A Graph-Assisted Framework for Multiple Graph Learning

Xiang Zhang, *Student Member, IEEE*, and Qiao Wang, *Senior Member, IEEE*

***Abstract*—In this paper, we endeavor to jointly learn multiple distinct but related graphs by exploiting the underlying topological relationships between them. The difficulty lies in how to design a regularizer that accurately describes the intricate topological relationships, especially without prior knowledge. This problem becomes more challenging for the scenarios where data for different graphs are stored separately and prohibited from being transmitted to an unreliable central server due to privacy concerns. To address these issues, we propose a novel regularizer termed pattern graph to flexibly describe our priors on topological patterns. Theoretically, we provide the estimation error upper bound of the proposed graph estimator, which characterizes the impact of some factors on estimation errors. Furthermore, an approach that can automatically discover relationships among graphs is proposed to handle awkward situations without priors. On the algorithmic aspect, we develop a decentralized algorithm that updates each graph locally without sending the private data to a central server. Finally, extensive experiments on both synthetic and real data are carried out to validate the proposed method, and the results demonstrate that our framework outperforms the state-of-the-art methods.**

***Index Terms*—Graph learning, decentralized algorithms, graph Laplacian, topological patterns, multiple graph learning.**

I. Introduction

GRAPHS are powerful tools for characterizing structured data and are widely used in numerous fields, e.g., machine learning [1], signal processing [2], and statistics [3], where vertices represent data entities, and edges describe hidden relationships between these entities [4]. Among these applications, constructing some graphs based on prior knowledge is of paramount importance, such as transport networks and social networks. However, prior graphs are not always available. Even if we have topological priors, they may fail to capture the intrinsic relationships between data entities accurately [4]. An alternative way is to learn a graph directly from raw high dimensional data for some downstream tasks, e.g., graph neural network [5]. Inferring graph topology from data, also known as graph learning [6], has been a hot research topic in recent years. In the literature, statistical models are one of the widely studied models, in which a precision matrix representing the graph topology is learned from the observed data [3], [7]. Recently, imposing Laplacian constraints on the learned precision matrices has received increasing attention [8], [9], [10]. Under the Laplacian constraints, graph learning can be formulated as a Laplacian constrained precision matrix estimation problem, which is similar to the notable models that learn graphs from smooth signals [11], [12]. Smoothness based models attempt to learn graphs from a signal processing perspective by exploiting the emerging graph signal processing (GSP) tools [13], [14]. A graph signal being smooth means that signal values of two connected vertices with large edge weights tend to be similar [12]. In real life, plenty of signals have the property of smoothness, including meteorological data [15], social network data [12], and medical data [16].

However, most of the existing literature focuses on learning a single graph. In many scenarios, we may collect heterogeneous data from multiple distinct but related graphs defined over the same node set. For instance, suppose that we collect brain fMRI data for a set of autistic and non-autistic individuals to learn the impact of autism on brain functional connectivity graphs. In this case, one would expect graphs for the two datasets to be different since some connections between brain functional regions will change due to autism. As such, it is no longer reasonable to learn a single graph for all these observations. On the other hand, graphs for the two datasets are also expected to be similar, as most brain functional connections of autistic subjects work properly. Therefore, estimating these graphs separately ignores the hidden topological relationships among them that may be critical to improving learning performance. Grounded on the above observations, a more realistic option is to jointly learn multiple graphs [17]. The philosophy behind multiple graph learning (MGL) is that we can borrow information from the data of other related graphs to improve inference performance. The commonly used methodology is to exploit the information from topological relationships in the form of regularization terms. Specifically, some regularizers describing the assumed topological patterns are designed and added to objective functions to obtain the desired graphs. A thorough literature review of MGL is included in Section II-A.

While there have been many excellent explorations of MGL, they may still suffer from the following limitations. (i) The designed regularizer is too naive to capture the intricate relationships among graphs in the real world. In this regard, the widely studied problem of learning time-varying graphs is a vivid example. The regularizers of current models [18], [19], [20]

Manuscript received 10 September 2022; revised 21 May 2023 and 6 December 2023; accepted 19 December 2023. Date of publication 10 January 2024; date of current version 14 February 2024. Part of this work has been published in EUSIPCO2022. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dorina Thanou. *(Corresponding author: Qiao Wang.)*

The authors are with the School of Information Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: xiangzhang369@seu.edu.cn; qiaowang@seu.edu.cn).

Digital Object Identifier 10.1109/TSIPN.2024.3352236

can be concluded as $\sum_{t=2}^{T}\psi(\mathbf{W}_t-\mathbf{W}_{t-1})$, where $\mathbf{W}_t$ is the adjacency matrix of the t-th time slot and ψ is a distance function, e.g., ℓ_1 norm. Observe that the constraints only occur between two temporally adjacent graphs, which may not be consistent with the real situation. For example, the periodic pattern of urban crowd flow networks suggests connections between two graphs at the same moment on two different days, even though they are not temporally adjacent. (ii) The designs of regularizers excessively rely on the prior knowledge about topological patterns. An excellent work is [17] which exploits a Gram matrix to describe the prior topological relationships. However, the construction of the Gram matrix depends on a subjective understanding of the underlying structural relationships among graphs, which may be inaccurate. (iii) Data privacy is completely ignored by existing MGL models. The data of different graphs may be collected separately by some individuals or organizations and stored in their corresponding local clients. The popular learning paradigm for existing methods is to collect data from different graphs into a central server to jointly learn multiple graphs. Albeit effective, we often encounter such pragmatic scenarios where sending data to an unreliable third-party server is not allowed due to privacy concerns. For instance, the brain fMRI data of autistic individuals collected by different medical institutions are privacy-sensitive, and patients may be averse to the leakage of their private data. Such specific constraint renders the widely used centralized learning paradigm impotent.

To this end, we shed some new light on MGL by addressing the three aforementioned concerns. Specifically, we propose a novel MGL framework to flexibly exploit the topological patterns, even without topological priors available. Moreover, an algorithm considering preserving data privacy is developed to collaboratively learn multiple graphs. The main contributions of this paper are summarized in detail as follows:

Firstly, we propose a graph-structured regularizer termed *pattern graph* to describe intricate topological patterns in the real world. In a pattern graph, connections between any two paired graphs can be added. This thus makes it flexible to incorporate the prior knowledge about topological patterns into the learning process. Based on the regularizer, we then propose a novel MGL model under the widely studied smoothness assumptions [11], which generalizes some classic MGL models, e.g., time-varying models [18], [19]. Moreover, we establish the estimation error upper bound for the proposed graph estimator, in which the impact of several factors on the estimation performance is revealed. Note that the proposed pattern graph is similar to the Markov random field (MRF), where probability distributions of random variables are associated with vertices, and relationships between the distributions are modeled using an undirect graph. However, the probability graphs used in MRF are different from the pattern graphs since they have no edge weights. Moreover, the edges in probability graphs describe the dependencies between random variables, while those in pattern graphs capture the topological relationships.

Secondly, we propose an approach to automatically discover the topological patterns behind multiple graphs. The approach bypasses the requirement of a handcrafted regularizer and is suitable for the case without topological priors. Furthermore, we consider the scenario where a new graph comes after learning all graphs. A method is proposed to incrementally learn the new graph using the previously learned graphs, which can effectively reduce the problem scale.

Thirdly, we develop a fully decentralized algorithm to learn multiple graphs without sending private data to a central server, where each client updates its corresponding graph locally with its own data. Data privacy can be preserved to some extent since the data will not leave local clients. The convergence analysis of our algorithm is also provided.

Finally, extensive experiments on both synthetic and real data are performed to validate the effectiveness of the proposed framework. The results show that our framework outperforms the start-of-the-art methods when faced with intricate relationships among graphs, especially when no priors are available.

Organization: The rest of this paper is organized as follows. We first describe some related works in Section II, followed by the background and the problem statement in Section III. We propose our MGL model in Section IV and the decentralized algorithm in Section V. Experiments are carried out in Section VI. Some concluding remarks are presented in Section VII.

Notations: Throughout this paper, vectors, matrices and sets are written in bold lowercase, bold uppercase letters and calligraphic capital letters, respectively. $\mathbf{y}_{[i]}$ and $\mathbf{Y}_{[ij]}$ denote the i-th and (i,j) entry of $\mathbf{y}$ and $\mathbf{Y}$. The ℓ_1, ℓ_2 and Frobenius norm are denoted as $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_{\mathrm{F}}$. The notations $\circ$, $\dagger$, and $\mathrm{Tr}(\cdot)$ denote Hadamard product, pseudo inverse, and trace operator. $\mathrm{diag}(\mathbf{y})$, $\mathrm{diag}(\mathbf{Y})$ and $\mathrm{diag}_0(\mathbf{Y})$ stand for converting a vector $\mathbf{y}$ to a diagonal matrix, converting the diagonal elements of $\mathbf{Y}$ to a vector and setting the diagonal entries of $\mathbf{Y}$ to 0. $\mathbf{y}\geq 0$ and $\mathbf{Y}\geq 0$ mean that all elements of $\mathbf{y}$ and $\mathbf{Y}$ are greater than 0. The constant vectors $\mathbf{1},\mathbf{0}$, and matrix $\mathbf{I}$ represent column the vector of ones, zeros, and identity matrix. The $Gauss(\mathbf{y},\mathbf{Y})$ denotes Gaussian distribution with mean vector $\mathbf{y}$ and covariance matrix $\mathbf{Y}$. Finally, $\mathbb{R}$ denotes the set of real values. The superscripts of $\mathbb{R}$ denote dimensions of vector (matrices) and subscript $+$ denotes nonnegative constraint.

II. Related Work

A. Multiple Graph Learning

The MGL models can be roughly divided into at least two categories, i.e., statistical models and GSP based models. The statistical models are mostly built on the Gaussian graphical models [21], [22]. Following the standard paradigm, these models attempt to learn multiple precision matrices by assuming some topological patterns, e.g., time-varying patterns [20], [23], group similarity [24], pairwise similarity [23], [24], common sparsity structure [25], and cluster structure [26]. Our proposed model differs from the statistical models in the sense that we learn valid graph Laplacian matrices instead of precision matrices. The other line of MGL is based on GSP, and a widely studied family of methods assumes that graph signals are stationary [27], [28]. Our model, however, comes from another assumption, i.e., the smoothness assumption. There is a profusion of studies on smoothness-based MGL, and most of them attempt to learn time-varying graphs [18], [19], [29], which only exploit the relationships between temporally adjacent graphs. By contrast,

our model focuses on a more general scenario where relationships between any two paired graphs can be described. Some studies [17], [30] are also able to describe complex topological patterns via Gram matrices. However, they need a handcrafted regularizer incorporating prior knowledge, while our model can automatically discover the relationships from data. Recently, a new scenario has arisen where we need to learn multiple graphs without knowing which graph the signals come from [30], [31], [32]. The goal of these works is to decouple the signals into groups and learn a graph for each of them. In our work, however, signal labels are given a priori. We learn multiple graphs jointly by using the underlying topological relationships between the graphs.

B. Multi-Task Learning

Learning multiple graphs jointly can be regarded as a multi-task learning (MTL) problem. In the literature, one of the main clues of MTL is to assume some relationships between different tasks, e.g., task similarity. One common tool to characterize task similarity is covariance matrices [33], [34]. Some other specific structures, e.g., multi-level structure [35] and tree structure [36], are also leveraged to describe task similarity. See [37] for a comprehensive review of MTL. Our model also stems from the similarity based MTL but exploits a graph structure to represent task relationships.

Recently, preserving privacy in MTL has been a surge of interest. One of the mainstream works is distributed MTL [38], where a personalized model is learned for each client [39]. The clients only transmit variables, instead of raw data, to their related clients. However, the task relationships of these models are determined by prior knowledge [40]. A framework that learns tasks and task relationships simultaneously is proposed in [41], but it requires a central server to learn task similarity. In [42], local tasks and their relationships are jointly learned in a decentralized way, which is close to our algorithm. However, our formulation is a generalized version of [42] since the metric measuring similarity between tasks is not limited to squared ℓ_2 norm as [42] does.

III. Background and Problem Statement

A. GSP Background

We focus on undirected graphs with nonnegative weights and no self-loops. For such a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with d vertices, where $\mathcal{V}$ and $\mathcal{E}$ are the set of vertices and edges of $\mathcal{G}$ respectively, the adjacency matrix $\mathbf{A}$ is a $d \times d$ symmetric matrix with zero diagonal entries and nonnegative off-diagonal entries. Another important matrix of $\mathcal{G}$, the Laplacian matrix $\mathbf{L}$, is then defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [2], in which degree matrix $\mathbf{D}$ is a $d \times d$ diagonal matrix with $\mathbf{D}_{[ii]} = \sum_{j=1}^{d} \mathbf{A}_{[ij]}$. Note that $\mathbf{A}$ and $\mathbf{L}$ can represent the topology of $\mathcal{G}$ since they have a one-to-one relationship. A graph signal $\mathbf{x} = [\mathbf{x}_{[1]}, \ldots, \mathbf{x}_{[d]}]^\top$ generating from graph $\mathcal{G}$ means that every dimension of $\mathbf{x}$ is regarded as a node in $\mathcal{G}$. Furthermore, edges of $\mathcal{G}$ describe the relationships between dimensions of $\mathbf{x}$, and weights quantify the "closeness" of these relationships. Our framework is grounded on the smoothness assumption, and hence the definition of smooth graph signals is provided first.

Definition 1: (Smoothness, [12]). Given a graph signal $\mathbf{x}$ and the Laplacian matrix $\mathbf{L}$ of a graph $\mathcal{G}$, the smoothness of $\mathbf{x}$ over $\mathcal{G}$ is defined as

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \frac{1}{2} \sum_{i,j} \mathbf{A}_{[ij]} \left(\mathbf{x}_{[i]} - \mathbf{x}_{[j]}\right)^2. \tag{1}$$

By definition, a smaller value of (1) corresponds to a smoother signal over $\mathcal{G}$. Therefore, for a smooth signal, the values of two dimensions connected with large edges weight of $\mathcal{G}$ need to be similar.

