Journal of the American Statistical Association

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uasa20

Learning Coefficient Heterogeneity over Networks: A Distributed Spanning-Tree-Based Fused-Lasso Regression

Xin Zhang, Jia Liu & Zhengyuan Zhu

To cite this article: Xin Zhang, Jia Liu & Zhengyuan Zhu (2022): Learning Coefficient Heterogeneity over Networks: A Distributed Spanning-Tree-Based Fused-Lasso Regression, Journal of the American Statistical Association, DOI: 10.1080/01621459.2022.2126363

To link to this article: https://doi.org/10.1080/01621459.2022.2126363

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2022, VOL. 00, NO. 0, 1–13: Theory and Methods
https://doi.org/10.1080/01621459.2022.2126363

Taylor & Francis
Taylor & Francis Group

Learning Coefficient Heterogeneity over Networks: A Distributed Spanning-Tree-Based Fused-Lasso Regression

Xin Zhang[a], Jia Liu[b], and Zhengyuan Zhu[a]

[a]Department of Statistics, Iowa State University, Ames, IA; [b]Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH

ABSTRACT

Identifying the latent cluster structure based on model heterogeneity is a fundamental but challenging task arises in many machine learning applications. In this article, we study the clustered coefficient regression problem in the distributed network systems, where the data are locally collected and held by nodes. Our work aims to improve the regression estimation efficiency by aggregating the neighbors' information while also identifying the cluster membership for nodes. To achieve efficient estimation and clustering, we develop a distributed spanning-tree-based fused-lasso regression (DTFLR) approach. In particular, we propose an adaptive spanning-tree-based fusion penalty for the low-complexity clustered coefficient regression. We show that our proposed estimator satisfies statistical oracle properties. Additionally, to solve the problem parallelly, we design a distributed generalized alternating direction method of multiplier algorithm, which has a simple node-based implementation scheme and enjoys a linear convergence rate. Collectively, our results in this article contribute to the theories of low-complexity clustered coefficient regression and distributed optimization over networks. Thorough numerical experiments and real-world data analysis are conducted to verify our theoretical results, which show that our approach outperforms existing works in terms of estimation accuracy, computation speed, and communication costs. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received July 2021
Accepted August 2022

KEYWORDS
Adaptive fused-lasso; Distributed generalized ADMM; Minimum spanning tree; MODIS; Sensornets

1. Introduction

In recent years, distributed statistical learning has attracted growing research interest due to its advantages in computation efficiency, data privacy and system scalability. In contrast to the traditional centralized statistical analysis where models are estimated with the data stored in a central server, in the context of distributed statistical learning, the data are locally collected and remain distributed over multiple local machines, which could be mobile devices, wireless sensors, hospital data systems, or some other local data sources (Damiani et al. 2015; Jochems et al. 2016; Guo et al. 2020; Cadena et al. 2021). Through mutually sharing the local statistics, all the local machines in the distributed system collaborate with each other to perform the global model estimation and statistical analysis. So far, various distributed statistical methods have been proposed and studied, such as distributed sparsity learning (Mateos, Bazerque, and Giannakis 2010), distributed nonparametric estimation (Lin, Wang, and Zhou 2020; Xu et al. 2021) and distributed Bayesian analysis (Xu et al. 2014), etc.

In this article, we consider a fundamental *distributed* regression problem with clustered coefficients over networks: Suppose there are K nodes in the network, each of which collects its local dataset $D_k = \{(\mathbf{x}_{k,i}, y_{k,i})\}_{i=1}^n$, where $\mathbf{x}_{k,i} \in \mathbb{R}^d$ and $y_{k,i} \in \mathbb{R}$ $(k = 1, \ldots, K)$ represent the ith covariate vector and response in the kth dataset, respectively; and n denotes the size of the dataset. For ease of exposition, the size of each dataset is assumed to be balanced (i.e., all nodes have n samples).[1] Hence, the total sample size in the network is $N = Kn$. We assume that there exist S unknown underlying clusters of the nodes. Further, the data pairs $(\mathbf{x}, y)$ from the sth cluster follow a common linear model:

$$y = \boldsymbol{\beta}_s^\top \mathbf{x} + \varepsilon, \quad s = 1, \ldots, S, \tag{1}$$

where $\boldsymbol{\beta}_s = [\boldsymbol{\beta}_{s,1}, \ldots, \boldsymbol{\beta}_{s,d}]^\top \in \mathbb{R}^d$ is the coefficient vector for the sth cluster, the independent error ε has a zero mean and a known variance σ^2. The above linear model varies across the underlying clusters, and the datasets in the same cluster s share the same coefficient $\boldsymbol{\beta}_s$, thus the name of clustered coefficient regression. In our problem, the number of clusters S, the nodes' cluster membership and their model coefficients are all unknown. Our goal is to identify the cluster membership of each node and estimate their corresponding coefficient. However, due to communication limitation or privacy restrictions, one cannot merge these datasets to a single node. Thus, the main challenge of this problem is to perform clustering and estimate the coefficients of each cluster in the network in a *distributed* fashion.

The above problem naturally arises in many machine learning applications. For example, a wireless sensor network is

[1] Our algorithms and results in this article can easily be extended to cases with datasets of unbalanced sizes.

CONTACT Zhengyuan Zhu zhuz@iastate.edu Department of Statistics, Iowa State University, Ames, IA
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA.

deployed in a large spatial domain to collect and learn the relationship between the soil temperature y and air temperature $\mathbf{x}$ (Lee, Zhu, and Toscas 2015). The spatial domain can be divided into several subregions due to the landcover types (e.g., forest, grassland, etc.), and temperature relationships may vary geographically: sensors in the same subregion may share the same regression relationship, and the coefficients vary across different subregions. Similar scenarios could also emerge in other applications, such as meta-analysis on medical data (Tang and Song 2016), federated learning on the speech analysis (Konecny, McMahan, and Ramage 2015), to name just a few.

Unfortunately, distributively clustering nodes based on regression model over networks is challenging as it includes two nontrivial *inter-dependent* and *conflicting* subtasks: (a) statistical estimator design and (b) distributed optimization under the proposed estimator. In the literature, there exist spanning-tree-based centralized estimator designs that achieve strong statistical performance guarantee with $\Theta(K)$ computational complexity (e.g., Li and Sang (2018), see Section 2 for detailed discussions). However, the spanning-tree-based penalty architectures make it difficult to design distributed optimization algorithms. On the other hand, there exist efficient distributed algorithms for solving related clustering problems over networks (e.g., Hallac, Leskovec, and Boyd (2015), Jiang, Mukherjee, and Sarkar (2016), and Wang et al. (2018), see Section 2 for details). However, it is unclear whether they could provide statistical performance guarantees, such as the selection consistency and estimation normality. Moreover, they all suffer $O(K^2)$ computational and communication costs.[2] In light of the limitations of these existing works, in this article, we ask the following fundamental question: *Could we develop a new distributed approach to achieve both strong statistical performance guarantees and* $\Theta(K)$ *computation and communication costs?* In other words, could we achieve the best of both worlds of the existing methods in the literature?

In this article, we show that the answer to the above question is *affirmative*. The main contribution of this article is that, for the first time, we develop a novel distributed spanning-tree-based fused lasso regression (DTFLR) approach for solving the clustered coefficient regression problem in the distributed network systems. Our approach enjoys oracle statistical performance and enables low-complexity distributed optimization algorithm design with *linear* convergence rate. The main results of this article are summarized as follows:

- *Low-Complexity Estimator Design:* We propose a new adaptive spanning-tree-based penalty function for the clustered coefficient regression problem with $\Theta(K)$ complexity. Specifically, by comparing the coefficient similarities between the nodes, we construct an adaptive minimum spanning tree from the original network graph and only the edges in the tree are considered in the penalty function. Under this approach, the number of terms in the penalty function is reduced to $K-1$ (hence, $\Theta(K)$ as opposed to $O(K^2)$).
- *Statistical Performance Guarantee:* Based on the spanning tree structure, we propose an adaptive lasso approach to penalize the linear model coefficient differences. We show that our proposed estimator enjoys elegant oracle properties (see, Fan and Li 2001), which means that our method can identify the nodes' cluster memberships almost surely and the estimators achieve asymptotic normality.
- *Distributed Optimization Algorithm Design:* Due to the restrictions imposed by the spanning-tree-based estimator design, traditional gradient- or alternating direction method of multipliers (ADMM)-type methods cannot be applied to distributively solve the optimization problem and find the nodes' cluster memberships. In this article, we develop a novel distributed generalized ADMM algorithm (DG-ADMM) for solving the spanning-tree-based fused lasso problem. Moreover, we show that our algorithm has a simple *node-based* structure that is easy to implement and also enjoys a *linear* convergence.

Collectively, our results in this article contribute to the theories of low-complexity model clustering over networks and distributed optimization. We validate our theoretical results with thorough simulation studies and real-world data analysis, which show that our method has advantages in the estimation accuracy, computation efficiency and communication cost. Due to space limitation, we relegate most of the proof details and part of numerical results to supplementary materials.

2. Related Work

In the literature, many approaches have been developed to cluster heterogeneous data, such as the mixture model (Hastie and Tibshirani 1996; Shen and He 2015), the spectral clustering (Rohe, Chatterjee, and Yu 2011), etc. However, most of the literature focuses on clustering the observation y, rather than the relationship between y and the covariate $\mathbf{x}$. The authors of Ma and Huang (2017) and Ma et al. (2020) are the first few to investigate the cluster structure based on model heterogeneity. Specifically, they considered the pairwise fusion penalty to cluster the intercepts and the regression coefficients, respectively. However, with K individuals to be clustered, the pairwise fusion penalty introduces an extra computational complexity at the order $O(K^2)$. In Tang and Song (2016), the authors proposed an ordering-based fused lasso regression method termed FLARCC to identify heterogeneity patterns of coefficients and to merge the homogeneous parameter clusters across multiple datasets with $\Theta(K)$ computational complexity. However, FLARCC does not exploit any spatial network structure to improve the estimation performance. Also, its ordering-based fusion penalty cannot be easily extended to the cases when the network relationship of the data is imposed. In our work, we consider to cluster the nodes and estimate their models in the network. To avoid the extra tremendous complexity, we focus on the spanning-tree-based fusion penalization. The authors of Li and Sang (2018) proposed a spatially clustered coefficient (SCC) regression method, which uses a spatial minimum spanning tree (MST) to capture the spatial relationships among the data. Nevertheless, as the SCC method constructs

[2]The O and Θ notation are the Bachmann–Landau notations Knuth (1976). $a_n = O(b_n)$ denotes that there exist positive constants C and n_0 with $|a_n| \le Cb_n$ for all $n \ge n_0$; $a_n = \Theta(b_n)$ denotes that there exist positive constants C, C' and n_0 with $C' \le a_n \le Cb_n$ for all $n \ge n_0$.

the fusion penalty based on the native spatial MST structure and the traditional lasso, it cannot achieve the clustering consistency and asymptotic normality for the estimation and leads to over-clustering in practical. By contrast, in our work, we develop an adaptive spanning-tree-based fusion penalty, in which the model similarity is used as adaptive weights for both determining the spanning-tree structure and constructing the adaptive fused lasso. Thanks to the adaptive fusion, our approach enhances the estimation efficiency and significantly reduces the algorithm complexities. Furthermore, unlike all the above methods that are implemented on a single centralized machine, a key distinguishing feature of our work is that we conduct coefficient clustering in a *distributed* fashion without data merging.

