Privacy-Preserving Causal Structure Inference in Probability Temporal Models

Bingyi Shi[1]
[1]School of Statistics and Data Science, Nankai University
m13818413106@gmail.com

Abstract

Past methodologies for learning the structure of Probability Temporal Networks (PTNs) have depended on centralized computation frameworks that require the complete aggregation of data. However, many real-world applications now involve federated temporal data managed by separate entities (e.g., IoT networks, industrial partners) that must collaboratively develop models while upholding strict data privacy standards. A key technical challenge arises from the inherent heterogeneity in client-specific time series distributions among decentralized participants. In this study, we introduce a federated optimization framework for estimating the structure of the networks from horizontally partitioned homogeneous sequences. We then extend this framework to address scenarios with distributional divergence by incorporating proximity-regulated regularization within client-adaptive learning protocols. The proposed `fPTN` and `cfPTN` frameworks utilize continuous optimization methods with gradient synchronization, ensuring confidentiality by exclusively communicating parameter updates during federation cycles. Controlled experiments and real-world case studies demonstrate that these frameworks outperform state-of-the-art methods, especially in multi-client settings characterized by a large number of participants and limited localized observations.

1 Introduction

Structure learning for **Probability Temporal Networks** (PTNs) is a fundamental approach in causal discovery and the analysis of temporal representations. As probabilistic graphical models, PTNs—also known as Probability Temporal Networks—play a critical role in uncovering inter-timeslice dependencies and evolving conditional independencies, thereby serving as essential tools for dynamic causal inference. Their practical applications extend to diverse areas, including decoding genetic regulatory dynamics [Lemoine *et al.*, 2021], cybersecurity threat profiling [Chockalingam *et al.*, 2017], industrial process optimization [Sun *et al.*, 2020], and human activity recognition [Zeng *et al.*, 2016].

Traditional methods for inducing PTNs typically rely on centralized frameworks that require complete data integration—a practice that is becoming increasingly unsuitable in today's distributed data environments, which are often driven by pervasive IoT infrastructures [Madakam *et al.*, 2015]. Modern observational datasets are commonly fragmented among independent entities (e.g., edge devices, corporate stakeholders, clinical networks) that operate under strict data sovereignty constraints. In many cases, individual clients suffer from sample scarcity, making independent PTN reconstruction infeasible and necessitating collaborative, privacy-preserving federated approaches. For example, in cross-institutional medical research, multiple hospitals may seek to jointly develop a PTN to identify symptom interdependencies, yet directly exchanging patient data would violate privacy regulations [hip, 2003]. Moreover, the challenge is compounded by systemic data divergence: clients may exhibit either distributional homogeneity or heterogeneous configurations, which include both domain-specific distribution mismatches and fundamental feature space differences (e.g., non-overlapping measurement variables). These discrepancies create significant obstacles for federated model convergence, underscoring the need for privacy-aware techniques that can reconcile inconsistent observational patterns across decentralized temporal repositories.

A robust privacy-preserving learning paradigm is therefore indispensable. Federated learning (FL) [McMahan *et al.*, 2016] has emerged as a transformative framework for decentralized model development under data sovereignty constraints. By confining data exchange to parametric updates (such as gradients or intermediate model weights) rather than raw data, FL enables collaborative efforts across multiple clients in settings such as IoT ecosystems [Kontar *et al.*, 2021], mechanical condition monitoring [Zhang *et al.*, 2024], and recommender systems [Chai *et al.*, 2020], as detailed in [Li *et al.*, 2020]. Moreover, Customized Federated Learning (PFL) further extends FL to accommodate non-IID environments by blending client-specific adaptation with collective knowledge transfer. Current PFL strategies encompass: 1) proximal regularization [Sahu *et al.*, 2018] to balance local and global model alignment; 2) distribution-aware clustering [Muhammad *et al.*, 2020] to group clients based on

*1

latent similarities; and 3) meta-learning architectures [Jiang *et al.*, 2019] that enable swift local adaptation. These innovations significantly improve model performance in heterogeneous federated regimes while rigorously maintaining data confidentiality.

A persistent challenge lies in reconciling score-based Temporal Probability Network (PTN) structure learning—traditionally dependent on discrete optimization—with the continuous optimization foundation inherent to federated learning. Seminal works by [Zheng *et al.*, 2018] and [Pamfil *et al.*, 2020] address this gap by implementing algebraic acyclicity constraints, thereby recasting PTN induction as a continuous optimization problem. Nonetheless, adapting prominent federated algorithms such as FedAvg [McMahan *et al.*, 2016] and Ditto [Li *et al.*, 2021] remains nontrivial, primarily due to the stringent structural acyclicity demands of PTNs. The interdependence of edge constraints across clients' temporal Belief Network fragments introduces unique synchronization challenges that are not encountered in conventional federated optimization tasks.

Contributions This work advances federated causal discovery through the following methodological and empirical contributions:

- **Federated PTN Learning Frameworks:**
  - `FPTN`: A **federated continuous optimization** framework for learning Temporal Probability Network structures across distributionally homogeneous clients, employing **ADMM (Alternating Direction Method of Multipliers)** to synchronize temporal dependencies without raw data exchange.
  - `CFPTN`: A **Customized extension** that incorporates proximal regularization to manage cross-client domain shifts in heterogeneous settings, thereby enabling client-specific causal discovery while preserving overall structural coherence.
- **Constraint-Agnostic Causal Discovery:** Both frameworks autonomously infer PTN topologies from horizontally fragmented temporal datasets, supporting **arbitrary node in-degrees** without presupposing graph sparsity.
- **Systematic Validation:** Comprehensive evaluations on synthetic and high-dimensional real-world datasets (including industrial sensors and biomedical signals) demonstrate:
  - **Scalability** to 100+ clients with variable local sample sizes,
  - **Structural fidelity** that surpasses centralized baselines in non-IID federated regimes,
  - **Computational efficiency** under constrained communication settings.
- **Foundational Advance:** To our knowledge, this represents the **first methodological framework** for federated PTN learning that explicitly tackles **cross-client distributional heterogeneity** in temporal causal discovery—an underexplored challenge in decentralized time-series analysis.

2 Related Work

Probability Temporal Networks (PTNs) extend static Belief Networks to capture temporal causal dependencies, modeling inter-timeslice conditional independencies. Structure learning, which involves inferring directed edges between temporally ordered variables, and parameter learning, which estimates transition probabilities, are essential components of PTN induction. [Murphy, 2002] later adapted expectation-maximization (EM) for temporal parameter estimation, combining static Bayesian methods with Markovian state transitions.

Recent advancements leverage continuous optimization: [Pamfil *et al.*, 2020] reformulated PTN structure learning as a constrained continuous problem using algebraic acyclicity certificates, circumventing the complexity of combinatorial search. In parallel, [Tank *et al.*, 2022] integrated neural network architectures into Granger causal frameworks, enabling the capture of nonlinear dependencies through autoencoder-driven PTNs. Applications in bioinformatics, as demonstrated by [Yu *et al.*, 2004], have further validated PTNs for inferring gene regulatory dynamics from siloed omics datasets.

However, these approaches assume centralized data aggregation, posing privacy concerns and scalability limitations, especially in sensitive fields such as healthcare [Yu *et al.*, 2004]. Federated learning addresses these challenges by enabling distributed, privacy-preserving collaboration, aligning with the growing need for cross-silo data sovereignty. Despite this, federated PTN learning remains unexplored, and no existing framework addresses the following issues:

- **Decentralized Temporal Acyclicity**: Achieving synchronization of edge constraints across clients without centralized coordination.
- **Causal Heterogeneity**: Reconciling client-specific dependencies within a unified PTN structure.
- **Communication Overhead**: Striking a balance between model fidelity and efficient network optimization.

Although federated learning of Belief Networks (BNs) has gained attention, methods for learning Probability Temporal Networks (PTNs) are still notably absent. Early distributed BN approaches [Gou *et al.*, 2007] relied on post hoc aggregation of locally inferred graphs, either through conditional independence reconciliation or majority voting. These methods overlooked iterative collaborative optimization, often leading to suboptimal topologies due to limited information exchange. Notably, [Ng and Zhang, 2022] introduced a continuous optimization framework using ADMM to jointly infer DAG structures across clients—an important step forward from heuristic aggregation. However, no work yet addresses the temporal interdependencies inherent in PTNs, where both intra- and inter-slice edge constraints must be globally synchronized within the context of decentralized data governance.

3 Problem Formulation

Consider a decentralized learning framework with K distributed agents, indexed by $k \in \{1, \ldots, K\}$. Each agent holds a locally stored dataset consisting of M distinct realizations of a weakly stationary multivariate temporal process. Formally, the repository of the k-th agent is given by:

$$\mathcal{D}^k = \{x_{m,t}^k\}_{t=0}^T, \quad x_{m,t}^k \in \mathbb{R}^d, \quad m \in \{1, \dots, M\},$$

where d represents the dimensionality of the observed variables. Direct transmission of raw sequences $\{x_{m,t}^k\}$ between agents is prohibited due to data sovereignty protocols. For simplicity, we omit the realization index m in the following notation (for a detailed treatment, refer to Suppl. §A.1), and focus on a representative trajectory $\{x_t^k\}_{t=0}^T$.

The temporal dynamics for each agent k are modeled using a Structural Vector Autoregressive (SVAR) system of order p, expressed as:

$$(x_t^k)^\top = (x_t^k)^\top W_k + \sum_{i=1}^{p} (x_{t-i}^k)^\top A_{k_i} + (u_t^k)^\top, \quad (1)$$

with the following specifications:

1. Temporal indexing $t \in \{p, p+1, \dots, T\}$, where p defines the system's lagged memory depth.
2. Innovation term $u_t^k \sim \mathcal{N}(0, I)$, exhibiting spatiotemporal independence.
3. Intra-temporal adjacency matrix $W_k \in \mathbb{R}^{d \times d}$, structured as a directed acyclic graph (DAG) to encode contemporaneous causal relationships.
4. Inter-temporal coupling matrices $\{A_{k_i}\}_{i=1}^p \in \mathbb{R}^{d \times d}$, capturing lagged multivariate dependencies across successive timesteps.

The parameter matrices W_k and A_{k_i} are subject to two operational configurations:

1. **Homogeneous Configuration**: Global parameter sharing, where $W_k = W$ and $A_{k_i} = A_i$ for all k.

2. **Heterogeneous Configuration**: Agent-specific adaptation, where $\{W_k, A_{k_i}\}$ are distinct for each agent k.

A uniform autoregressive order p is assumed across all K agents. In the homogeneous setup, the system-wide DAG W and lagged dependencies A_i remain consistent, while in the heterogeneous configuration, individual agents are allowed to have unique causal structures. Each agent k processes $n_k = T + 1 - p$ temporal observations.

Federated Inference Objective: Given the distributed temporal dataset $X = \bigcup_{k=1}^K \mathcal{D}^k$, the goal is to estimate either: - Homogeneous parameters $W, \{A_i\}_{i=1}^p$, or - Heterogeneous parameters $\{W_k, \{A_{k_i}\}_{i=1}^p\}_{k=1}^K$,

while adhering to privacy constraints that restrict direct data sharing. Collaboration is achieved through the secure exchange of parameter estimates or topological metadata, rather than raw data sequences. The collective sample size is the sum of individual agent sample sizes, given by $n = \sum_{k=1}^K n_k$.

Partial Participation Protocol: To accommodate resource heterogeneity, each communication cycle engages a subset $j \leq K$ of agents. Inactive agents preserve local parameter states without affecting: - Homogeneous parameters: Remain unchanged during inactivity. - Heterogeneous parameters: Require synchronized updates only when active.

This protocol ensures elastic scalability while maintaining continuity in federated optimization.

4 Methodology

We introduce two federated paradigms: Federated Probability Temporal Networks (fPTN) and their Customized variant (cfPTN). Figure 1 illustrates their structural distinction in a tripartite agent system modeling $d = 3$ variables with $p = 2$.

4.1 Decentralized Probability Temporal Networks Formulation

This section presents the framework for learning Probability Temporal Networks (PTNs) in decentralized, time-dependent processes, extending the continuous optimization method introduced by [Pamfil *et al.*, 2020]. The goal is to adapt structural learning for a federated environment, formulating it as a constrained optimization problem with added sparsity regularization. The composite objective function we optimize combines three essential components:

1. **Temporal Autoregressive Loss**: This term quantifies the model's adherence to the temporal structure in the data, following the SVAR model.

2. **Sparsity Regularization**: Represented by ℓ_1-norms, this term encourages a sparse solution by penalizing excessive connections.

3. **Acyclicity Constraint**: This ensures the interactions between variables within the same time slice adhere to a directed acyclic graph (DAG).

