Time-Varying Graph Learning Under Structured Temporal Priors

Xiang Zhang
School of Information Science and Engineering
Southeast University
Nanjing, China
xiangzhang369@seu.edu.cn

Qiao Wang
School of Information Science and Engineering
Southeast University
Nanjing, China
qiaowang@seu.edu.cn

***Abstract*—This paper endeavors to learn time-varying graphs by using structured temporal priors that assume underlying relations between arbitrary two graphs in the graph sequence. Different from many existing methods that only describe variations between two consecutive graphs, we propose a structure named *temporal graph* to characterize the underlying real temporal relations. Under this framework, classic priors like temporal homogeneity are actually special cases of our temporal graph. To address computational issue, we further develop a distributed algorithm based on Alternating Direction Method of Multipliers (ADMM) to solve the induced optimization problem. Numerical experiments on synthetic and real data demonstrate the superiorities of our method.**

***Index Terms*—ADMM, graph learning, structured temporal prior, time-varying graphs**

I. Introduction

Inferring the topology from data containing (hidden) structure, which is also called graph learning [1]–[4], has become a hot research topic since that prior graphs are usually unavailable for graph-based models in many applications, e.g., graph neural networks [5]. In parallel with statistical models [6], [7], graph signal processing (GSP) [8] plays a pivotal role in graph learning, which attempts to learn graphs from perspective of signal processing. One notable assumption that GSP based models leverage is smoothness, under which signal values of two connected vertices with large edge weights tend to be similar [3]. On the other hand, a typical feature existing in most models is that the environment is assumed to be static such that one can learn merely a single graph from all observed data. However, relationships between entities are usually time-varying in real world. Therefore, learning a series of time-varying graphs with timestamps is a more reasonable choice.

Current time-varying graph learning methods attempt to jointly learn graphs of all time slots by exploiting prior assumptions about evolutionary patterns of dynamic graphs [9]. One may find that the most used assumptions here is temporal homogeneity [10], under which only a small number of edges are allowed to change between two consecutive graphs. The essence of prior assumptions like temporal homogeneity is to establish temporal relations between graphs of different time slots using prior knowledge, which are crucial for learning time-varying graphs since they actually bring structural information, in addition to data, to learning process.

Albeit interesting, assumption of temporal homogeneity only cares about variations between graphs in neighboring time slots and treats them equally. Obviously, this assumption is simple enough but may be inconsistent with the real temporal relations in some applications. Here we take crowd flow networks of urban area as an example. The variations of crowd flow networks at different time periods in a day are not uniform due to the difference of travel behaviour [11]. For example, the patterns in early morning (1 a.m.–5 a.m.) are apparently different from those in rush hours (7 a.m.–9 a.m.). Thus, it is not reasonable to treat all variations equally. Furthermore, common knowledge tells us that networks in the same time period of two different working days, e.g., 10 a.m. in Monday and Tuesday, are also similar. Capturing this periodic pattern is beyond the ability of the temporal homogeneity assumption.

To this end, a more general time-varying graph learning method should be proposed by generalizing the assumption of temporal homogeneity. In this paper, a flexible structure named *temporal graph* is leveraged to describe structured temporal relations of time-varying graphs. In temporal graph, relations between graphs of any paired time slots, not limited to adjacent time slots, can be established, and we use weights to measure the "closeness" of these relations. Therefore, temporal homogeneity assumption [10] is a special case of our framework. Furthermore, the algorithm for solving the classic model [10] suffers from increasing complexity as the number of time slots. To address computational issue, a distributed algorithm based on Alternating Direction Method of Multipliers (ADMM) is developed to solve the induced optimization problem, which can save considerable time when the number of time periods is large. Numerical tests illustrate that our method outperforms the state-of-the-art methods in face of intricate temporal structures.