Our work also contributes to the theory of distributed optimization over networks, which have attracted a significant amount of recent research (see, e.g., Nedic and Ozdaglar 2009; Shi et al. 2014; Yuan, Ling, and Yin 2016; Eisen, Mokhtari, and Ribeiro 2017). In the general framework of distributed optimization, all K nodes in a connected network distributively and collaboratively solve an optimization problem in the form of: $\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) \triangleq \sum_{i=1}^{K} f_i(\boldsymbol{\beta})$, where each f_i is the objective function observable only to the ith node and $\boldsymbol{\beta}$ is a globally common model coefficient across all nodes, thus, also known as distributed consensus learning. Interestingly, opposite to traditional distributed consensus algorithms that focus on reaching the same model coefficient, our work considers whether there exists underlying *disagreements* among the nodes' local coefficients: the nodes are to be classified to several clusters and only the nodes in each cluster share the same coefficient. We note that the authors of Jiang, Mukherjee, and Sarkar (2016) and Wang et al. (2018) also focused on discovering the clustering patterns among the nodes in distributed network. However, they adopted a graph-based ridge penalty to obtain consensus of the inner-cluster coefficients, which can be reformulated as the well-known Laplacian penalty (Ando and Zhang 2007). However, a main limitation of the Laplacian penalty is that it cannot shrink the differences of the coefficient estimates to zero and thus fails to recover the cluster structure.

The most related work to ours is Hallac, Leskovec, and Boyd (2015), which also considered the network lasso method. In Hallac, Leskovec, and Boyd (2015), the authors adopted an ℓ_2 group lasso penalty for each edge in the network graph and also proposed a distributed ADMM algorithm to solve the network lasso problem. Our work differs from Hallac, Leskovec, and Boyd (2015) in the following key aspects: (a) The number of the penalty terms in Hallac, Leskovec, and Boyd (2015) depends on the number of edges in the network graph, which yields an $O(K^2)$ computation complexity and is *unscalable* for the large-sized networks. In this article, we consider *an adaptive spanning-tree-based* penalty, which contains exactly $K-1$ penalty terms; (b) The penalty in Hallac, Leskovec, and Boyd (2015) adopted the traditional lasso penalty for $||\boldsymbol{\beta}_i - \boldsymbol{\beta}_j||$, while we consider an *adaptive* lasso penalty for the coefficient differences, which enjoys elegant *oracle* properties (i.e., the selection consistency and the asymptotic normality); (c) The algorithm in Hallac, Leskovec, and Boyd (2015) is based on the classic ADMM algorithm with two constraints on each edge of the original network, while we propose a new distributed generalized ADMM method with only *one* constraint on each edge of the adaptive spanning-tree, which significantly reduces the algorithm's implementation complexity; (d) We rigorously prove the model consistency and algorithm convergence of our proposed approach, both of which were not studied in Hallac, Leskovec, and Boyd (2015).

3. Model and Problem Statement

Given a network system with graph structure $G = (V, E)$, where V and E represent the node and edge sets, respectively, our goal is to estimate the coefficients $\{\boldsymbol{\beta}_i\}_{i=1}^{K}$ and determine the cluster membership for each node. This problem can be formulated as minimizing the following objective function:

$$L_{\text{Graph}}(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{K} ||\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}_i||^2 + \sum_{(v_i,v_j)\in E} P_\lambda(\boldsymbol{\beta}_i - \boldsymbol{\beta}_j), \quad (2)$$

where $v_i \in V$ denotes the ith node in the network; $\mathbf{y}_i = [y_{i,1}, \ldots, y_{i,n}]^\top \in \mathbb{R}^n$ and $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n}]^\top \in \mathbb{R}^{n\times d}$ represent the response and design matrix at the ith node, respectively; and P_λ is a penalty function with tuning parameter λ. Note that the objective function in Equation (2) consists of two parts: the first part is an ordinary least square (OLS) problem for all the coefficients $\boldsymbol{\beta} \triangleq [\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_K^\top]^\top \in \mathbb{R}^{Kd}$; the second term is a penalty designed to shrink the difference of any two coefficient vectors if the corresponding nodes are connected in the network. Note that the second term in Equation (2) depends on the network topology. Thus, we make the following assumption that is necessary to guarantee that the problem is well-defined in terms of estimation accuracy:

Assumption 1. For any two nodes v_i and v_j in the given connected network $G = (V, E)$, if they are from the same cluster, then there exists a path connecting them such that all nodes on the path belong to the same cluster.

Under Assumption 1, each node is connected with its cluster members if the cluster size is larger than one: for any node v_i from a cluster with more than two nodes, there exists another node v_j from the same cluster and a path in the graph connecting them. Without loss of generality, we can suppose the edge (v_i, v_j) belongs to E; Otherwise, we can find another node v_k in the path that $(v_i, v_k) \in E$ and v_k is from the same cluster as v_i. This assumption guarantees that the nodes within the same cluster will not be separated by other clusters. Hence, by removing inter-cluster edges, that is, identifying edges with *nonzero* coefficient difference, the original network graph can be reduced into S subgraphs, which are the subgroup clusters. In our work, the network topology G is defined by the distributed system, and there is a tradeoff between system cost and the risk of violating Assumption 1: For example, in the application of wireless sensor networks, the sensors' communication power determines the network topology. Larger communication range can improve the connectivity of the network and ensure Assumption 1. However, it requires higher signal strength of the sensors and thus costs more electrical energy for communication. For the objective in Equation (2), two important remarks are in order:

Remark 1. First, the penalty in Equation (2) is defined by all edges in the network graph. If the penalty function is chosen as $P_\lambda(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j) = \lambda||\boldsymbol{\beta}_i - \boldsymbol{\beta}_j||_2$, then Equation (2) has the same form as in the network lasso method (Hallac, Leskovec, and Boyd 2015). Second, the objective in Equation (2) can also be viewed as a variant of the method proposed in Ma and Huang (2017), where the penalization terms are based on all pairwise coefficient differences among the nodes, and hence the total number of the penalization terms is exactly $(K-1)K/2$. Although in Equation (2) the number of penalization terms is reduced to exactly $|E|$, the value of $|E|$ still implies that the number of penalization terms could scale as $O(K^2)$ if the network graph is dense, which will in turn result in heavy computational load as the network size gets large. To address the problem, we will propose a simplified spanning-tree-based penalty function in Section 4.

4. Problem Reformulation: An Adaptive Spanning Tree Approach

As mentioned earlier and has been long noted in statistics (see, e.g., Tang and Song 2016; Li and Sang 2018) and optimization (see, e.g., Chow et al. 2016) communities, directly including all edges in penalization terms will incur high model complexity. To reduce the redundant penalization terms, several strategies have been proposed, including the ordering-based method in Ke, Fan, and Wu (2015) and Tang and Song (2016) and the MST-based approach in Li and Sang (2018). Specifically, in Ke, Fan, and Wu (2015) and Tang and Song (2016), the authors first determined the initial estimation of the coefficients and then ordered the coefficients. They then presumed that similar coefficients will be neighbors with high probability and only regularized terms associated with the adjacent coefficients are considered. By contrast, the authors of Li and Sang (2018) used the *spatial* distance to construct an MST, and the penalization terms in the tree are preserved. In essence, these two strategies are spanning-tree-based, with the only difference being the definitions of distance measure for the tree: the first one uses model similarity, while the second one uses spatial distances. In this article, we propose a new spanning-tree-based approach, where the distance measure for the tree can be viewed as intelligently integrating the above two measures. Surprisingly, we will show that this new distance measure achieves significant performance gains.

Specifically, we construct a spanning-tree as follows: First, *local* OLS estimators are determined in each node individually: $\widehat{\boldsymbol{\beta}}_{i,\mathrm{OLS}} = [\mathbf{X}_i^\top \mathbf{X}_i]^{-1}[\mathbf{X}_i^\top \mathbf{y}_i]$. Then, the weight for two nodes is defined based on *both* their local model similarity and their connection relationship in the graph as follows:

$$\widetilde{s}_{i,j} = \begin{cases} ||\widehat{\boldsymbol{\beta}}_{i,\mathrm{OLS}} - \widehat{\boldsymbol{\beta}}_{j,\mathrm{OLS}}||, & \text{if } (v_i, v_j) \in E, \\ \infty, & \text{otherwise.} \end{cases} \tag{3}$$

The weight $\widetilde{s}_{i,j}$ in (3) contains two important pieces of information. The first one is the network topology, which is characterized by spatial distances (e.g., in a sensor network, the nodes can only be connected within a certain communication range); the second one is the local model similarity that implies the likelihood of two nodes being in the same cluster. Based on (3), an MST can be created, so that only terms associated with the tree are considered in the objective: $L_{\mathrm{MST}_s}(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^K ||\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}_i||^2 + \sum_{(v_i,v_j)\in \mathrm{MST}_s} P_\lambda(\boldsymbol{\beta}_i - \boldsymbol{\beta}_j)$, where the notation MST_s signifies that the MST is based on the model similarity. Note that the estimation efficiency and clustering accuracy significantly depend on the penalty. Toward this end, we first prove the following key lemma that guarantees that the nodes in the same cluster are connected in the MST_s based on our weight defined in (3):

Lemma 1 (Inclusion of MST_s). Given the MST_s based on the weights in Equation (3) and consider any two nodes v_i and v_j in the same cluster. Under Assumption 1, as the local sample size $n \to \infty$, with probability 1, there exists a path in the MST_s connecting v_i and v_j such that all the nodes on the path belongs to the same cluster.