The optimization problem is formally expressed as:

$$\begin{aligned} \min_{W,A} \quad & \ell(W, A) + \lambda_W \|W\|_1 + \lambda_A \|A\|_1, \\ \text{subject to} \quad & \text{DAG}(W), \\ & \ell(W, A) = \frac{1}{2n} \left\| X_t - X_t W - X_{[t-p:t-1]} A \right\|_F^2. \end{aligned}$$

where $X_{[t-p:t-1]} = [X_{t-1} \mid \cdots \mid X_{t-p}]$ is a matrix that concatenates the p-lagged observations from the dataset, with each lagged term $X_{t-i} \in \mathbb{R}^{n \times d}$. The matrix A, which contains the inter-slice relationships, is vertically concatenated from the transposed blocks corresponding to each lag, as shown:

$$A = \begin{bmatrix} A_1^\top \\ \vdots \\ A_p^\top \end{bmatrix} \in \mathbb{R}^{pd \times d}.$$

Here, $X_{[t-p:t-1]} \in \mathbb{R}^{n \times pd}$ captures the dependencies between variables across multiple time lags, and A maps those lagged dependencies to their corresponding contemporaneous interactions.

Acyclicity Enforcement: To ensure that the graph structure remains acyclic, we use the spectral condition defined in [Zheng *et al.*, 2018], expressed as:

$$h(W) = \text{tr}\left(e^{W \odot W}\right) - d = 0,$$

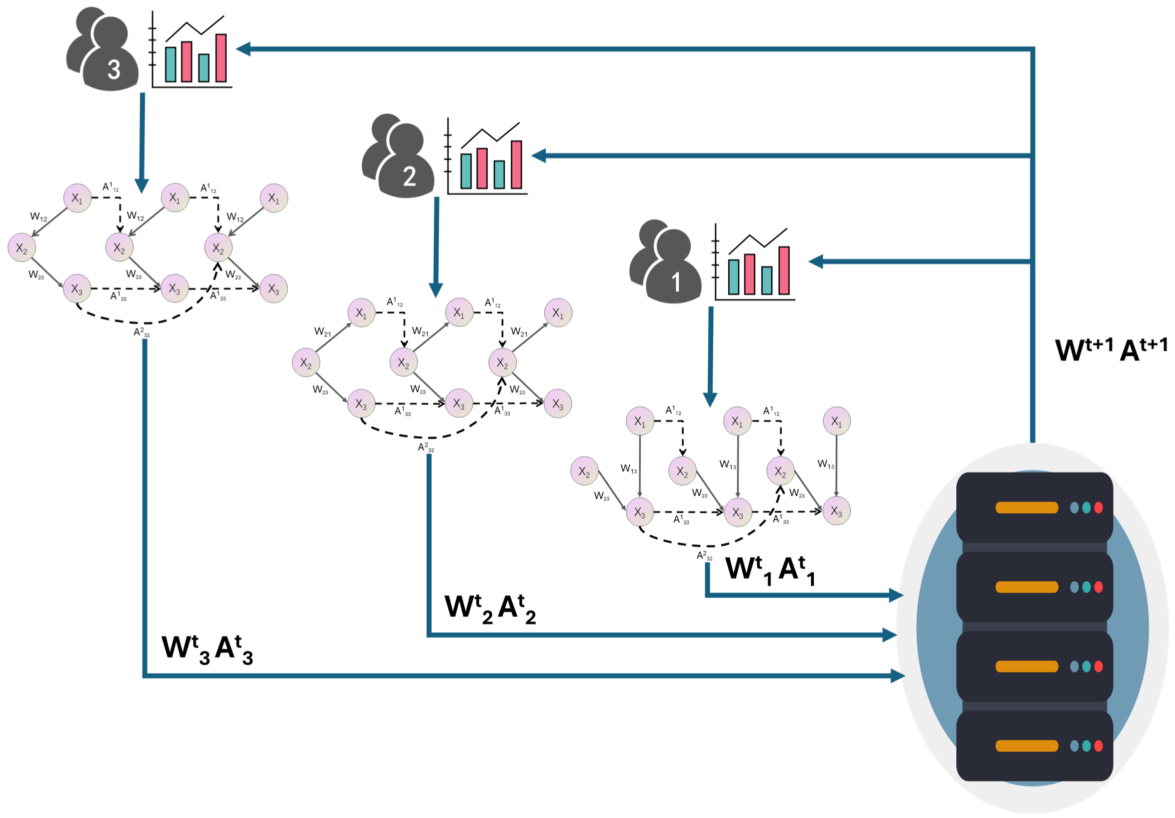

Figure 1: Architectural comparison between fPTN and cfPTN frameworks. fPTN enforces complete parameter homogeneity ($W_1 = W_2 = W_3$, $A_{k_i} = A_i\,\forall k$), while cfPTN accommodates agent-specific DAGs and temporal coupling matrices ($W_k \neq W_{k'}$, $A_{k_i} \neq A_{k'_i}$) while maintaining collaborative learning.

where $\odot$ denotes element-wise multiplication. This condition guarantees that the learned graph is a DAG, and the resulting penalty is differentiable, allowing gradient-based optimization methods to be applied. Thus, the original combinatorial DAG constraint is transformed into a differentiable penalty term, streamlining the optimization process.

In the federated learning context, where data encapsulation prevents the direct application of traditional optimization methods, we use the **Alternating Direction Method of Multipliers (ADMM)** [Boyd *et al.*, 2011], a powerful optimization technique known for breaking down complex problems into smaller, more solvable subproblems. ADMM is particularly effective for large-scale optimization involving intricate constraints. In this framework, we decompose the constrained problem in equation (4.1) into a series of subproblems, applying an iterative message-passing approach to converge toward the optimal solution. The efficiency of ADMM is particularly evident when the subproblems have closed-form solutions, which we derive for the first subproblem.

To align the problem with the ADMM framework, we introduce local variables $B_1, \ldots, B_K \in \mathbb{R}^{d\times d}$ for intra-slice matrices and $D_1, \ldots, D_K \in \mathbb{R}^{pd\times d}$ for inter-slice matrices. Additionally, global variables $W \in \mathbb{R}^{d\times d}$ and $A \in \mathbb{R}^{pd\times d}$ are shared across clients and represent a reformulated version of the problem:

$$\begin{aligned}\min_{B_k, D_k, W, A} \quad & \sum_{k=1}^{K} \ell_k(B_k, D_k) + \lambda_W \|W\|_1 + \lambda_A \|A\|_1 \\ \text{subject to} \quad & h(W) = 0, \\ & B_k = W, \quad k = 1, \ldots, K, \\ & D_k = A, \quad k = 1, \ldots, K.\end{aligned}$$

In this formulation, the local variables B_k and D_k represent the model parameters specific to each client, while the global variables W and A ensure consistency across the clients. These constraints ensure that all clients share the same structure, enforcing consistency in the learned model parameters. The setup is analogous to the global variable consensus framework often used in ADMM. The ADMM method uses dual decomposition and the augmented Lagrangian method, facilitating the separation of the objective function into subproblems, which can then be solved independently and in parallel.

To transform the constrained problem into a sequence of unconstrained subproblems, we apply the **augmented Lagrangian method**, which can be written as:

$$
\begin{aligned}
&\mathcal{L}\Big(\{B_k, D_k\}_{k=1}^K, W, A, \alpha, \{\beta_k, \gamma_k\}_{k=1}^K; \rho_1, \rho_2\Big) = \\
&\quad \sum_{k=1}^K \Bigg[\ell_k(B_k, D_k) + \text{tr}\left(\beta_k^\top (B_k - W)\right) + \frac{\rho_2}{2}\|B_k - W\|_F^2 \\
&\qquad + \text{tr}\left(\gamma_k^\top (D_k - A)\right) + \frac{\rho_2}{2}\|D_k - A\|_F^2 \Bigg] \\
&\qquad + \lambda_W \|W\|_1 + \lambda_A \|A\|_1 + \alpha h(W) + \frac{\rho_1}{2} h(W)^2,
\end{aligned}
$$

where $\beta_k \in \mathbb{R}^{d\times d}$, $\gamma_k \in \mathbb{R}^{pd\times d}$, and α are estimates of the Lagrange multipliers, while ρ_1 and ρ_2 are penalty coefficients. The Frobenius norm is denoted as $\|\cdot\|_F$.

The update rules for the ADMM iterations are as follows:

Local Updates for B_k and D_k

$$
\begin{aligned}
&(B_k^{(t+1)}, D_k^{(t+1)}) = \\
&\quad \arg\min_{B_k, D_k} \Bigg[\ell_k(B_k, D_k) + \text{tr}\left(\beta_k^{(t)\top}(B_k - W^{(t)})\right) \\
&\qquad + \frac{\rho_2^{(t)}}{2}\|B_k - W^{(t)}\|_F^2 + \text{tr}\left(\gamma_k^{(t)\top}(D_k - A^{(t)})\right) \\
&\qquad + \frac{\rho_2^{(t)}}{2}\|D_k - A^{(t)}\|_F^2 \Bigg].
\end{aligned} \tag{2}
$$

Global Updates for W and A

$$
\begin{aligned}
&(W^{(t+1)}, A^{(t+1)}) \\
&= \arg\min_{W,A} \Bigg[\alpha^{(t)} h(W) + \frac{\rho_1^{(t)}}{2} h(W)^2 + \lambda_W \|W\|_1 + \lambda_A \|A\|_1 \\
&+ \sum_{k=1}^K \left(\text{tr}\left(\beta_k^{(t)\top}(B_k^{(t+1)} - W)\right) + \frac{\rho_2^{(t)}}{2}\|B_k^{(t+1)} - W\|_F^2 \right) \\
&+ \sum_{k=1}^K \left(\text{tr}\left(\gamma_k^{(t)\top}(D_k^{(t+1)} - A)\right) + \frac{\rho_2^{(t)}}{2}\|D_k^{(t+1)} - A\|_F^2 \right) \Bigg].
\end{aligned} \tag{3}
$$

Updating the Dual Variables

$$
\begin{aligned}
\beta_k^{(t+1)} &= \beta_k^{(t)} + \rho_2^{(t)}(B_k^{(t+1)} - W^{(t+1)}), \\
\gamma_k^{(t+1)} &= \gamma_k^{(t)} + \rho_2^{(t)}(D_k^{(t+1)} - A^{(t+1)}), \\
\alpha^{(t+1)} &= \alpha^{(t)} + \rho_1^{(t)} h(W^{(t+1)}), \\
\rho_1^{(t+1)} &= \phi_1 \rho_1^{(t)}, \\
\rho_2^{(t+1)} &= \phi_2 \rho_2^{(t)},
\end{aligned} \tag{4}
$$

where $\phi_1, \phi_2 \in \mathbb{R}$ are hyperparameters that determine how quickly the coefficients ρ_1 and ρ_2 are updated. As mentioned earlier, ADMM excels when the optimization subproblems have closed-form solutions. The subproblem in equation (2) is a standard proximal minimization problem, widely studied in numerical optimization [Combettes and Pesquet, 2011; Parikh and Boyd, 2014a]. For clarity, we define the following matrices:

$$
\begin{aligned}
S &= \frac{1}{n_k} X_t^{k\top} X_t^k, \\
M &= \frac{1}{n_k} X_t^{k\top} X_{(t-p):(t-1)}^k, \\
N &= \frac{1}{n_k} X_{(t-p):(t-1)}^{k\top} X_{(t-p):(t-1)}^k, \\
P &= S + \rho_2^{(t)} I, \\
Q &= N + \rho_2^{(t)} I,
\end{aligned}
$$

and the vectors:

$$
\begin{aligned}
b_1 &= S - \beta_k^{(t)} + \rho_2^{(t)} W^{(t)}, \\
b_2 &= M^\top - \gamma_k^{(t)} + \rho_2^{(t)} A^{(t)}.
\end{aligned}
$$

By calculating the gradient, we obtain the following closed-form solutions:

$$
B_k^{(t+1)} = \left(P - MQ^{-1}M^\top\right)^{-1}\left(b_1 - MQ^{-1}b_2\right).
$$

$$
D_k^{(t+1)} = \left(Q - M^\top P^{-1}M\right)^{-1}\left(b_2 - M^\top P^{-1}b_1\right).
$$

For a more detailed explanation, the procedure is outlined in the supplementary material §A.4. The presence of the acyclicity term $h(W)$ prevents a closed-form solution for the problem in equation (3). Consequently, this optimization problem can be solved using first-order methods, such as gradient descent, or second-order methods, such as L-BFGS [Byrd *et al.*, 2003]. In this approach, we represent W as $W_+ - W_-$ (and similarly for A) to manage the ℓ_1 penalty term, and use the L-BFGS method to solve the optimization. The entire procedure is summarized in Algorithm 1.