B. Smoothness Based Graph Learning

For given N observations $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_N]$, the objective of graph learning is to infer the hidden graph topology, i.e., $\mathbf{A}$ or $\mathbf{L}$, directly from the observational signals. Smoothness-based graph learning aims to learn those graphs that make the graph signals smoothest, whose common paradigm is

$$\min_{\mathbf{L} \in \mathcal{L}} \sum_{i=1}^{N} \frac{1}{N} \mathbf{x}_i^\top \mathbf{L}\mathbf{x}_i + g_L(\mathbf{L}), \tag{2}$$

where $g_L(\mathbf{L})$ is a regularization term of $\mathbf{L}$ to obtain some desired property of graphs, e.g., sparsity. Moreover, $\mathcal{L}$ is the set containing all Laplacian matrices

$$\mathcal{L} \triangleq \left\{\mathbf{L} : \mathbf{L} = \mathbf{L}^\top, \mathbf{L}\mathbf{1} = \mathbf{0}, \mathbf{L}_{[ij]} \leq 0 \text{ for } i \neq j\right\}. \tag{3}$$

By following [11], we select $g_L(\mathbf{L})$ as $-\alpha \mathbf{1}^\top \log(\text{diag}(\mathbf{L})) + \frac{\beta}{2}\|\text{diag}_0(\mathbf{L})\|_{\text{F}}^2$, where α and β are predefined parameters. The first term of $g_L(\mathbf{L})$ controls the degrees of each node, and the second term controls the sparsity of edges [11]. With $g_L(\mathbf{L})$, (2) can be expressed in the form of an adjacency matrix $\mathbf{A}$ [11], i.e.,

$$\min_{\mathbf{A} \in \mathcal{A}} \frac{1}{2^\frown N} \|\mathbf{A} \circ \mathbf{C}\|_1 - \alpha \mathbf{1}^\top \log(\mathbf{A}\mathbf{1}) + \frac{\beta}{2}\|\mathbf{A}\|_{\text{F}}^2, \tag{4}$$

where $\mathcal{A}$ is the set defined as

$$\mathcal{A} = \left\{\mathbf{A} : \mathbf{A} \in \mathbb{R}_+^{d \times d}, \mathbf{A} = \mathbf{A}^\top, \text{diag}(\mathbf{A}) = \mathbf{0}\right\}. \tag{5}$$

Furthermore, $\mathbf{C} \in \mathbb{R}^{d \times d}$ in (4) is a pairwise distance matrix. Let $\mathbf{X}^\top = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_d]$, where $\tilde{\mathbf{x}}_i^\top \in \mathbb{R}^N$ is the i-th row vector of $\mathbf{X}$. The (i, j) entry of $\mathbf{C}$ is equivalent to

$$\mathbf{C}_{[ij]} = \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2. \tag{6}$$

Since $\mathbf{A}$ is a symmetric matrix with diagonal entries equal to zero, the number of free variables of $\mathbf{A}$ is in fact $\frac{d(d-1)}{2}$ [11]. For the sake of brevity, let $p \triangleq \frac{d(d-1)}{2}$ and define a vector $\mathbf{w} \in \mathbb{R}^p$ whose entries are the upper triangle variables of $\mathbf{A}$. With the definition of $\mathbf{w}$, (4) can be rewritten as the following form [11]

$$\min_{\mathbf{w} \geq 0} f(\mathbf{w}) = \min_{\mathbf{w} \geq 0} \frac{1}{N} \mathbf{r}^\top \mathbf{w} \underbrace{-\alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}) + \beta \|\mathbf{w}\|_2^2}_{g(\mathbf{w})}, \tag{7}$$

where $\mathbf{S}$ is a linear operator satisfying $\mathbf{S}\mathbf{w} = \mathbf{A}\mathbf{1}$, and $\mathbf{r}$ is the vector form of the upper triangle elements in $\mathbf{C}$.

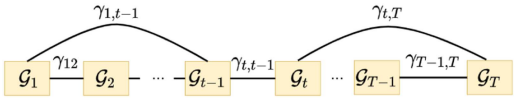


Fig. 1. Illustration of pattern graph.

C. Problem Statement

Suppose there are T datasets containing graph signals $\mathbf{X}_1, \ldots, \mathbf{X}_T$ generated from T graphs $\mathcal{G}_1, \ldots, \mathcal{G}_T$ defined over the same node set, where $\mathbf{X}_t \in \mathbb{R}^{d \times N_t}$ and N_t is the data size of the t-th dataset. The graph signals are stored separately and prohibited from leaving their local clients due to data privacy. We are required to infer the topology of these graphs jointly without sending all data to a central server. The task is grounded on two assumptions. (i) The collected signals $\mathbf{X}_t$ are smooth over the corresponding graph $\mathcal{G}_t$; (ii) The T graphs are topologically related. The first technical concern of this study is how to exploit the latent topological relationships to boost learning performance, especially without topological prior. The second technical concern is how to jointly learn the T graph under the constraints of data silos.

IV. Modeling and Formulation

A. MGL With Prior Topological Patterns

We first assume that the structured relationships across different graphs are known a priori. The relationship between two graphs is interpreted as the underlying similarity between them, and we need to incorporate the prior similarities into the objective function. To this end, we propose a general and flexible regularizer named *pattern graph* to describe the relationships among these graphs. A simple example of pattern graphs is given in Fig. 1. The pattern graph $\mathcal{G}_N$ is an undirected and weighted graph whose nodes represent $\mathcal{G}_1, \ldots, \mathcal{G}_T$, respectively. The edges in $\mathcal{G}_N$ describe the relationships between any two paired graphs, which are represented by some constraint functions $\psi(\cdot)$ that penalize the difference between the two connected graphs. Additionally, edge weight γ_{ij} is used to measure the "closeness" of the corresponding connections. Formally, a pattern graph is denoted as $\mathcal{G}_N = \{\mathcal{V}_N, \mathcal{E}_N\}$, where $\mathcal{V}_N$ is the node set containing T graphs, and $\mathcal{E}_N$ is the edge set containing all connections between these graphs. We assume there are S edges in $\mathcal{G}_N$, i.e., $|\mathcal{E}_N| = S$, and the proposed regularizer is expressed as

$$\sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi\left(\mathbf{w}_i - \mathbf{w}_j\right), \tag{8}$$

where $\mathbf{w}_i$ is the vector form of the adjacency matrix of $\mathcal{G}_i$. Given $\mathcal{G}_N$, the proposed model is formulated as

$$\begin{aligned} &\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T\geq 0} \sum_{t=1}^{T} f_t(\mathbf{w}_t) + \eta \sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi(\mathbf{w}_i - \mathbf{w}_j) \\ = &\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T\geq 0} \sum_{t=1}^{T} \frac{1}{N_t}\mathbf{r}_t^\top \mathbf{w}_t - \alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}_t) + \beta\|\mathbf{w}_t\|_2^2 \\ &+ \eta \sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi(\mathbf{w}_i - \mathbf{w}_j), \end{aligned} \tag{9}$$

where η is used to scale the edge objectives relative to the node objectives in (9). The first term f_t is the basic graph learning formulation in (7), where $\mathbf{r}_t$ is the vectorized pairwise distance matrix of the t-th graph. Note that f_t depends on t because of r_t. The second term of (9) is the regularizer that encodes prior topological relationships using a pattern graph. In our setup, we let $\gamma_{ij} = \gamma_{ji}$ since $\mathcal{G}_N$ is undirected. Note that (i, j) and (j, i) are referred to the same edge in pattern graphs, and we will not distinguish them.

1) Selection of ψ: Since (i, j) and (j, i) represent the same edge, we suppose ψ is convex and satisfies $\psi(\mathbf{w}_i - \mathbf{w}_j) = \psi(\mathbf{w}_j - \mathbf{w}_i)$. Practically, ψ can be selected by cross-validation or incorporating domain knowledge about topological patterns [20]. For example, ℓ_1 norm is suitable for the case where few edges are allowed to change between two graphs [43], while ℓ_2 norm may be appropriate when there exist topological switches [19]. More choices of ψ can be found in [20].

2) Determination of η: Note that in (9), η is not necessary since it can be incorporated into γ_{ij}. However, as [44] states, we had better view them separately. The advantage is that we can regard γ_{ij} as a relative weight and only need to tune η to yield different global results. Indeed, η is a trade-off parameter between using information from data and topological patterns. When $\eta = 0$, all graphs are independently learned only using the information from the data. The information of structural relationships is gradually added to the objective functions as the increase of η. In the extreme case when $\eta \to \infty$, if $\mathcal{G}_N$ is a connected graph, (9) will boil down to

$$\min_{\mathbf{w}\geq 0} \sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}), \tag{10}$$

where a consensus graph is obtained from all data since the second term of (9) forces all graphs to be the same. In this study, we select the η that achieves the best performance by grid search. More details will be introduced in Section VI.

3) Design of $\mathcal{G}_N$: The design of $\mathcal{G}_N$ depends on the topological relationships among graphs. The edges in $\mathcal{G}_N$ connect two supposedly related graphs, and the corresponding weights are determined by the relative "closeness" of these connections. Without loss of generality, we can suppose that $\sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij} = 1$ and assign weights to different edges using some prior knowledge. Some existing works, e.g., [28], also use a pattern graph to integrate prior topological relationships into the learning process. However, when no priors are available, we will propose a data-driven method to learn the underlying $\mathcal{G}_N$ in Section IV-B.

4) Relationship to previous models: The proposed pattern graph is a general regularizer that can include many existing models, e.g., time-varying models that learn $\mathcal{G}_1, \ldots, \mathcal{G}_T$ with time stamps. The penalties of these models impose constraints on two temporally adjacent graphs. Therefore, if we build $\mathcal{G}_N$ as a chain with equal edge weights, our model will reduce to the existing time-varying models according to different ψ. Specifically, if we select ψ as ℓ_1, ℓ_2, and squared ℓ_2 norms, pattern graphs will reduce to the regularizers of the temporal

homogeneity assumption [20], [43], the abrupt change assumption [19], and the smooth change assumption [18], respectively. In addition to time-varying models, if we build $\mathcal{G}_N$ as a graph with weighted edges and select ψ as ℓ_1 norm, our penalty will boil down to the regularizers of [27], [28].

5) Theoretical guarantee: In this part, our goal is to establish estimation error bounds for the proposed graph estimator, from which the influence of some factors on the estimation performance can be revealed. Some critical technical assumptions are first made before we go any further.

Assumption 1: The collected data $\mathbf{x}$ is bounded, as well as the corresponding distance vector $\mathbf{r}$, i.e., there exists a constant C_r such that $\|\mathbf{r}\|_2 \leq C_r$.

Assumption 2: The gradient of $g(\mathbf{w})$ are bounded, i,e, there exist a constant C_g such that $\|\nabla g_{\mathbf{w}}(\mathbf{w})\|_2 \leq C_g$, for all possible $\mathbf{w}$.

Assumption 1 naturally holds in real-world applications. Assumption 2 is common in theoretic analysis, and a similar assumption is made in [45]. The possible graphs in Assumption 2 are those where all node degrees are greater than zero since the log-barrier term in $g(\mathbf{w})$ avoids isolated vertices. Without loss of generality, we assume the data sizes of T graphs are the same, i.e., $N_t = N$ for $t = 1, \ldots, T$. The general case where different graphs have different numbers of data can be similarly analyzed. First, suppose that the distance vector $\mathbf{r}_t$ satisfies

$$(\mathbf{r}_t)_{[i]} = (\mathbf{r}_t^*)_{[i]} + (\mathbf{e}_t)_{[i]}, \tag{11}$$

where $\mathbf{r}_t^*$ is the true distance vector of t-th graph, and $\mathbf{e}_t$ is an error vector caused by noisy measurements. We assume that $(\mathbf{e}_t)_{[i]} \sim Gauss(0, \sigma_e^2)$ for $i = 1, \ldots, p$, and all noises are assumed to be independent of each other. We denote $\widehat{\mathbf{W}} = [\widehat{\mathbf{w}}_1, \ldots, \widehat{\mathbf{w}}_T]$ as the graphs learned by our model and $\mathbf{W}^* = [\mathbf{w}_1^*, \ldots, \mathbf{w}_T^*]$ as the real graphs. We then analyze the upper bound on the estimation error between graphs learned from noisy data using our model and the real graphs. The result is presented in the following theorem, and we only provide the analysis of $\psi = \ell_1$ and ℓ_2 norms due to space limitation.

Theorem 1: Under Assumptions 1-2, given $\delta, \eta, \beta > 0$, and a pattern graph $\mathcal{G}_N$ with an adjacency matrix $\mathbf{A}_N$ describing the relationships among T graphs, we have probability at least $1 - \exp(-\frac{1}{2}(\delta - pT\log(1 + \frac{\delta}{pT})))$ such that

$$\begin{aligned} &\|\widehat{\mathbf{W}} - \mathbf{W}^*\|_{\mathrm{F}} \\ &\leq \frac{C_g\sqrt{T} + \eta\sqrt{S\|\mathbf{L}_\gamma\|_2}}{\beta} + \frac{\sigma_e\sqrt{pT+\delta} + \sqrt{T}C_r}{N\beta}, \end{aligned} \tag{12}$$

where $\mathbf{L}_\gamma = \mathbf{D}_\gamma - \mathbf{A}_\gamma$. Moreover, $\mathbf{A}_\gamma = \mathbf{A}_N \circ \mathbf{A}_N$ and $\mathbf{D}_\gamma$ is a diagonal matrix with $(\mathbf{D}_\gamma)_{[ii]} = \sum_{j=1}^{T}(\mathbf{A}_\gamma)_{[ij]}$.

Proof: See Appendix A. □

Theorem 1 provides an upper bound on the estimation error of our proposed model. For a specific model, i.e., given model parameters α, β, η, and pattern graph $\mathcal{G}_N$, the estimation error of learning T graphs using this model should not exceed the upper bound. The theorem characterizes the estimation error bound w.r.t. some key parameters, such as the number of graphs T, sample size N, the number of free variables p, and the measurement noise level σ_e. The upper bound consists of two parts, and the first part is determined by the collected data, i.e., $\frac{\sigma_e\sqrt{pT+\delta}+\sqrt{T}C_r}{N\beta}$, which will diminish to zero with N. The second part is determined by the regularizers of our model, i.e., $g(\mathbf{w})$ and the pattern graph $\mathcal{G}_N$. Compared with IGL, the proposed pattern graph will cause an additional term in the upper bound, $\eta\sqrt{S\|\mathbf{L}_\gamma\|_2}/\beta$, which depends on the topology of $\mathcal{G}_N$. The regularizer-dependent part does not decrease to zero with N because it is data-independent, meaning that we cannot reduce the estimation errors caused by improper pattern graphs by increasing the amount of data. This motivates us to devise a method for building an accurate pattern graph, especially when no topological prior is available. In the literature, some other MGL models, such as [17], [23], [30], also derive their estimation error bounds. Similar to our work, their error bounds are related to data dimension, data size, the number of graphs to estimate, and some other factors. However, their results are derived in a noise-free setting, whereas our results account for data perturbations.