Several important remarks are in order: (a) With Lemma 1, the number of inter-cluster edges is $S-1$. Thus, the MST_s is a connected graph with the *smallest* possible number of inter-cluster edges. (b) We note that there exist distributed methods to find MST_s (e.g., the GHS algorithm Gallager, Humblet, and Spira 1983) and their implementation details are beyond the scope of this article.

5. Statistical Model: An Adaptive Fused-Lasso-Based Approach

For convenience, we use $[\mathbf{v}]_p$ to denote the pth element of vector $\mathbf{v}$. Based on the MST_s, we specialize the objective by adopting the following adaptive lasso penalty:

$$L_{\mathrm{MST}_s}(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^K ||\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}_i||^2 + \frac{\lambda}{2}\sum_{i=1}^K \sum_{j\in\mathcal{N}_i}\sum_{p=1}^d [\widehat{\boldsymbol{\pi}}_{i,j}]_p \big|[\boldsymbol{\beta}_i]_p - [\boldsymbol{\beta}_j]_p\big|, \tag{4}$$

where $\mathcal{N}_i$ represents the set of the neighboring nodes of node i in the MST_s, $\widehat{\boldsymbol{\pi}}_{i,j} \in \mathbb{R}^d$ is an adaptive weight vector defined as $[\widehat{\boldsymbol{\pi}}_{i,j}]_p \triangleq 1/\big|[\widehat{\boldsymbol{\beta}}_{i,\mathrm{OLS}}]_p - [\widehat{\boldsymbol{\beta}}_{j,\mathrm{OLS}}]_p\big|^\gamma$ for some constant $\gamma > 0$. Therefore, our proposed estimator is $\widehat{\boldsymbol{\beta}}_{\mathrm{MST}_s} = \arg\min_{\boldsymbol{\beta}} L_{\mathrm{MST}_s}(\boldsymbol{\beta})$.

Remark 2. Here, our use of an adaptive lasso penalty is motivated by: (a) Adaptive lasso is known to be an oracle procedure for related variable selection problems Zou (2006); (b) With an adaptive lasso penalty, the objective in Equation (4) is strongly convex as long as the design matrix $\mathbf{X}$ is of full row rank, so that the optimal point of Equation (4) is unique. In Ma and Huang (2017) and Ma et al. (2020), similar methods were proposed based on the minimax concave penalty (MCP) and the smoothly clipped absolute deviations (SCAD) penalty, both of which have been shown to be statistically efficient. However, from optimization perspective, the two concave penalties will render the objective nonconvex, which leads to intractable algorithm design.

For more compact notation in the subsequent analysis, we rewrite the objective function Equation (4) in the following matrix form:

$$L_{\text{MST}_s}(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda_N \sum_{p=1}^{d(K-1)} [\widehat{\boldsymbol{\pi}}]_p |[(\mathbf{H}\otimes \mathbf{I}_K)\boldsymbol{\beta}]_p|, \tag{5}$$

where $\mathbf{y} \triangleq [\mathbf{y}_1^\top, \ldots, \mathbf{y}_K^\top]^\top$, $\mathbf{X} \triangleq \text{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_K)^\top$ and $\boldsymbol{\beta} \triangleq [\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_K^\top]^\top$ are the response vector, the design matrix, and coefficient vector, respectively; and $\otimes$ denotes the Kronecker product. In Equation (5), $\mathbf{H}$ is the incident matrix of the MST_s, which is row full rank and each entry in $\mathbf{H}$ defined as $[\mathbf{H}]_{l,i} = \begin{cases} 1, & \text{if } i = s(l), \\ -1, & \text{if } i = e(l), \\ 0, & \text{otherwise,} \end{cases}$ where $s(l)$ and $e(l)$ denote the starting and ending node indices of edge l in the MST_s, respectively, with $s(l) < e(l)$. In Equation (5), $[\widehat{\boldsymbol{\pi}}]_p \triangleq 1/[\underline{\mathbf{H}} \cdot \boldsymbol{\beta}_{\text{OLS}}]_p^\gamma$, where $\underline{\mathbf{H}} \triangleq \mathbf{H} \otimes \mathbf{I}_K$ and $\boldsymbol{\beta}_{\text{OLS}} \triangleq [\boldsymbol{\beta}_{1,\text{OLS}}^\top, \ldots, \boldsymbol{\beta}_{K,\text{OLS}}^\top]^\top$ is the vector form of the OLS estimations. Note that adding one more row to $\mathbf{H}$, we can form a square and full rank matrix: $\widetilde{\mathbf{H}} \triangleq \begin{bmatrix} \mathbf{H} \\ \frac{1}{\sqrt{K}}\mathbf{1}^\top \end{bmatrix}$ Li and Sang (2018), and the objective in Equation (5) can be equivalently rewritten as $L_{\text{MST}_s}(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda_N \sum_{p=1}^{dK} [\widehat{\boldsymbol{\pi}}]_p |[\underline{\widetilde{\mathbf{H}}}\boldsymbol{\beta}]_p|$, where $\underline{\widetilde{\mathbf{H}}} \triangleq \widetilde{\mathbf{H}} \otimes \mathbf{I}_K$ is a full rank square matrix. Define $\boldsymbol{\Delta} \triangleq \underline{\widetilde{\mathbf{H}}}\boldsymbol{\beta}$ as the difference of the connected nodes' coefficients. It then follows that the above objective $L_{\text{MST}_s}(\boldsymbol{\beta})$ can be rewritten in terms of $\boldsymbol{\Delta}$ as $L_{\text{MST}_s}(\boldsymbol{\Delta}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\underline{\widetilde{\mathbf{H}}}^{-1}\boldsymbol{\Delta}||^2 + \lambda_N \sum_{p=1}^{dK} [\widehat{\boldsymbol{\pi}}]_p |[\boldsymbol{\Delta}]_p|$. Our estimator then becomes: $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s} = \arg\min_{\boldsymbol{\Delta}} L_{\text{MST}_s}(\boldsymbol{\Delta})$. Since there is a one-to-one transformation between $\widehat{\boldsymbol{\beta}}_{\text{MST}_s}$ and $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$ (i.e., $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s} = \underline{\widetilde{\mathbf{H}}}\widehat{\boldsymbol{\beta}}_{\text{MST}_s}$), we can instead focus on the theoretical properties of $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$. Denote the true coefficients as $\boldsymbol{\beta}_* = [\boldsymbol{\beta}_{1,*}^\top, \ldots, \boldsymbol{\beta}_{K,*}^\top]^\top$, and $\boldsymbol{\Delta}_* = \underline{\widetilde{\mathbf{H}}}\boldsymbol{\beta}_*$. Note that if the two connected nodes are from the same cluster, the corresponding elements in $\boldsymbol{\Delta}_*$ are zero. We denote the set of nonzero elements in $\boldsymbol{\Delta}_*$ as $\mathcal{A}_*$. Similarly, the set of nonzero elements in $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$ is denoted as $\widehat{\mathcal{A}}$. To prove the oracle properties of $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$, we need the following assumptions for the linear model in Equation (1):

Assumption 2. For the linear model in Equation (1): (a) the errors are iid with zero mean and variance σ^2; (b) $\frac{1}{N}(\mathbf{X}\underline{\widetilde{\mathbf{H}}}^{-1})^\top \mathbf{X}\underline{\widetilde{\mathbf{H}}}^{-1} \xrightarrow{p} \boldsymbol{C}$ for some positive definite matrix $\boldsymbol{C}$ as $N \to \infty$.

In Condition (ii) of Assumption 2, since $\underline{\widetilde{\mathbf{H}}}$ is full rank, $\boldsymbol{C}$ is positive definite if $\mathbf{X}$ is full column rank. Now, we state the oracle properties of $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$ as follows:

Theorem 1 (Oracle Properties). Suppose that λ satisfies $\lambda/\sqrt{N} \to 0$ and $\lambda N^{(\gamma-1)/2} \to \infty$. Under Assumptions 1 and 2, our estimator satisfies the following two properties: (a) (Clustering Consistency) $\lim_{N\to\infty} \mathbb{P}(\widehat{\mathcal{A}} = \mathcal{A}_*) = 1$ and thus $\mathbb{P}([\widehat{\boldsymbol{\beta}}_{\text{MST}_s}]_i = [\widehat{\boldsymbol{\beta}}_{\text{MST}_s}]_j) = 1$ if $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j$; (b) (Asymptotic Normality) $\sqrt{N}([\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}]_{\mathcal{A}_*} - [\boldsymbol{\Delta}_*]_{\mathcal{A}_*}) \xrightarrow{d} \mathcal{N}(0, \sigma^2 \boldsymbol{C}_{\mathcal{A}_*}^{-1})$ as $N \to \infty$, where $\boldsymbol{C}_{\mathcal{A}_*}$ is the submatrix of $\boldsymbol{C}$ corresponding to $\mathcal{A}_*$.

The clustering consistency (also known as selection consistency) is a direct consequence of Lemma 1, which says that our method correctly clusters the nodes, that is, $\mathbb{P}([\widehat{\boldsymbol{\beta}}_{\text{MST}_s}]_i = [\widehat{\boldsymbol{\beta}}_{\text{MST}_s}]_j) = 1$ if nodes i and j are from the same cluster. The asymptotic normality of $\widehat{\boldsymbol{\beta}}_{\text{MST}_s}$ follows immediately from the linear transformation, that is, $\widehat{\boldsymbol{\beta}}_{\text{MST}_s} = \underline{\widetilde{\mathbf{H}}}^{-1}\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$. We relegate the proof details of Theorem 1 to the supplementary materials. Practically, the cluster membership of each node can be identified by checking $\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}$: if $[\widehat{\boldsymbol{\Delta}}_{\text{MST}_s}]_l = \mathbf{0}$, then the two nodes connected by edge l are from the same cluster.