Algorithm 1 Federated Temporal Probability Network Learning

```
Require:  • Initial parameters: $\rho_1, \rho_2, \alpha^{(1)}, \{\beta_k^{(1)}, \gamma_k^{(1)}\}_{k=1}^K$
          • Multiplicative factors: $\phi_1, \phi_2 > 1$
          • Initial points: $W^{(1)}, A^{(1)}$
Ensure: Learned model parameters $W, A$
1: for round $t = 1, 2, \ldots$ do
2:     Clients (parallel):
           – Solve local problem (2)
3:     Server:
           – Collect $\{B_k^{(t+1)}, D_k^{(t+1)}\}_{k=1}^K$ from clients
           – Solve global problem (3)
           – Broadcast $W^{(t+1)}, A^{(t+1)}$ to clients
4:     Parameter Updates:
           – Server: Update $\alpha^{(t+1)}, \rho_1^{(t+1)}, \rho_2^{(t+1)}$ via (4)
           – Clients: Update $\beta_k^{(t+1)}, \gamma_k^{(t+1)}$ via (4)
5: end for
```

4.2 Customized Federated Temporal Probability Network Learning

In this section, we introduce the Customized Federated PTN Learning (`cfPTN`) method, which is designed specifically for heterogeneous time series data. This approach builds upon the proximal operator, employed to reformulate Problem (4.1), as described by [Parikh and Boyd, 2014b]. The formal definition of the proximal operator is as follows:

Definition 1 (Proximal Operator)**.** *Let $\mathcal{X}$ be a vector space with norm $\|.\|_{\mathcal{X}}$. The proximal operator is defined as:*

$$prox_{\mu f}(X) = \arg\min_{Z} \mu||Z - X||_{\mathcal{X}}^2 + f(Z)$$

To define the objective function for `cfPTN`, we express it as:

$$f_k(W_k, A_k) = \ell_k(W_k, A_k) + \lambda_W \|W_k\|_1 + \lambda_A \|A_k\|_1 \quad (5)$$

for the k-th client. In the context of `cfPTN`, A_k is occasionally represented as $\left[A_{k_1}^\top \ \cdots \ A_{k_p}^\top\right]^\top$.

Next, following the method outlined by [Sahu *et al.*, 2018], we replace f_k with its proximal operator. The structure of the resulting optimization problem is formulated as:

$$\begin{aligned} F_k(W, A) &:= prox_{\mu f}(W, A) \\ &= \min_{W_k, A_k} \mu||W_k - W||_F^2 + \mu||A_k - A||_F^2 + f_k(W_k, A_k) \\ &= \min_{W_k, A_k} \ell(W_k, A_k) + \mu||W_k - W||_F^2 + \mu||A_k - A||_F^2 \\ &+ \lambda_W \|W_k\|_1 + \lambda_A \|A_k\|_1. \end{aligned} \quad (6)$$

The hyperparameter μ controls the trade-off between the global model (W, A) and the customized model (W_k, A_k). A larger value of μ helps clients with less reliable data by incorporating aggregated information from other clients, while a smaller μ emphasizes personalization for clients with substantial high-quality data. Thus, we define the following optimization problem:

$$\begin{aligned} &\min_{W, A} \sum_{k=1}^{K} F_k(W, A) \quad \text{subject to} \quad W \text{ is acyclic}, \\ &F_k(W, A) = \min_{W_k, A_k} \ell(W_k, A_k) + \mu||W_k - W||_F^2 \\ &\qquad + \mu||A_k - A||_F^2 + \lambda_W \|W_k\|_1 + \lambda_A \|A_k\|_1. \end{aligned} \quad (7)$$

Problem (7) represents a bi-level optimization problem, typically tackled using iterative first-order gradient methods. In these methods, the lower-level problem is solved first to approximate W_i, and this solution is then used in the upper-level problem to optimize W, repeating the process until convergence. While this approach is simple, it often yields suboptimal solutions and faces difficulties in fine-tuning hyperparameters, such as the learning rate. To address these challenges, we propose a relaxed formulation of Problem (7) to improve both efficiency and solution accuracy.

$$\begin{aligned} &\min_{\{W_k, A_k\}_{k=1}^K, W, A} \\ &\sum_{k=1}^{K} \ell_k(W_k, A_k) + \mu \left(||W_k - W||_F^2 + ||A_k - A||_F^2\right) \quad (8) \\ &\quad + \lambda_W \|W_k\|_1 + \lambda_A \|A_k\|_1 \\ &\text{subject to } h(W_k) = 0. \end{aligned}$$

The objective is to learn an optimal customized model (W_k, A_k) for each client k and an optimal global model (W, A) that jointly minimize Problem (8), ensuring a balance between personalization and global knowledge aggregation. Since ADMM is a primal-dual method known for its enhanced iteration stability and faster convergence compared to gradient-based approaches, we propose utilizing ADMM to solve Problem (8). To achieve this, we introduce auxiliary variables $\{\tilde{W}_k, \tilde{A}_k\}_{k=1}^K$, enabling us to reformulate Problem (8) into a separable structure by partitioning the variables into distinct blocks. This transformation results in a more efficient optimization framework, as outlined below:

$$\begin{aligned} &\min_{\{W_k, A_k, \tilde{W}_k, \tilde{A}_k\}_{k=1}^K, W, A} \\ &\sum_{k=1}^{K} \ell_k(W_k, A_k) + \mu \left(||W_k - \tilde{W}_k||_F^2 + ||A_k - \tilde{A}_k||_F^2\right) \\ &\quad + \lambda_W \|W_k\|_1 + \lambda_A \|A_k\|_1 \\ &\text{subject to } h(W_k) = 0, \\ &\quad \tilde{W}_k = W, \\ &\quad \tilde{A}_k = A, \quad k = 1, 2, \ldots, K. \end{aligned} \quad (9)$$

In this context, $\{\tilde{W}_k, \tilde{A}_k\}_{k=1}^K$ represents the local model of client k. Problem (9) is equivalent to Problem (8) in that both share the same optimal solutions. Since Problem (9) involves linear constraints and multiple block variables, we can solve it using the exact penalty method within the ADMM framework, which is well-suited for such optimization structures. Additionally, we assume $h(W_k) = 0$ for all clients, as we need to enforce the Directed Acyclic Graph (DAG) constraint across all clients. To further facilitate this, we apply the augmented Lagrangian method to convert the problem into a series of unconstrained subproblems. The corresponding augmented Lagrangian function is defined as follows:

$$\begin{aligned} &\mathcal{L}\Big(\{W_k, A_k, \tilde{W}_k, \tilde{A}_k\}_{k=1}^K,\, W,\, A,\, \{\beta_k, \gamma_k\}_{k=1}^K;\, \alpha,\, \rho_1,\, \rho_2,\, \mu\Big) = \\ &\sum_{k=1}^{K} \Big[\ell_k(W_k, A_k) \;+\; \mu\, \|W_k - \tilde{W}_k\|_F^2 \;+\; \mu\, \|A_k - \tilde{A}_k\|_F^2 \\ &\quad +\; \lambda_W\, \|W_k\|_1 \;+\; \lambda_A\, \|A_k\|_1 \\ &\quad +\; \mathrm{tr}\Big(\beta_k^\top(\tilde{W}_k - W)\Big) \;+\; \frac{\rho_2}{2}\, \|\tilde{W}_k - W\|_F^2 \\ &\quad +\; \mathrm{tr}\Big(\gamma_k^\top(\tilde{A}_k - A)\Big) \;+\; \frac{\rho_2}{2}\, \|\tilde{A}_k - A\|_F^2\Big] \\ &+\; \alpha\, h(W_k) \;+\; \frac{\rho_1}{2}\, \big[h(W_k)\big]^2. \end{aligned}$$

where $\{\beta_k\}_{k=1}^K \in \mathbb{R}^{d\times d}$, $\{\gamma_k\}_{k=1}^K \in \mathbb{R}^{pd\times d}$, and $\alpha \in \mathbb{R}$ are estimates of the Lagrange multipliers; ρ_1 and ρ_2 are penalty coefficients, and $\|\cdot\|_F$ denotes the Frobenius norm. It is important to note that we use a single pair of λ_W, λ_A for all clients, as an effective pair can generally be found for datasets with the same dimension, based on `fPTN` and previous studies [Pamfil *et al.*, 2020]. Due to the Customized nature of the problem, each W_k, A_k may have different dimensions or structural properties (e.g., connectivity). In this study, we consider cases where all clients share identical dimensions and connectivity. However, extending this framework to accommodate different dimensions or varying connectivity structures represents an interesting direction for future research, as discussed in § 7.

Next, we define the iterative update rules of ADMM. Starting with initial values for $\{W_k, A_k, \tilde{W}_k, \tilde{A}_k\}_{k=1}^K$, W, A, and the dual variables, the updates are as follows:

For W_k, A_k:

$$\begin{aligned}
&({W_k}^{(t+1)}, A_k^{(t+1)}) = \\
&\quad \arg\min_{W_k, A_k} \Big[\ell_k(W_k, A_k) + \mu\|W_k - \tilde{W}_k^{(t)}\|^2 \\
&\quad + \mu\|A_k - \tilde{A}_k^{(t)}\|^2 + \alpha^{(t)}\, h(W_k) + \frac{\rho_1^{(t)}}{2}\,[h(W_k)]^2 \\
&\quad + \lambda_W\|W_k\|_1 + \lambda_A\|A_k\|_1\Big].
\end{aligned} \tag{10}$$

For $\tilde{W}_k, \tilde{A}_k$:

$$\begin{aligned}
&({\tilde{W}_k}^{(t+1)}, \tilde{A}_k^{(t+1)}) = \\
&\quad \arg\min_{\tilde{W}_k, \tilde{A}_k} \Big[\mu\|W_k^{(t+1)} - \tilde{W}_k\|^2 \\
&\quad + \mu\|A_k^{(t+1)} - \tilde{A}_k\|^2 + \mathrm{tr}\big(\beta_k^{(t)\top}(\tilde{W}_k - W^{(t)})\big) \\
&\quad + \frac{\rho_2^{(t)}}{2}\|\tilde{W}_k - W^{(t)}\|^2 + \mathrm{tr}\big(\gamma_k^{(t)\top}(\tilde{A}_k - A^{(t)})\big) \\
&\quad + \frac{\rho_2^{(t)}}{2}\|\tilde{A}_k - A^{(t)}\|^2\Big].
\end{aligned} \tag{11}$$

For W, A:

$$\begin{aligned}
&(W^{(t+1)}, A^{(t+1)}) = \\
&\quad \arg\min_{W,A}\Big[\sum_{k=1}^K\Big(\mathrm{tr}\big(\beta_k^{(t)\top}(\tilde{W}_k^{(t+1)} - W)\big) \\
&\quad + \frac{\rho_2^{(t)}}{2}\|\tilde{W}_k^{(t+1)} - W\|^2 + \sum_{k=1}^K\Big(\mathrm{tr}\big(\gamma_k^{(t)\top}(\tilde{A}_k^{(t+1)} - A)\big) \\
&\quad + \frac{\rho_2^{(t)}}{2}\|\tilde{A}_k^{(t+1)} - A\|^2\Big)\Big].
\end{aligned} \tag{12}$$

For Dual Variables:

$$\begin{aligned}
\beta_k^{(t+1)} &= \beta_k^{(t)} + \rho_2^{(t)}(\tilde{W}_k^{(t+1)} - W^{(t+1)}), \\
\gamma_k^{(t+1)} &= \gamma_k^{(t)} + \rho_2^{(t)}(\tilde{A}_k^{(t+1)} - A^{(t+1)}), \\
\alpha^{(t+1)} &= \alpha^{(t)} + \rho_1^{(t)}\left(\frac{1}{K}\sum_{k=1}^K h(W_k^{(t+1)})\right), \\
\rho_1^{(t+1)} &= \phi_1\rho_1^{(t)}, \\
\rho_2^{(t+1)} &= \phi_2\rho_2^{(t)},
\end{aligned} \tag{13}$$

In the practical implementation of `cfPTN`, we use closed-form expressions for $\tilde{W}_k, \tilde{A}_k$ derived as follows:

$$\tilde{W}_k^{(t+1)} = \frac{2\mu\, W_k^{(t+1)} + \rho_2^{(t)}\, W^{(t)} - \beta_k^{(t)}}{2\mu + \rho_2^{(t)}}.$$

$$\tilde{A}_k^{(t+1)} = \frac{2\mu\, A_k^{(t+1)} + \rho_2^{(t)}\, A^{(t)} - \gamma_k^{(t)}}{2\mu + \rho_2^{(t)}}.$$

For a comprehensive overview of the procedure, we have summarized it in the supplementary material § A.5. In this study, we use the L-BFGS method to solve the optimization problem. The procedure for `cfPTN` is outlined in Algorithm 2.