B. MGL With Automatically Learned Topological Patterns

To alleviate the limitation that (9) requires a handcrafted $\mathcal{G}_N$, we propose a data-driven approach that can learn $\mathcal{G}_N$ automatically. As the previous statement, it is preferred that the learned $\mathcal{G}_N$ is well-aligned with the similarity between $\mathcal{G}_1, \ldots, \mathcal{G}_T$. Specifically, large γ_{ij} should be assigned to the edge connecting two closer graphs. Fortunately, this is close to the idea of smoothness stated in Section III-A. Therefore, a natural way is to learn $\mathcal{G}_N$ by borrowing the formulation of the smoothness based graph learning. A closer look at (9) shows that the term $\psi(\mathbf{w}_i - \mathbf{w}_j)$ can be regarded as a "distance" between $\mathbf{w}_i$ and $\mathbf{w}_j$. Thus, similar to (6), we first define a distance matrix $\mathbf{C}_\psi \in \mathbb{R}^{T\times T}$ as

$$(\mathbf{C}_\psi)_{[ij]} = \psi(\mathbf{w}_i - \mathbf{w}_j). \tag{13}$$

The regularizer $\sum_{(i,j)\in\mathcal{G}_N}\gamma_{ij}\psi(\mathbf{w}_i - \mathbf{w}_j)$ can be then rewritten as $\boldsymbol{\gamma}^\top\boldsymbol{\psi}$, where $\boldsymbol{\gamma} \in \mathbb{R}^{T(T-1)/2}$ represents the vectorized adjacency matrix of $\mathcal{G}_N$, and $\boldsymbol{\psi} \in \mathbb{R}^{T(T-1)/2}$ contains the upper triangle elements of $\mathbf{C}_\psi$, same as the definition of $\mathbf{r}$. The $\boldsymbol{\gamma}^\top\boldsymbol{\psi}$ has the same form as the smoothness based term in (7), but differs in that $\mathbf{r}$ is calculated using squared ℓ_2 norm, while $\boldsymbol{\psi}$ is based on any distance metric function ψ. Borrowing the formulation of (7), given T graphs $\mathbf{w}_1, \ldots, \mathbf{w}_T$, we aim to learn $\mathcal{G}_N$ using

$$\min_{\boldsymbol{\gamma}\geq 0} \boldsymbol{\gamma}^\top\boldsymbol{\psi} + g_\gamma(\boldsymbol{\gamma}), \tag{14}$$

where $g_\gamma(\boldsymbol{\gamma})$ is used to ensure the learned $\mathcal{G}_N$ acquire some desired properties. Following $g(\mathbf{w})$, we choose $g_\gamma(\boldsymbol{\gamma}) = -\alpha_\gamma\mathbf{1}^\top\log(\widetilde{\mathbf{S}}\boldsymbol{\gamma}) + \beta_\gamma\|\boldsymbol{\gamma}\|_2^2$. The definitions of $\widetilde{\mathbf{S}}$, α_γ and β_γ are the same as those in $g(\mathbf{w})$. Since we need to learn $\mathcal{G}_1, \ldots, \mathcal{G}_T$ and $\mathcal{G}_N$ simultaneously, we combine (9) and (14) and obtain

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T\geq 0,\boldsymbol{\gamma}\geq 0} \sum_{t=1}^{T} f_t(\mathbf{w}_t) + \eta\left(\boldsymbol{\gamma}^\top\boldsymbol{\psi} + g_\gamma(\boldsymbol{\gamma})\right). \tag{15}$$

The problem is bi-convex w.r.t. $\boldsymbol{\gamma}$ and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_T]$.

An extension: If we have learned $\mathcal{G}_1, \ldots, \mathcal{G}_T$ through (15), now given a new set of samples $\mathbf{X}_{T+1}$, how can we learn the new graph $\mathcal{G}_{T+1}$? Clearly, it is computationally expensive to resolve (15) with all $T+1$ datasets. As a remedy, we propose an approach to incrementally learn $\mathcal{G}_{T+1}$ based on the already learned graphs, $\mathbf{w}_1, \ldots, \mathbf{w}_T$, i.e.,

$$\min_{\mathbf{w}_{T+1}\geq 0, \gamma_{T+1,t}\geq 0} f_{T+1}(\mathbf{w}_{T+1}) + \eta \left(\sum_{t=1}^{T} \gamma_{T+1,t}\, \psi(\mathbf{w}_t - \mathbf{w}_{T+1}) - \alpha_\gamma \log\left(\sum_{t=1}^{T} \gamma_{T+1,t} \right) + \beta_\gamma \sum_{t=1}^{T} \gamma_{T+1,t}^2 \right). \tag{16}$$

The formulation is similar to that in (15), but we only learn the $(T+1)$-th graph $\mathbf{w}_{T+1}$ and its connections with the already learned graphs, i.e., $\gamma_{T+1,t}$, where $t = 1, \ldots T$.

Remark 1: The parameters of (9) and (15) are $\alpha, \beta, \eta, \alpha_\gamma$ and β_γ. As stated in Proposition 2 in [11], there exists a scaling effect when we tune α w.r.t. β. We can fix α as a constant and search only β. This also holds for α_γ and β_γ in (15). Thus, the free parameters are β, η, and β_γ.

V. Proposed Algorithm

In this section, we develop an algorithm for solving the problem (15). Our algorithm is based on the block coordinate descent (BCD) algorithm, i.e., updating $\mathcal{G}_1, \ldots, \mathcal{G}_T$ and $\mathcal{G}_N$ alternatively. The algorithm solving each subproblem should be fully decentralized since we are reluctant to allow a third-party central server to access all the data.

A. The Decentralized Algorithm

For the sake of brevity, we omit the iteration number K of the outer loop of the BCD algorithm, and only discuss the update in one iteration.

1) Fix $\mathcal{G}_N$ and update $\mathcal{G}_1, \ldots, \mathcal{G}_T$: We utilize the ADMM algorithm to solve the subproblem of fixing $\mathcal{G}_N$, i.e., the problem (9). We employ the ADMM algorithm for the following reasons. First, compared with conventional algorithms, the ADMM algorithm can update T graphs in parallel depsite they are coupled with each other in the regularizer $\sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi(\mathbf{w}_i - \mathbf{w}_j)$. This can improve the efficiency of the algorithm especially when T is large. Second, the ADMM provides an efficient message-passing approach to jointly learn multiple graphs under privacy constraints. We will explain in detail below.

To solve our problem via the ADMM algorithm, we introduce a copy of $\mathbf{w}_i$, denoted as $\mathbf{z}_{ij}$, at each edge $(i,j) \in \mathcal{E}_N$. The same edge also has a $\mathbf{z}_{ji}$, which is a copy of $\mathbf{w}_j$. The subproblem (9) is equivalent to the following problem

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T\geq 0} \sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}_t) + \eta \sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi(\mathbf{z}_{ij} - \mathbf{z}_{ji}) \quad \text{s.t. } \mathbf{w}_i = \mathbf{z}_{ij}, \text{ for } i = 1, \ldots, T \text{ and } j \in \mathcal{N}(i), \tag{17}$$

where $\mathcal{N}(i)$ denotes the set containing all neighboring nodes of i. We define a matrix $\mathbf{W} \in \mathbb{R}^{p\times T} \triangleq [\mathbf{w}_1, \ldots, \mathbf{w}_T]$ containing all primal variables. Similarly, matrices of the consensus variables $\mathbf{Z} \in \mathbb{R}^{p\times 2S}$ and the dual variables $\mathbf{U} \in \mathbb{R}^{p\times 2S}$ are also defined. For the n-th edge (i,j) in $\mathcal{G}_N$, $n = 1, \ldots, S$, the corresponding consensus variable vectors $\mathbf{z}_{ij}$ and $\mathbf{z}_{ji}$ are the $(2n-1)$-th and $2n$-th columns of $\mathbf{Z}$. This also holds for the matrix of dual variables $\mathbf{U}$. In our setup, we suppose $\mathbf{w}_1, \ldots, \mathbf{w}_T$ are located in T local clients respectively, and the client t also stores all variables $\mathbf{z}_{ti}$ and $\mathbf{u}_{ti}$ for $i \in \mathcal{N}(t)$. We then write the scaled augmented Lagrangian of (17) as

$$\begin{aligned} & L_\rho(\mathbf{W}, \mathbf{Z}, \mathbf{U}) \\ &= \sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}_t) + \eta \sum_{(i,j)\in\mathcal{E}_N} \gamma_{ij}\psi(\mathbf{z}_{ij} - \mathbf{z}_{ji}) \\ &\quad + \sum_{(i,j)\in\mathcal{E}_N} \Big(-\frac{\rho}{2}\left(\|\mathbf{u}_{ij}\|_2^2 + \|\mathbf{u}_{ji}\|_2^2\right) \\ &\quad + \frac{\rho}{2}\left(\|\mathbf{w}_i - \mathbf{z}_{ij} + \mathbf{u}_{ij}\|_2^2 + \|\mathbf{w}_j - \mathbf{z}_{ji} + \mathbf{u}_{ji}\|_2^2\right) \Big), \end{aligned} \tag{18}$$

where $\rho > 0$ is an ADMM penalty parameter [46]. The update procedure of the ADMM framework consists of

$$\mathbf{W}^{k+1} = \operatorname*{argmin}_{\mathbf{W}\geq 0} L_\rho\left(\mathbf{W}, \mathbf{Z}^k, \mathbf{U}^k\right), \tag{19}$$

$$\mathbf{Z}^{k+1} = \operatorname*{argmin}_{\mathbf{Z}} L_\rho\left(\mathbf{W}^{k+1}, \mathbf{Z}, \mathbf{U}^k\right), \tag{20}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{W}^{k+1} - \mathbf{Z}^{k+1}. \tag{21}$$

In the following, we solve the problem (17) by iteratively updating $\mathbf{W}, \mathbf{Z}$ and $\mathbf{U}$.

Update $\mathbf{W}$: The update of the columns of $\mathbf{W}$ is

$$\begin{aligned} & \left(\mathbf{w}_t^{k+1}, 1 \leq t \leq T\right) \\ &= \operatorname*{argmin}_{\mathbf{w}_1,\ldots,\mathbf{w}_T\geq 0} \sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}_t) \\ &\quad + \frac{\rho}{2} \sum_{(i,j)\in\mathcal{E}_N} \left(\|\mathbf{w}_i - \mathbf{z}_{ij}^k + \mathbf{u}_{ij}^k\|_2^2 + \|\mathbf{w}_j - \mathbf{z}_{ji}^k + \mathbf{u}_{ji}^k\|_2^2\right). \end{aligned} \tag{22}$$

Obviously, we can update each $\mathbf{w}_t^{k+1}$ separately,

$$\mathbf{w}_t^{k+1} = \operatorname*{argmin}_{\mathbf{w}_t\geq 0} f_t(\mathbf{w}_t) + \frac{\rho}{2} \sum_{j\in\mathcal{N}(t)} \|\mathbf{w}_t - \mathbf{z}_{tj}^k + \mathbf{u}_{tj}^k\|_2^2. \tag{23}$$

Let $\boldsymbol{\theta}_t^k \triangleq \frac{1}{n_t}\sum_{j\in\mathcal{N}(t)}(\mathbf{z}_{tj}^k - \mathbf{u}_{tj}^k)$, where $n_t = |\mathcal{N}(t)|$, (23) can be reformulated as

$$\begin{aligned} \mathbf{w}_t^{k+1} &= \operatorname*{argmin}_{\mathbf{w}_t\geq 0} f_t(\mathbf{w}_t) + \frac{n_t\rho}{2}\|\mathbf{w}_t - \boldsymbol{\theta}_t^k\|_2^2 \\ &\triangleq \operatorname*{argmin}_{\mathbf{w}_t\geq 0} \breve{f}_t(\mathbf{w}_t). \end{aligned} \tag{24}$$

We use a projected gradient descent (PGD) algorithm [47] to solve this problem. Specifically, the gradient of the objective function of (24) is as follows

$$\nabla_{\mathbf{w}_t} \breve{f}_t(\mathbf{w}_t) = \mathbf{r}_t/N_t + \nabla_{\mathbf{w}_t} g(\mathbf{w}_t) + n_t\rho\left(\mathbf{w}_t - \boldsymbol{\theta}_t^k\right), \tag{25}$$

where $\nabla_{\mathbf{w}_t} g(\mathbf{w}_t) = -\alpha\mathbf{S}^\top(\mathbf{S}\mathbf{w}_t)^{\cdot(-1)} + 2\beta\mathbf{w}_t$, and $.(-1)$ is an element-wise operator. With this gradient, we solve (24) in an

iterative way. In iteration k, we use a dummy variable $\widetilde{\mathbf{w}}_t$ and set $\widetilde{\mathbf{w}}_t^0 = \mathbf{w}_t^k$. We then update $\widetilde{\mathbf{w}}_t^r$ using

$$\widetilde{\mathbf{w}}_t^{r+1} = \max\left(\widetilde{\mathbf{w}}_t^r - \nu_w^r \nabla_{\mathbf{w}_t} \breve{f}_t(\widetilde{\mathbf{w}}_t^r),\ 0\right) \tag{26}$$

until it converges to $\widetilde{\mathbf{w}}_t^*$, where r denotes the iteration number of PGD, and ν_w is the stepsize. Here, we use the line search method to determine ν_w [48]. Finally, we let $\mathbf{w}_t^{k+1} = \widetilde{\mathbf{w}}_t^*$.

Update $\mathbf{Z}$*:* For each edge $(i,j) \in \mathcal{E}_N$, we update the corresponding column vectors $\mathbf{z}_{ij}, \mathbf{z}_{ji}$ of $\mathbf{Z}$ as follows

$$\begin{aligned}&\mathbf{z}_{ij}^{k+1}, \mathbf{z}_{ji}^{k+1}\\ &= \underset{\mathbf{z}_{ij},\mathbf{z}_{ji}}{\operatorname{argmin}}\ \eta\gamma_{ij}\psi(\mathbf{z}_{ij} - \mathbf{z}_{ji})\\ &\quad + \frac{\rho}{2}\left(\left\|\mathbf{w}_i^{k+1} - \mathbf{z}_{ij} + \mathbf{u}_{ij}^k\right\|_2^2 + \left\|\mathbf{w}_j^{k+1} - \mathbf{z}_{ji} + \mathbf{u}_{ji}^k\right\|_2^2\right).\end{aligned} \tag{27}$$

It is difficult to solve (27) due to that variables $\mathbf{z}_{ij}$ and $\mathbf{z}_{ji}$ are coupled with each other in $\psi(\mathbf{z}_{ij} - \mathbf{z}_{ji})$. Inspired by the method proposed in [20], we define a function $\widetilde{\psi}$ satisfying

$$\widetilde{\psi}\left(\begin{bmatrix}\mathbf{z}_{ij}\\ \mathbf{z}_{ji}\end{bmatrix}\right) = \psi(\mathbf{z}_{ji} - \mathbf{z}_{ij}). \tag{28}$$

With this definition, (27) can be solved separately,

$$\begin{bmatrix}\mathbf{z}_{ij}^{k+1}\\ \mathbf{z}_{ji}^{k+1}\end{bmatrix} = \operatorname{prox}_{\frac{\eta\gamma_{ij}}{\rho}\widetilde{\psi}}\left(\begin{bmatrix}\mathbf{u}_{ij}^k + \mathbf{w}_i^{k+1}\\ \mathbf{u}_{ji}^k + \mathbf{w}_j^{k+1}\end{bmatrix}\right), \tag{29}$$

where $\operatorname{prox}_{\frac{\eta\gamma_{ij}}{\rho}\widetilde{\psi}}(\cdot)$ is the proximal operator of function $\frac{\eta\gamma_{ij}}{\rho}\widetilde{\psi}$ [49]. However, the closed form of the proximal operator is unknown, and thus we introduce the following property [20].