6. A Distributed Generalized ADMM Algorithm

In this section, we will design a distributed algorithm for minimizing Equation (4). Due to the penalty structure in Equation (4), a natural idea is to use the popular ADMM method (Boyd et al. 2011), which has been shown to be particularly suited for solving lasso related problems (e.g., Ma and Huang 2017; Wahlberg et al. 2012; Zhu 2017; Ma et al. 2020). However, in what follows, we will first illustrate why it is challenging to use a regular ADMM approach to solve the MST_s-based fused-lasso problem in a distributed fashion.

(1) Challenges in Distributed Optimization Algorithm Design: To see why it is *nontrivial* to design a distributed ADMM-based algorithm, we first note that the penalty in (4) can be written as $\frac{1}{2}\sum_{i=1}^K \sum_{j\in\mathcal{N}_i} \sum_{p=1}^d [\widehat{\boldsymbol{\pi}}_{i,j}]_p |[\boldsymbol{\beta}_i]_p - [\boldsymbol{\beta}_j]_p| = \sum_{e_l\in\text{MST}_s} \sum_{p=1}^d [\widehat{\boldsymbol{\pi}}_l]_p |[\boldsymbol{\beta}_{s(l)}]_p - [\boldsymbol{\beta}_{e(l)}]_p|$, where e_l represents the lth edge in MST_s. In the above, $s(l)$ and $e(l)$ denote the starting and ending node indices of edge l, respectively, with $s(l) < e(l)$; and $\widehat{\boldsymbol{\pi}}_l = \widehat{\boldsymbol{\pi}}_{s(l),e(l)}$ is the corresponding adaptive weight vector for the lth edge. With the same notation as in Section 5, the weight difference at edge l is $\boldsymbol{\Delta}_l = \boldsymbol{\beta}_{s(l)} - \boldsymbol{\beta}_{e(l)}$ and $\boldsymbol{\Delta} = [\Delta_1^\top, \ldots, \Delta_{K-1}^\top]^\top = \underline{\mathbf{H}}\boldsymbol{\beta}$. Note that there are $K-1$ edges in MST_s. Thus, the problem of minimizing the objective in (4) can be reformulated as

$$\min \frac{1}{2}\sum_{i=1}^K ||\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}_i||^2 + \lambda_N \sum_{l=1}^{K-1}\sum_{p=1}^d [\widehat{\boldsymbol{\pi}}_l]_p |[\boldsymbol{\Delta}_l]_p|,$$
$$\text{s.t.} \boldsymbol{\Delta} = \underline{\mathbf{H}}\boldsymbol{\beta}. \tag{6}$$

Following classic ADMM, we can construct an augmented Lagrangian with penalty parameter $\tau > 0$ as $L_\tau(\boldsymbol{\beta}, \boldsymbol{\Delta}, \mathbf{z}) = \frac{1}{2}\sum_{i=1}^K ||\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta}_i||^2 + \lambda_N \sum_{l=1}^{K-1}\sum_{p=1}^d [\widehat{\boldsymbol{\pi}}_l]_p |[\boldsymbol{\Delta}_l]_p| - \langle \mathbf{z}, \underline{\mathbf{H}}\boldsymbol{\beta} - \boldsymbol{\Delta}\rangle + \frac{\tau}{2}||\underline{\mathbf{H}}\boldsymbol{\beta} - \boldsymbol{\Delta}||^2$, where $\mathbf{z} \in \mathbb{R}^{d(K-1)}$ is the vector of dual variables corresponding to the $K-1$ edges. In what follows, we derive the updating rules for $(\boldsymbol{\beta}^{t+1}, \boldsymbol{\Delta}^{t+1}, \mathbf{z}^{t+1})$. First, given the primal and dual pair $\boldsymbol{\beta}^t, \mathbf{z}^t$, for the lth edge with end nodes $s(l)$ and $e(l)$, to determine the weight difference $\boldsymbol{\Delta}^{t+1}$, we solve the subproblem $\boldsymbol{\Delta}^{t+1} = \arg\min_{\boldsymbol{\Delta}} L_\tau(\boldsymbol{\beta}^t, \boldsymbol{\Delta}, \mathbf{z}^t)$, and hence it can be shown that:

$$\boldsymbol{\Delta}_l^{t+1} = S_{\lambda_N\widehat{\boldsymbol{\pi}}_l/\tau}\Big(\boldsymbol{\beta}_{s(l)}^t - \boldsymbol{\beta}_{e(l)}^t - \frac{1}{\tau}\mathbf{z}_l^t\Big), \tag{7}$$

where $S_{\lambda_N\widehat{\boldsymbol{\pi}}_l/\tau}$ is the coordinate-wise soft-thresholding operator with $[\lambda_N\widehat{\boldsymbol{\pi}}_l/\tau]_p = \lambda_N[\widehat{\boldsymbol{\pi}}_l]_p/\tau$. Next, we derive the updating rule for $\boldsymbol{\beta}^{t+1}$. Similar to the classic ADMM method, it can be shown that :

$$\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta}} L_\tau(\boldsymbol{\beta}, \boldsymbol{\Delta}^{t+1}, \mathbf{z}^t) \tag{8}$$

$$= [\mathbf{X}^\top\mathbf{X} + \tau\mathbf{L}\otimes\mathbf{I}_d]^{-1}[\mathbf{X}^\top\mathbf{y} + \underline{\mathbf{H}}^\top(\tau\boldsymbol{\Delta}^{t+1} + \mathbf{z}^t)],$$

where $\mathbf{L} = \mathbf{H}^\top\mathbf{H}$ is the Laplacian matrix of the MST$_s$. Unfortunately, the matrix inverse in (8) *cannot be computed in a distributed fashion* due to the coupled structure of the Laplacian matrix $\mathbf{L}$.

(2) Our Solution Approach: To address the above challenges, our key idea is to leverage the (centralized) generalized ADMM in Deng and Yin (2016) to derive a new updating rule for $\boldsymbol{\beta}^{t+1}$, so that it can be implemented in a parallel fashion. To this end, instead of directly solving Equation (8), we add a quadratic term $\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^t)^\top\mathbf{P}(\boldsymbol{\beta}-\boldsymbol{\beta}^t)$ in the subproblem ($\mathbf{P}$ is a positive semidefinite matrix to be specified later):

$$\begin{aligned}\boldsymbol{\beta}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{\beta}} L_\tau(\boldsymbol{\beta}, \boldsymbol{\Delta}^{t+1}, \mathbf{z}^t) + \frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^t)^\top\mathbf{P}(\boldsymbol{\beta}-\boldsymbol{\beta}^t) \quad (9)\\ &= [\mathbf{X}^\top\mathbf{X}+\tau\mathbf{L}\otimes\mathbf{I}_d+\mathbf{P}]^{-1}[\mathbf{X}^\top\mathbf{y}+\underline{\mathbf{H}}^\top(\tau\boldsymbol{\Delta}^{t+1}+\mathbf{z}^t)+\mathbf{P}\boldsymbol{\beta}^t].\end{aligned}$$

Now, the *key step* is to recognize that we can choose $\mathbf{P}$ as $\mathbf{P} = -\tau\mathbf{L}\otimes\mathbf{I}_d + \mathbf{D}$, where $\mathbf{D} = \text{diag}(D_1,\ldots,D_K)\otimes\mathbf{I}_d$ with positive scalars D_i for node i (the choice of D_i will be specified soon). It then follows that $\boldsymbol{\beta}^{t+1} = [\mathbf{X}^\top\mathbf{X}+\mathbf{D}]^{-1}[\mathbf{X}^\top\mathbf{y} + \underline{\mathbf{H}}^\top(\tau\boldsymbol{\Delta}^{t+1} + \mathbf{z}^t) + \mathbf{P}\boldsymbol{\beta}^t]$. By comparing our proposed $\boldsymbol{\beta}$'s updates (9) with the traditional one (8), it can be seen that our method uses a diagonal matrix $\mathbf{D}$ to approximate the graph Laplacian matrix $\mathbf{L}$ for the design matrix mixing (i.e., calculating the matrix inverse $(\mathbf{X}^\top\mathbf{X}+\mathbf{D})^{-1}$ rather than $(\mathbf{X}^\top\mathbf{X}+\mathbf{L})^{-1}$). Then, to correct the approximation error, our algorithm involves a neighboring coefficient mixing (i.e., the term $\mathbf{P}\boldsymbol{\beta}^t = (\mathbf{D}-\tau\mathbf{L}\otimes\mathbf{I}_d)\boldsymbol{\beta}^t$). Based on Equation (9), we have the following local update at each node i:

$$\begin{aligned}\boldsymbol{\beta}_i^{t+1} = [\mathbf{X}_i^\top\mathbf{X}_i + D_i\mathbf{I}_d]^{-1}\Big[\mathbf{X}_i^\top\mathbf{y}_i + \sum_{v_i\in e_l}[\mathbf{H}]_{li}(\tau\boldsymbol{\Delta}_l^{t+1}+\mathbf{z}_l^t)\\ + (D_i - \tau\deg(i))\boldsymbol{\beta}_i^t + \tau\sum_{j\in\mathcal{N}_i}\boldsymbol{\beta}_j^t\Big], \quad (10)\end{aligned}$$

where $v_i\in e_l$ means that node v_i is an end node of edge e_l, and $\deg(i) \triangleq |\mathcal{N}_i|$ is the degree of the node v_i. Thus, the update of $\boldsymbol{\beta}_i^{t+1}$ only requires the local connected neighbors' information, which *implies distributed implementation*.

Note that it remains to specify how to choose the values of D_i, $i = 1,\ldots,K$, for $\mathbf{P}$. According to Proposition 2 (to be proved later), in order for our algorithm to achieve a desirable linear convergence rate, the matrix $\mathbf{P}$ needs to be positive definite. Note that $\mathbf{P} = \mathbf{D}-\tau\mathbf{L}\otimes\mathbf{I}_d = [\text{diag}(D_1,\ldots,D_K)-\tau\mathbf{L}]\otimes\mathbf{I}_d$. To guarantee $\mathbf{P}\succ 0$, based on the Gershgorin circle theorem, it can be readily verified that choosing $D_i > 2\deg(i)$, $\forall i = 1,\ldots,K$, suffices.

