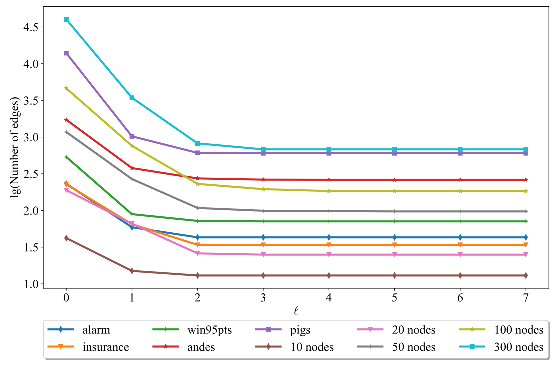


Fig. 7. The value of ℓ and the convergence of FedPC

4.5 Analysis of the ratio of clients.

In Phase 3 of the FedSkele subroutine, a threshold ratio of 30% is chosen to remove or keep an edge in an aggregated skeleton. This threshold ratio is chosen after considering that if the ratio is too low, many incorrect edges may be kept in the skeleton, while if it is too high, many correct edges may be removed from the skeleton.

To further analyze the effect of the threshold ratio, experiments were conducted with ratios ranging from 20% to 90%. The SHD values of the aggregated skeleton using all 10 datasets were observed, and the values were normalized using min-max normalization to scale them to a range of $(0, 1)$. The normalized values of SHD with different ratios are shown in Figure 8.

Empirical results showed that, in general, the SHD value first decreases and then increases as the ratio of clients increases. However, the minimum value of SHD was reached on most of the datasets when the ratio was up to 30%. This suggests that a threshold ratio of 30% is an appropriate value for achieving good performance in FedSkele.