II. Preliminaries

We will learn undirected graphs $\mathcal{G}$ with non-negative weights. Given N observed signals $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ generated from $\mathcal{G}$, graph learning is aimed to infer the adjacency matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ of $\mathcal{G}$. Under smoothness priors, it is equivalent to solving the following problem [2]

$$\min_{\mathbf{A}\in\mathcal{A}} \|\mathbf{A}\circ\mathbf{R}\|_1 - \alpha\mathbf{1}^\top\log(\mathbf{A}\mathbf{1}) + \frac{\beta}{2}\|\mathbf{A}\|_{\mathrm{F}}^2, \qquad (1)$$

where $\circ$ is Hadamard product and $\mathbf{1} = [1, \ldots, 1]^\top \in \mathbb{R}^d$ is a column vector of ones. Parameters α and β are predefined constants. Furthermore, $\mathcal{A}$ is the set defined as [2]

$$\mathcal{A} = \left\{ \mathbf{A} : \mathbf{A} \in \mathbb{R}_+^{d \times d}, \mathbf{A} = \mathbf{A}^\top, \mathrm{diag}(\mathbf{A}) = \mathbf{0} \right\}, \quad (2)$$

where $\mathbb{R}_+$ is the set of nonnegative real numbers and $\mathbf{0} \in \mathbb{R}^d$ is a column vector of zeros. For data matrix $\mathbf{X} \in \mathbb{R}^{d \times N} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_d]^\top$, the pairwise distance matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ in (1) is defined as

$$\mathbf{R}_{[ij]} = \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2, \quad (3)$$

where $\mathbf{R}_{[ij]}$ is the (i, j) entry of $\mathbf{R}$. The first term of (1) is the smoothness of the observed signals over $\mathcal{G}$. Besides, the second and third term control the degree of each node and sparsity of edges [2]. Note that $\mathbf{A}$ is a symmetric matrix with diagonal entries equal to zero, and hence the number of free variables of $\mathbf{A}$ is $p \triangleq \frac{d(d-1)}{2}$. We define a vector $\mathbf{w} \in \mathbb{R}^p$ whose entries are the upper right variables of $\mathbf{A}$. Therefore, problem (1) can be rewritten as [2]

$$\min_{\mathbf{w} \geq 0} f(\mathbf{w}) = \min_{\mathbf{w} \geq 0} 2\mathbf{r}^\top \mathbf{w} - \alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}) + \beta \|\mathbf{w}\|_2^2, \quad (4)$$

where the linear operator $\mathbf{S}$ satisfies $\mathbf{S}\mathbf{w} = \mathbf{A}\mathbf{1}$ and $\mathbf{r}$ is the vector form of the upper right variables of $\mathbf{R}$.

Under these notations, the time-varying graph learning will learn a series of graphs $\mathbf{w}_1, \ldots, \mathbf{w}_T$ using signals $\mathbf{X}_1, \mathbf{X}_2, ..\mathbf{X}_T$ collected during T time periods, where $\mathbf{X}_t \in \mathbb{R}^{d \times N}$ is the data matrix of time slot t. Specifically, temporal homogeneity assumption based model is formulated as [10]

$$\begin{aligned} &\min_{\mathbf{w}_t \geq 0} \sum_{t=1}^{T} f_t(\mathbf{w}_t) + \eta \sum_{t=2}^{T} \|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1 \\ &= \min_{\mathbf{w}_t \geq 0} \sum_{t=1}^{T} 2\mathbf{r}_t^\top \mathbf{w}_t - \alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}_t) + \beta \|\mathbf{w}_t\|_2^2 \\ &\quad + \eta \sum_{t=2}^{T} \|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1, \end{aligned} \quad (5)$$

where η is a global parameter controls the weight of temporal priors and $\mathbf{r}_t$ is calculated using $\mathbf{X}_t$. The last term of (5) indicates that only a small number of edges are allowed to change between two consecutive graphs.

III. Proposed Framework

Observe (5) and we can find that temporal homogeneity prior only imposes constraints on variations of graphs between two adjacent time slots equally, which is too simple and may fail to characterize the temporal relations in real world. In this paper, we suggest a general structure named *temporal graph* to describe temporal relations of time-varying graphs. The temporal graph $\mathcal{G}_N$ is a graph structure whose nodes represent graphs of T time slots and edges indicate the relationships between the connected nodes, i.e, constraints on variations between the corresponding graphs. As shown in Fig.1, temporal graph is undirected but with nonnegative weighted edges. Any two nodes can be connected in temporal graph, e.g., γ_{1t} in Fig.1, instead of allowing merely two consecutive graphs connection. Furthermore, we give up treating these temporal constraints equally and use edge weights to measure the

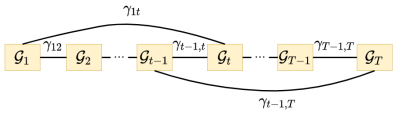


Fig. 1: A prototype of temporal graph structure

"closeness" of them. Clearly, temporal graph is more general and able to describe intricate temporal structures in real world.