Lastly, the dual variables $\mathbf{z}^{t+1}$ can be updated as $\mathbf{z}^{t+1} = \mathbf{z}^t - \tau(\underline{\mathbf{H}}\boldsymbol{\beta}^{t+1} - \boldsymbol{\Delta}^{t+1})$, and hence for the lth edge, the corresponding dual update is:

$$\mathbf{z}_l^{t+1} = \mathbf{z}_l^t - \tau\Big(\boldsymbol{\beta}_{s(l)}^{t+1} - \boldsymbol{\beta}_{e(l)}^{t+1} - \boldsymbol{\Delta}_l^{t+1}\Big). \quad (11)$$

Note, however, that the update rules (7) and (11) are *edge-based* while (10) is *node-based*. To make the update rules consistent, we define several additional notations: At node $s(l)$, we let $\boldsymbol{\Delta}_{s(l)}^t = \boldsymbol{\Delta}_l^t$ and $\mathbf{z}_{s(l)}^t = \mathbf{z}_l^t$; At node $e(l)$, we let $\boldsymbol{\Delta}_{e(l)}^t = -\boldsymbol{\Delta}_l^t$ and $\mathbf{z}_{e(l)}^t = -\mathbf{z}_l^t$. With some derivations, it can be verified that if $\boldsymbol{\Delta}_{s(l)}^t = -\boldsymbol{\Delta}_{e(l)}^t = \boldsymbol{\Delta}_l^t$ and $\mathbf{z}_{s(l)}^t = -\mathbf{z}_{e(l)}^t = \mathbf{z}_l^t$ are satisfied in iteration t, then in iteration $t+1$, $\boldsymbol{\Delta}_{s(l)}^{t+1} = -\boldsymbol{\Delta}_{e(l)}^{t+1} = \boldsymbol{\Delta}_l^{t+1}$ and $\mathbf{z}_{s(l)}^{t+1} = -\mathbf{z}_{e(l)}^{t+1} = \mathbf{z}_l^{t+1}$ still hold based on the following node-based updating rules: $\forall i\in\{s(l),e(l)\}$ and $j = \{s(l),e(l)\}/\{i\}$,

$$\begin{cases}\boldsymbol{\Delta}_i^{t+1} = S_{\lambda_N\widehat{\boldsymbol{\pi}}_l/\tau}\Big(\boldsymbol{\beta}_i^t - \boldsymbol{\beta}_j^t - \frac{1}{\tau}\mathbf{z}_i^t\Big),\\ \boldsymbol{\beta}_i^{t+1} = [\mathbf{X}_i^\top\mathbf{X}_i + D_i\mathbf{I}_d]^{-1}\Big[\mathbf{X}_i^\top\mathbf{y}_i + \sum_{v_i\in e_l}(\tau\boldsymbol{\Delta}_i^{t+1}+\mathbf{z}_i^t)\\ \qquad +(D_i-\tau\deg_i)\boldsymbol{\beta}_i^t + \tau\sum_{j\in\mathcal{N}_i}\boldsymbol{\beta}_j^t\Big],\\ \mathbf{z}_i^{t+1} = \mathbf{z}_i^t - \tau\Big(\boldsymbol{\beta}_i^{t+1} - \boldsymbol{\beta}_j^{t+1} - \Delta_i^{t+1}\Big).\end{cases} \quad (12)$$

Thus, we can set $\boldsymbol{\Delta}_{s(l)}^0 = -\boldsymbol{\Delta}_{e(l)}^0 = \boldsymbol{\beta}_s(l)^0 - \boldsymbol{\beta}_e(l)^0$ and $\mathbf{z}_{s(l)} = \mathbf{z}_{e(l)} = \mathbf{0}$, $\forall l$, which satisfy the above conditions. Our method is summarized in Algorithm 1. The outputs of the algorithm are the estimated coefficient $\widehat{\boldsymbol{\beta}}$ and the coefficient difference $\widehat{\boldsymbol{\Delta}}$. Whether two nodes are in the same cluster can be determined by checking $\widehat{\boldsymbol{\Delta}}$: $\widehat{\boldsymbol{\Delta}}_{s(l)} = \widehat{\boldsymbol{\Delta}}_{e(l)} = \mathbf{0}$ if $s(l)$ and $e(l)$ are in the same cluster. The following proposition guarantees the linear convergence rate of our proposed DG-ADMM Algorithm.

Proposition 2 (Linear Convergence). Denote the KKT point for (6) as $\mathbf{u}_* = (\boldsymbol{\beta}_*^\top, \boldsymbol{\Delta}_*^\top, \mathbf{z}_*^\top)^\top$. With a proper $\mathbf{D}$ such that $\mathbf{P}\succ 0$, the iterates $\{\mathbf{u}^t\}_{t=1}^\infty$ converge to $\mathbf{u}_*$ in the sense of $\boldsymbol{G}$-norm: $||\mathbf{u}^t-\mathbf{u}_*||_{\boldsymbol{G}}\to 0$, where $||\cdot||_{\boldsymbol{G}}$ represents the semi-norm $||\mathbf{x}||_{\boldsymbol{G}}^2 \triangleq \mathbf{x}^\top\boldsymbol{G}\mathbf{x}$ and $\boldsymbol{G}$ is defined as $\boldsymbol{G} \triangleq \text{diag}(\mathbf{D},\mathbf{0},\frac{1}{\tau}\mathbf{I}_{d(K-1)})$ Further, the convergence rate is linear, that is, there exists $\delta>0$, such that $||\mathbf{u}^{t+1}-\mathbf{u}_*||_{\boldsymbol{G}}^2 \le (1+\delta)^{-1}||\mathbf{u}^t-\mathbf{u}_*||_{\boldsymbol{G}}^2$.

Due to space limitation, we provide the derivation details and the proof details in the supplementary materials.

(3) Distributed Tuning Parameter Selection: The tuning parameter λ plays an important role in the estimation performance of DTFLR. In this part, we present the method for the tuning parameter selection in the distributed system. Our selection method is based on the Bayesian information criterion (BIC), which has been widely used in many related works (Ke, Fan, and Wu 2015; Tang and Song 2016; Ma and Huang 2017; Li and Sang 2018; Ma et al. 2020). Similar to DG-ADMM, our selection method only requires local statistics exchanges instead of the data flooded across the nodes. Given λ, the conventional BIC for DTFLR is defined as $\text{BIC}(\lambda) = \log[\frac{1}{K}\sum_{k=1}^K\sum_{i=1}^n\frac{1}{n}\big(y_{k,i}-\widehat{\boldsymbol{\beta}}_i(\lambda)^\top\mathbf{x}_{k,i}\big)^2] + \frac{\log(Kn)}{Kn}\cdot\widehat{S}(\lambda_N)d$, where $\widehat{\boldsymbol{\beta}}_i(\lambda)$ is the estimated coefficient with the tuning parameter λ and $\widehat{S}(\lambda)$ is the corresponding estimated cluster number. Under our distributed setting, the nodes can first calculate the local prediction error $\sum_{i=1}^n\frac{1}{n}\big(y_{k,i}-\widehat{\boldsymbol{\beta}}_i(\lambda)^\top\mathbf{x}_{k,i}\big)^2$ and determine the cluster membership by checking $\widehat{\boldsymbol{\Delta}}$. Then, all the nodes submit the local information to the root node in the MST$_s$. By aggregating all the local information, the root node can calculate BIC(λ). To select the best parameter value, the nodes repeat the DG-ADMM algorithm against a grid of candidates $\{\lambda_l\}_{l=1}^L$. After that, the root node collects the local information for BIC calculation and chooses the best value by $\lambda_{\text{BIC}*} = \arg\min_{\lambda_l}\{\text{BIC}(\lambda_l)\}_{l=1}^L$. $\lambda_{\text{BIC}*}$ will be broadcast to all the nodes for determining the final model. Note that the total computation and communication complexities are scaled with the number of candidate values L. In practice, L is a constant number and thus

Algorithm 1 Distributed spanning tree based fused lasso regression (DTFLR) method.

Input: Data $\{\mathbf{X}_i, \mathbf{y}_i\}_{i=1}^K$, a series of tuning parameters $\{\lambda_l\}_{l=1}^L$;

Each node finds the local OLS estimation and sets $\boldsymbol{\beta}_i^0 = \boldsymbol{\beta}_{i,\text{OLS}}$;

Each node sends $\boldsymbol{\beta}_i^0$ to its neighboring nodes and calculates the weight (3);

The network constructs an MST_s based on $\boldsymbol{\beta}_i^0$;

for $\lambda \in \{\lambda_l\}_{l=1}^L$ **do**

  The nodes $s(l)$ and $e(l)$ of edge l set $\mathbf{z}_{s(l)}^0 = \mathbf{z}_{e(l)}^0 = \mathbf{0}$ and $\boldsymbol{\Delta}_{s(l)}^0 = -\boldsymbol{\Delta}_{e(l)}^0 = \boldsymbol{\beta}_{s(l)}^0 - \boldsymbol{\beta}_{e(l)}^0$;

  while not converged **do**

    Each node sends its current $\boldsymbol{\beta}_i^t$ to its neighboring nodes in the MST_s;

    Each node updates the primal and dual variables using the rules in (12);

  end while

  The nodes compute prediction errors and aggregate information for BIC(λ);

end for

Output: The estimated model $\boldsymbol{\beta}_{\text{BIC}*}$ based on $\lambda_{\text{BIC}*} = \arg\min_{\lambda_l}\{\text{BIC}(\lambda_l)\}_{l=1}^L$.

it would not affect the algorithm performance in the sense of convergence order.

7. Numerical Simulation

In this section, we empirically examine the estimation performance of our proposed DTFLR method and estimator $\widehat{\boldsymbol{\beta}}_{\text{MSTs}}$. In Section 7.1, we first study the impact of sample size on our estimator. In Section 7.2, we compare algorithm complexities under different network topology and penalty functions. Another two comparisons are provided in supplementary materials to study the impact of the node number on decentralized estimation and the impact of underlying cluster number, respectively.