Property 1 ([20]): If a function l_1 is a composition of another function l_2 with an orthogonal affine transformation,i.e., $l_1(\mathbf{a}) = l_2(\mathbf{G}\mathbf{a} + \mathbf{b})$, and $\mathbf{G}\mathbf{G}^\top = \frac{1}{\lambda}\mathbf{I}$, then

$$\operatorname{prox}_{l_1}(\mathbf{a}) = (\mathbf{I} - \lambda\mathbf{G}^\top\mathbf{G})\mathbf{a} + \lambda\mathbf{G}^\top\left(\operatorname{prox}_{\frac{l_2}{\lambda}}(\mathbf{G}\mathbf{a} + \mathbf{b}) - \mathbf{b}\right). \tag{30}$$

In our problem, $l_1 = \widetilde{\psi}$, $l_2 = \psi$, $\mathbf{G} = [-\mathbf{I}\ \ \mathbf{I}]$, $\mathbf{b}$ is a vector of zeros, and $\lambda = \frac{1}{2}$. With Property 1, the following update can be easily reached

$$\begin{aligned}\begin{bmatrix}\mathbf{z}_{ij}^{k+1}\\ \mathbf{z}_{ji}^{k+1}\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}\mathbf{u}_{ij}^k + \mathbf{w}_i^{k+1} + \mathbf{u}_{ji}^k + \mathbf{w}_j^{k+1}\\ \mathbf{u}_{ij}^k + \mathbf{w}_i^{k+1} + \mathbf{u}_{ji}^k + \mathbf{w}_j^{k+1}\end{bmatrix}\\ &\quad + \frac{1}{2}\begin{bmatrix}-\operatorname{prox}_{\frac{2\eta\gamma_{ij}}{\rho}\psi}\left(\mathbf{w}_j^{k+1} + \mathbf{u}_{ji}^k - \mathbf{w}_i^{k+1} - \mathbf{u}_{ij}^k\right)\\ \operatorname{prox}_{\frac{2\eta\gamma_{ji}}{\rho}\psi}\left(\mathbf{w}_j^{k+1} + \mathbf{u}_{ji}^k - \mathbf{w}_i^{k+1} - \mathbf{u}_{ij}^k\right)\end{bmatrix}.\end{aligned} \tag{31}$$

When updating $\mathbf{z}_{ij}$, client i requires the variables $\mathbf{w}_j$ and $\mathbf{u}_{ji}$ from its neighboring clients $j \in \mathcal{N}(i)$.

Update $\mathbf{U}$*:* For each edge $(i,j) \in \mathcal{E}_N$, the corresponding columns of $\mathbf{U}$ can be updated by

$$\begin{aligned}\mathbf{u}_{ij}^{k+1} &= \mathbf{u}_{ij}^k + \mathbf{w}_i^{k+1} - \mathbf{z}_{ij}^{k+1}\\ \mathbf{u}_{ji}^{k+1} &= \mathbf{u}_{ji}^k + \mathbf{w}_j^{k+1} - \mathbf{z}_{ji}^{k+1}.\end{aligned} \tag{32}$$

Algorithm 1: The Algorithm for (15).

Input:
$\alpha, \beta, \alpha_r, \beta_r, \eta, \rho, \epsilon_w, k_{\max}, \epsilon_W$, signals $\mathbf{X}_1$,..., $\mathbf{X}_T$

Initialize $\mathbf{w}_t^{(0,0)}$ for $t \in \mathcal{V}_N$, $\mathbf{z}_{ij}^{(0,0)}$ and $\mathbf{u}_{ij}^{(0,0)}$ for all $(i,j) \in \mathcal{E}_N$, $\boldsymbol{\gamma}^{(0,0)}$, $\boldsymbol{\gamma}^0$, $error > \epsilon_W$, set $k, K = 0$
while $error > \epsilon_W$ **do**
  // Fix $\mathcal{G}_N$ *and update* $\mathcal{G}_1, \ldots, \mathcal{G}_T$
  All clients determine their neighboring nodes via $\boldsymbol{\gamma}^K$
  while iterates not converge, all clients separately **do**
    Update $\mathbf{w}_t^{(K,k+1)}$ using PGD in client t, $t = 1, \ldots, T$
    Each client t sends $\mathbf{w}_t^{(K,k+1)}, \mathbf{u}_{tj}^{(K,k)}$ to its neighbouring clients, where $t = 1, \ldots, T$ and $j \in \mathcal{N}(t)$
    Update $\mathbf{z}_{tj}^{(K,k+1)}$ using (31) in client t, $t = 1, \ldots, T$ and $j \in \mathcal{N}(t)$
    Update $\mathbf{u}_{tj}^{(K,k+1)}$ using (32) in client t, $t = 1, \ldots, T$ and $j \in \mathcal{N}(t)$
    $k = k + 1$
  end while
  Let $\mathbf{w}_t^{K+1} = \mathbf{w}_t^{(K+1,0)} = \mathbf{w}_t^{(K,k)}$, and then set $k = 0$
  // Fix $\mathcal{G}_1, \ldots, \mathcal{G}_T$ *and update* $\mathcal{G}_N$
  while iterates not converge **do**
    Select an client $m \in \{1, \ldots, T\}$ and form the set $\mathcal{M}$
    The client m requests the latest degree deg_t and graph $\mathbf{w}_t^{K+1}$ from the clients $t \in \mathcal{M}$
    Update $\boldsymbol{\gamma}_{[\mathcal{T}]}^{(K,k+1)}$ using (34) in client m
    Send the updated element $(m,t) \in \mathcal{T}$ of $\boldsymbol{\gamma}_{[\mathcal{T}]}^{(K,k+1)}$ to the associated client $t \in \mathcal{M}$
    $k = k + 1$
  end while
  Let $\boldsymbol{\gamma}^{K+1} = \boldsymbol{\gamma}^{(K+1,0)} = \boldsymbol{\gamma}^{(K,k)}$, and then set $k = 0$
  Calculate $error = \|\mathbf{W}^{K+1} - \mathbf{W}^K\|_{\mathrm{F}} / \|\mathbf{W}^K\|_{\mathrm{F}}$
  $K = K + 1$
end while
return $\mathbf{w}_1^K$, ..., $\mathbf{w}_T^K$ and $\boldsymbol{\gamma}^K$

We update $\mathbf{W}, \mathbf{Z}$ and $\mathbf{U}$ alternatively until the solution converges. The adopted stop criterion is that the relative differences $\|\mathbf{w}_t^{k+1} - \mathbf{w}_t^k\|_2 / \|\mathbf{w}_t^k\|_2, t = 1, \ldots, T$, are all smaller than a predefined tolerance ϵ_w.

2) Fix $\mathcal{G}_1, \ldots, \mathcal{G}_T$ *and update* $\mathcal{G}_N$*:* Given $\mathbf{w}_1, \ldots, \mathbf{w}_T$, the subproblem becomes that

$$\min_{\boldsymbol{\gamma} \geq 0} h(\boldsymbol{\gamma}) = \min_{\boldsymbol{\gamma} \geq 0} \eta\boldsymbol{\gamma}^\top\boldsymbol{\psi} + \eta g_\gamma(\boldsymbol{\gamma}). \tag{33}$$

This problem is similar to the classic graph learning problem (7). Therefore, a feasible solution is to collect all local graphs to a central server and learn $\mathcal{G}_N$ by some widely-used algorithms like PDS algorithm in [11]. However, in this study, we exploit a decentralized algorithm to solve the problem to avoid an unreliable center server collecting all learned graphs.

In the k-th iteration, client t only stores the entries of $\boldsymbol{\gamma}^k$ corresponding to edges in $\{(t,i) : i = 1, \ldots, T\}$. We denote the set containing all the above entries as $\mathcal{S}(t)$, and $\boldsymbol{\gamma}_{[\mathcal{S}(t)]}^k$ is the

sub-vector of $\boldsymbol{\gamma}^k$ stored at client t. Following the framework of peer sampling [42], [50], at iteration k, we randomly select a client m and sample without replacement a set $\mathcal{M}$ of M clients from the set $\{1,\ldots,T\}\backslash\{m\}$. We only update those entries of $\boldsymbol{\gamma}$ corresponding to the edge set $\mathcal{T}=\{(m,t):t\in\mathcal{M}\}$ at client m, which is denoted as $\boldsymbol{\gamma}_{[\mathcal{T}]}$. At the start of this iteration, the client m first requests the degree $deg_t^k=\mathbf{1}^\top\boldsymbol{\gamma}^k_{[\mathcal{S}(t)]}$ from all clients $t\in\mathcal{M}$ to construct a vector $\mathbf{d}^k_{\mathcal{M}}=(deg_t^k)_{t\in\mathcal{M}}$. The client m also requests the learned graphs $\mathbf{w}_t$ from clients $t\in\mathcal{M}$ and calculates the distance vector $\boldsymbol{\psi}^k_{\mathcal{T}}=(\psi(\mathbf{w}_m-\mathbf{w}_t))_{(m,t)\in\mathcal{T}}$, where the subscript denotes the entries that only related to $\mathcal{T}$. The client m then performs the following update

$$\boldsymbol{\gamma}^{k+1}_{[\mathcal{T}]}=\max\left(\boldsymbol{\gamma}^k_{[\mathcal{T}]}-\nu^k_\gamma\nabla h\left(\boldsymbol{\gamma}^k_{[\mathcal{T}]}\right),0\right), \tag{34}$$

where ν_γ is a stepsize and selected in the same way as ν_w. The gradient $\nabla h(\boldsymbol{\gamma}^k_{[\mathcal{T}]})$ can be calculated as

$$\nabla h\left(\boldsymbol{\gamma}^k_{[\mathcal{T}]}\right)=\eta\left(\boldsymbol{\psi}^k_{\mathcal{T}}-\alpha_\gamma\widetilde{\mathbf{S}}^\top_{\mathcal{M},\mathcal{T}}\left(\frac{1}{\mathbf{d}^k_{\mathcal{M}}}\right)+2\beta_\gamma\boldsymbol{\gamma}^k_{[\mathcal{T}]}\right), \tag{35}$$

where $\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}$ means selecting the rows and columns of $\widetilde{\mathbf{S}}$ corresponding to $\mathcal{M}$ and $\mathcal{T}$, respectively. After the update, the client m sends the updated entries $(m,t)\in\mathcal{T}$ of $\boldsymbol{\gamma}^{k+1}_{[\mathcal{T}]}$ to the associated client t. The algorithm stops after reaching a predetermined maximum number of iterations $k_{\max}$.

B. Privacy Analysis

When we update $\mathcal{G}_1,\ldots,\mathcal{G}_T$ based on the ADMM algorithm, the updates of $\mathbf{W}$ and $\mathbf{U}$ only require the data and variables stored in the corresponding local client, without any privacy leakage. In the step of updating $\mathbf{Z}$, client i needs variables $\mathbf{w}_j$ and $\mathbf{u}_{ji}$ from its adjacent client $j\in\mathcal{N}(i)$. However, exchanging variables does not involve any raw data. On the other hand, when we update $\mathcal{G}_N$, the commuted data are the stored degrees and graphs of the sampled clients, which does not violate privacy constraints. In summary, our algorithm can jointly learn all graphs in a decentralized manner without private data leaving the local clients.

C. Convergence Analysis

Firstly, we focus on the algorithm for solving (9). Note that the problem (9) is convex, and thus the ADMM based algorithm is guaranteed to converge to the global optimum of (9); see [46] for the detailed proof. Next, we provide the convergence analysis of the algorithm for updating $\boldsymbol{\gamma}$.

Proposition 1: Let k be the number of running iterations when updating $\boldsymbol{\gamma}$ in the outer iteration K. We have $\mathbb{E}[h(\boldsymbol{\gamma}^{(K,k)})-h^*]\leq\mu^k(h(\boldsymbol{\gamma}^{(K,0)})-h^*)$, where h^* is the optimal objective value of (33) for given $\mathbf{w}_1,..,\mathbf{w}_T$, and $\mu=1-\frac{2MC_s}{T(T-1)L_h}$. The constant $C_s=2\eta\beta_\gamma$ and $L_h=\eta\alpha_\gamma\sqrt{(M+1)(T+M-2)}/deg^2_{\min}+2\eta\beta_\gamma$, where $deg_{\min}$ is the smallest degree of $\mathcal{G}_N$ when updating $\boldsymbol{\gamma}$.

Proof: See Appendix B. □

From Proposition 1, we find that the convergence speed is dependent on M, i.e., the number of the sampled clients per iteration. More sampled clients mean faster convergence speed, as well as more transmitted data per iteration. However, the trade-off between the convergence speed and the communication cost is not the focus of this study, and we simply set $M=\lfloor\frac{T}{2}\rfloor$, where $\lfloor\cdot\rfloor$ means round down. Next, we establish the convergence analysis of the BCD based algorithm. The following assumptions are first made.

Proposition 2: The alternate updating of $\mathbf{W}=[\mathbf{w}_1,\ldots,\mathbf{w}_T]$ and $\boldsymbol{\gamma}$ in Algorithm 1 monotonically decrease the objective function value of (15) in each iteration until convergence. Furthermore, when ψ is differentiable, the algorithm will converge to a stationary point.

Proof: Suppose $F(\mathbf{W},\boldsymbol{\gamma})$ is the objective function of (15). At K-th iteration, when we fix $\boldsymbol{\gamma}$ and update $\mathbf{W}$, the subproblem is strictly convex, and the proposed ADMM algorithm is guaranteed to find the unique minimum [46]. Therefore, the overall objective function of (15) decreases monotonically, i.e.

$$F(\mathbf{W}^{K+1},\boldsymbol{\gamma}^K)\leq F(\mathbf{W}^K,\boldsymbol{\gamma}^K). \tag{36}$$

On the other hand, when we fix $\mathbf{W}$ and update $\boldsymbol{\gamma}$, the subproblem (33) is also strictly convex. Proposition 1 proves that the proposed algorithm can achieve the unique minimum. Then, we can obtain the following inequality

$$F(\mathbf{W}^{K+1},\boldsymbol{\gamma}^{K+1})\leq F(\mathbf{W}^{K+1},\boldsymbol{\gamma}^K). \tag{37}$$

As a result, the overall objective function value of (15) decreases monotonically in each iteration until Algorithm 1 converges. Furthermore, when ψ is differentiable, the objective function $F(\mathbf{W},\boldsymbol{\gamma})$ is also continuously differentiable. Also, each subproblem of $F(\mathbf{W},\boldsymbol{\gamma})$ can attain its unique minimum. Therefore, we can directly use Proposition 3.7.1 of [51] and conclude that the algorithm can converge to a stationary point. □

Remark 2: The algorithm for learning a new graph incrementally (16) is similar to Algorithm 1, and we place the algorithm in Appendix C. Moreover, although the algorithm is decentralized, it is also applicable to centralized scenarios.