Formally, we define a temporal graph $\mathcal{G}_N = \{\mathcal{V}_N, \mathcal{E}_N\}$, where $\mathcal{V}_N$ is the node set containing graphs of all time slots and $\mathcal{E}_N$ is the edge set containing connections between these graphs. In this paper, we suppose that there are T nodes and s edges in $\mathcal{G}_N$, i.e., $|\mathcal{V}_N| = T$ and $|\mathcal{E}_N| = s$. Time-varying graph learning using temporal graph is then formulated as

$$\min_{\mathbf{w}_t \geq 0} \sum_{t \in \mathcal{V}_N} f_t(\mathbf{w}_t) + \eta \sum_{(i,j) \in \mathcal{E}_N} \gamma_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_1, \quad (6)$$

where γ_{ij} is the relative weight between the i-th and j-th time slots. Parameter η is used to scale the weight of edge objectives relative to node objectives. Note that, (6) will reduce to (5) if we build $\mathcal{G}_N$ as a chain structure with equal weights.

The design of $\mathcal{G}_N$ is based on our prior knowledge of temporal relations. We only need to care about the relative closeness of two graphs i.e., γ_{ij}, which is quantified by experience or other auxiliary knowledge. Global weight η can be determined by cross validation. The main contribution of temporal graph $\mathcal{G}_N$ is providing a more flexible structure to describe temporal relations fusing our prior knowledge. In the worst case, when we have no structured temporal priors, our framework can still boil down to model (5) but provides a more efficient algorithm introduced in the next section.

IV. ADMM based algorithm

The algorithm in [10] for solving (5) attempts to learn $\mathcal{G}_1, ..., \mathcal{G}_T$ in a centralized fashion, resulting in increasing complexity as T. We hence develop a novel distributed algorithm based on ADMM framework [12]–[14] to solve (6). The algorithm is able to learn graphs of different time slots in parallel, showing its efficiency when T is large. For an edge $(i, j) \in \mathcal{E}_N$, we first introduce a consensus variable of $\mathbf{w}_i$, denoted as $\mathbf{z}_{ij}$. In fact, $\mathbf{z}_{ij}$ represents the connection starting from i to j. For the same edge, $\mathbf{z}_{ji}$ is the consensus variable of $\mathbf{w}_j$. With consensus variables, (6) is equivalent to the following problem

$$\begin{aligned} &\min_{\mathbf{w}_t \geq 0} \sum_{t \in \mathcal{V}_N} f_t(\mathbf{w}_t) + \eta \sum_{(i,j) \in \mathcal{E}_N} \gamma_{ij} \|\mathbf{z}_{ij} - \mathbf{z}_{ji}\|_1 \\ &\text{s.t. } \mathbf{w}_i = \mathbf{z}_{ij}, \text{ for } i = 1, ..., T \text{ and } j \in \mathcal{M}(i), \end{aligned} \quad (7)$$

where $\mathcal{M}(i)$ denotes the set of all the nodes that are connected with node i in $\mathcal{G}_N$. We define a matrix $\mathbf{W} \in \mathbb{R}^{p \times T} \triangleq [\mathbf{w}_1, ..., \mathbf{w}_T]$ containing all primal variables. In addition, matrices of consensus variables $\mathbf{Z} \in \mathbb{R}^{p \times 2s}$ and dual variables $\mathbf{U} \in \mathbb{R}^{p \times 2s}$ are also defined. For the n-th edge (i, j) in temporal graph $\mathcal{G}_N$, $n = 1, \ldots, s$, the corresponding consensus variable vectors $\mathbf{z}_{ij}, \mathbf{z}_{ji}$ are the $(2n-1)$-th and $2n$-th columns

of $\mathbf{Z}$, respectively. This also holds true for matrix $\mathbf{U}$. The scaled form of augmented Lagrangian of (7) is obtained as