7.1. Impact of Sample Size

In this part of simulation, we consider the following network settings (see Figure 1(a)–(b)): The nodes are uniformly located in the space $[-1, 1]^2$ and the numbers of the nodes are 50 and 100, respectively. There are five underlying clusters separated by the solid lines, as shown in Figure 1(a)–(b). The covariate $\mathbf{x}$ are generated from multivariate normal distribution with zero mean and covariance $\text{cov}(\mathbf{x}_i, \mathbf{x}_j) = 0.5^{|i-j|}$. The random error ε follows the standard normal distribution. The true coefficients are randomly generated as $\boldsymbol{\beta}_{G_1,*} = [4.59, 2.60, -5.12]^\top$, $\boldsymbol{\beta}_{G_2,*} = [-2.88, 1.51, 0.59]^\top$, $\boldsymbol{\beta}_{G_3,*} = [3.04, 0.53, -4.74]^\top$, $\boldsymbol{\beta}_{G_4,*} = [-8.09, -3.20, -2.45]^\top$ and $\boldsymbol{\beta}_{G_5,*} = [-0.28, -4.25, -1.28]^\top$. In the network graph, if the distance of two nodes is smaller than 0.5, then there is an communication edge between them, which is shown a dashed line. Note that with 0.5 as the radius, Assumption 1 is satisfied (see in Figure 1(a)–(b)).

We focus on four different methods: (a) the Laplacian graph-based method (Wang et al. 2018), in which the penalty can be regarded as a variant of ridge penalty; (b) the Graph ℓ_1 method with the penalty in (2), which considers all the edges in the graph; (c) the SCC method proposed in Li and Sang (2018); (d) our DTFLR method with the MST_s ℓ_1 penalty. We study the cases with different numbers of node K and local sample n: (a) $K = 50$ and $n = 50$; (b) $K = 50$ and $n = 100$; (c) $K = 100$ and $n = 50$. Note that Cases (1) and (2) have different local sample sizes,

We compare in terms of the following performance metrics: (a) the accuracy of model estimation, $\text{MSE}(\widehat{\boldsymbol{\beta}}) = \frac{1}{K}\sum_{i=1}^K ||\widehat{\boldsymbol{\beta}}_i - \boldsymbol{\beta}_{i,*}||_2^2$; (b) the estimated group number $\widehat{S}$; (c) sensitivity, which measures the proportion of node pairs from the same cluster that are correctly identified; (d) specificity, which measures the proportion of node pairs from the different clusters that are correctly identified. Note that the values of sensitivity and specificity are in the range $[0, 1]$. The closer to 1, the better the prediction is. The simulation results are reported in Table 1 and Figure 2.

From Table 1 and Figure 2, we can see that our DTFLR method outperforms the other methods under all the three circumstances: First of all, we can see that the MSE from the Laplacian gragh-based method is higher than those of the other ℓ_1 based penalty and also the Laplacian graph-based method cannot find the nodes' membership. This is because the Laplacian penalty, which is a variant of ridge penalty, cannot shrink the coefficient difference to zero when two nodes are from the same cluster. Compared to the Graph ℓ_1 method, our method improves the efficiency by reducing about 21%, 36%, 49% in

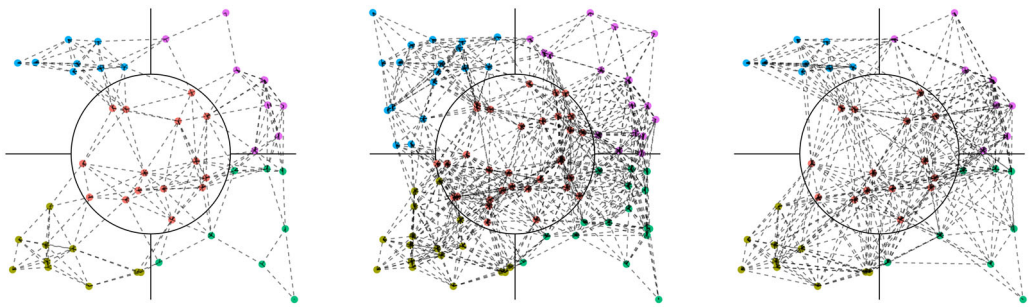


Figure 1. Simulation network settings: (a) random design with 50 nodes and the radius 0.5; (b) random design with 100 nodes and the radius 0.5;(c) random design with 50 nodes and the radius 0.75. The solid lines and nodes' color show the underlying partition and the dashed lines represent the communication edges among the nodes in network graphs.

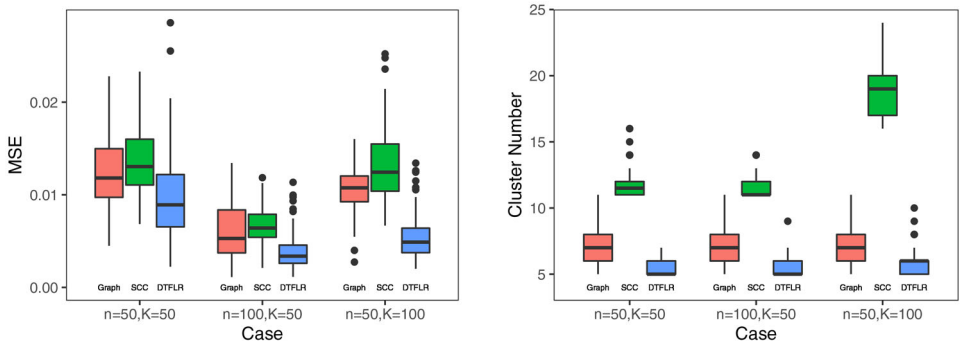


Figure 2. The boxplots of MSEs of $\widehat{\boldsymbol{\beta}}$ and the estimated group numbers $\widehat{S}$ using the three ℓ_1 penalty methods under three cases.

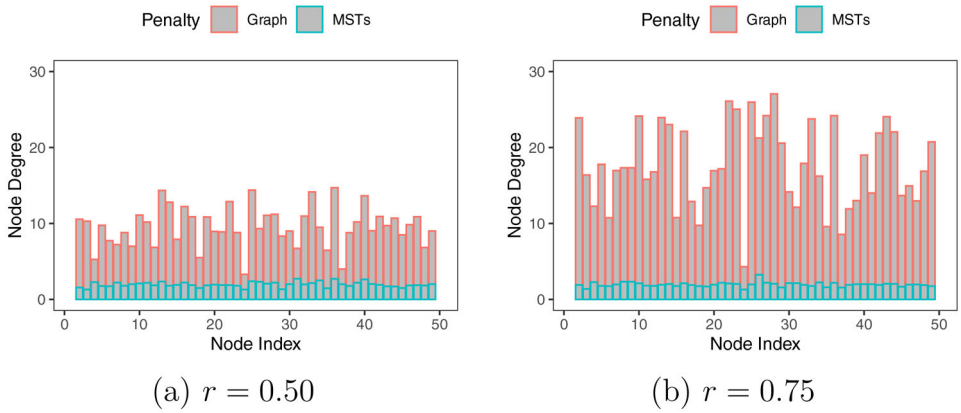


Figure 3. The barcharts of the node degrees for the two settings in Simulation 2.

Table 1. The results of Simulation 2.

Case	Method	MSE($\widehat{\boldsymbol{\beta}}$)	$\widehat{S}$	Sensitivity	Specificity
n=50 K=50	Laplacian	0.0329	NA	NA	NA
	Graph	0.0123	7.28	0.9325	1
	SCC	0.0134	11.87	0.6777	1
	DTFLR	0.0097	5.55	0.9681	1
n=100 K=50	Laplacian	0.0154	NA	NA	NA
	Graph	0.0061	6.83	0.9449	1
	SCC	0.0067	11.58	0.7051	1
	DTFLR	0.0039	5.35	0.9759	1
n=50 K=100	Laplacian	0.0331	NA	NA	NA
	Graph	0.0107	6.94	0.9717	1
	SCC	0.0132	18.62	0.3855	1
	DTFLR	0.0055	5.89	0.9140	1

NOTE: The results are based on 100 repetitions.

MSE for the three cases, respectively, while the estimated cluster numbers are closer to five, which is the true group number. Keeping the sample nodes and doubling the local samples in Case 2, the MSE of our proposed regularization reduces to the half of Case 1, which validates our Theorem 1. Comparing Cases 1 and 2, the estimation efficiencies for all three methods are improved. This is because by adding more nodes, the total sample size is larger. However, for Cases 2 and 3, although they have the same total sample size, the estimation gets better with fewer node and simpler network topology. Additionally, note that the estimated cluster numbers of the SCC method is much worse than the Graph ℓ_1 method and our DTFLR method. This is because the MST constructed by the spatial distant cannot guarantee that the nodes from the same group are connected in the tree. This encourages us to use model similarity as the weights when constructing the MST in our DTFLR method.

7.2. Impact of Network Complexity

In this section, we use simulations to illustrate the impact of the choice of penalty on the accuracy, computation time and communication cost. The computations are performed on a Windows computer with a 2.93 GHz Intel(R) Core(TM) i7 CPU processor and 16.0 GB memory. We compare two method: the Graph ℓ_1 method and our DTFLR method. In the distributed algorithm, the nodes need to update and store the local $\boldsymbol{\beta}_i$, $\boldsymbol{\Delta}_{i,l}$, and $\mathbf{z}_{i,l}$ in each iteration. Note that the numbers of $\{\boldsymbol{\Delta}_{i,l}\}_l$ and $\{\mathbf{z}_{i,l}\}_l$ are the same as those the penalization terms associated with node i. Meanwhile, the nodes are required to send the local $\boldsymbol{\beta}_i$ to their neighbor nodes in the graph or tree. Clearly, the amount of data being transmitted grows as the graph becomes denser. Here, we consider 50 nodes with the same setting as in Simulation 7.1. Each node contains 50 samples. We adjust the network denseness by changing the connection radius threshold value r. Two setting are compared, $r = 0.50$ and $r = 0.75$ (See Figure 1(a) and Figure 1(c)).