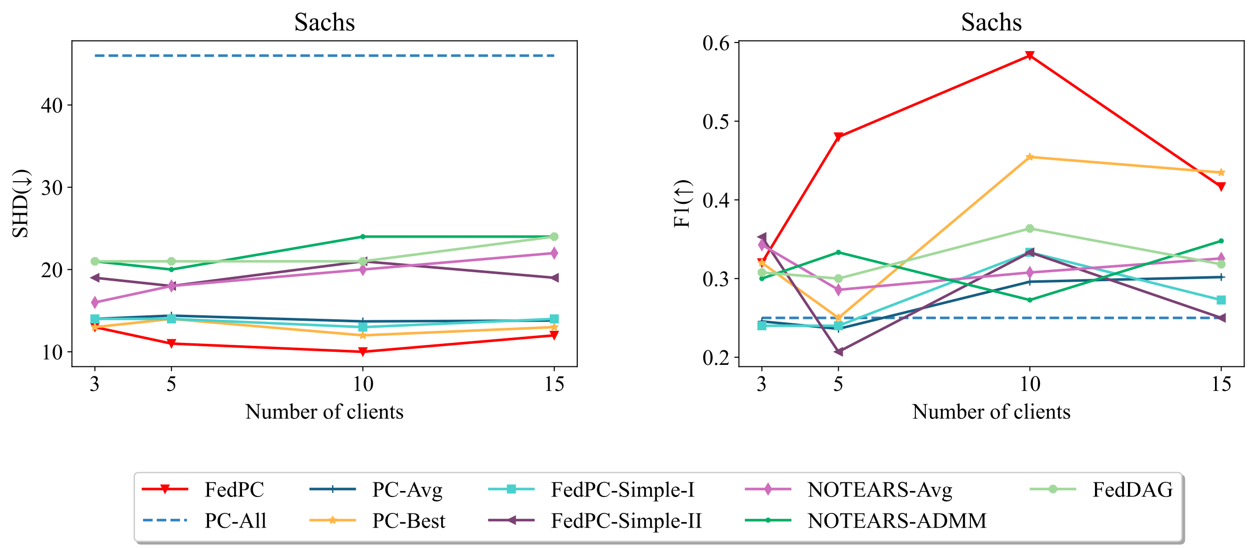


Fig. 6. Results of the real dataset

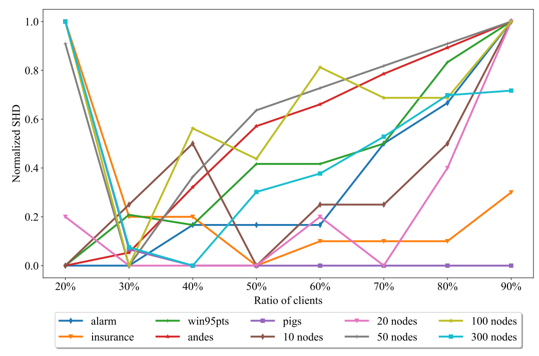


Fig. 8. Normalized SHD and the ratio of clients

5 Conclusion and Future Work

Learning causal structures while preserving data privacy brings many challenges to traditional causal structure learning methods. In this paper, we have studied the privacy-aware causal structure learning problem in the federated learning setting and have developed a novel FedPC algorithm that seamlessly have the classic PC algorithm into the federated learning paradigm. Compared to 8 competing algorithms, FedPC achieves promising performance using various datasets. Furthermore, the idea of FedPC can be applied to existing PC-derived algorithms for designing new algorithms for tackling data privacy.

Meanwhile, we briefly discuss a few possible research directions for future work. Firstly, it is interesting to systematically analyze the different strategies for identifying separation sets for identifying v structures in the FedOrien subroutine. One strategy is to calculate the p-value for all possible separation sets of two variables in an unshielded triple and send them to the server to compute the average p-value (across different clients) for each separation set, then select the separation set with the highest p-value. Another strategy is that we find all separation sets at each client that meets a given threshold, and send these separation sets to the server to perform a voting across these separation sets collected for getting a suitable separation set. Secondly, in the practical setting, multiple privacy-preserving datasets may contain hidden variables or heterogeneous, we plan to extend FedPC to deal with either hidden variables or heterogeneous data. Thirdly, we will combine FedPC with IDA [37] for studying privacy-aware cause effect estimation for robust machine learning. Finally, in practical scenarios, the variables may differ across clients. It is worth studying that each client has a different set of variables but all clients share a common set of variables in federated setting.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (under grant 2021ZD0111801) and the National Natural Science Foundation of China (under grant 62176082).

References

[1] J. Pearl, *Causality*. Cambridge university press, 2009.
[2] S. Yang, K. Yu, F. Cao, L. Liu, H. Wang, and J. Li, "Learning causal representations for robust domain adaptation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2750–2764, 2023.
[3] K. Kuang, L. Li, Z. Geng, L. Xu, K. Zhang, B. Liao, H. Huang, P. Ding, W. Miao, and Z. Jiang, "Causal inference," *Engineering*, vol. 6, no. 3, pp. 253–263, 2020.
[4] X. Guo, K. Yu, L. Liu, F. Cao, and J. Li, "Causal feature selection with dual correction," *IEEE Transactions on Neural Networks and Learning Systems*, 2022. [Online]. Available: https://doi.org/10.1109/TNNLS.2022.3178075
[5] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, prediction, and search*. MIT press, 2000.
[6] R. Cai, J. Qiao, Z. Zhang, and Z. Hao, "Self: structural equational likelihood framework for causal discovery," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[7] S. Nie, C. de Campos, and Q. Ji, "Learning bayesian networks with bounded tree-width via guided search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
[8] J. M. Mooij, S. Magliacane, and T. Claassen, "Joint causal inference from multiple contexts," *Journal of Machine Learning Research*, vol. 21, pp. 1–108, 2020.
[9] S. Yang, H. Wang, K. Yu, F. Cao, and X. Wu, "Towards efficient local causal structure learning," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1592–1609, 2021.

[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[12] D. Colombo, M. H. Maathuis *et al.*, “Order-independent constraint-based causal structure learning.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3741–3782, 2014.

[13] M. J. Vowels, N. C. Camgoz, and R. Bowden, “Dya like dags? a survey on structure learning and causal discovery,” *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.

[14] D. M. Chickering, “Optimal structure identification with greedy search,” *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 507–554, 2002.

[15] T. Gao, K. Fadnis, and M. Campbell, “Local-to-global bayesian network structure learning,” in *Proceedings of International Conference on Machine Learning*. PMLR, 2017, pp. 1193–1202.

[16] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, “The max-min hill-climbing bayesian network structure learning algorithm,” *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.

[17] X. Guo, Y. Wang, X. Huang, S. Yang, and K. Yu, “Bootstrap-based causal structure learning,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 656–665.

[18] X. Guo, K. Yu, L. Liu, P. Li, and J. Li, “Adaptive skeleton construction for accurate DAG learning,” *IEEE Transactions on Knowledge and Data Engineering*, 2023. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3265015

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

[20] I. Ng, A. Ghassami, and K. Zhang, “On the role of sparsity and dag constraints for learning linear DAGs,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 943–17 954, 2020.