$$\begin{aligned} &L_\rho(\mathbf{W},\mathbf{Z},\mathbf{U}) \\ &=\sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}_t)+\eta\sum_{(i,j)\in\mathcal{E}_N}\gamma_{ij}\left\|\mathbf{z}_{ij}-\mathbf{z}_{ji}\right\|_1 \\ &\quad+\sum_{(i,j)\in\mathcal{E}_N}\Bigg(\frac{\rho}{2}\left(\|\mathbf{u}_{ij}\|_2^2+\|\mathbf{u}_{ji}\|_2^2\right) \\ &\quad+\frac{\rho}{2}\left(\|\mathbf{w}_i-\mathbf{z}_{ij}+\mathbf{u}_{ij}\|_2^2+\|\mathbf{w}_j-\mathbf{z}_{ji}+\mathbf{u}_{ji}\|_2^2\right)\Bigg), \end{aligned} \tag{8}$$

where $\rho>0$ is an ADMM penalty parameter [14]. Following ADMM framework, we alternately update $\mathbf{W},\mathbf{U}$ and $\mathbf{Z}$.

1) Update $\mathbf{W}$: For $\mathbf{W}=[\mathbf{w}_1,\ldots,\mathbf{w}_T]$, the update of $\left(\mathbf{w}_t^{k+1},1\le t\le T\right)$ is as follows

$$\begin{aligned} &\left(\mathbf{w}_t^{k+1},1\le t\le T\right) \\ &=\underset{\mathbf{w}_t\ge 0}{\operatorname{argmin}}\sum_{t\in\mathcal{V}_N} f_t(\mathbf{w}_t) \\ &\quad+\frac{\rho}{2}\sum_{(i,j)\in\mathcal{E}_N}\left(\|\mathbf{w}_i-\mathbf{z}_{ij}^k+\mathbf{u}_{ij}^k\|_2^2+\|\mathbf{w}_j-\mathbf{z}_{ji}^k+\mathbf{u}_{ji}^k\|_2^2\right). \end{aligned} \tag{9}$$

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

$$\mathbf{w}_t^{k+1}=\underset{\mathbf{w}_t\ge 0}{\operatorname{argmin}}\ f_t(\mathbf{w}_t)+\frac{\rho}{2}\sum_{j\in\mathcal{M}(t)}\|\mathbf{w}_t-\mathbf{z}_{tj}^k+\mathbf{u}_{tj}^k\|_2^2. \tag{10}$$

If we let $\boldsymbol{\theta}_t^k\triangleq\frac{1}{m}\sum_{j\in\mathcal{M}(t)}\left(\mathbf{z}_{tj}^k-\mathbf{u}_{tj}^k\right)$, where $m=|\mathcal{M}(t)|$, (10) can be reformulated as

$$\begin{aligned} \mathbf{w}_t^{k+1}&=\underset{\mathbf{w}_t\ge 0}{\operatorname{argmin}}\ f_t(\mathbf{w}_t)+\frac{m\rho}{2}\|\mathbf{w}_t-\boldsymbol{\theta}_t^k\|_2^2 \\ &\triangleq\underset{\mathbf{w}_t\ge 0}{\operatorname{argmin}}\ g_t(\mathbf{w}_t). \end{aligned} \tag{11}$$

In this paper, we use projected gradient descent (PGD) algorithm [16] to solve problem (11). The gradient of the objective function of (11) is as follows

$$\nabla g_t(\mathbf{w}_t)=2\mathbf{r}_t+2\beta\mathbf{w}_t+m\rho(\mathbf{w}_t-\boldsymbol{\theta}_t^k)-\alpha\mathbf{S}^\top\left(\mathbf{S}\mathbf{w}_t\right)^{\cdot(-1)}, \tag{12}$$

where $.(-1)$ is an elementwise reciprocal operator. We set $\mathbf{y}^0=\mathbf{w}_t^k$ and iteratively update $\mathbf{y}^r$ using

$$\mathbf{y}^{r+1}=\left(\mathbf{y}^r-\epsilon\nabla g_t(\mathbf{y}^r)\right)_+, \tag{13}$$

until it converges to $\mathbf{y}^*$ with a certain precision, where $(\cdot)_+\triangleq\max(\cdot,0)$, r is the number of iterations of PGD algorithm and ϵ the step size. After obtaining the solution $\mathbf{y}^*$ of (11), we set $\mathbf{w}_t^{k+1}=\mathbf{y}^*$. Note that all $\mathbf{w}_t$ can be updated in parallel.