Algorithm 2 Customized Federated cfPTN Learning

```
Require:  • Initial parameters: $\rho_1^{(1)}, \rho_2^{(1)}, \alpha^{(1)}, \{\beta_k^{(1)}, \gamma_k^{(1)}\}_{k=1}^K$
          • Scaling factors: $\phi_1, \phi_2 > 1$
          • Variables: $\{W_k, A_k, \tilde{W}_k, \tilde{A}_k\}_{k=1}^K, W, A$
          • Client subset per round: $j \le K$
Ensure: Global model parameters $W, A$
 1: for round $t = 1, 2, \dots$ do
 2:     Clients (parallel):
            – Solve (10) and (11)
            – Upload $\tilde{W}_k^{(t+1)}, \tilde{A}_k^{(t+1)}$ to server
 3:     Server:
            – Aggregate $\{\tilde{W}_k^{(t+1)}, \tilde{A}_k^{(t+1)}\}$
            – Solve (12) to update $W^{(t+1)}, A^{(t+1)}$
            – Broadcast $W^{(t+1)}, A^{(t+1)}$ to clients
 4:     Parameter Updates:
            – Server: $\alpha^{(t+1)} \leftarrow (13)$, $\rho_i^{(t+1)} \leftarrow \phi_i\rho_i^{(t)}$
            – Client $k$: $\beta_k^{(t+1)}, \gamma_k^{(t+1)} \leftarrow (13)$
 5: end for
```

5 Experiments

In this section, we first evaluate the performance of `fPTN` on simulated data generated from a linear SVAR structure [Gong *et al.*, 2023]. We then compare it with three baseline methods using three evaluation metrics to demonstrate the effectiveness of our proposed method. Figure 2 provides an illustrative example using homogeneous time series data. Subsequently, we examine the performance of `cfPTN` on heterogeneous time series data generated by a linear SVAR structure across different clients and assess its performance under partial client participation by uniformly sampling a subset of clients.

Benchmark Methods. We compare our federated approach, referred to as `fPTN` in Sec. 4.1, with three other methods. The first baseline, denoted as `Ave`, computes the average of the weighted adjacency matrices estimated by DYNOTEARS [Pamfil *et al.*, 2020] from each client's local dataset, followed by thresholding to determine the edges. The second baseline, referred to as `Best`, selects the best graph from among those estimated by each client based on the lowest Structural Hamming Distance (SHD) to the ground truth. Although this method is impractical in real-world scenarios (since it assumes access to the ground truth), it serves as a useful reference. Additionally, we consider applying

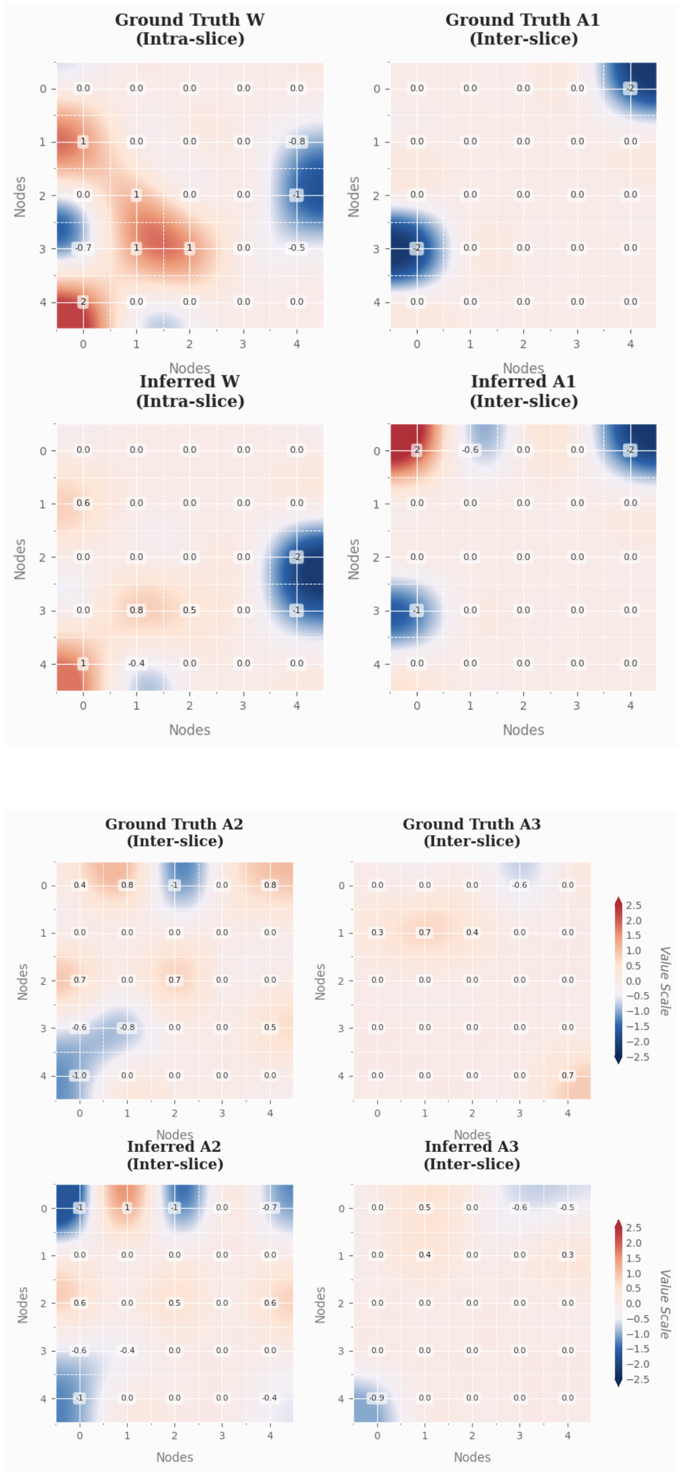

Figure 2: An example result using `fPTN` for Gaussian noise data with $n = 500$ samples, $d = 5$ variables, an autoregressive order $p = 3$, and $K = 10$ clients. All clients have the same W, A. We set the thresholds $\tau_w = \tau_a = 0.3$. Our algorithm recovers weights close to the ground truth.

DYNOTEARS to the combined dataset from all clients, denoted as `Alldata`. Note that the final graphs produced by `Ave` may contain cycles for W, and we do not apply any post-processing to remove them, as doing so could degrade performance. Since DYNOTEARS does not provide official source code, we reimplement it using `numpy` and `scipy` in a simplified, self-contained version with approximately one hundred lines of code. This implementation improves readability and reusability compared to existing versions available on GitHub. For the Customized federated approach, `cfPTN`, we compare its performance against `fPTN` on heterogeneous time series data. Since no existing methods are specifically designed for learning heterogeneous PTNs, we focus on evaluating the improvements of `cfPTN` over `fPTN` in a single example and demonstrate that `cfPTN` yields reasonable results.

Evaluation Metrics. To assess performance, we employ three key metrics: Structural Hamming Distance (SHD), True Positive Rate (TPR), and False Discovery Rate (FDR). SHD quantifies the disparity between the estimated graph and the true graph, accounting for missing edges, additional edges, and incorrectly oriented edges [Tsamardinos *et al.*, 2006]. A lower SHD signifies a closer match to the ground truth. The TPR (also referred to as sensitivity or recall) measures the fraction of true edges that are correctly identified. It is calculated as the ratio of true positives to the sum of true positives and false negatives [Glymour *et al.*, 2019]. A higher TPR reflects the model's ability to correctly identify more true edges. The FDR measures the proportion of false positives among all predicted edges. It is computed as the ratio of false positives to the total of false positives and true positives [Benjamini and Hochberg, 1995]. A lower FDR indicates a more reliable model, with fewer false edges being identified. Together, these metrics provide a well-rounded evaluation of the model's effectiveness in inferring graph structures. For heterogeneous time series data, where we have K distinct ground truth matrices W_k and A_{k_i}, we evaluate performance individually for each client. Thus, we report the mean SHD (mSHD), mean TPR (mTPR), and mean FDR (mFDR) across all clients, offering a comprehensive view of performance across the entire dataset.

5.1 Federated Probability Temporal Network Learning

Data Generation & Settings. For `fPTN`, we generate data based on the Structural Equation Model outlined in Eq. (**??**). The data generation process consists of four steps: (1) Constructing the weighted graphs G_W and G_A; (2) Generating data matrices X and Y consistent with these graphs; (3) Dividing the data among K clients into X_{client} and Y_{client}; (4) Applying all algorithms to the client data (X_{client} and Y_{client}, or X and Y) and evaluating their performance. The specifics of steps (1) and (2) can be found in §A.2. We introduce Gaussian noise with a standard deviation of 1. The intra-slice DAG follows an Erdős-Rényi (ER) graph with an average degree of 4, while the inter-slice DAG is also modeled as an ER graph with a mean out-degree of 1. Despite the sparsity of the graphs at higher d, they remain connected under our settings as per [Erdős and Rényi, 1960]. The exponential decay factor for the inter-slice weights is set to $\eta = 1.5$. We optimize the hyperparameters λ_w and λ_a by generating heatmaps to identify their optimal values, as documented in the Supplementary Materials (§A.3). We set $\phi_1 = 1.6$ and $\phi_2 = 1.1$ with initial values $\rho_1 = \rho_2 = 1$, and the initial Lagrange multipliers are set to zero. For DYNOTEARS (and consequently for `Ave` and `Best`), we follow the authors' recommended hyperparameters. To avoid confusion, we summarize the data dimensions used in `fPTN` as follows:

- For `fPTN`, X_{client} and Y_{client} have sizes $K \times n_k \times d$ and $K \times n_k \times pd$, respectively, where K is the number of clients, n_k is the sample size for the k-th client, d is the dimensionality, and p is the autoregressive order.

Experiment Settings. We consider two experimental scenarios. First, we keep the number of clients K fixed while increasing the number of variables d, ensuring the total sample size n remains constant. This allows us to examine how well the methods scale with increasing dimensionality. In the second scenario, we fix the total sample size $n \in \{256, 512\}$ and vary the number of clients K from 2 to 64. This setup evaluates the adaptability and robustness of the methods as the data is distributed among more clients.

Varying Number of Variables

In this section, we investigate PTN with $n = 5d$ samples distributed evenly across $K = 10$ clients for $d = 10, 20$, and $n = 6d$ for $d \in \{5, 15\}$. Note that setting $n = 5d$ or $6d$ ensures each client receives an integer number of samples. We generate datasets for each of these configurations, keeping in mind that each client typically has a limited number of samples, which presents a challenging scenario.

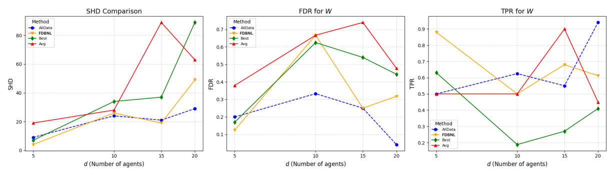

Figure 3: Structure learning results for W in a PTN with Gaussian noise for $d = 5, 10, 15, 20$ variables, an autoregressive order $p = 1$, and $K = 10$ clients. Each metric value represents the mean performance across 10 different simulated datasets.

Figure 3 presents the results for the inferred W. The corresponding results for A can be found in §A.8. Across all tested values of d, `fPTN` consistently achieves the lowest SHD compared to both `Ave` and `Best`, and even outperforms `Alldata` for $d = 5, 15$. This advantage may arise from the more refined hyperparameter tuning required by `Alldata`. In addition to the lower SHD, `fPTN` achieves a high TPR, comparable to `Alldata` and higher than `Best`, suggesting that `fPTN` correctly identifies most true edges. Furthermore, `fPTN` demonstrates a lower FDR than `Ave` and `Best`, highlighting its superior reliability in edge identification, even in this challenging setup.

Varying the Number of Clients

We next consider experiments where a constant total number of samples is partitioned among different numbers of clients. For $d \in \{10, 20\}$, we generate $n = 512$ samples with an autoregressive order $p = 1$. These samples are distributed uniformly among $K \in \{2, 4, 8, 16, 32, 64\}$ clients. The hyperparameters λ_w and λ_a for DYNOTEARS, `Ave`, and `Best` are set following the guidelines in [Pamfil *et al.*, 2020]. For `fPTN`, we search for the optimal regularization parameters within the range $[0.05, 0.5]$ (incremented by 0.05) by minimizing the SHD.