[21] Y. Yu, T. Gao, N. Yin, and Q. Ji, “DAGs with no curl: An efficient dag structure learning approach,” in *Proceedings of International Conference on Machine Learning*. PMLR, 2021, pp. 12 156–12 166.

[22] Y. Yu, J. Chen, T. Gao, and M. Yu, “Dag-gnn: DAG structure learning with graph neural networks,” in *Proceedings of International Conference on Machine Learning*. PMLR, 2019, pp. 7154–7163.

[23] S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien, “Gradient-based neural DAG learning,” in *International Conference on Learning Representations*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=rklbKA4YDS

[24] X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. Xing, “Learning sparse nonparametric dags,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 3414–3425.

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

[26] C. Glymour, K. Zhang, and P. Spirtes, “Review of causal discovery methods based on graphical models,” *Frontiers in Genetics*, vol. 10, p. 524, 2019.

[27] B. Huang, K. Zhang, M. Gong, and C. Glymour, “Causal discovery from multiple data sets with non-identical variable sets,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 153–10 161.

[28] S. Triantafillou and I. Tsamardinos, “Constraint-based causal discovery from multiple interventions over overlapping variable sets,” *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 2147–2205, 2015.

[29] R. Xiong, A. Koenecke, M. Powell, Z. Shen, J. T. Vogelstein, and S. Athey, “Federated causal inference in heterogeneous observational data,” *arXiv preprint arXiv:2107.11732*, 2021.

[30] E. Gao, J. Chen, L. Shen, T. Liu, M. Gong, and H. Bondell, “FedDAG: Federated DAG structure learning,” *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=MzWgBjZ6Le

[31] I. Ng and K. Zhang, “Towards federated bayesian network structure learning with continuous optimization,” in *Proceedings of International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 8095–8111.

[32] C. Meek, “Causal inference and causal explanation with background knowledge,” in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 403–410.

[33] D. Kalainathan, O. Goudet, and R. Dutta, “Causal discovery toolbox: Uncovering causal relationships in python,” *Journal of Machine Learning Research*, vol. 21, pp. 37:1–37:5, 2020.

[34] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf, “Kernel-based conditional independence test and application in causal discovery,” *arXiv preprint arXiv:1202.3775*, 2012.

[35] K. Sachs, O. Perez, D. Pe’er, D. A. Lauffenburger, and G. P. Nolan, “Causal protein-signaling networks derived from multiparameter single-cell data,” *Science*, vol. 308, no. 5721, pp. 523–529, 2005.

[36] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[37] M. H. Maathuis, M. Kalisch, and P. Bühlmann, “Estimating high-dimensional intervention effects from observational data,” *The Annals of Statistics*, vol. 37, no. 6A, pp. 3133–3164, 2009.

Jianli Huang received the B.S. degree in computer science from Jiangsu University, Zhenjiang, China, in 2021. She is currently pursuing the masters degree with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei.

Her current research interests focus on causal discovery and federated learning.

Xianjie Guo received the B.S. degree from Anhui Normal University, Wuhu, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China.

His current research interests include causal discovery and federated learning.

Kui Yu (Member, IEEE) received the Ph.D. degree in computer science from the Hefei University of Technology, Hefei, China, in 2013.

From 2013 to 2018, he was a research fellow of computer science with the University of South Australia, Adelaide, Australia and Simon Fraser University, Burnaby, Canada. He is a full Professor with the School of Computer Science and Information Engineering, Hefei University of Technology. His main research interests include causal discovery and machine learning.

Fuyuan Cao received the M.S. and Ph.D. degrees in computer science from Shanxi University, Taiyuan, China, in 2004 and 2010, respectively. He is currently a professor with the school of computer and information technology, Shanxi University, China.

His current research interests include machine learning and clustering analysis.

Jiye Liang (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Xian Jiaotong University, Xian, China, in 1990 and 2001, respectively. He is currently a professor with the School of Computer and Information Technology, Shanxi University, Taiyuan, China, where he is also the director of the Key Laboratory of Computational Intelligence and Chinese Information Processing of the Ministry of Education.

He has authored more than 170 journal papers in his research fields. His current research interests include computational intelligence, granular computing, data mining, and knowledge discovery.