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

$$\begin{aligned} &\mathbf{z}_{ij}^{k+1},\mathbf{z}_{ji}^{k+1} \\ &=\underset{\mathbf{z}_{ij},\mathbf{z}_{ji}}{\operatorname{argmin}}\ \eta\gamma_{ij}\left\|\mathbf{z}_{ij}-\mathbf{z}_{ji}\right\|_1 \\ &\quad+\frac{\rho}{2}\left(\left\|\mathbf{w}_i^{k+1}-\mathbf{z}_{ij}+\mathbf{u}_{ij}^k\right\|_2^2+\left\|\mathbf{w}_j^{k+1}-\mathbf{z}_{ji}+\mathbf{u}_{ji}^k\right\|_2^2\right). \end{aligned} \tag{14}$$

It is difficult to solve (14) due to that $\mathbf{z}_{ij}$ and $\mathbf{z}_{ji}$ are coupled with each other in $\left\|\mathbf{z}_{ij}-\mathbf{z}_{ji}\right\|_1$. Inspired by the method proposed in [17], we define a function $\tilde{\psi}$

$$\tilde{\psi}\left(\begin{bmatrix}\mathbf{z}_{ij}\\ \mathbf{z}_{ji}\end{bmatrix}\right)=\left\|\mathbf{z}_{ij}-\mathbf{z}_{ji}\right\|_1, \tag{15}$$

Algorithm 1 ADMM based algorithm

Input:
α, β, η, ρ, the predefined $\mathcal{G}_N$, signals $\mathbf{X}_1$,..., $\mathbf{X}_T$
Output:
The learned graph $\mathbf{w}_1,\ldots,\mathbf{w}_T$

```
Initialize w_t^0, z_ij^0 and u_ij^0 for t ∈ V_N, (i,j) ∈ E_N, set k = 0
while stop criterion not satisfied do
    Update w_1^{k+1}, ...,w_T^{k+1} using PGD in parallel
    Update z_ij^{k+1}, z_ji^{k+1} for (i,j) ∈ E_N using (18) in parallel
    Update u_ij^{k+1},u_ji^{k+1} for (i,j) ∈ E_N using (19) in parallel
    k = k + 1
end while
return w_1^k, w_2^k, ...,w_T^k
```

with which (14) can be solved by

$$\begin{bmatrix}\mathbf{z}_{ij}^{k+1}\\ \mathbf{z}_{ji}^{k+1}\end{bmatrix}=\operatorname{prox}_{\frac{\eta\gamma_{ij}}{\rho}\tilde{\psi}}\left(\begin{bmatrix}\mathbf{u}_{ij}^k+\mathbf{w}_i^{k+1}\\ \mathbf{u}_{ji}^k+\mathbf{w}_j^{k+1}\end{bmatrix}\right), \tag{16}$$

where $\operatorname{prox}_{\frac{\eta\gamma_{ij}}{\rho}\tilde{\psi}}(\cdot)$ is the proximal operator of function $\tilde{\psi}$ [18]. However, we have no knowledge of the closed form of the operator $\operatorname{prox}_{\frac{\eta\gamma_{ij}}{\rho}\tilde{\psi}}(\cdot)$. Hence a property of proximal operators mentioned in [17] might be introduced here.

Property 1. *If a function* $h_1(\mathbf{v})=h_2(\mathbf{G}\mathbf{v}+\mathbf{H})$*, and* $\mathbf{G}\mathbf{G}^\top=\frac{1}{\lambda}\mathbf{I}$*, where* $\mathbf{I}$ *is an identity matrix, then*

$$\begin{aligned} &\operatorname{prox}_{h_1}(\mathbf{v}) \\ &=(\mathbf{I}-\lambda\mathbf{G}^\top\mathbf{G})\mathbf{v}+\lambda\mathbf{G}^\top(\operatorname{prox}_{\frac{1}{\lambda}h_2}(\mathbf{G}\mathbf{v}+\mathbf{H})-\mathbf{H}). \end{aligned} \tag{17}$$