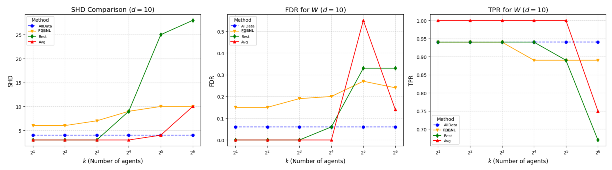

Figure 4: Structure learning outcomes for W in a PTN under Gaussian noise for $d = 10$ variables, with $p = 1$, and various client counts. A total of $n = 512$ samples is evenly divided among $K \in \{2, 4, 8, 16, 32, 64\}$. Each reported metric is averaged over 10 independent simulations.

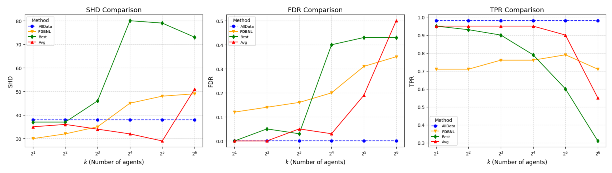

Figure 5: Structure learning outcomes for W in a PTN under Gaussian noise for $d = 20$ variables, with $p = 1$, and varying client counts. A total of $n = 512$ samples is evenly split among $K \in \{2, 4, 8, 16, 32, 64\}$. Each metric is computed as the mean over 10 independent simulations.

Figures 4 and 5 present the results for W with $d = 10$ and $d = 20$, respectively (see §A.8 for the corresponding A results). As the number of clients K increases, the TPRs of `Ave` and `Best` drop sharply, leading to elevated SHD values. Although the TPR of `fPTN` also decreases with larger K, it consistently remains higher than that of the competing methods. For instance, with $d = 20$ and $K = 64$, `fPTN` attains a TPR of 0.7, whereas `Ave` and `Best` only reach 0.5 and 0.3, respectively. This underscores the benefit of information sharing in the optimization process, which allows `fPTN` to accurately recover the PTN structure even when data is highly fragmented.

We further examine the scenario with $n = 256$ samples. The W-based outcomes for this setting are provided below, with A-based results available in §A.8.

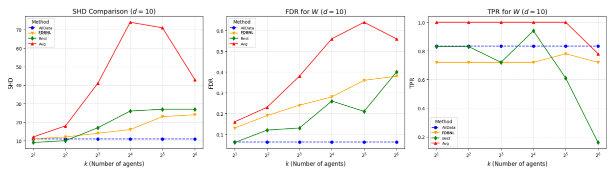
Figure 6: Structure learning outcomes for W in a PTN under Gaussian noise for $d = 10$ variables, with $p = 1$, and various client counts. Here, $n = 256$ samples are evenly distributed among $K \in \{2, 4, 8, 16, 32, 64\}$. Each reported metric is the average over 10 simulation runs.

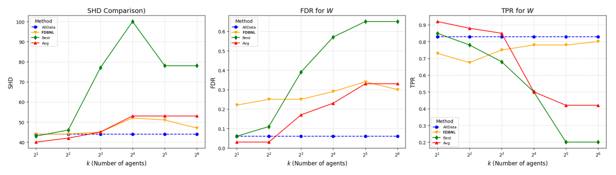
Figure 7: Structure learning outcomes for W in a PTN under Gaussian noise for $d = 20$ variables, with $p = 1$, and varying numbers of clients. A total of $n = 256$ samples is evenly partitioned among $K \in \{2, 4, 8, 16, 32, 64\}$. Each metric is averaged over 10 independent simulations.

From Figures 6 and 7, fPTN consistently achieves the lowest SHD in comparison with Best and Ave. In the case of $d = 20$, while the TPRs of Ave and Best continue to decline as K increases, fPTN maintains a higher TPR for all $K \geq 8$. In extremely distributed settings, such as $K = 64$ with $d = 20$ (where each client only has 4 samples), both Ave and Best fail to converge. Thus, we focus on comparing metrics at $K = 32$ for a fair evaluation. For $d = 10$, although Ave exhibits the highest TPR among all methods across all values of K, its FDR is considerably higher than those of the other approaches. This indicates that the thresholding strategy in Ave may lead to an excessive number of false positives. Nonetheless, as K increases, the TPR of fPTN for $d = 10$ remains comparable to that achieved by Alldata.

In some settings, when either K or d is low, alternative methods may surpass fPTN. For instance, with $d = 10$ and $n = 512$, both Ave and Best yield lower SHDs than fPTN when $K \leq 16$. This outcome is understandable since a smaller number of clients and lower dimensionality mean that each client has sufficient samples to reliably learn the structure on their own. However, as the problem complexity increases (for example, when $d = 20$), fPTN demonstrates superior performance by maintaining a lower FDR and efficiently aggregating distributed information. These observations underscore the significance of choosing the appropriate method and fine-tuning hyperparameters, and they suggest promising avenues for future work aimed at enhancing the efficiency of information exchange in PTN learning.

5.2 Customized Federated Probability Temporal Network Learning

In this section, we compare the performance of cfPTN with that of fPTN. For our experiment, we generate six pairs of (W, A) for every client and simulate 10 datasets with each k-th client receiving $n_k = 30$ samples. We then calculate the average mSHD, mTPR, and mFDR over these datasets after applying both cfPTN and fPTN.

As shown in Table 1, the mSHD values for both W and A under cfPTN are approximately half of those observed for fPTN. Additionally, the mTPR achieved by cfPTN is roughly double that of fPTN. Regarding mFDR, cfPTN exhibits a slightly lower value, with mFDR decreasing by about a factor of two. These results indicate that cfPTN markedly enhances performance across all evaluation metrics relative to fPTN. To the best of our knowledge, no existing technique can learn Probability Temporal Networks (PTNs) from heterogeneous time series data. In light of this gap, cfPTN offers a novel solution that significantly improves accuracy.

Metric		cfPTN	fPTN
mSHD	W	6.2	10.2
	A	5.4	12.7
mTPR	W	0.64	0.35
	A	0.51	0.24
mFDR	W	0.55	0.60
	A	0.19	0.33

Table 1: Performance comparison between cfPTN and fPTN with $d = 5$, $K = 6$, and $n_k = 30$ for each client. The expected degree per node is 4 (in both directions) with $p = 1$. Each value represents the average performance across 10 simulated datasets.

Data Generation & Settings. For cfPTN, data is generated according to the Structural Equation Model in Eq. (**??**) through the following steps: (1) constructing individual weighted graphs G_{W_k} and $G_{A_{k_i}}$ for each client; (2) producing the corresponding data matrices X_{client} and Y_{client}; and (3) applying cfPTN to assess performance. The procedures for steps (1) and (2) mirror those used in fPTN, except that dataset partitioning is not necessary. Gaussian noise with a standard deviation of 1 is added, and the exponential decay base for inter-slice weights is set to $\eta = 1.5$. Through experimentation, we found that setting $\lambda_W = \lambda_A = 0.1$ works well for $d \in \{5, 10\}$, whereas for $d \in \{15, 20\}$ a setting of $\lambda_W = \lambda_A = 0.01$ paired with $\mu = 0.1$ is effective. We choose $\phi_1 = 1.6$ and $\phi_2 = 1.1$ with initial values $\rho_1 = \rho_2 = 1$, and initialize all Lagrange multipliers to zero. The data dimensions are identical to those in the fPTN experiments, though each client now possesses distinct W_k and A_{k_i}.

Experiment Settings. We adopt two experimental scenarios analogous to those in the fPTN experiments. In the first scenario, we keep the number of clients K constant while increasing the number of variables d, with the total sample size n held fixed. This scenario tests the scalability of the methods as dimensionality increases. In the second scenario, we fix the total sample size at $n = 512$ and vary the number of clients K from 2 to 32, evaluating how well the methods adapt when data is distributed more thinly across clients.

Varying the Number of Variables

In this section, we examine the performance of learning W_k and A_{k_i} when each client receives $n_k = Kd$ samples, with d taking values from $\{5, 10, 15, 20\}$ across $K = 6$ clients. As in our earlier example, we first generate the matrices W_k and A_{k_1} for every client using $p = 1$. Then, for each client, we generate a dataset of size n_k. Additionally, we test our `cfPTN` approach under two distinct network connectivity scenarios: one representing low connectivity and the other high connectivity. In this study, we do not consider mixed connectivity scenarios, which we leave as a future research direction (see §7). For the low-connectivity setup, we set the expected degree of each node to be $\lfloor \frac{d}{2} \rfloor - \frac{d}{5}$, ensuring it is below half of the total variables. Conversely, in the high-connectivity scenario, we define the expected degree as $d - \frac{d}{5}$.

Metric	Var.	Dimension (d) 5	10	15	20
Low Connectivity					
mSHD	W	4.4	16.8	44.7	79.2
	A	2.8	10.9	29.6	32.2
mTPR	W	0.61	0.49	0.55	0.53
	A	0.81	0.53	0.31	0.31
mFDR	W	0.72	0.67	0.51	0.53
	A	0.35	0.47	0.23	0.34
High Connectivity					
mSHD	W	6.2	13.6	52.0	110.0
	A	5.4	8.3	41.0	31.0
mTPR	W	0.64	0.64	0.71	0.76
	A	0.51	0.52	0.31	0.27
mFDR	W	0.55	0.54	0.44	0.59
	A	0.19	0.21	0.36	0.19

Table 2: Performance metrics (mSHD, mTPR, mFDR) for low and high network connectivity across dimensions $d = 5, 10, 15, 20$. Values are averages over 10 simulations.

Tables 2 reveal that for the W matrices, the mSHD is considerably lower in the low-connectivity regime compared to the high-connectivity case. In contrast, the mSHD for A remains relatively consistent regardless of connectivity. With respect to mTPR, the high-connectivity setting yields significantly better results than the low-connectivity one. Notably, while the mTPR for W in the high-connectivity scenario stays steady as d increases, the mTPR for A declines with increasing d. Moreover, in low connectivity, A's mTPR reaches 0.81, markedly higher than the 0.51 observed in high connectivity—an expected outcome since detecting relationships in sparser networks is inherently more challenging. Regarding mFDR for A, the high-connectivity case holds steady at around 0.2 for $d \in \{5, 10, 20\}$, while in the low-connectivity case, it rises to roughly 0.3. For W, mFDR in the high-connectivity scenario is around 0.54, which is notably lower than the corresponding values for low connectivity when $d \in \{5, 10\}$. Overall, the high-connectivity configuration demonstrates superior mFDR performance.

We further investigate the case of partial participation for $d = 10$. Given that there are only six clients in total, we vary the number of participating clients j over the values $1, 2, 3, 4, 5$, while all other settings remain unchanged. As shown in Table 3, performance metrics remain consistent regardless of how many clients participate. Specifically, the mSHD for W is in the range of 13 to 14, while that for A hovers around 8. Similarly, the mTPR is approximately 0.60 for W and 0.56 for A, and the mFDR exhibits a similar trend.

Metric		$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
mSHD	W	14.2	13.6	14.3	14.9	14.5
	A	8.1	8.3	8.7	7.7	8.5
mTPR	W	0.59	0.61	0.60	0.58	0.60
	A	0.56	0.55	0.55	0.58	0.56
mFDR	W	0.50	0.49	0.53	0.55	0.54
	A	0.22	0.21	0.23	0.23	0.23

Table 3: Results under partial participation for $d = 10$. Each metric is averaged over 10 simulated datasets.

Varying the Number of Clients

Next, we consider situations in which a fixed total number of samples is allocated among different numbers of clients. For $d \in \{10, 20\}$, a total of $n = 512$ samples is generated and evenly distributed across $K \in \{2, 4, 8, 16, 32\}$ clients. This corresponds to each client receiving $n_k = \{256, 128, 64, 32, 16\}$ samples, respectively. The intra-slice DAG is designed to have an average degree of 4 (counting edges in both directions). This scenario is particularly challenging, as learning distinct PTNs for as many as 32 clients with only 16 samples per client poses significant difficulties.

Metric	Variable	Number of Clients (K) 2	4	8	16	32
Dimension $d = 10$						
mSHD	W	11.3	20.6	17.4	18.2	19.4
	A	6.7	10.5	10.7	10.8	14.8
mTPR	W	0.68	0.44	0.51	0.51	0.49
	A	0.63	0.52	0.48	0.48	0.36
mFDR	W	0.49	0.62	0.54	0.54	0.59
	A	0.17	0.31	0.23	0.28	0.37
Dimension $d = 20$						
mSHD	W	56.2	57.5	54.3	52.1	56.1
	A	33.2	36.1	40.8	55.6	56.3
mTPR	W	0.45	0.46	0.46	0.34	0.32
	A	0.38	0.31	0.33	0.25	0.17
mFDR	W	0.40	0.45	0.37	0.49	0.51
	A	0.18	0.16	0.16	0.21	0.27

Table 4: Performance metrics (mSHD, mTPR, mFDR) for dimensions $d = 10$ and $d = 20$ as the number of clients K varies, computed as averages over 10 simulations.