D. Complexity Analysis

We only analyze the computational cost of one outer iteration in Algorithm 1. The analysis of the ADMM based algorithm is first provided. When we update $\mathbf{w}_t$, the most time-consuming operation is calculating $\nabla_{\mathbf{w}_t}\breve{f}(\mathbf{w}_t)$, which costs order $\mathcal{O}(d^2)$ flops. Suppose that the maximum number of iterations taken to reach convergence in the PGD algorithm is k_1 for all clients $t=1,\ldots,T$. The total cost of updating $\mathbf{w}_t$ in T local clients is at most $\mathcal{O}(Tk_1\tilde{}d^2)$ per iteration. When updating $\mathbf{z}$ and $\mathbf{u}$, all operations are element-wise, including the proximal operator $\text{prox}(\cdot)$. Thus, the required cost is $\mathcal{O}(Sp)$. The overall cost of ADMM algorithm is then $\mathcal{O}(k_2(Tk_1\tilde{}d^2+Sp))$, where k_2 is the required number of iterations for the ADMM to converge. As illustrated in [46], the ADMM algorithm can converge to a modest accuracy quite quickly, indicating that k_2 is usually a small number. When updating $\boldsymbol{\gamma}$, we need to update $\boldsymbol{\gamma}_{\mathcal{T}}$ using (34). The computational burden mainly lies in (35), which costs $\mathcal{O}(Md+M^2)$ per iteration. The overall cost for updating $\boldsymbol{\gamma}$ is hence $\mathcal{O}(k_{\max}(Md+M^2))$.

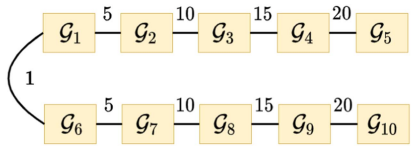


Fig. 2. Topological relationships among graphs in Pattern 1.

VI. Experiments

In this section, the performance of our proposed framework is validated by both synthetic and real-world data. In the real-world data experiments, we consider some scenarios where data are privacy-sensitive. Firstly, let's introduce some experimental setups.

A. Experimental Setups

1) Multiple graphs generation: Two topological patterns are tested in this study. We generate graphs in Pattern 1 following the structure displayed in Fig. 2. An initial graph $\mathcal{G}_1$ is first generated, and five edges of $\mathcal{G}_1$ are randomly changed to obtain $\mathcal{G}_2$. We then generate $\mathcal{G}_3$ by changing ten edges of $\mathcal{G}_2$, as indicated in Fig. 2. Following this way, we can build the remaining graphs in Pattern 1. Different from chain structure, $\mathcal{G}_6$ is generated based on $\mathcal{G}_1$ instead of $\mathcal{G}_5$. We then construct prior $\mathcal{G}_N$ of this pattern with the same connections shown in Fig. 2. Edge weights of this $\mathcal{G}_N$ are inverse proportion with the number of changed edges and satisfy $\sum_{(i,j)\in\mathcal{G}_N}\gamma_{ij}=1$ as previously stated. For Pattern 2, we generate graphs belonging to three clusters. Each cluster contains three graphs, i.e., $\mathcal{G}_1,\mathcal{G}_4,\mathcal{G}_7$ in cluster 1, $\mathcal{G}_2,\mathcal{G}_5,\mathcal{G}_8$ in cluster 2, and $\mathcal{G}_3,\mathcal{G}_6,\mathcal{G}_9$ in cluster 3. In each cluster, we generate an initial graph and change five edges of the initial graph randomly to generate the remaining graphs. The prior $\mathcal{G}_N$ of this pattern has three clusters, and graphs within clusters are fully connected with edge weights equal to 1, while graphs of two different clusters are not connected. Two types of initial random graphs are generated, which are Gaussian radial basis function (RBF) random graphs [12] and Erdös–Rényi random graphs (ER) [52]. The initial RBF graph is generated by following the same way as in [12]. The kernel width of the RBF function is set to 0.5. As for ER graph, we set the connection probability between two nodes to 0.25. The number of nodes of all graphs in our synthetic data experiments is 20.

2) Smooth graph signals generation: For t-th graph $\mathcal{G}_t$, we first calculate the Laplacian matrix $\mathbf{L}_t$ and then generate N_t graph signals from the Gaussian distribution defined by [12]

$$\mathbf{x}\sim\text{Gauss}\left(\mathbf{0},\mathbf{L}_t^{\dagger}+\sigma_w^2\mathbf{I}\right), \tag{38}$$

where σ_w^2 is the noise level. We set $\sigma_w=0.1$ except for the experiment of convergence (see below). As demonstrated in [11], graph signals generated in this way are smooth over the corresponding graph. In our setup, we simply set $N_t=N$, i.e., the data sizes of all graphs are assumed to be the same.

3) Evaluation metric: In topology inference, determining whether two vertices are connected can be regarded as a binary classification problem. As a result, we employ the following three metrics to evaluate the classification results,

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}},\quad \text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}},$$
$$\text{MCC}=\frac{\text{TP}\cdot\text{TN}-\text{FP}\cdot\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}, \tag{39}$$

where TP is true positive rate, TN is true negative rate, FP is the false positive rate and FN denotes false negative rate. Matthews correlation coefficient (MCC) [53] is a comprehensive evaluation metric since it summarizes all information from the TP, TN, FP, andFN quantities. The value range of MCC is $[-1,+1]$. The value $+1$ means that we exactly learn the existence of all edges in the ground-truth, while -1 is interpreted as a total misidentification. The last metric is the relative error (RE) for evaluating edge weights, i.e.,

$$\text{RE}=\frac{\|\widehat{\mathbf{A}}-\mathbf{A}_{\text{gt}}\|_{\text{F}}}{\|\mathbf{A}_{\text{gt}}\|_{\text{F}}}, \tag{40}$$

where $\widehat{\mathbf{A}}$ is the learned adjacency matrix and $\mathbf{A}_{\text{gt}}$ is the ground-truth. It is worth noting that all the values of the above metrics in ensuing experiments are the average of all T graphs.

4) Baselines: Six baselines are employed and listed in Table I, where IGL means all T graphs are learned independently based on classic smoothness based model (7). L-Precision is a statistical model, where multiple Laplacian constrained precision matrices are independently estimated. Methods 3-5 are time-varying models based on the smoothness assumption, whose $\mathcal{G}_N$ are chain structures. We also implement methods 3-5 using the decentralized algorithm since they are special cases of our framework. JEMGL is a recently proposed model that leverages Gram matrices to describe topological patterns [17]. However, we should mention that the design of the Gram matrix is determined by prior knowledge [17]. In our experiment, we build the Gram matrix according to the handcrafted $\mathcal{G}_N$. The algorithm for JEMGL in [17] does not consider data privacy, but we still take it as a baseline to compare the effectiveness of different regularizers. Ours-1 and Ours-2 are our proposed models (9) with the prior $\mathcal{G}_N$. Ours-J-1 and Ours-J-2 are the proposed models that jointly learn graphs and topological patterns.

5) Determination of parameters: As stated in Remark 1, we fix α and α_γ as 2 and grid search for the best β and β_γ as in [11]. The parameters η are selected as those that reach the maximum

TABLE I
Baselines and the Proposed Method

Index	Method	Paper	$\psi(\cdot)$
1	IGL	[11]	–
2	L-Precision	[9]	–
3	TVGL-Tikhonov	[18]	squared ℓ_2 norm
4	TVGL-1	[43]	ℓ_1 norm
5	TVGL-2	[19]	ℓ_2 norm
6	JEMGL	[17]	–
7	Ours-1	This paper (9)	ℓ_1 norm
8	Ours-2	This paper (9)	ℓ_2 norm
9	Ours-J-1	This paper (15)	ℓ_1 norm
10	Ours-J-2	This paper (15)	ℓ_2 norm

TABLE II
PERFORMANCE OF THE LEARNED GRAPHS IN PATTERN 1

		$N = 10$				$N = 40$				$N = 80$			
		Precision	Recall	MCC	RE	Precision	Recall	MCC	RE	Precision	Recall	MCC	RE
RBF	IGL	0.714	0.591	0.534	1.248	0.758	0.776	0.660	0.838	0.767	0.867	0.726	0.709
	L-Precision	0.620	0.593	0.348	1.223	0.661	0.744	0.477	0.810	0.685	0.855	0.577	0.658
	TVGL-Tikhonov	0.763	0.725	0.630	0.965	0.841	0.811	0.748	0.663	0.865	0.845	0.791	0.583
	TVGL-1	0.737	0.764	0.639	0.873	0.840	0.849	0.772	0.610	0.862	0.862	0.798	0.565
	TVGL-2	0.766	0.658	0.592	1.133	0.852	0.788	0.741	0.722	0.873	0.825	0.784	0.602
	JEMGL	**0.780**	0.620	0.579	1.213	**0.858**	0.737	0.712	0.754	0.881	0.806	0.777	0.610
	Ours-1	0.732	**0.826**	**0.666**	**0.865**	0.815	**0.922**	**0.801**	**0.541**	0.839	0.930	0.827	0.519
	Ours-2	0.749	0.715	0.613	1.026	0.824	0.847	0.760	0.658	0.865	0.896	0.825	0.578
	Ours-J-1	0.756	0.694	0.605	1.196	0.832	0.898	0.800	0.586	0.846	**0.933**	**0.835**	**0.520**
	Ours-J-2	0.757	0.659	0.585	1.211	0.835	0.849	0.769	0.689	**0.874**	0.888	0.827	0.586
ER	IGL	0.487	0.448	0.262	1.199	0.601	0.639	0.452	0.738	0.693	0.736	0.590	0.648
	L-Precision	0.543	0.570	0.321	1.385	0.616	0.777	0.513	0.804	0.659	0.888	0.618	0.657
	TVGL-Tikhonov	0.466	0.552	0.279	0.975	0.582	0.772	0.506	0.617	0.650	0.829	0.603	0.525
	TVGL-1	0.459	0.643	0.308	0.887	0.569	0.834	0.525	0.544	0.660	0.859	0.631	0.480
	TVGL-2	0.485	0.482	0.274	1.114	0.593	0.712	0.484	0.661	0.679	0.812	0.623	0.548
	JEMGL	0.462	**0.721**	0.345	1.353	0.558	**0.892**	0.540	0.621	0.644	**0.919**	0.647	0.464
	Ours-1	**0.557**	0.544	**0.370**	**0.867**	**0.625**	0.781	0.557	**0.492**	0.701	0.898	0.661	**0.421**
	Ours-2	0.476	0.549	0.289	1.010	0.590	0.763	0.505	0.602	0.722	0.819	0.639	0.503
	Ours-J-1	0.494	0.512	0.295	1.138	0.613	0.812	**0.560**	0.582	**0.774**	0.876	**0.664**	0.434
	Ours-J-2	0.486	0.479	0.274	1.193	0.609	0.701	0.492	0.652	0.707	0.783	0.633	0.509

MCC metric. Some other parameters ρ, ϵ_w, ϵ_W and $k_{\max}$ are set to 0.5, 10^{-4}, 10^{-3} and 100, respectively. The parameters of all baselines are also selected as those that reach the best MCC performance.

B. Synthetic Data

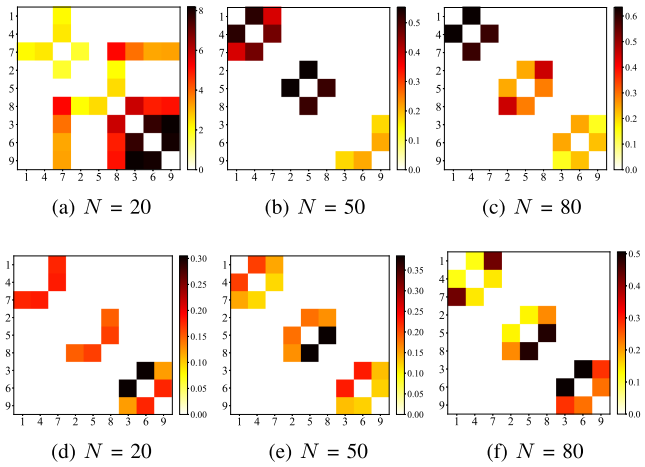
1) Pattern 1: We first examine the impact of data sizes on the estimation performance. Table II displays the results with different data sizes for both RBF and ER graphs. We can find that IGL and L-Precision have inferior performance because they learn all graphs independently and ignores the topological relationships among them. As such, they only make use of the information from the data. When N is small, the available information extracted from data is too scarce to learn a reliable graph, resulting in poor performance. Note that the performance of models with ℓ_1 norm in penalty term outperforms that of ℓ_2 norm overall. This trend can be explained by that the differences between graphs in Pattern 1 are actually small, which is roughly consistent with the constraints of ℓ_1 norm. Therefore, ℓ_1 norm is a more suitable penalty function under this circumstance. Intriguingly, the performance of TVGL-1 is inferior to that of Ours-J-1. The reason for this trend is that TVGL-1 is based on a chain structure that fails to capture the real topological relationships in Pattern 1. On the contrary, the pattern graph provides a flexible tool to characterize the weighted non-chain relationships in Fig. 2. From the above observations, we conclude that the extra structured priors do improve learning performance. Another interesting point is that our methods without prior $\mathcal{G}_N$ perform poorly when the data size is small since we need to learn $\mathcal{G}_N$ from data directly. As N increases, the performance of Ours-J-1 and Ours-J-2 starts to approach that of models with prior $\mathcal{G}_N$ because more reliable $\mathcal{G}_N$ can be obtained from data.

2) Pattern 2: We next provide the results of Pattern 2. As shown in Table III, IGL and L-Precision still perform poorly for the same reason mentioned before. On the contrary, our models outperform the other baselines for both MCC and RE. In Pattern 2, our models with ℓ_2 norm reach the best performance when N is large. This may be caused by that the ℓ_2 norm is more suitable for this pattern. Another point worth noting is that the models with chain-structured priors do not perform as well as IGL. This can be explained by that the real topological pattern of Pattern 2 is far away from the assumed $\mathcal{G}_N$, e.g., chain structure. Thus, some misleading information is brought into the learning process, leading to poor performance. Our models, however, can flexibly describe the complex topological relationships thanks to pattern graphs. In Fig. 3, we further provide the $\mathcal{G}_N$ learned by our models without prior topological patterns. We select the experiments of ER graph in Pattern 2. We can observe from Fig. 3 that our method can effectively learn the cluster structures among these graphs, especially with large data sizes. Thus, our proposed model still works for the cases where no priors are available.