In our problem, $h_1=\tilde{\psi},h_2=\ell_1$ norm, $\mathbf{G}=[-\mathbf{I}\ \mathbf{I}]$, $\mathbf{H}$ is zero matrix and $\lambda=\frac{1}{2}$. According to Property 1, the following update can be easily reached for (16),

$$\begin{aligned} \begin{bmatrix}\mathbf{z}_{ij}^{k+1}\\ \mathbf{z}_{ji}^{k+1}\end{bmatrix}&=\frac{1}{2}\begin{bmatrix}\mathbf{u}_{ij}^k+\mathbf{w}_i^{k+1}+\mathbf{u}_{ji}^k+\mathbf{w}_j^{k+1}\\ \mathbf{u}_{ij}^k+\mathbf{w}_i^{k+1}+\mathbf{u}_{ji}^k+\mathbf{w}_j^{k+1}\end{bmatrix} \\ &+\frac{1}{2}\begin{bmatrix}-\operatorname{prox}_{\frac{2\eta\gamma_{ij}}{\rho}\|\cdot\|_1}\left(\mathbf{w}_j^{k+1}+\mathbf{u}_{ji}^k-\mathbf{w}_i^{k+1}-\mathbf{u}_{ij}^k\right)\\ \operatorname{prox}_{\frac{2\eta\gamma_{ji}}{\rho}\|\cdot\|_1}\left(\mathbf{w}_j^{k+1}+\mathbf{u}_{ji}^k-\mathbf{w}_i^{k+1}-\mathbf{u}_{ij}^k\right)\end{bmatrix}. \end{aligned} \tag{18}$$

Now each column of $\mathbf{Z}$ can be updated in parallel. Finally, we can update $\mathbf{u}$ in parallel using

$$\begin{aligned} \mathbf{u}_{ij}^{k+1}&=\mathbf{u}_{ij}^k+\mathbf{w}_i^{k+1}-\mathbf{z}_{ij}^{k+1} \\ \mathbf{u}_{ji}^{k+1}&=\mathbf{u}_{ji}^k+\mathbf{w}_j^{k+1}-\mathbf{z}_{ji}^{k+1}. \end{aligned} \tag{19}$$

In summary, our algorithm can be implemented in a distributed fashion since the columns of $\mathbf{W},\mathbf{Z}$ and $\mathbf{U}$ can all be updated in parallel. The global convergence is also guaranteed by ADMM framework since (6) is a convex problem. Furthermore, the stopping criterion of ADMM framework can be referred in [14].

V. Numerical Experiments

A. Synthetic Data

We first validate the strengths of our framework and algorithm using synthetic data. The temporal structure we use

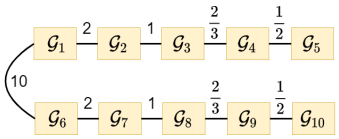


Fig. 2: Non-chain structure

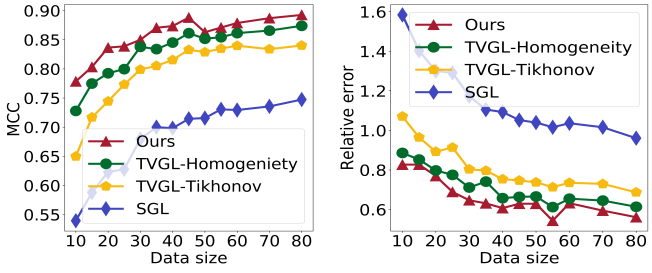
TABLE I: Running time (s) v.s. T: all results take the form of logarithm ($\log_{10}$)

T	2	5	10	15	20	25	30	35
PDS	1.575	2.562	3.231	3.665	3.901	4.083	4.187	4.304
Ours	0.802	1.473	1.947	2.099	2.136	2.202	2.523	2.502