Tables 4 reveal several trends. For $d = 10$, the mSHD for W remains between 10 and 20, while that for A is approximately 10. Notably, the mTPR for W stays relatively stable even as K increases—for example, at $K = 32$, the mTPR values for W and A are 0.49 and 0.36, respectively. This stability is promising, especially in cases where the per-client sample size is very low. In contrast, the mFDR for A is around 0.2–0.3, whereas for W it is roughly 0.5, indicating that the A matrix is more conservative with fewer false positives.

For $d = 20$, the mSHD for W stabilizes at around 55, but the mSHD for A worsens from roughly 30 to 50 as K increases, suggesting that A is more sensitive to higher dimensions. A similar pattern is seen in the mTPR: while W achieves about 0.45 for $K \in \{2, 4, 8\}$, its value drops to around 0.3 for larger K. This decline is expected, given that the number of samples per client diminishes while the complexity of the ground-truth matrix remains high. Overall, however, `cfPTN` demonstrates robust and consistent performance across varying client numbers, even in the demanding case where $d = 20$, $K = 32$, and $n_k = 16$.

6 Applications

6.1 Blood Oxygenation Level-Dependent (BOLD)

In this experiment, we apply the proposed learning methods to estimate connections in the human brain using simulated blood oxygenation level-dependent (BOLD) imaging data [Smith *et al.*, 2011]. The dataset consists of 28 independent datasets, each containing a number of observed variables $d \in \{5, 10, 15\}$. Each dataset includes 50 subjects (i.e., 50 ground-truth networks) with 200 time steps. To conduct the experiments, we simulate time series measurements corresponding to five different human subjects for each value of d and compute the Average AUROC using the `sklearn` package.

For the `fPTN` configuration, we divide the 200 time steps into five clients ($K = 5$). More details about the data can be found at https://www.boldb.ox.ac.uk/datasets/netsim/index.html. In our experiments, we evaluate the proposed method for $d \in \{5, 10, 15\}$. For $d = 5$, we use subjects 3, 6, 9, 12, and 15 from `Sim-1.mat`. For $d = 10$, we use subjects 2, 4, 6, 8, and 10 from `Sim-2.mat`. For $d = 15$, we use subjects 1, 3, 5, 7, and 9 from `Sim-3.mat`. For the `cfPTN` configuration, we do not partition the data; instead, each subject is treated as a client. Therefore, for $d \in \{5, 10, 15\}$, we have five clients, each with 200 time steps. For $d = 5$, we use subjects 2, 4, 6, 8, and 10 from `Sim-1.mat`, while for $d = \{10, 15\}$, we use subjects 1, 3, 5, 7, and 9 from `Sim-2.mat` and `Sim-3.mat`. To clarify the distinction between the two methods: (1) `fPTN` applies to each subject individually, partitioned into five clients, requiring five repetitions. (2) `cfPTN` is applied directly to all five subjects without partitioning. Further details of the experimental setup can be found in the supplementary materials (see §A.6).

We compare our approach with the Economy Statistical Recurrent Units (eSRU) model [Khanna and Tan, 2019] for Granger causality inference, as well as other existing methods based on Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) networks [Tank *et al.*, 2022], and a Convolutional Neural Network (CNN)-based model, the Temporal Causal Discovery Framework (TCDF) [Nauta *et al.*, 2019], specifically for multivariate time series data with $d = 15$, as previously examined by [Khanna and Tan, 2019]. From the results in Table 5, our proposed `fPTN` achieves an AUROC of 0.74, outperforming the LSTM-based approach and approaching the performance of the CNN-based TCDF method. While `fPTN` does not surpass the MLP-based and eSRU-based methods, it is notable that its federated version outperforms or closely matches several well-established non-federated benchmarks. Similarly, `cfPTN` achieves an AUROC of 0.74, showing superior performance compared to the LSTM-based method and approaching the CNN-based TCDF method. Importantly, `cfPTN` can be applied directly to multiple subjects, significantly reducing the computational overhead associated with data preparation. While `cfPTN` does not exceed the performance of MLP-based and eSRU-based methods, its Customized version consistently outperforms or closely aligns with several established non-Customized benchmarks. This result is expected, as MLP and eSRU methods rely on deep architectures that excel at capturing complex structural dependencies.

Most importantly, our `fPTN` and `cfPTN` methods introduce a new perspective on this problem by ensuring data security through their federated and Customized approaches. This capability is crucial in scenarios where data privacy is a concern, as it enables effective analysis without compromising sensitive or distributed datasets. We explore this aspect further in Section 7.

Method	**Averaged AUROC**
MLP [Tank *et al.*, 2022]	0.81±0.04
LSTM [Tank *et al.*, 2022]	0.70±0.03
TCDF [Nauta *et al.*, 2019]	0.75±0.04
eSRU [Khanna and Tan, 2019]	0.84±0.03
fPTN	**0.74**±0.04
cfPTN	**0.74**±0.05

Table 5: Mean AUROC comparison of different methods for $d = 15$.

6.2 SysGenSIM

Leveraging the SysGenSIM Challenge datasets [Marbach *et al.*, 2009; Stolovitzky *et al.*, 2007; Stolovitzky *et al.*, 2009], our work concentrates on the time-series track of the *InSilico_Size100* subchallenge. In this setting, gene expression data from SysGenSIM are used to reconstruct gene regulatory networks. The *InSilico_Size100* dataset comprises 5 independent datasets, each containing 10 time-series of 100 genes measured over 21 time steps. We interpret each dataset as originating from a distinct institution (e.g., different hospitals or research centers), which motivates our federated framework. Additional details and data can be found at https://sysgensim.sourceforge.net/datasets.html.

Let $X^g_{t,r}$ represent the expression level of gene g at time $t \in \{0, 1, \ldots, 20\}$ for replicate $r \in \{0, 1, \ldots, R\}$. Here, R varies depending on the dataset—if a single dataset is used,

$R = 10$. Hence, $X_{t,r} \in \mathbb{R}^{100}$ and $X_t \in \mathbb{R}^{R\times 100}$. For this experiment, we set the autoregressive order $p = 1$ to be consistent with the VAR-based method used in the SysGenSIM Challenge by [Lu *et al.*, 2021]. In the federated configuration of our `fPTN` approach, we assign $K = 5$ clients, each receiving $R = 2$ replicates. Consequently, each client has a time-series dataset of 100 genes spanning 42 time steps. We discovered that small regularization parameters ($\lambda_W = \lambda_A = 0.0025$) yield robust performance across all SysGenSIM datasets. For the Customized approach `cfPTN`, we treat each dataset as a distinct client; hence, each client comprises 200 time steps with $d = 100$ and $K = 5$. For this setting, we use $\lambda_W = \lambda_A = 0.001$ and set $\mu = 0.1$.

In prior work [Lu *et al.*, 2021], various methods were compared for network inference, including approaches based on Mutual Information (MI), Granger causality, dynamical systems modeling, Decision Trees, Gaussian Processes (GPs), and Temporal Probability Networks (PTNs). We did not include DYNOTEARS in our comparisons since its publicly available source code is lacking and our own implementation did not yield competitive results. The methods were evaluated in terms of AUPR (Area Under the Precision-Recall Curve) and AUROC (Area Under the Receiver Operating Characteristic Curve) across the five datasets.

Table 6 summarizes the comparative performance. The GP-based technique attains the highest mean AUPR (0.208) and AUROC (0.728). In terms of AUPR, our `fPTN` method surpasses the TSNI (ODE-based) method in its standalone, non-federated version. Moreover, the AUPR achieved by `fPTN` is on par with those reported for Ebdnet (PTN-based), GCCA (VAR-based), and ARACNE (MI-based) approaches. Notably, the mean AUROC of `fPTN` exceeds that of GCCA, ARACNE, and TSNI, and is comparable to Ebdnet and VBSSMa (both PTN-based). We also applied `fPTN` directly to the entire dataset (denoted as `fPTN-d`) using the same settings as for `cfPTN`. The direct application produced subpar results, which led us to partition each dataset to satisfy i.i.d. conditions required by `fPTN`. Our experiments with `cfPTN` in heterogeneous scenarios are promising: its AUPR is 0.034, outperforming the non-Customized TSNI method. Furthermore, the AUPR for `fPTN` is only marginally higher than that of `cfPTN`. Additionally, `cfPTN` performs similarly to Ebdnet, GCCA, and ARACNE, while its mean AUROC is superior to that of ARACNE and nearly matches those of TSNI and GCCA.

In summary, `fPTN` and `cfPTN` deliver performance that is competitive with, and sometimes even exceeds, that of other non-federated benchmarks. Nonetheless, the GP-based approach remains the best in terms of both AUPR and AUROC. This advantage likely stems from its ability to model nonlinearities and capture complex temporal dynamics—challenges that federated methods face due to data heterogeneity and varying conditions across sites. Extending GP-based techniques to a federated framework for structure learning represents a promising direction for future work, which we discuss further in Section 7.

The methods used below were mentioned in the following past works:Ebdnet [Rau *et al.*, 2010];VBSSMa,GCCA,TSNI [Penfold and Wild, 2011];CSId [Penfold *et al.*, 2015];ARACNE [Margolin *et al.*, 2006]

Method	AUPR/AUROC	Notes
fPTN	0.040/0.600	Federated
cfPTN	0.034/0.561	Customized federated
fPTN-d	0.023/0.512	Direct application
Ebdnet	0.043/0.640	PTN-based method
VBSSMa	0.086/0.620	PTN-based method
CSId	0.208/0.728	GP-based method
GCCA	0.050/0.584	VAR-based method
TSNI	0.026/0.566	ODE-based method
ARACNE	0.046/0.558	MI-based method

Table 6: Comparison of mean AUPR and mean AUROC scores on the SysGenSIM dataset, along with brief descriptions of each method.

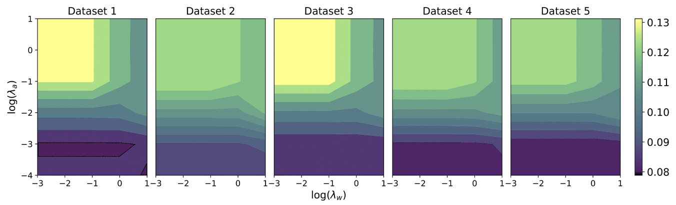

Figure 8: Heatmaps of cross-validation RMSE for the 5 SysGenSIM datasets across a range of A and W parameters.

7 Conclusion

In this work, we have introduced a federated approach for learning Probability Temporal Networks (PTNs) on homogeneous time-series data, as well as a Customized federated method for heterogeneous time-series data. Specifically, we proposed a federated PTN learning method utilizing ADMM, where only model parameters are exchanged during optimization. Our experimental results demonstrate that this federated framework excels, particularly in settings where numerous clients with small sample sizes collaborate—a common scenario in federated learning. We also developed a Customized federated PTN learning approach by incorporating a proximal operator, leveraging ADMM for enhanced accuracy. Our results indicate that the Customized model maintains robust performance even with limited sample sizes and numerous clients. Furthermore, we found that partial client participation does not lead to a significant loss in performance. Below, we discuss the limitations of our approach and outline possible avenues for future research.

Assumptions. Our method assumes that the PTN structure, represented by W and A, is static over time and shared across all time series within the dataset. Relaxing this assumption could offer substantial benefits, such as allowing the structure to evolve smoothly over time, which has been explored in previous work [Song *et al.*, 2009]. Another potential avenue for future research is investigating the behavior of the algorithm in non-stationary or cointegrated time series settings [Malinsky and Spirtes, 2019], or addressing challenges arising from confounding variables in the data [Huang *et al.*, 2015].

Federated Learning. Our `fPTN` method relies on ADMM for federated learning, which assumes that every client participates in every round of communication and optimization. Although we incorporated partial client participation in the `cfPTN` framework, this approach has limited impact, as it only adjusts which clients are involved, without modifying the underlying optimization process. This "always-on" requirement can be a significant challenge in real-world applications, particularly in large-scale systems where clients—such as mobile devices or IoT sensors—may experience intermittent connectivity, limited power, or varying availability. Ensuring that all devices participate synchronously in every round is often impractical and could result in performance bottlenecks. One promising direction for future work is to investigate alternative federated optimization strategies, including asynchronous approaches, that allow clients to operate statelessly and support cross-device learning. Additionally, the ADMM procedure requires sharing model parameters with a central server, raising potential privacy concerns. Research has demonstrated that such parameters can leak sensitive information in some contexts, such as image data [Phong *et al.*, 2018]. Therefore, exploring differential privacy techniques [Dwork and Roth, 2014] to secure shared model parameters is a key area for future investigation.