3) The impact of η: As alluded to earlier, η is a global parameter controlling the scale of edge weights of $\mathcal{G}_N$. We can regard it as a trade-off between using information from data and topological relationships. In this experiment, we vary η from 10^{-3} to 10^2 to learn the impact of η. Specifically, we fix $N = 100$ and select Pattern 2 as an example to show the impact of η. In Fig. 4, the performance, both MCC and RE, first increases as η grows and then reaches its peak at a certain η^*. When η continues to increase, the performance starts to decrease until it becomes stable and hardly changes with η anymore. This can be explained by that, as η increases, information on topological relationships is gradually taken into consideration, which boosts the estimation performance. There exists an η^* that balances

TABLE III
PERFORMANCE OF THE LEARNED GRAPHS IN PATTERN 2

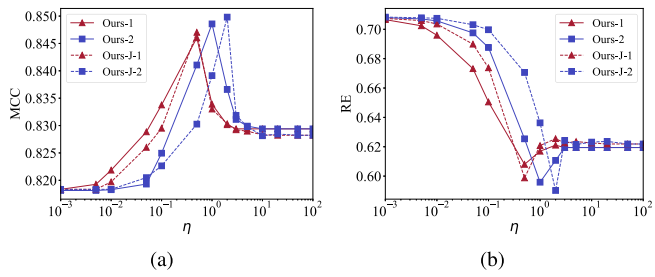
		$N = 10$				$N = 40$				$N = 80$			
		Precision	Recall	MCC	RE	Precision	Recall	MCC	RE	Precision	Recall	MCC	RE
RBF	IGL	0.685	0.594	0.502	1.141	0.732	**0.926**	0.724	0.761	0.873	0.828	0.789	0.650
	L-Precision	0.427	0.624	0.353	1.081	0.513	0.847	0.541	0.618	0.531	0.896	0.588	0.512
	TVGL-Tikhonov	0.669	0.701	0.544	0.901	0.777	0.878	0.743	0.619	0.785	0.909	0.770	0.570
	TVGL-1	0.639	0.754	0.547	0.865	0.755	0.847	0.706	0.685	0.738	**0.941**	0.749	0.639
	TVGL-2	0.697	0.632	0.531	1.051	0.832	0.822	0.754	0.690	0.826	0.869	0.778	0.598
	JEMGL	0.722	0.612	0.538	1.462	0.844	0.805	0.752	0.872	0.778	0.898	0.772	0.698
	Ours-1	0.712	**0.720**	**0.593**	**0.834**	0.847	0.861	0.788	0.572	0.839	0.890	0.804	0.500
	Ours-2	0.730	0.665	0.576	0.897	0.862	0.867	0.805	**0.567**	0.848	0.890	0.811	0.501
	Ours-J-1	**0.750**	0.561	0.529	1.037	0.894	0.765	0.762	0.568	0.842	0.888	0.806	**0.493**
	Ours-J-2	0.739	0.564	0.522	1.032	**0.900**	0.849	**0.821**	0.585	**0.890**	0.870	**0.830**	0.498
ER	IGL	0.445	0.550	0.272	1.107	0.555	0.768	0.492	0.752	0.587	0.880	0.584	0.631
	L-Precision	0.417	0.640	0.310	1.076	0.480	0.885	0.495	0.701	0.488	0.949	0.538	0.609
	TVGL-Tikhonov	0.414	0.697	0.287	0.939	0.455	0.889	0.437	0.700	0.449	0.956	0.462	0.634
	TVGL-1	0.386	0.728	0.256	0.919	0.426	0.928	0.415	0.765	0.422	0.973	0.434	0.725
	TVGL-2	0.436	0.615	0.286	1.044	0.496	0.843	0.464	0.719	0.515	0.943	0.541	0.621
	JEMGL	**0.510**	0.541	0.339	1.415	0.550	0.826	0.521	0.770	0.551	0.962	0.588	0.560
	Ours-1	0.456	0.715	**0.350**	**0.865**	0.532	0.894	0.533	**0.581**	0.614	**0.963**	0.658	0.508
	Ours-2	0.427	**0.759**	0.325	0.902	**0.582**	0.828	0.552	0.586	**0.623**	0.959	**0.668**	**0.506**
	Ours-J-1	0.468	0.562	0.308	1.177	0.565	0.822	0.531	0.629	0.619	0.928	0.646	0.513
	Ours-J-2	0.449	0.546	0.276	1.121	0.538	**0.939**	**0.565**	0.659	0.593	0.956	0.652	0.507



(a) $N = 20$ (b) $N = 50$ (c) $N = 80$

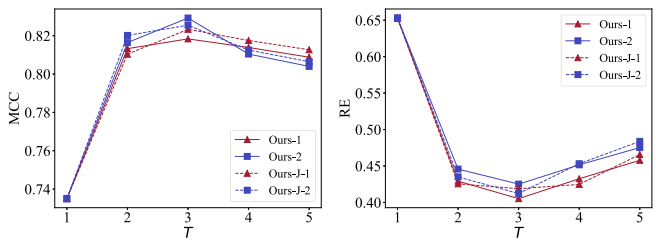
(d) $N = 20$ (e) $N = 50$ (f) $N = 80$

Fig. 3. $\mathcal{G}_N$ learned by our models: the top row shows the $\mathcal{G}_N$ learned by Ours-J-1, while the bottom row shows the $\mathcal{G}_N$ learned by Ours-J-2. The numbers on the axes represent graph numbers in Pattern 2.



(a) (b)

Fig. 4. Impact of η on the performance of our methods.



(a) (b)

Fig. 5. Impact of T on the performance of our methods.

information from data and topological priors the most properly. When η is too large, some consensus graphs are obtained and will not change anymore as discussed in Section IV.

4) The impact of T: We then check the impact of the number of graphs. We fix $N = 100$ and vary T from 1 to 5. The graphs are generated in the same way as the first five graphs in Pattern 1. The results are displayed in Fig. 5. When $T = 1$, all data are assumed to come from the same distribution, and our model will reduce to IGL learning a single graph. We can observe from Fig. 5 that the performance of $T > 1$ is significantly better than that of $T = 1$ since MGL uses information from T data clusters to jointly learn graphs, while IGL only uses data from one cluster. But with the increase of T, the performance is slightly improved or even decreased. This may be caused by that the gain of using all data may have a marginal effect, but increasing T makes it difficult to describe topological relationships, which may bring additional estimation errors.

5) Incremental graph learning: We also test the incremental graph learning formulation in (16) based on the already learned graphs. We first learn the graphs in Pattern 2 using Ours-J-1. We then generate a new graph that lies in the first cluster of Pattern 2. The new graph is learned by our proposed approach in (16) based on the previously learned graphs by Ours-J-1. The IGL

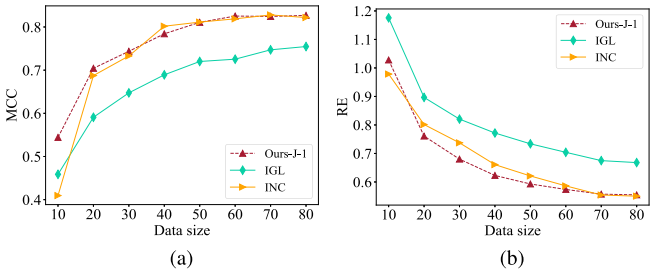


Fig. 6. Results of incremental graph learning.

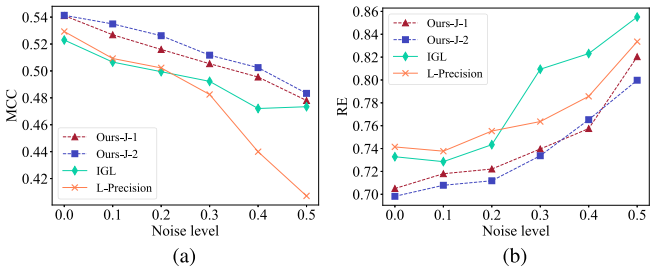


Fig. 7. Results of unlabeled signals.

and Our-J-1 with all data are taken as baselines. As displayed in Fig. 6, the incremental learning, denoted as INC, achieves comparable performance to Ours-J-1, but with a much smaller problem scale. Moreover, it outperforms IGL, meaning that it benefits from the learned topological relationships.

6) Unlabelled signals: Furthermore, we consider the case of unlabeled signals. Specifically, five graphs with chain structures are built, after which we generate 200 signals for each graph using (38). However, the labels of the signals, i.e. which Gaussian distribution they come from, are assumed to be unknown. We hope to decouple the signals into multiple groups and learn a graph for each group. There have been some excellent works on this problem [30], [31], [32]. The common idea of these models is to alternately label the signal and use the labeled signal to learn multiple graphs. Given labeled signals, existing models usually learn the graphs independently [31], [32] or with some topological priors [30]. We aim to test whether our model—which can jointly learn multiple graphs without topological priors—can improve the performance of MGL for unlabeled signals. In this experiment, we employ the framework proposed in [32] to alternately label signals and learn multiple graphs. However, we replace the step of learning graphs given labeled signals with IGL, L-Precision, Ours-J-1, and Ours-J-2. In this way, we can remove the effect of labeling signals, which is not the focus of this study. Fig. 7 displays the results of different noise levels. We can observe that our models (Ours-J-1 and Ours-J-2) are superior to IGL and L-Precision since our model can utilize the topological relationships to jointly learn all graphs using the labeled signals. Moreover, accurately learned graphs can in turn improve labeling accuracy [30]. The experimental results demonstrate that our model can be flexibly integrated into existing MGL frameworks for unlabeled signals and achieve competitive performance, as it can jointly learn all graphs without any priors.

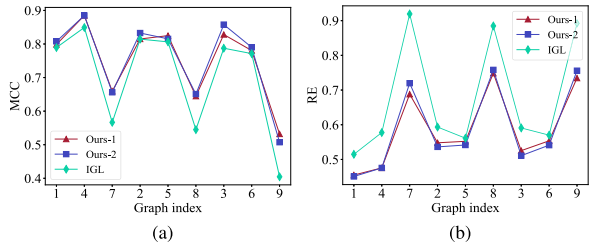


Fig. 8. Generalization performance of the learned pattern graph.

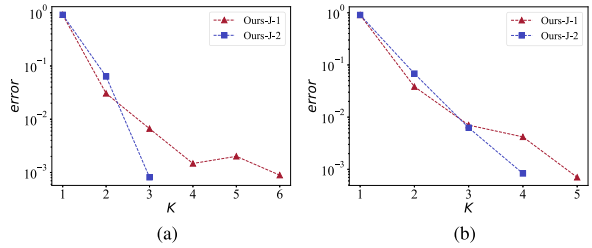


Fig. 9. Results of convergence.

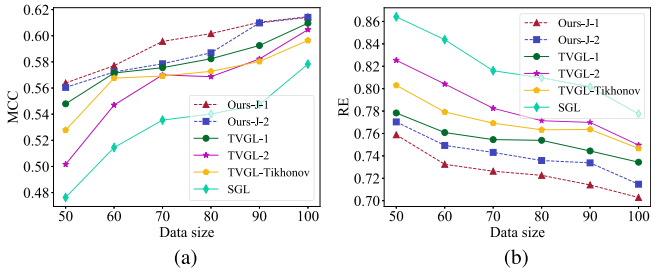
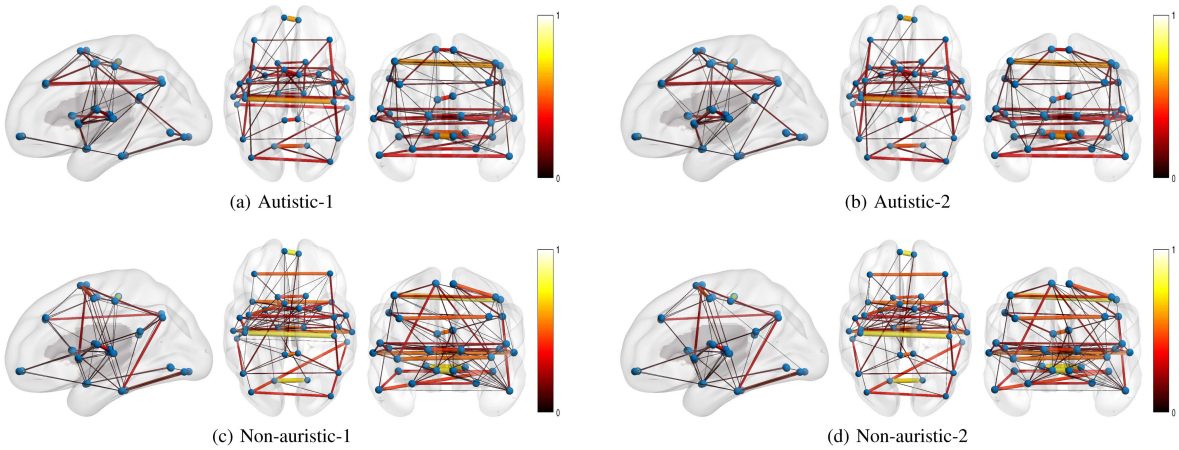


Fig. 10. Results of experiments on social network data.

7) Generalization performance: In this experiment, we consider the generalization performance of the learned pattern graphs. Specifically, we first jointly learn multiple graphs and a pattern graph of Pattern 2. Then, we change five fixed edges for each graph. For graphs in each cluster of Pattern 2, we generate 100 signals for the first two of them and 20 signals for the remaining one. We use the learned pattern graph to jointly learn new graphs from the new signals. Fig. 8 depicts the performance of each graph instead of their average. It is observed that even if the underlying graph topology changes, the learned pattern graph still helps to improve the learning performance. The reason is that the graphs change in a consistent manner, and the changed graphs still have similar topological relationships as before, which can be described by the pattern graph. The results prove the generalization performance of the learned pattern graph. The figure also reveals another interesting point. For the graphs corresponding to small data sizes ($\mathcal{G}_7$, $\mathcal{G}_8$, $\mathcal{G}_9$), the performance gap between MGL and IGL is larger than that of graphs corresponding to large data size. This can be interpreted as that under the MGL framework, graphs can benefit from "borrowing" information from other datasets, especially for graphs with small data sizes.

8) Convergence: Finally, we test the convergence of our proposed algorithm. We learn graphs in Pattern 1 and Pattern 2 with $N = 100$, respectively. As depicted in Fig. 9, as the number



(a) Autistic-1 (b) Autistic-2

(c) Non-auristic-1 (d) Non-auristic-2

Fig. 11. Graphs of brain functional connectivity learned by Ours-J-1. The graph of each individual is displayed in three views, i.e., Sagittal, Axial, and Coronal, respectively.

of iterations K increases, the relative error between two consecutive iterations, i.e., $error = \|\mathbf{W}^{K+1} - \mathbf{W}^{K}\|_{\mathrm{F}}/\|\mathbf{W}^{K}\|_{\mathrm{F}}$, decreases to less than the threshold ϵ_W quickly. Therefore, our algorithm can converge in only a few iterations.

C. Real-World Data

1) Social network: We first consider social networks from the Ljubljana student network dataset.[1] The dataset contains 12 networks whose nodes represent the same 32 students. The edges capture interactions among these students. The interactions are built based on the students' answers to different questions that might be related to privacy. A total of 12 questions are asked, corresponding to 12 graphs. We select 5 networks, and these networks should be related since all answers come from the same students. We suppose the questions are asked by different organizations that are reluctant to share their data. Therefore, we learn these graphs in a decentralized way without delivering all data to a central server. Note that the dataset contains no graph signals, and we generate signals for each network using (38). The data sizes of each network are the same and are ranging from 50 to 100. We evaluate the results by MCC and RE because we have ground-truth graphs. Due to that we have no knowledge about the underlying relationships between these graphs, we give up using the baselines that require a pre-designed $\mathcal{G}_N$. As displayed in Fig. 10, Ours-J-1 reaches the best performance, followed by Our-J-2. The baselines with $\mathcal{G}_N$ of chain structure still outperform IGL, implying that these models benefit from learning all graphs jointly. However, the performance of the chain-structured models is inferior to ours, since our model can learn a $\mathcal{G}_N$ that better fits the real topological patterns.