As discussed above, the costs for computation and communication depend on the node degrees of the nodes in the graph or tree. Based on the simulation setting, the connected degrees are shown in Figure 3. The node degrees are deterministic for the graph ℓ_1 method. For MST$_s$ ℓ_1 penalty in DTFLR, the node

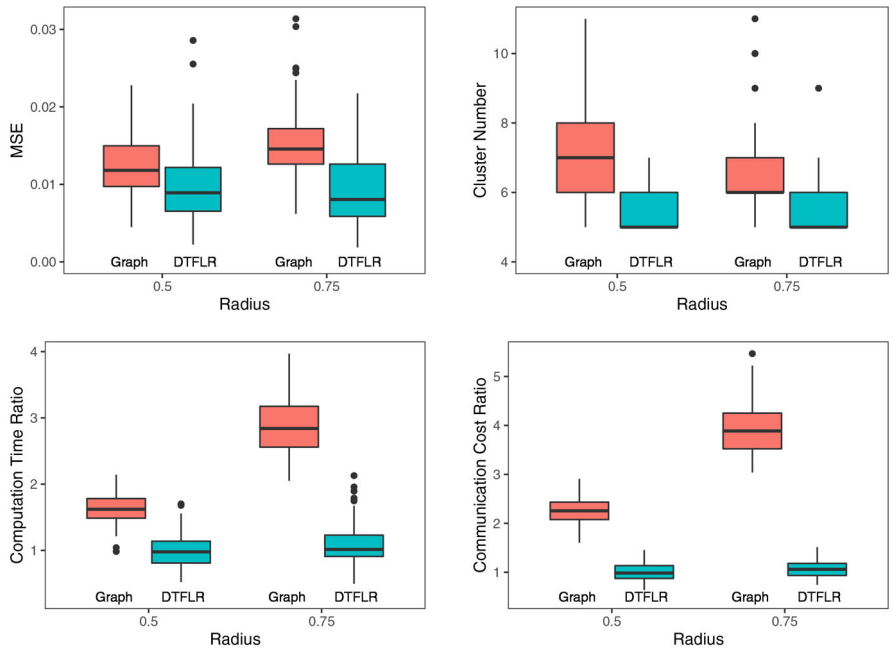


Figure 4. The boxplots of MSEs of $\widehat{\boldsymbol{\beta}}$, the estimated group numbers $\widehat{S}$, the computation time ratio and the communication cost ratio of the graph ℓ_1 method and DTFLR.

degrees are stochastic because the trees are varying with the local samples. Thus, we repeat 100 trials and compute the average degrees for the nodes. In the case with $r = 0.50$, the maximum degrees for the graph ℓ_1 and MST$_s$ ℓ_1 penalties are 12 and 2.72, respectively; while in the case with $r = 0.75$, the corresponding maximum degrees are 25 and 3.25, respectively.

Next, we compare the accuracy and costs for the two methods under different networks. The MSEs and $\widehat{S}$ are used to measure the accuracy. Here the computation time approximation are considered. Note that the node with more edges take longer time to calculate more variables. Thus, the computation time for each iteration is the time for the nodes with the maximum node degree, and the total computation time is the summation of the running times of all iterations. The communication cost is defined as the total amount of transmitted messages, which is proportional to the product of the iterations and the edges. We set the baseline as the average computation time and the average communication cost for DTFLR under $r = 0.50$. The results are shown in Figure 4. We can see that our DTFLR method outperforms in all aspects. By pruning redundant edges, our DTFLR method enjoys both lower computation and communication costs, as well as the higher estimation accuracy. In contrast, for the Graph ℓ_1 method, more edges in the graph result in longer computation time and higher communication cost, as well as less accurate estimation.

8. Real Data Study

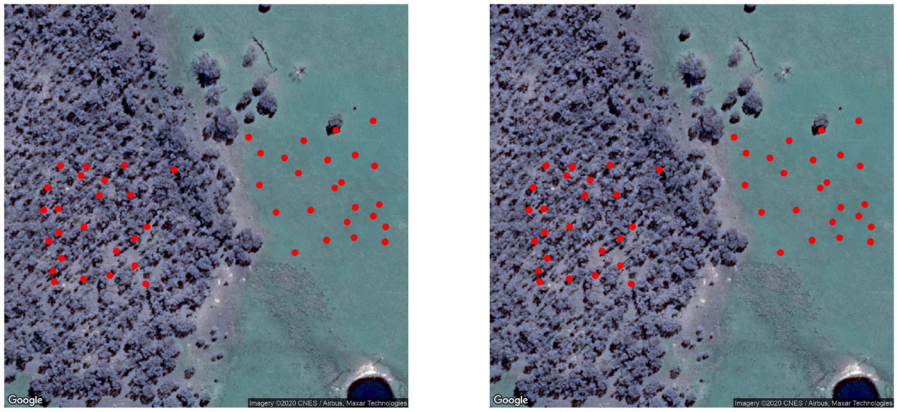
In this section, we'd like to demonstrate the use of our method on two real datasets from the wireless sensor network (Lee, Zhu, and Toscas 2015) and the remote sensing (Li et al. 2018b). We compare the following methods: (a) our proposed DTFLR method; (b) the graph ℓ_1 method shown in (2); (c) k-Means method on the local estimates.

8.1. Sensornets CSIRO Sensor Network Data

This dataset comes from the Sensornets project[3] of the Commonwealth Scientific and Industrial Research Organization (CSIRO). The project developed a family of sensor network nodes known as FLECK ©, which are capable of sensing, computation and wireless communications, to collect the local soil and air temperatures from 00:00 November 9, 2012 until 23:55 January 7, 2013. In our study, we focus on two networks deployed near Yass, New South Wales, Australia, with 48 and 49 valid sensors, respectively (See in Figure 5). In each network, the sensors were divided into two landcover types, that is, grassland and forest. We'd like to study the relationship between the soil temperature y and air temperature x and check the landcover effect on the relationship. Here we focus on the first five days' data.

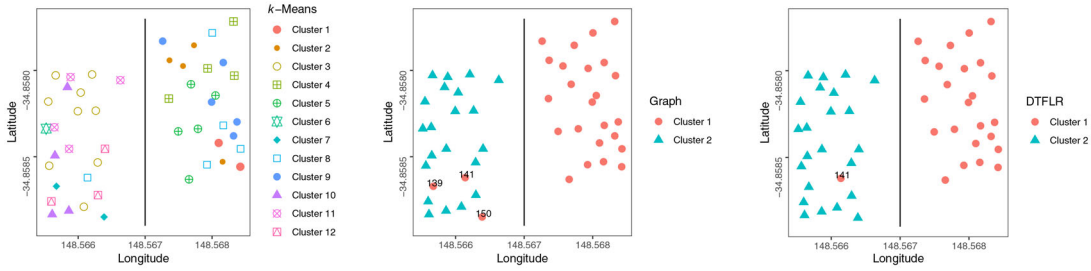
We consider the following linear model: $y_{it} = \beta_0 + \beta_1 x_{it} + \epsilon_{it}$, where y_{it}, x_{it}, and ϵ_{it} are the soil, air temperature and measurement error from the ith sensor at time t, respectively, and β_0 and β_1 are the unknown coefficients. We apply the three methods to the data and the clustering results are shown in Figure 6. It can be seen that the k-Means method has more clusters than the other two methods and the clustering patterns are not interpretable. Both the Graph ℓ_1 method and our DTFLR method can divide the sensors into two groups which almost follow the landcover types. However, in site 1, the two methods have misclassified some sensors: One sensor for our DTFLR Method and three for the Graph ℓ_1 method. The sensor with ID 141 is misclassified under both two methods. From Figure 5(a), we can see that though this node is in the general forest area, it specific location is at the gap among the tree cover. This is the reason why its pattern is similar to those in the grassland area. However, the Graph ℓ_1 method also misclassified the

[3] *http://www.sensornets.csiro.au/*

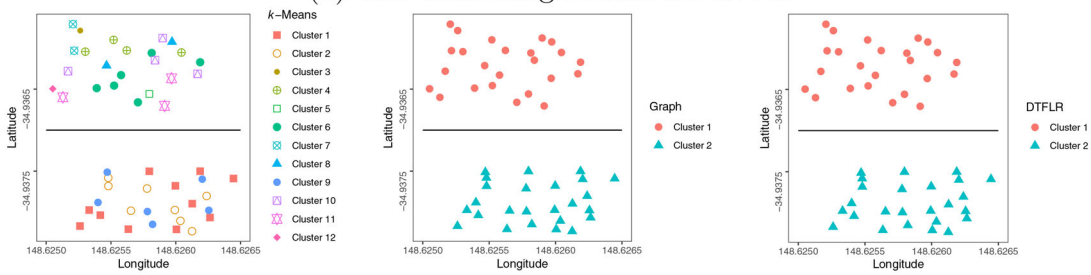


(a) Site 1 (b) Site 2

Figure 5. The wireless sensor networks of the two sites: the left (a) is the site 1 and the right (b) is the site 2. Red points present the sensors. The map data is from Google Map.

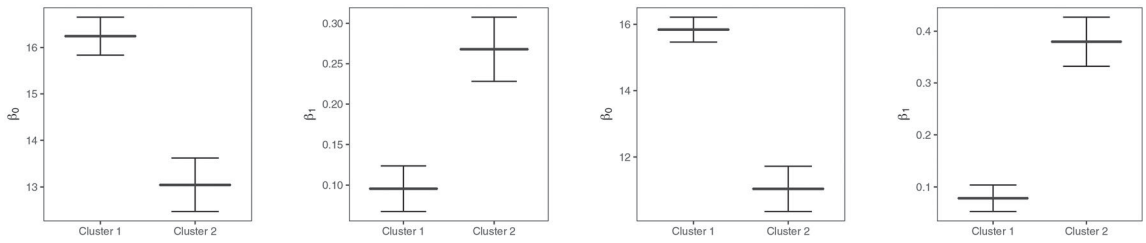


(a) The clustering results for site 1.



(b) The clustering results for site 2.

Figure 6. The clustering results for the sensors: The first column is for *k*-Means, the second column is for the Graph ℓ_1 method and the third column is for our DTFLR.



(a) The estimated coefficients for site 1. (b) The estimated coefficients for site 2.

Figure 7. The clusters' estimated coefficients with 95% confidence intervals.

other two sensors with ID 139 and 150, which are covered by the forest. Thus, we think our DTFLR method has a better clustering performance than the Graph ℓ_1 method and the *k*-Means method. Also we show the clusters' coefficient estimates of DTFLR in Figure 7, which are significantly different across landcover types. Thus, we reach the conclusion that the soil and air temperature relationship will vary with the different landcover type and the estimation efficiency can be improved by aggregating the neighboring information in the wireless sensor network.