is shown in Fig.2. It is an unchained structure where $\mathcal{G}_6$ is connected with $\mathcal{G}_1$ instead of $\mathcal{G}_5$. To obtain time-varying graphs, an initial RBF graph $\mathcal{G}_1$ with 20 vertices is generated in the same way as [3]. After that, $\mathcal{G}_2$ is obtained by changing edges in $\mathcal{G}_1$ randomly and the number of the changed edges is inverse proportion with the edge weights of $\mathcal{G}_N$ in Fig.2. Following this way, we can generate other graphs sequentially. We emphasis that $\mathcal{G}_6$ is generated based on $\mathcal{G}_1$ instead of $\mathcal{G}_5$. Smooth graph signals $\mathbf{X}_t$ of each $\mathcal{G}_t$ are generated from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{L}_t^\dagger)$, where $\mathbf{L}_t$ is the Laplacian matrix of $\mathcal{G}_t$ and $\dagger$ is the pseudo inverse. See [3] for more details. The adopted evaluation metrics are Matthews correlation coefficient (MCC) [19] and relative error, each averaged over all time. MCC is a metric representing the accuracy of the estimated graph topology and its value is between -1 and 1 (-1 represents completely wrong detection while +1 means completely right detection). Relative error is defined as $\|\mathbf{A}^* - \mathbf{A}_{\text{gt}}\|_{\text{F}}/\|\mathbf{A}_{\text{gt}}\|_{\text{F}}$, where $\mathbf{A}^*$ is the learned adjacency matrix and $\mathbf{A}_{\text{gt}}$ is the groundtruth. Three baselines are leveraged, i.e., SGL (learn graphs of each time periods independently), TVGL-Tikhonov [9] and TVGL-Homogeneity [10]. The last two are time-varying models with a chained temporal structure. Following the method of parameter selection in [2], we fix $\alpha = 2$ and find the best β by grid search [2]. Furthermore, we choose η that maximizes MCC, which is 2.5. In ADMM framework, ρ is set to be 0.5 and tolerance values are set to be 10^{-3} (both relative and absolute tolerance) [14]. The parameters of baselines are all selected as the ones corresponding to the best MCC values. The following results are the average of 20 independent experiments. All algorithms are implemented by python and run on an Intel(R) Xeon(R) CPU with 2.10GHz clock speed and 256GB of RAM.

Figure 3 shows the performance of different data size N of each time slots. We can observe that SGL reaches the worst performance since no temporal priors are exploited. The performance of TVGL-Tikhonov and TVGL-Homogeneity is inferior to ours due to that their chain structure fails to characterize the real temporal structure depicted in Fig.3. On the contrary, our framework is able to describe the unchained structure easily thanks to the strong representation ability of temporal graph. Therefore, our method is superior to the other baselines when faced with intricate temporal structures.

We also compare the efficiency of our algorithm with that of



(a) (b)

Fig. 3: Performance of the learned graph with different sample size (a) MCC; (b) Relative error

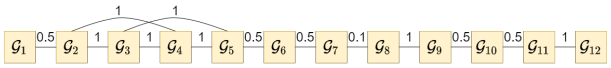


Fig. 4: The designed temporal structure

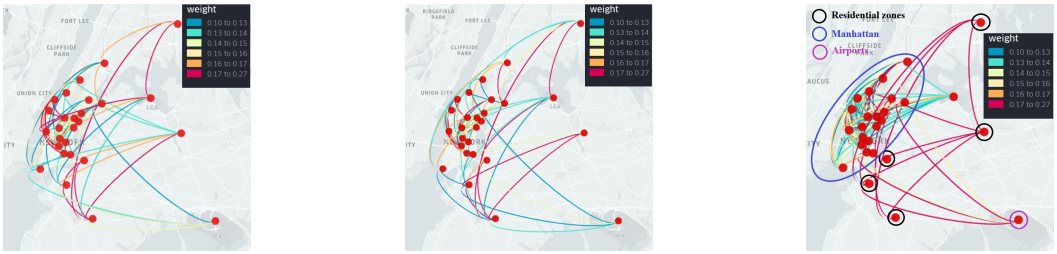
centralized PDS algorithm in [10]. We fix $d = 100$ and apply our method to chain temporal structure problems defined in (5). This is feasible since chain structure is a special case of our framework. We implement our algorithm in a distributed way. Our code is run on different cores of a single machine, and 25 cores are used. The results of two algorithms are listed in Table I. We take logarithm ($\log_{10}$) on the results for ease of presentation. We find that the running time of PDS is significantly greater than ours especially when T is large. The runtime of our algorithm increases slowly with T thanks to the distributed feature. A great increase occurs in our algorithm when $T = 30$, which is caused by that the number of used cores is 25, and additional waiting time is required when $T > 25$.