2) COIL-20 data: We next employ the COIL-20 dataset,[2] which is a collection of gray-scale images including 20 objects taken from 360 degrees. The size of each image is 32×32, and each object has 72 images (5 degrees an image). We randomly select 4 objects and divide the images of each object into 6 views. Each view contains images taken in consecutive 60 degrees, e.g., $[0°, 55°]$. Therefore, each view contains 48 images taken from 4 objects, i.e., 12 images per object. We aim to learn the graphs representing the relationships among these 48 images. One basic assumption is that the 48 images of different views are defined on the same node set. We further assume that the images of each view are taken by different photographers, and they are reluctant to share their photos. Thus, we cannot process all the images on a central server. The image itself is taken as graph signals, i.e., $\mathbf{X}_i \in \mathbb{R}^{48\times 1024}$. It is reasonable to assume that the graphs of different views are related owing to that they come from the same objects. In addition, the learned graphs should have four clusters since all images belong to four objects. To evaluate the learned graphs, we adopt the classic Louvain method [54] to detect the communities in the learned graphs. Three commonly used metrics, i.e., normalized mutual information (NMI), Fowlkes and Mallows index (FMI) [55], and Rand Index (RI) are adopted to evaluate the detection results. The labels of the images are taken as the ground-truth. The final results are the average of all views. Similarly, we do not consider the baselines that need pre-designed $\mathcal{G}_N$. The detection results are listed in Table IV. Our models, both Ours-J-1 and Ours-J-2, succeed to reach the best performance, meaning that the graphs learned by our models can better restore the clusters. The underlying topological relationships between different views provide useful information for us to learn all graphs jointly.

3) The fMRI data: Finally, we employ blood-oxygenation-level-dependent (BOLD) time series extracted from fMRI data

TABLE IV
COMMUNITY DETECTION RESULTS

	Ours-J-1	Ours-J-2	TVGL-1	TVGL-2	TVGL-Tikhonov	IGL
NMI	**0.790**	**0.790**	0.727	0.753	0.750	0.703
RI	**0.846**	**0.846**	0.830	0.845	0.835	0.817
FMI	**0.753**	**0.753**	0.711	0.736	0.728	0.693

[1]The data is available at http://vladowiki.fmf.uni-lj.si/doku.php?id= pajek:data:pajek:students

[2]The data are available at https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

to learn the brain functional connectivity of autism. The dataset[3] contains 539 autistic subjects and 573 typical controls. We select 3 autistic subjects and 3 controls. For each individual, we select 34 functional regions of interest (ROI) from 90 standard regions of the Anatomical Automatic Labeling (AAL) template. The length of the time series of each ROI is 196, i.e., $\mathbf{X}_i \in \mathbb{R}^{34\times 196}$. We aim to learn the connectivity graphs of these 34 functional regions. The basic assumption is that autism may change the functional region connectivity, and if we learn about the differences in brain functional connectivity graphs between autistic and non–autistic subjects, it may help us better understand the mechanism of autism. Thus, we use Ours-J-2 to learn the brain functional connectivity graphs of the selected subjects jointly. We suppose the data are collected by different medical institutions and are forbidden to leave their local clients due to privacy concerns. Therefore, we execute our algorithm in a decentralized manner. As depicted in Fig. 11, graphs in the autistic group are similar to each other, and the same for the control group. Specifically, graphs in the Autism class show fewer edges and smaller edge weights, implying weakened connectivity between functional regions due to the effects of Autism. The observation is consistent with the current study [56]. However, our knowledge does not allow us to conduct a deeper analysis from a medical perspective. In the future, a domain expert may be helpful.

VII. Conclusion

In this paper, we have presented a framework for learning multiple distinct but related graphs jointly. For this problem, the underlying topological relationships between the graphs to be learned may help improve learning performance. Therefore, we proposed a novel regularizer termed pattern graph to flexibly encode prior topological relationships into learning process. Furthermore, we put forth a data-driven approach to automatically learn pattern graphs to handle the situation when no priors are available. Algorithmically, we developed a decentralized algorithm to learn multiple graphs without sending private data to a central server. Extensive experiments were carried out and showed the superiority of our proposed framework. Future research directions may include generalizing our framework to directed pattern graphs as well as other graph learning methods except for smoothness assumption.

Appendix A
Proof of Theorem 1

To prove Theorem 1, we first provide the following lemmas.

Lemma 1: The $g(\mathbf{w})$ is a $2\beta-$strongly convex function.

Proof: For any $\mathbf{w} \geq 0$, we can calculate the Hessian matrix of $g(\mathbf{w})$ as $\nabla_{\mathbf{ww}} g(\mathbf{w}) = 2\beta\mathbf{I} + \alpha\mathbf{S}^\top \text{diag}((\mathbf{Sw})^{\cdot(-2)})\mathbf{S}$. It is easy to check that $\nabla_{\mathbf{ww}} g(\mathbf{w}) \succ 2\beta\mathbf{I}$. Hence, $g(\mathbf{w})$ is $2\beta-$strongly convex. □

Lemma 2: For any $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$, we have that $\Psi(\mathbf{Y}) \leq \eta\sqrt{S\|\mathbf{L}_\gamma\|_2}\|\mathbf{Y}\|_{\text{F}}$, where $\Psi(\mathbf{Y}) = \eta\sum_{(i,j)\in\mathcal{E}_N}\gamma_{ij}\psi(\mathbf{y}_i - \mathbf{y}_j)$, and $\psi = \ell_1$ or ℓ_2 norm.

[3]http://preprocessed-connectomes-project.org/abide/

Proof: Firstly, we define that $\widetilde{\mathbf{y}}_i \in \mathbb{R}^T$ is the i-th row vector of $\mathbf{Y}$. When $\psi = \ell_1$ norm, we obtain that

$$\Psi(\mathbf{Y}) \leq \eta\sqrt{S}\sqrt{\sum_{(i,j)\in\mathcal{E}_N}\gamma_{ij}^2\left(\mathbf{y}_i - \mathbf{y}_j\right)^2} = \eta\sqrt{S}\sqrt{\sum_{i=1}^{p}\widetilde{\mathbf{y}}_i^\top\mathbf{L}_\gamma\widetilde{\mathbf{y}}_i}$$
$$\leq \eta\sqrt{S}\sqrt{\sum_{i=1}^{p}\|\mathbf{L}_\gamma\|_2\|\widetilde{\mathbf{y}}_i\|_2^2} = \eta\sqrt{S\|\mathbf{L}_\gamma\|_2}\|\mathbf{Y}\|_{\text{F}}. \quad (41)$$

When $\psi = \ell_2$, the proof is the same as ℓ_1 norm. □

The following proof is suitable for both $\psi = \ell_1$ and ℓ_2 norm. We first rewrite the objective function in a matrix form. Define $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_T] \in \mathbb{R}^{p\times T}$, and we rewrite (9) as

$$\min_{\mathbf{W}\geq 0}\ (1/N)\left\langle\mathbf{R}, \mathbf{W}\right\rangle + G(\mathbf{W}) + \Psi(\mathbf{W}), \quad (42)$$

where $G(\mathbf{W}) = \sum_{t=1}^{T} g(\mathbf{w}_t)$. It is not difficult to show that

$$(1/N)\left\langle\mathbf{R}, \widehat{\mathbf{W}}\right\rangle + G(\widehat{\mathbf{W}}) + \Psi(\widehat{\mathbf{W}})$$
$$\leq (1/N)\left\langle\mathbf{R}, \mathbf{W}\right\rangle + G(\mathbf{W}) + \Psi(\mathbf{W}), \quad (43)$$

which yields that

$$(1/N)\left\langle\mathbf{R}^*, \widehat{\mathbf{W}} - \mathbf{W}\right\rangle + G(\widehat{\mathbf{W}}) - G(\mathbf{W})$$
$$\leq \Psi(\mathbf{W}) - \Psi(\widehat{\mathbf{W}}) + (1/N)\left\langle\mathbf{E}, \mathbf{W} - \widehat{\mathbf{W}}\right\rangle, \quad (44)$$

where $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_T]$ is the error matrix. We then define a variable $v = \sum_{t=1}^{T}\sum_{i=1}^{p}\frac{1}{\sigma_e^2}(\mathbf{e}_t)_{[i]}^2 = \frac{1}{\sigma_e^2}\|\mathbf{E}\|_{\text{F}}^2$ which follows a chi-squared distribution with the degree of freedom as pT. Using the Wallace inequality [57], for $\delta > 0$, we have

$$\Pr(v \geq pT + \delta) \leq \exp\left(-\frac{1}{2}\left(\delta - pT\log\left(1 + \frac{\delta}{pT}\right)\right)\right). \quad (45)$$

Therefore, we conclude that

$$\Pr\left((1/N)\|\mathbf{E}\|_{\text{F}} \leq (\sigma_e/N)\sqrt{pT+\delta}\right)$$
$$\geq 1 - \exp\left(-\frac{1}{2}\left(\delta - pT\log\left(1 + \frac{\delta}{pT}\right)\right)\right). \quad (46)$$

With probability $1 - \exp\left(-\frac{1}{2}\left(\delta - pT\log\left(1+\frac{\delta}{pT}\right)\right)\right)$,

$$(1/N)\left\langle\mathbf{E}, \mathbf{W} - \widehat{\mathbf{W}}\right\rangle \leq (1/N)\|\mathbf{E}\|_{\text{F}}\|\mathbf{W} - \widehat{\mathbf{W}}\|_{\text{F}}$$
$$\leq (\sigma_e/N)\sqrt{pT+\delta}\|\mathbf{W} - \widehat{\mathbf{W}}\|_{\text{F}}. \quad (47)$$

We next focus on the terms related to G in (44) and obtain

$$G(\widehat{\mathbf{W}}) - G(\mathbf{W}) = \sum_{t=1}^{T} g(\widehat{\mathbf{w}}_t) - g(\mathbf{w}_t)$$
$$\geq \sum_{t=1}^{T}\left\langle\nabla g(\mathbf{w}_t), \widehat{\mathbf{w}}_t - \mathbf{w}_t\right\rangle + \beta\left\|\widehat{\mathbf{w}}_t - \mathbf{w}_t\right\|_2^2$$
$$\geq \sum_{t=1}^{T} -\left\|\nabla g(\mathbf{w}_t)\right\|_2\left\|\widehat{\mathbf{w}}_t - \mathbf{w}_t\right\|_2 + \beta\left\|\widehat{\mathbf{w}}_t - \mathbf{w}_t\right\|_2^2$$

$$\geq -C_g\sqrt{T}\left\|\widehat{\mathbf{W}}-\mathbf{W}\right\|_{\mathrm{F}}+\beta\left\|\widehat{\mathbf{W}}-\mathbf{W}\right\|_{\mathrm{F}}^2. \tag{48}$$

The first inequality holds due to that g is a strongly convex function as stated in Lemma 1. The last inequality holds due to Assumption 2 and the fact that $\sum_{t=1}^{T}\|\widehat{\mathbf{w}}_t-\mathbf{w}_t\|_2\leq\sqrt{T}\|\widehat{\mathbf{W}}-\mathbf{W}\|_{\mathrm{F}}$. For $\mathbf{R}$ related terms, we obtain that

$$\begin{aligned}(1/N)\left\langle\mathbf{R}^*,\widehat{\mathbf{W}}-\mathbf{W}\right\rangle&\geq-(1/N)\left\|\mathbf{R}^*\right\|_{\mathrm{F}}\left\|\widehat{\mathbf{W}}-\mathbf{W}\right\|_{\mathrm{F}}\\&\geq-\left(\sqrt{T}C_r/N\right)\left\|\widehat{\mathbf{W}}-\mathbf{W}\right\|_{\mathrm{F}}.\end{aligned} \tag{49}$$

The second inequality holds due to Assumption 1. Finally, we focus on the Ψ related terms of (44) and obtain that

$$\begin{aligned}&\Psi(\mathbf{W})-\Psi(\widehat{\mathbf{W}}) \qquad (50)\\&=\eta\sum_{(i,j)\in\mathcal{E}_N}\gamma_{ij}\left(\psi(\mathbf{w}_i-\mathbf{w}_j)-\psi(\widehat{\mathbf{w}}_i-\widehat{\mathbf{w}}_j)\right)\\&\leq\eta\sum_{(i,j)\in\mathcal{E}_N}\gamma_{ij}\psi\left((\mathbf{w}_i-\widehat{\mathbf{w}}_i)-(\mathbf{w}_j-\widehat{\mathbf{w}}_j)\right)\\&=\Psi(\mathbf{W}-\widehat{\mathbf{W}})\leq\eta\sqrt{S\|\mathbf{L}_\gamma\|_2}\|\mathbf{W}-\widehat{\mathbf{W}}\|_{\mathrm{F}}, \qquad (51)\end{aligned}$$

The last inequality holds due to Lemma 2. Plugging (47), (51), (48) and (49) into (44), and let $\mathbf{W}=\mathbf{W}^*$, we have

$$\begin{aligned}\beta\left\|\widehat{\mathbf{W}}-\mathbf{W}^*\right\|_{\mathrm{F}}^2&\leq C_g\sqrt{T}\left\|\widehat{\mathbf{W}}-\mathbf{W}^*\right\|_{\mathrm{F}}+\frac{\sqrt{T}C_r}{N}\left\|\widehat{\mathbf{W}}-\mathbf{W}^*\right\|_{\mathrm{F}}\\&+\frac{\sigma_e}{N}\sqrt{pT+\delta}\|\widehat{\mathbf{W}}-\mathbf{W}^*\|_{\mathrm{F}}+\eta\sqrt{S\|\mathbf{L}_\gamma\|_2}\|\mathbf{W}^*-\widehat{\mathbf{W}}\|_{\mathrm{F}}.\end{aligned} \tag{52}$$

From (52), we can reach the conclusion of Theorem 1.

Appendix B
Proof of Proposition 1

The following lemma is proposed to facilitate the proof.

Lemma 3: The function $h(\boldsymbol{\gamma})$ is $2\eta\beta_\gamma$ -strongly convex, and for any selected block $\mathcal{T}$, the corresponding gradient $\nabla h(\boldsymbol{\gamma}_{[\mathcal{T}]})$ is L_h- Lipschitz continuous, where $L_h=\eta(\alpha_\gamma\sqrt{(M+1)(T+M-2)}/deg_{\min}^2+2\beta_\gamma)$.