Customized Learning. Heterogeneous settings are increasingly relevant to real-world applications, where clients often differ in terms of computational power, communication bandwidth, and local data distributions. These differences create challenges for model convergence and performance. Our work introduces the first methodology for learning Customized PTNs via a proximal operator, with the potential to extend this framework to non-temporal settings, such as NOTEARS [Zheng *et al.*, 2018]. Additionally, several Customized federated learning frameworks, such as meta-learning and clustered federated learning, can be integrated with our approach. A promising research direction is to address scenarios where the number of variables, d, differs across clients. Similarly, clients may have varying time-lagged dependencies, p, that they wish to explore, which presents another interesting challenge. Mixed connectivity levels in networks also warrant attention, as sparse networks may lead to confounding, with missing edges in high-connectivity networks. To address such challenges, Customized federated learning methods such as FedNova [Wang *et al.*, 2020], which normalizes aggregated updates to account for heterogeneous local computational workloads, could be applied. Asynchronous techniques are particularly critical in heterogeneous scenarios. While our asynchronous technique does not degrade accuracy, incorporating more advanced asynchronous methods remains essential, especially in the context of reinforcement learning [Espeholt *et al.*, 2018].

Nonlinear Dependencies. Our methodology assumes linear dependencies, chosen for simplicity to emphasize the temporal and dynamic aspects of the problem. However, more complex nonlinear dependencies can be captured using techniques like Gaussian Processes [Gnanasambandam *et al.*, 2024; **?**] or neural networks. Additionally, substituting the least squares loss function with logistic loss or other exponential family log-likelihoods could allow modeling of binary data. Combining continuous and discrete data is another important consideration [Andrews *et al.*, 2019], especially for many real-world applications. Exploring these directions could greatly enhance the applicability of our methods in diverse domains.

Full implementation and benchmarks: https://github.com/SBY7219/Fed_PTN

References

[Andrews *et al.*, 2019] Bryan Andrews, Joseph Ramsey, and Gregory F. Cooper. Learning high-dimensional directed acyclic graphs with mixed data-types. In *Proceedings of Machine Learning Research*, volume 104 of *Proceedings of Machine Learning Research*, pages 4–21. PMLR, 05 Aug 2019.

[Benjamini and Hochberg, 1995] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[Byrd *et al.*, 2003] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208, 2003.

[Chai *et al.*, 2020] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. *IEEE Intelligent Systems*, 35(1):30–38, August 2020.

[Chockalingam *et al.*, 2017] Sabarathinam Chockalingam, Wolter Pieters, André Teixeira, and P.H.A.J.M. Gelder. *Bayesian Network Models in Cyber Security: A Systematic Review*, pages 105–122. 11 2017.

[Combettes and Pesquet, 2011] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer New York, 2011.

[Dwork and Roth, 2014] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[Erdős and Rényi, 1960] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[Espeholt *et al.*, 2018] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. 02 2018.

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

[Gnanasambandam *et al.*, 2024] Raghav Gnanasambandam, Bo Shen, Andrew Chung Chee Law, Chaoran Dou, and Zhenyu Kong. Deep gaussian process for enhanced bayesian optimization and its application in additive manufacturing. *IISE Transactions*, pages 1–14, 2024.

[Gong *et al.*, 2023] Chang Gong, Di Yao, Chuzhe Zhang, Wenbin Li, and Jingping Bi. Causal discovery from temporal data: An overview and new perspectives. *Journal of Artificial Intelligence Research*, 75:231–268, 2023.

[Gou *et al.*, 2007] Kui Xiang Gou, Gong Xiu Jun, and Zheng Zhao. Learning bayesian network structure from distributed homogeneous data. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, volume 3, pages 250–254, 2007.

[hip, 2003] Hipaa privacy rule: Summary of the privacy rule. Code of Federal Regulations, 45 CFR Parts 160 and 164, 2003. Available at: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

[Huang *et al.*, 2015] Biwei Huang, Kun Zhang, and Bernhard Schölkopf. Identification of time-dependent causal model: a gaussian process treatment. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3561–3568. AAAI Press, 2015.

[Jiang *et al.*, 2019] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning, 09 2019.

[Khanna and Tan, 2019] Saurabh Khanna and Vincent Yan Fu Tan. Economy statistical recurrent units for inferring nonlinear granger causality. *ArXiv*, abs/1911.09879, 2019.

[Kontar *et al.*, 2021] Raed Kontar, Naichen Shi, Xubo Yue, Seokhyun Chung, Eunshin Byon, Mosharaf Chowdhury, Jionghua Jin, Wissam Kontar, Neda Masoud, Maher Nouiehed, Chinedum E. Okwudire, Garvesh Raskutti, Romesh Saigal, Karandeep Singh, and Zhi-Sheng Ye. The internet of federated things (ioft). *IEEE Access*, 9:156071–156113, 2021.

[Lemoine *et al.*, 2021] Guillaume G. Lemoine, Marie-Pier Scott-Boyer, Baptiste Ambroise, et al. GWENA: gene co-expression networks analysis and extended modules characterization in a single Bioconductor package. *BMC Bioinformatics*, 22:267, 2021.

[Li *et al.*, 2020] Tian Li, Ananda K. Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020.

[Li *et al.*, 2021] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6357–6368. PMLR, 18–24 Jul 2021.

[Lu *et al.*, 2021] Jing Lu, Bianca Dumitrascu, Ian C. McDowell, Byung-Jun Jo, Andres Barrera, Lichen K. Hong, and et al. Causal network inference from gene transcriptional time-series response to glucocorticoids. *PLoS Computational Biology*, 17(1):e1008223, 2021.

[Madakam *et al.*, 2015] Somayya Madakam, R Ramaswamy, and Siddharth Tripathi. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3:164–173, 04 2015.

[Malinsky and Spirtes, 2019] Daniel Malinsky and Peter Spirtes. Learning the structure of a nonstationary vector autoregression. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2986–2994. PMLR, 16–18 Apr 2019.

[Marbach *et al.*, 2009] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.

[Margolin *et al.*, 2006] Alexander A. Margolin, Ilya Nemenman, Karen Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7:S7, 2006.

[McMahan *et al.*, 2016] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[Muhammad *et al.*, 2020] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. Fedfast: Going beyond average for faster training of federated recommender systems. pages 1234–1242, 08 2020.

[Murphy, 2002] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.d. thesis, University of California, Berkeley, 2002.

[Nauta *et al.*, 2019] Meike Nauta, Doina Bucur, and Christin Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):312–340, 2019.

[Ng and Zhang, 2022] Ignavier Ng and Kun Zhang. Towards federated bayesian network structure learning with continuous optimization. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151, Valencia, Spain, 2022. PMLR.

[Pamfil *et al.*, 2020] Razvan Pamfil, Stefan Bauer, Bernhard Schölkopf, and Joachim M. Buhmann. DYNOTEARS: Structure learning from time-series data. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2020.

[Parikh and Boyd, 2014a] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, January 2014.

[Parikh and Boyd, 2014b] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[Penfold and Wild, 2011] Christopher A. Penfold and David L. Wild. How to infer gene networks from expression profiles, revisited. *Interface Focus*, 1:857–870, 2011.

[Penfold *et al.*, 2015] Christopher A. Penfold, Ahamed Shifaz, Philip E. Brown, Alexander Nicholson, and David L. Wild. Csi: a nonparametric bayesian approach to network inference from multiple perturbed time series gene expression data. *Statistical Applications in Genetics and Molecular Biology*, 14(3):307–310, Jun 2015.

[Phong *et al.*, 2018] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *Trans. Info. For. Sec.*, 13(5):1333–1345, May 2018.

[Rau *et al.*, 2010] Andrea Rau, Florence Jaffrézic, Jean-Louis Foulley, and Rebecca W. Doerge. An empirical bayesian method for estimating biological networks from temporal microarray data. *Statistical Applications in Genetics and Molecular Biology*, 9:Article 9, 2010. Epub 2010 Jan 15.

[Sahu *et al.*, 2018] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv: Learning*, 2018.

[Smith *et al.*, 2011] Stephen M Smith, Karla L Miller, Gholamreza Salimi-Khorshidi, Mathew Webster, Christian F Beckmann, Thomas E Nichols, Joseph D Ramsey, and Mark W Woolrich. Network modelling methods for fmri. *NeuroImage*, 54(2):875–891, 2011.

[Song *et al.*, 2009] Le Song, Mladen Kolar, and Eric P. Xing. Time-varying dynamic bayesian networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 1732–1740, Red Hook, NY, USA, 2009. Curran Associates Inc.

[Stolovitzky *et al.*, 2007] Gustavo Stolovitzky, Don Monroe, and Andrea Califano. Dialogue on reverse-engineering assessment and methods: The dream of high-throughput pathway inference. *Annals of the New York Academy of Sciences*, 1115:11–22, 2007.

[Stolovitzky *et al.*, 2009] Gustavo Stolovitzky, Robert J Prill, and Andrea Califano. Lessons from the dream2 challenges. *Annals of the New York Academy of Sciences*, 1158:159–195, 2009.

[Sun *et al.*, 2020] Yanning Sun, Wei Qin, and Zilong Zhuang. Quality consistency analysis for complex assembly process based on bayesian networks. *Procedia Manufacturing*, 51:577–583, 2020. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).

[Tank *et al.*, 2022] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. B. Fox. Neural granger causality. *IEEE Trans-*

actions on Pattern Analysis and Machine Intelligence, 44(8):4267–4279, 2022.

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

[Wang *et al.*, 2020] Jianyu Wang, Zachary Charles, Brendan Childers, Yu-Xiang Sun, Suhas Sreehari, Sebastian U Stich, Mikhail Dmitriev, Ameet Talwalkar, and Michael I Jordan. Federated learning with normalized averaging. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[Yu *et al.*, 2004] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.

[Zeng *et al.*, 2016] Zhiya Zeng, Xia Jiang, and Richard Neapolitan. Discovering causal interactions using bayesian network scoring and information gain. *BMC Bioinformatics*, 17:221, 2016.

[Zhang *et al.*, 2024] Zihan Zhang, Shancong Mou, Mostafa Reisi Gahrooei, Massimo Pacella, and Jianjun Shi. Federated multiple tensor-on-tensor regression (fedmtot) for multimodal data under data-sharing constraints. *Technometrics*, pages 1–26, 2024.

[Zheng *et al.*, 2018] Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, pages 9472–9483. Curran Associates, Inc., 2018.

A Supplementary Materials

A.1 Multivariate Time-Series Notation

To introduce the index m for the M realizations, we can organize the data for client k. For each client k, define the effective sample size as $n_k = M(T+1-p)$. The time series model for each client can then be expressed as:

$$X_t^k = X_t^k W + X_{t-1}^k A_1 + \cdots + X_{t-p}^k A_p + Z^k,$$

where the terms are defined as follows:

- X_t^k is an $n_k \times d$ matrix, with each row corresponding to $(x_{m,t}^k)^\top$, where $m = 1, \ldots, M$;
- X_{t-i}^k represents the time-lagged matrices for $i = 1, \ldots, p$;
- Z^k is the matrix of noise terms, where each entry corresponds to $(u_{m,t}^k)^\top$.

This setup allows us to handle multiple realizations per client while preserving the structure of the SVAR model across both time and variables. The subsequent learning and optimization steps follow the method described in the main paper.

A.2 Simulation Data Generation

Intra-slice Graph: We generate a random directed acyclic graph (DAG) using the *Erdős-Rényi (ER) model* with a target mean degree pr. In this model, edges are formed independently by performing i.i.d. Bernoulli trials with a probability pr/dr, where dr is the total number of nodes. The graph is initially represented by an adjacency matrix and then oriented to ensure acyclicity by imposing a lower triangular structure, resulting in a valid DAG. Finally, the nodes are randomly permuted to remove trivial ordering, creating a randomized and realistic structure suitable for subsequent analysis.

Inter-slice Graph: The inter-slice graph is also generated using the *ER model*. Here, edges are directed from node i_{t-1} at time $t-1$ to node j_t at time t. The binary adjacency matrix A_{bin} is constructed as:

$$A_{i_{t-1},j_t} = \begin{cases} 1 & \text{with probability } \frac{pr}{dr} \quad \text{for edges from } i_{t-1} \text{ to } j_t, \\ 0 & \text{otherwise.} \end{cases}$$

Assigning Weights: Once the binary adjacency matrix is created, we assign weights to the edges drawn from a *uniform distribution*. For W, the weights are in the range $[-0.5, -0.3] \cup [0.3, 0.5]$, and for A, the weights are in the range $[-0.5\alpha, -0.3\alpha] \cup [0.3\alpha, 0.5\alpha]$, where:

$$\alpha = \frac{1}{\eta^{p-1}},$$

and $\eta \geq 1$ is a decay parameter that governs the reduction in edge influence as time steps increase.