B. Real Data

Our framework is also applied to the Yellow Taxi Trip data of New York city[1] to learn the time-varying travel relationships between different taxi zones. The data record timestamps and locations of pickups of taxi orders. We focus on data from 0 a.m. to 12 a.m. and learn a graph for each hour, which means that 12 time slots, as well as graphs, are finally obtained. The city are divided into 27 zones and the number of taxi pickups of each zones within 15 minutes are taken as signals for that zone. A total of 80 graphs signals for each zone are collected, i.e., $\mathbf{X}_t \in \mathbb{R}^{27\times 80}$ for each t, since we only select data of 20 workdays in September of 2018.

We then design a temporal structure $\mathcal{G}_N$, which is shown in Fig.4, based on our prior knowledge of the variations of crowd flow networks. Note that temporal structure in Fig.4 is not a chain structure since the variations of crowd flow patterns at different time duration in one day are not uniform due to the diversity of travel behaviour. In the early morning, most people are in sleep and the crowd mobility patterns may stay static. Therefore, we connect graphs in early morning with each other, i.e., from $\mathcal{G}_2$ to $\mathcal{G}_5$, even they are not adjacent in time. Additionally, it is common sense that crowd mobility

[1]The data is available at https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data.



(a) $\mathcal{G}_3$: 2 a.m.-3 a.m. (b) $\mathcal{G}_5$: 4 a.m.-5a.m. (c) $\mathcal{G}_9$: 8 a.m.-9 a.m.

(d) $\mathcal{G}_3$: 2 a.m.-3 a.m. (e) $\mathcal{G}_5$: 4 a.m.-5 a.m. (f) $\mathcal{G}_9$: 8 a.m.-9 a.m.

Fig. 5: The learned graphs of taxi zones of New York. The upper row (a-c) shows results of our model and the lower row (d-e) shows the results of TV-Homogeneity model.

patterns change significantly in rush hours. Hence we set the smallest weight between $\mathcal{G}_7$ and $\mathcal{G}_8$.

We observe from Fig.5 that $\mathcal{G}_3$ and $\mathcal{G}_5$ learned by our method are similar despite they are not in consecutive time slots. It makes sense since the travel patterns in early morning should almost stay unchanged. However, temporal homogeneity assumption fails to capture this temporal characteristic. Compared with graphs of $\mathcal{G}_3$ and $\mathcal{G}_5$, $\mathcal{G}_9$ learned by our method shows the following changes. 1) Connections between residential zones are strengthened. This is caused by the fact that most people travel to work from home in rush hours. Therefore, the travel patterns of these zones are similar. 2) Connections between zones in Manhattan area are also strengthened due to the fact that more people take taxi to work area in Manhattan. However, these changes of TV-Homogeneity are less obvious than ours since it treats all variations equally.

VI. Conclusion

In this paper, we propose a general time-varying graph learning framework, under which temporal graph is employed to describe temporal structures. A distributed algorithm using ADMM framework is developed to solve the induced optimization problem. Experimental results show that our framework outperforms the state-of-the-art methods when facing complicated temporal structures.

References

[1] 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, 2019.

[2] V. Kalofolias, "How to learn a graph from smooth signals," in *Artif. Intel. and Stat. (AISTATS)*, 2016, pp. 920–929.

[3] 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, 2016.

[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, 2019.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw Learn Syst.*, vol.32, no. 1, pp. 4–24, 2020.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[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] A. Ortega, P. Frossard, and J. Kovačević, J. Moura and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[9] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2017, pp. 2826–2830.

[10] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," *arXiv preprint arXiv:2001.03346 [eess.SP]*, 2020.

[11] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Proc. Netw.*, vol. 3, no. 3, pp. 484–499, 2017.

[12] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[13] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[14] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011.

[15] N. Komodakis and J. Pesquet, "Playing with duality: An overview of recent primal dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no.6, pp. 31–54, 2015.

[16] P. Calamai and J. More "Projected gradient methods for linearly constrained problems," *Math Program.*, vol. 39, no. 1, pp. 93–116, 1987.

[17] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying Graphical Lasso," in *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 205–213.

[18] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, p. 127–239, 2014.

[19] D. Powers, "Evaluation: from precision, recalland f-measure to roc, informedness, markedness andcorrelation," *arXiv preprint arXiv:2010.16061*, 2020.