Proof: The proof of strong convexity is the same as that of Lemma 1. We next calculate the Lipschitz constant of $\nabla h(\boldsymbol{\gamma}_{[\mathcal{T}]})$. For arbitrary two vectors $\mathbf{a},\mathbf{b}\geq 0$, we have that

$$\begin{aligned}&\|\nabla h(\boldsymbol{a}_{\mathcal{T}})-\nabla h(\boldsymbol{b}_{\mathcal{T}})\|_2\\&=\eta\left\|\alpha_\gamma\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}^\top\left(\frac{1}{\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{b}}-\frac{1}{\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{a}}\right)+2\beta_\gamma\left(\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\right)\right\|_2\\&\leq 2\eta\beta_\gamma\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\|_2+\eta\alpha_\gamma\left\|\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}^\top\right\|_2\left\|\frac{1}{\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{b}}-\frac{1}{\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{a}}\right\|_2\\&\leq 2\eta\beta_\gamma\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\|_2+\frac{\eta\alpha_\gamma\left\|\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}^\top\right\|_2}{deg_{\min}^2}\left\|\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{a}-\widetilde{\mathbf{S}}_{\mathcal{M}}\mathbf{b}\right\|_2\\&\leq 2\eta\beta_\gamma\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\|_2+\frac{\eta\alpha_\gamma\left\|\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}^\top\right\|_2}{deg_{\min}^2}\left\|\widetilde{\mathbf{S}}_{\mathcal{M}}\right\|_2\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\|_2\\&=\eta\left(2\beta_\gamma+\alpha_\gamma\sqrt{(M+1)(T+M-2)}/deg_{\min}^2\right)\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\|_2\\&=L_h\left\|\mathbf{a}_{\mathcal{T}}-\mathbf{b}_{\mathcal{T}}\right\|_2,\end{aligned} \tag{53}$$

where the second equality holds since $\|\widetilde{\mathbf{S}}_{\mathcal{M},\mathcal{T}}^\top\|_2=\sqrt{M+1}$ and $\|\widetilde{\mathbf{S}}_{\mathcal{M}}\|_2=\sqrt{T+M-2}$. The calculations of spectral norm of the two matrices are similar with Lemma 1 of [58]. □

With Lemma 3, we next prove Theorem 1 by referring to Theorem 2 of [42]. Plugging the Lipschitz constant L_h and the constant of strong convexity into Theorem 2 of [42], we can reach our conclusion in Theorem 1.

Appendix C
Algorithm for Incremental Graph Learning

For simplicity, some notations are slightly abused here. Some notations are shared with those of Algorithm 1. Similar with the proposed algorithm for (15), the algorithm for (16) is based on BCD framework, i.e., we update $\mathbf{w}_{T+1}$ and $\gamma_{T+1,t},t=1,\ldots,T+1$ iteratively.

1) Fix $\gamma_{T+1,t}$ and Update $\mathbf{w}_{T+1}$

We also develop the algorithm under ADMM framework. We first write the following equivalent form of the original problem:

$$\begin{aligned}&\min_{\mathbf{w}_{T+1}\geq 0}\ f_{T+1}(\mathbf{w}_{T+1})+\eta\sum_{t=1}^{T}\gamma_{T+1,t}\psi(\mathbf{z}_t-\mathbf{w}_t)\\&\text{s.t. }\mathbf{z}_t=\mathbf{w}_{T+1},\ \text{for } t=1,\ldots,T\end{aligned} \tag{54}$$

where $\mathbf{z}_t$ is the consensus variables. We define $\mathbf{Z}=[\mathbf{z}_1,\ldots,\mathbf{z}_T]\in\mathbb{R}^{p\times T}$, and $\mathbf{U}=[\mathbf{u}_1,\ldots,\mathbf{u}_T]\in\mathbb{R}^{p\times T}$, where $\mathbf{u}_t,t=1,\ldots,T$ are dual variables. The augment Lagrangian form of (54) is

$$\begin{aligned}L_\rho(\mathbf{w}_{T+1},\mathbf{Z},\mathbf{U})&=f_{T+1}(\mathbf{w}_{T+1})+\eta\sum_{t=1}^{T}\gamma_{T+1,t}\psi(\mathbf{z}_t-\mathbf{w}_t)\\&+\sum_{t=1}^{T}\left(\frac{\rho}{2}\|\mathbf{w}_{T+1}-\mathbf{z}_t+\mathbf{w}_t\|_2^2-\frac{\rho}{2}\|\mathbf{u}_t\|_2^2\right)\end{aligned} \tag{55}$$

Following the ADMM framework, we alternatively update $\mathbf{w}_{T+1}$, $\mathbf{Z}$, $\mathbf{U}$.

Update $\mathbf{w}_{T+1}$: We solve the following problem to update $\mathbf{w}_{T+1}$:

$$\mathbf{w}_{T+1}^{k+1}=\operatorname*{argmin}_{\mathbf{w}_{T+1}\geq 0}\ f_{T+1}(\mathbf{w}_{T+1})+\sum_{t=1}^{T}\frac{\rho}{2}\|\mathbf{w}_{T+1}-\mathbf{z}_t^k+\mathbf{u}_t^k\|_2^2 \tag{56}$$

Define $\boldsymbol{\theta}^k=\frac{1}{T}\sum_{t=1}^{T}(\mathbf{z}_t^k-\mathbf{u}_t^k)$, and we have

$$\mathbf{w}_{T+1}^{k+1}=\operatorname*{argmin}_{\mathbf{w}_{T+1}\geq 0}\ f_{T+1}(\mathbf{w}_{T+1})+\frac{T\rho}{2}\|\mathbf{w}_{T+1}-\boldsymbol{\theta}^k\|_2^2. \tag{57}$$

We use the PGD algorithm to solve (57), which is the same as that of Algorithm 1, and we will omit here.

Update $\mathbf{Z}$*:* The of updating each column of $\mathbf{Z}$ is

$$\begin{aligned}&\mathbf{z}_t^{k+1}\\&= \underset{\mathbf{z}_t\geq 0}{\operatorname{argmin}}\ \eta\,\gamma_{T+1,t}\psi\left(\mathbf{z}_t-\mathbf{w}_t\right)+\frac{\rho}{2}\|\mathbf{w}_{T+1}^{k+1}-\mathbf{z}_t+\mathbf{u}_t^k\|_2^2\\&= \underset{\mathbf{z}_t\geq 0}{\operatorname{argmin}}\ \frac{1}{2}\|\mathbf{z}_t-(\mathbf{w}_{T+1}^{k+1}+\mathbf{u}_t^k)\|_2^2+\frac{\eta\gamma_{T+1,t}}{\rho}\psi\left(\mathbf{z}_t-\mathbf{w}_t\right)\\&= \operatorname{prox}_{\frac{\eta\gamma_{T+1,t}}{\rho}\psi}\left(\mathbf{w}_{T+1}^{k+1}+\mathbf{u}_t^k-\mathbf{w}_t\right)+\mathbf{w}_t. \qquad (58)\end{aligned}$$

Update $\mathbf{U}$*:* The update of $\mathbf{U}$ is straight, i.e.,

$$\mathbf{u}_t^{k+1}=\mathbf{u}_t^k+\mathbf{w}_{T+1}^{k+1}-\mathbf{z}_t^{k+1} \qquad (59)$$

We update $\mathbf{w}_{T+1}$, $\mathbf{Z}$ and $\mathbf{U}$ alternatively until the algorithm converges. Note that client $T+1$ only requests $\mathbf{w}_1,\ldots,\mathbf{w}_T$ from local clients $1,\ldots,T$, without leakage of privacy.

2) *Fix* $\mathbf{w}_{T+1}$ *and Update* $\gamma_{T+1,t}$

The optimization problem of updating $\gamma_{T+1,t}$ is

$$\begin{aligned}\min_{\gamma_{T+1,t}\geq 0}\ &\sum_{t=1}^{T}\gamma_{T+1,t}\,\psi(\mathbf{w}_t-\mathbf{w}_{T+1})-\alpha_\gamma\log\left(\sum_{t=1}^{T}\gamma_{T+1,t}\right)\\&+\beta_\gamma\sum_{t=1}^{T}\gamma_{T+1,t}^2. \qquad (60)\end{aligned}$$

It is a classic graph learning framework, and many algorithms can be resorted to solve it [11].

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their very insightful and valuable reviews.

REFERENCES

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv:1709.05584*.
[2] L. Stanković, M. Daković, and E. Sejdić, "Introduction to graph signal processing," in *Vertex-Frequency Analysis of Graph Signals*. Berlin, Germany: Springer, 2019, pp. 3–108.
[3] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical Lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.
[4] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, May 2019.
[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[6] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, May 2019.
[7] M. Yuan and Y. Lin, "Model selection and estimation in the Gaussian graphical model," *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.
[8] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.
[9] L. Zhao, Y. Wang, S. Kumar, and D. P. Palomar, "Optimization algorithms for graph Laplacian estimation via ADMM and MM," *IEEE Trans. Signal Process.*, vol. 67, no. 16, pp. 4231–4244, Aug. 2019.
[10] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6350–6354.
[11] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2016, pp. 920–929.
[12] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.
[13] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
[14] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," in *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.
[15] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 6508–6512.
[16] X. Pu, T. Cao, X. Zhang, X. Dong, and S. Chen, "Learning to learn graph topologies," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 4249–4262.
[17] Y. Yuan, D. W. Soh, K. Guo, Z. Xiong, and T. Q. S. Quek, "Joint network topology inference via structural fusion regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, 2023.
[18] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 2826–2830.
[19] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," 2020, *arXiv:2001.03346*.
[20] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2017, pp. 205–213.
[21] F. Huang and S. Chen, "Joint learning of multiple sparse matrix gaussian graphical models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2606–2620, Nov. 2015.
[22] A. R. Gonçalves, F. J. Von Zuben, and A. Banerjee, "Multi-task sparse structure learning with Gaussian copula models," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1205–1234, 2016.
[23] X. Yang, M. Sheng, Y. Yuan, and T. Q. Quek, "Network topology inference from heterogeneous incomplete graph signals," *IEEE Trans. Signal Process.*, vol. 69, pp. 314–327, 2020.
[24] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *J. Roy. Statist. Soc. Ser. B-Stat. Methodol.*, vol. 76, no. 2, pp. 373–397, 2014.
[25] L. Gan, X. Yang, N. Narisetty, and F. Liang, "Bayesian joint estimation of multiple graphical models," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9802–9812.
[26] B. Hao, W. W. Sun, Y. Liu, and G. Cheng, "Simultaneous clustering and estimation of heterogeneous graphical models," *J. Mach. Learn. Res.*, vol. 18, pp. 1–58, 2018.
[27] S. Segarra, Y. Wang, C. Uhler, and A. G. Marques, "Joint inference of networks from stationary graph signals," in *Proc. IEEE Asilomar Conf. Signals Syst. Comput.*, 2017, pp. 975–979.
[28] M. Navarro, Y. Wang, A. G. Marques, C. Uhler, and S. Segarra, "Joint inference of multiple graphs from matrix polynomials," *J. Mach. Learn. Res.*, vol. 23, pp. 3302–3336, 2022.
[29] X. Zhang and Q. Wang, "Time-varying graph learning under structured temporal priors," in *Proc. 30th Eur. Signal Process. Conf.*, 2021, pp. 2141–2145.
[30] Y. Yuanet al., "GRACGE: Graph signal clustering and multiple graph estimation," *IEEE Trans. Signal Process.*, vol. 70, pp. 2015–2030, 2022.
[31] H. P. Maretic and P. Frossard, "Graph Laplacian mixture model," *IEEE Trans.Signal Inf. Process. Netw.*, vol. 6, pp. 261–270, 2020.
[32] H. Araghi, M. Sabbaqi, and M. Babaie-Zadeh, "k-graphs: An algorithm for graph signal clustering and multiple graph learning," *IEEE Signal Process. Lett.*, vol. 26, no. 10, pp. 1486–1490, Oct. 2019.
[33] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," 2012, *arXiv:1203.3536*.
[34] Y. Zhang and D.-Y. Yeung, "A regularization approach to learning task relationships in multitask learning," *ACM Trans. Knowl. Discov. Data*, vol. 8, no. 3, pp. 1–31, 2014.
[35] L. Han and Y. Zhang, "Learning multi-level task groups in multi-task learning," in *Proc. Natl. Conf. Artif. Intell.*, 2015, vol. 29, no. 1.
[36] L. Han and Y. Zhang, "Learning tree structure in multi-task learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2015, pp. 397–406.
[37] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, 2022.

[38] L. Xie, I. M. Baytas, K. Lin, and J. Zhou, "Privacy-preserving distributed multi-task learning with asynchronous updates," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 1195–1204.
[39] W. Wang, J. Wang, M. Kolar, and N. Srebro, "Distributed stochastic multi-task learning with graph regularization," 2018, *arXiv:1802.03830*.
[40] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2017, pp. 509–517.
[41] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
[42] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully decentralized joint learning of personalized models and collaboration graphs," in *Proc. Int. Conf. Artif. Intell. Stat. AISTATS.*, 2020, pp. 864–874.
[43] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning based on sparseness of temporal variation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5411–5415.
[44] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2015, pp. 387–396.
[45] A. Natali, E. Isufi, M. Coutino, and G. Leus, "Learning time-varying graphs from online data," *IEEE Open J. Signal Process.*, vol. 3, pp. 212–228, 2022.
[46] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Hanover, MD, USA: Now Publishers Inc, 2011.
[47] P. H. Calamai and J. J. Moré, "Projected gradient methods for linearly constrained problems," *Math. Prog.*, vol. 39, no. 1, pp. 93–116, 1987.
[48] J. Nocedal and S. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 2006.
[49] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.
[50] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Trans. Comput. Syst.*, vol. 25, no. 3, pp. 8–es, 2007.
[51] D. P. Bertsekas, "Nonlinear Programming," Belmont, MA, USA: Athena Scientific, 1999.
[52] E. N. Gilbert, "Random graphs," *Ann. Inst. Statist. Math.*, vol. 30, no. 4, pp. 1141–1144, 1959.
[53] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020, *arXiv:2010.16061*.
[54] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no. 3-5, pp. 75–174, 2010.
[55] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, Netherlands: Elsevier, 2011.
[56] R. K. Kana, L. E. Libero, and M. S. Moore, "Disrupted cortical connectivity theory as an explanatory model for Autism spectrum disorders," *Phys. Life Rev*, vol. 8, no. 4, pp. 410–437, 2011.
[57] D. L. Wallace, "Bounds on normal approximations to student's and the Chi-square distributions," *Ann. Inst. Statist. Math.*, vol. 30, pp. 1121–1130, 1959.
[58] S. S. Saboksayr, G. Mateos, and M. Cetin, "Online discriminative graph learning from multi-class smooth signals," *Signal Process.*, vol. 186, 2021, Art. no. 108101.

Xiang Zhang (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in aircraft design and engineering from the Nanjing University of Aeronautics and Astronautics, Jiangsu, China, in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree with the Department of Information Science and Engineering, Southeast University, Nanjing, China. His research interests include graph signal processing and graph machine learning.

Qiao Wang (Senior Member, IEEE) was born in Anqing, Anhui, China, in 1966. He received the B.S., M.S., and Ph.D. degrees in mathematics from Wuhan University, Wuhan, China, in 1988, 1994, and 1997, respectively. In 1997, he joined the School of Information Science and Engineering, Southeast University, Nanjing, China, and was appointed as an Associate Professor, in 1999, and then a Full Professor in 2001. From 2003 to 2004, he was a Visiting Scientist with Harvard University, Cambridge, MA, USA. His research interests include urban science, data science, multimedia, applied mathematics, and information theory. He was the recipient of the Excellence Design by the International Society of the Built Environment, in 2019, First Prize of China Construction Science and Technology Award in 2020 for research in urban design, 2021 National Excellent Urban Design First Prize, and Second Prize of the Science and Technology Progress Award of the Ministry of Education of China in 2021 for his research on data analysis of the diagnosis and treatment of depression. He is also an Executive Editor of the ICT Express of Elsevier.