Table 7: Optimal values for λ_a and λ_w for varying d.

	$d=5$	$d=10$	$d=15$	$d=20$
λ_a	0.5	0.5	0.5	0.5
λ_w	0.5	0.5	0.5	0.5

A.3 Hyperparameter Analysis

This section details the optimal hyperparameter values used in our simulations. The following table presents the best values of λ_a and λ_w for different numbers of variables (d). Generally, $\lambda_a = 0.5$ and $\lambda_w = 0.5$ perform well across all configurations. However, when $\lambda_a, \lambda_w > 0.5$, the algorithm sometimes produces zero matrices.

The following tables summarize the optimal values for λ_a and λ_w for varying numbers of clients, with a fixed sample size of $n = 256$. In general, values of λ_a between 0.4 and 0.5 and λ_w between 0 and 0.5 work effectively. Simulations suggest that the optimal range for both λ_a and λ_w is between 0.05 and 0.5, depending on the network topology. Within this range, the error in Structural Hamming Distance (SHD) is generally less than 5 units from the optimal value.

Table 8: **Sample size with** $n = 256$ **and** $d = 10$

	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$	$k=64$
λ_a	0.05	0.5	0.5	0.4	0.35	0.4
λ_w	0.3	0.5	0.45	0.5	0.25	0.3

Table 9: **Sample size with** $n = 256$ **and** $d = 20$

	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$	$k=64$
λ_a	0.3	0.2	0.5	0.5	0.5	0.05
λ_w	0.4	0.5	0.5	0.5	0.5	0.25

A.4 Closed-Form Solution for B_k and D_k

We minimize the objective function with respect to B_k and D_k:

$$\begin{aligned} J(B_k, D_k) = \ell_k(B_k, D_k) &+ \operatorname{Tr}\left(\beta_k^t (B_k - W^{(t)})^\top\right) \\ &+ \frac{\rho_2^t}{2} \left\|B_k - W^{(t)}\right\|_F^2 \\ &+ \operatorname{Tr}\left(\gamma_k^t (D_k - A^{(t)})^\top\right) + \frac{\rho_2^t}{2} \left\|D_k - A^{(t)}\right\|_F^2 \end{aligned}$$

where:

$$\ell_k(B_k, D_k) = \frac{1}{2n} \left\|X_t - X_t B_k - X_{(t-p):(t-1)} D_k\right\|_F^2.$$

By taking the gradients and setting them to zero, we derive the following conditions for optimality:

Gradient with respect to B_k**:**

$$\nabla_{B_k} J = -\frac{1}{n} X_t^\top \left(X_t - X_t B_k - X_{(t-p):(t-1)} D_k \right) + \beta_k^t + \rho_2^t (B_k - W^{(t)}) = 0.$$

Simplifying:

$$\begin{aligned}
&(-S + SB_k + MD_k) + \beta_k^t + \rho_2^t (B_k - W^{(t)}) = 0 \\
&\implies (S + \rho_2^t I) B_k + MD_k = S - \beta_k^t + \rho_2^t W^{(t)} \\
&\implies PB_k + MD_k = b_1.
\end{aligned}$$

Gradient with respect to D_k:

$$\nabla_{D_k} J = -\frac{1}{n} X_{(t-p):(t-1)}^\top \left(X_t - X_t B_k - X_{(t-p):(t-1)} D_k \right) + \gamma_k^t + \rho_2^t (D_k - A^{(t)}) = 0.$$

Simplifying:

$$\begin{aligned}
&(-M^\top + M^\top B_k + ND_k) + \gamma_k^t + \rho_2^t (D_k - A^{(t)}) = 0 \\
&\implies M^\top B_k + (N + \rho_2^t I) D_k = M^\top - \gamma_k^t + \rho_2^t A^{(t)} \\
&\implies M^\top B_k + QD_k = b_2.
\end{aligned}$$

A.5 Closed form for Customized PTN learning

Update for $\mathbf{B}_k$

$$f_{\tilde{W}_k}(\tilde{W}_k) = \mu \|W_k^{(t+1)} - \tilde{W}_k\|^2 + \operatorname{tr}\big(\beta_k^{(t)\top} (\tilde{W}_k - W^{(t)})\big) + \frac{\rho_2^{(t)}}{2} \|\tilde{W}_k - W^{(t)}\|^2 \tag{14}$$

Hence,

$$\nabla_{\tilde{W}_k} f_{\tilde{W}_k}(\tilde{W}_k) = 2\mu\,(\tilde{W}_k - W_k^{(t+1)}) + \beta_k^{(t)} + \rho_2^{(t)}\,(\tilde{W}_k - W^{(t)}).$$

By setting $\nabla_{\tilde{W}_k} f_{\tilde{W}_k}(\tilde{W}_k) = 0$, we have

$$2\mu(\tilde{W}_k - W_k^{(t+1)}) + \beta_k^{(t)} + \rho_2^{(t)}(\tilde{W}_k - W^{(t)}) = 0.$$

$$\big(2\mu + \rho_2^{(t)}\big)\,\tilde{W}_k = 2\mu\, W_k^{(t+1)} + \rho_2^{(t)}\, W^{(t)} - \beta_k^{(t)}.$$

$$\boxed{\tilde{W}_k^{(t+1)} = \frac{2\mu\, W_k^{(t+1)} + \rho_2^{(t)}\, W^{(t)} - \beta_k^{(t)}}{2\mu + \rho_2^{(t)}}.}$$

Update for $\mathbf{D}_k$

$$f_{\tilde{A}_k}(\tilde{A}_k) = \mu \|A_k^{(t+1)} - \tilde{A}_k\|^2 + \operatorname{tr}\big(\gamma_k^{(t)\top} (\tilde{A}_k - A^{(t)})\big) + \frac{\rho_2^{(t)}}{2} \|\tilde{A}_k - A^{(t)}\|^2.$$

Hence,

$$\nabla_{\tilde{A}_k} f_{\tilde{A}_k}(\tilde{A}_k) = 2\mu\,(\tilde{A}_k - A_k^{(t+1)}) + \gamma_k^{(t)} + \rho_2^{(t)}\,(\tilde{A}_k - A^{(t)}).$$

By $\nabla_{\tilde{A}_k} f_{\tilde{A}_k}(\tilde{A}_k) = 0$, we have

$$2\mu(\tilde{A}_k - A_k^{(t+1)}) + \gamma_k^{(t)} + \rho_2^{(t)}(\tilde{A}_k - A^{(t)}) = 0.$$

$$\big(2\mu + \rho_2^{(t)}\big)\,\tilde{A}_k = 2\mu\, A_k^{(t+1)} + \rho_2^{(t)}\, A^{(t)} - \gamma_k^{(t)}.$$

$$\boxed{\tilde{A}_k^{(t+1)} = \frac{2\mu\, A_k^{(t+1)} + \rho_2^{(t)}\, A^{(t)} - \gamma_k^{(t)}}{2\mu + \rho_2^{(t)}}.}$$

A.6 Application in BOLD

In this section, we present the recorded AUROC values for our `fPTN` method, as shown in Table.10.

AUROC	1	2	3	4	5
$d = 15$	0.68	0.75	0.71	0.77	0.77
$d = 10$	0.69	0.77	0.71	0.69	0.83
$d = 5$	0.70	0.75	0.76	0.78	0.70

Our model produces two matrices, W and A, which represent strong and weak connections, respectively. The final weight matrix is derived by summing these two matrices element-wise. Through repeated experiments, we determined that setting $\lambda_W = 0.05$ and $\lambda_A = 0.01$ yields optimal performance across all datasets.

Below is the table for the `cfPTN` results:

Table 10: AUROC values for `cfPTN` across various datasets.

AUROC	1	2	3	4	5
$d = 15$	0.67	0.77	0.71	0.78	0.77
$d = 10$	0.78	0.71	0.70	0.76	0.75
$d = 5$	0.73	0.78	0.70	0.83	0.78

A.7 Application to SysGenSIM

In Table 11, we present the AUPR and AUROC for each dataset in the SysGenSIM challenge. Similar to the BOLD data, our method produces two matrices, W and A, but here they represent fast-acting and slow-acting influences, respectively. The final weight matrix is obtained by combining these matrices through element-wise addition. Based on the hyperparameter analysis by [Pamfil *et al.*, 2020], we found that $\lambda_W = 0.0025$ and $\lambda_A = 0.0025$ work well across all datasets.

Table 11: AUPR and AUROC for each dataset in the SysGenSIM challenge.

Dataset	AUPR	AUROC
1	0.054	0.64
2	0.032	0.58
3	0.041	0.60
4	0.034	0.58
5	0.035	0.62
Mean ± Std	0.040 ± 0.008	0.60 ± 0.022

A.8 Results for A

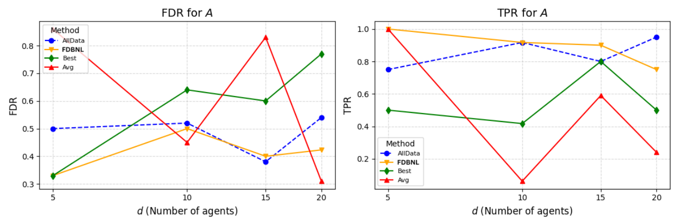

Figure 9: Learning results for A in a PTN with Gaussian noise for varying numbers of variables ($d = 5, 10, 15, 20$), with autoregressive order $p = 1$, and $K = 10$ clients. Each value represents the average performance across 10 different simulated datasets.

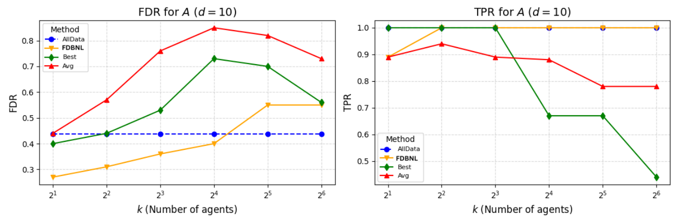

Figure 10: Structure learning for A in a PTN with Gaussian noise, $d = 10$, autoregressive order $p = 1$, and varying numbers of clients. A total of $n = 256$ samples are distributed evenly across $K \in \{2, 4, 8, 16, 32, 64\}$ clients. Each value reflects the mean performance over 10 different datasets.

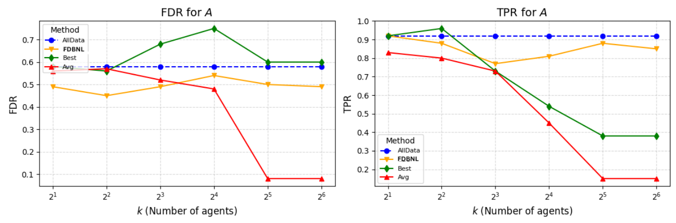

Figure 11: Structure learning for A in a PTN with Gaussian noise, $d = 20$, autoregressive order $p = 1$, and varying numbers of clients. A total of $n = 256$ samples are distributed evenly across $K \in \{2, 4, 8, 16, 32, 64\}$ clients. Each value represents the mean performance over 10 different datasets.

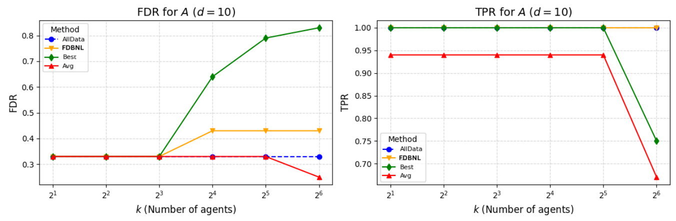

Figure 12: Learning for A in a PTN with Gaussian noise for $d = 10$, autoregressive order $p = 1$, and varying client numbers. A total of $n = 512$ samples are distributed evenly across $K \in \{2, 4, 8, 16, 32, 64\}$ clients. Each value reflects the mean performance across 10 simulated datasets.

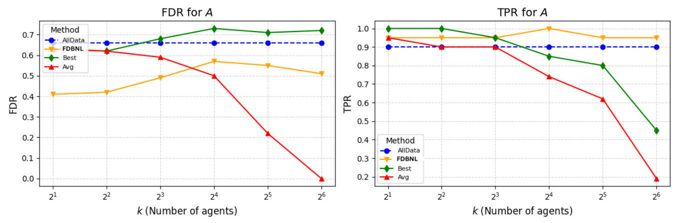

Figure 13: Structure learning for A in a PTN with Gaussian noise, $d = 20$, autoregressive order $p = 1$, and varying client numbers. A total of $n = 512$ samples are evenly distributed across $K \in \{2, 4, 8, 16, 32, 64\}$ clients. Each value reflects the mean performance across 10 simulated datasets.