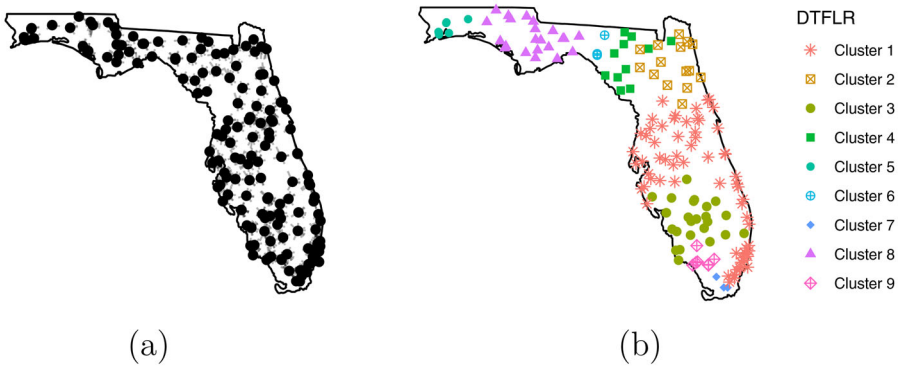


Figure 8. (a) is the network topology of the weather stations in Florida: The points present the station locations and the dashed lines present the connection between two stations. (b) is the clustering results from the proposed DTFLR regularization method.

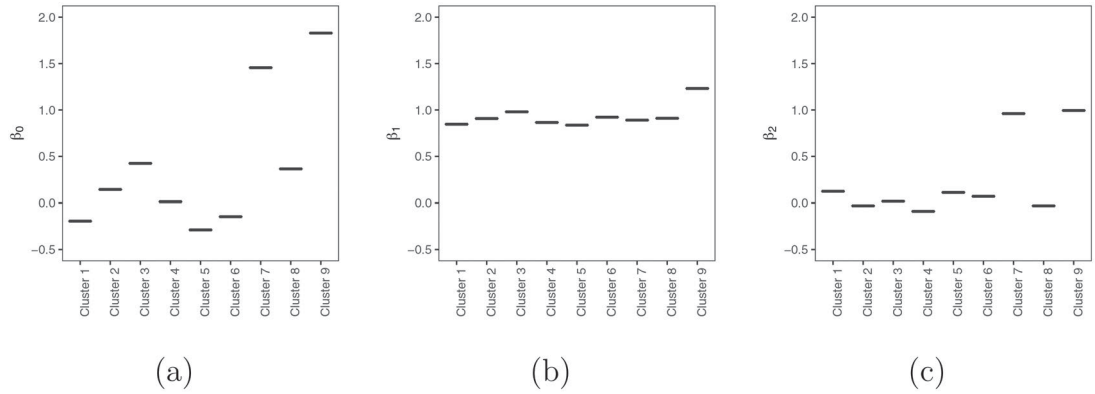


Figure 9. The coefficients' estimation over the clusters based on our DTFLR method.

8.2. High Resolution Air Temperature Remote Sensing Data

We consider the Global Historical Climatology Network-Daily (GHCN-D) dataset (Menne et al. 2012). The maximum air temperature (AT) observations are provided in the GHCN-D dataset from more than 10,000 stations in the conterminous U.S. on approximately 30 days in 2010, which are considered as the response y. Two auxiliary datasets, the gapfilled MODIS daily land surface temperature (LST) data (Li et al. 2018a) and the weather stations' elevation information (Li et al. 2018b), are used as covariate $\mathbf{x}$. In our study, we focus on the 157 stations in Florida with 3937 observations. The stations' network topology is shown in Figure 8(a), in which two stations are connected if their distance is smaller than 36 miles. Note that here 36 miles is the threshold with which the network is connected and the edge number is moderate.

In our study, we'd like to investigate the linear relationship between the daily maximum AT y and the daily LST $\mathbf{x}_1$, the station's elevation $\mathbf{x}_2$, and see whether the stations can be clustered based on the linear relationship. We consider the model, $y_{it} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \mathbf{x}_{1,it} + \boldsymbol{\beta}_2 \mathbf{x}_{2,it} + \epsilon_{it}$, in which the subscript i presents the ith station and t means the tth record. However, the above model is locally unidentifiable, because each location only has the fixed elevation. Thus, the initial estimation is found by the graph Laplacian method, which can be simply implemented in the distributed fashion and the corresponding estimates are also consistent (Wang et al. 2018). Note that in our method, the initial estimates are used for the MST_s construction and as the adaptive weights in the penalty. It can be easily checked that Lemma 1 and Theorem 1 still hold, as long as the initial estimates are consistent. The similar discussion on determining the adaptive weight is also provided in Zou (2006).

Figure 8(b) shows the clustering results on the stations. In the results, we have nine spatially contiguous clusters. To better understand the clustering result from DTFLR, we compare it with the National Land Cover Database[4] (NLCD) in 2011, because we believe the relationship between AT and LST varies with the landcover reflection. The NLCD provides the characteristics of the land surface on 30-meter resolution over seven main categories. We summarize the category percentages of each cluster in Table 2.

Table 2. The landcover category percentages (%) of each cluster based on the DTFLR clustering results and NLCD 2011.

Cluster	Developed	Forest	Planted	Shrubland	Wetlands
#1	80	3	12	0	5
#2	60	27	13	0	0
#3	56	0	28	8	8
#4	45	45	0	10	0
#5	100	0	0	0	0
#6	67	33	0	0	0
#7	0	33	0	0	67
#8	41	23	0	4	32
#9	0	0	0	0	100

[4] *https://www.usgs.gov/centers/eros/science/national-land-cover-database*

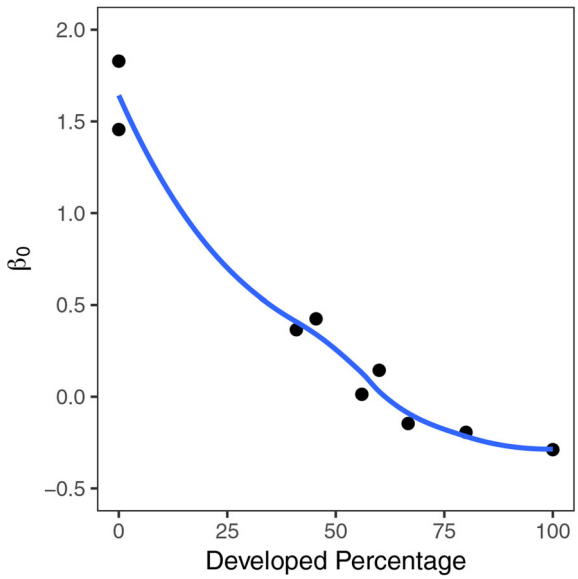


Figure 10. The estimated intercept $\boldsymbol{\beta}_0$ against the developed space percentage (%). The blue line is based on the local regression fitting.

Now, based on our clustering result, we analyze the pattern of the coefficient $\boldsymbol{\beta}$ over the landcover type. The coefficients for each cluster are shown in Figure 9. It can be seen that the main difference is on the intercept $\boldsymbol{\beta}_0$ in Figure 9(a). Here we show the intercept $\boldsymbol{\beta}_0$ against the developed space percentage in Figure 10. We can see that with the increasing of the developed space percentage, the intercept $\boldsymbol{\beta}_0$ are decreasing. It implies that with the same LST $\mathbf{x}_1$ and the elevation $\mathbf{x}_2$, the highly developed space has the lower air temperature y. Then for the LST coefficient $\boldsymbol{\beta}_1$ (see in Figure 9(b)), all the estimates are almost the same, around the value 1. This means that the LST effect is almost fixed across the clusters, and with one unit increasing in the LST, the AT also increases one unit. Lastly, we can see from Figure 9(c) that the elevation effect are around 0, except Clusters 7 and 9. We find that these two clusters are at the south part of Florida, and are mostly the wetland with the lowest elevation. It can be believed that these characteristics make these two clusters distinguished from the others.

9. Discussion

We considered the problem of distributively learning the regression coefficient heterogeneity over networks. We developed a new adaptive spanning-tree-based fused-lasso model and a low-complexity distributed generalized ADMM algorithm to solve the problem. We term our method as develop a distributed spanning-tree-based fused lasso regression (DTFLR). We investigated the theoretical properties of both the model consistency and algorithm convergence. Our theoretical results were validated by the thorough numerical experiments and real data analysis, which show that our approach outperforms the existing works in terms of estimation accuracy, computation speed and communication costs.

In this work, we focus on the case that the node number K is fixed and the local sample size increases. However, it is often the case that the number of the spatial location K increase and each location holds limited samples, that is, n is fixed, in many real spatial problems. Though our proposed optimization algorithm still works for that case, the statistical properties of our model need to be further investigated. Also, our framework assumed a simple balanced design that all nodes have local sample size going to infinity. It would be worth studying the imbalanced design where only a subset of nodes have $\mathbf{n} \to \infty$. Second, we leverages Assumption 1 for the consistency analysis. Highly connected network graph can guarantee the assumption while introduces intensive computation and communication costs. Thus, it is worth investigating the practical guideline for choosing the spatial network. Lastly, generalizing our framework to a more general class of regression problems, including generalized and semi-parametric linear model, will be an interesting future topic.

Supplementary Materials

The supplementary materials contain detailed derivations for the proposed algorithm, proofs for the theoretical results, as well as the additional numerical simulations. The implementation code is also provided in the supplementary materials.

Disclosure Statement

The authors report there are no competing interests to declare.

Funding

This work has been supported in part by NSF grants CAREER CNS-2110259, CNS-2112471, CCF-1934884, CCF-2110252, and a Google Faculty Research Award.

ORCID

Zhengyuan Zhu http://orcid.org/0000-0002-2266-0646

References

Ando, R. K., and Zhang, T. (2007), "Learning on Graph with Laplacian Regularization," in *Advances in Neural Information Processing Systems*, pp. 25–32. [3]

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011), "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, 3, 1–122. [5]

Cadena, J., Ray, P., Chen, H., Soper, B., Rajan, D., Yen, A., and Goldhahn, R. (2021), "Stochastic Gradient-based Distributed Bayesian Estimation in Cooperative Sensor Networks," *IEEE Transactions on Signal Processing*, 69, 1713–1724. [1]

Chow, Y.-T., Shi, W., Wu, T., and Yin, W. (2016), "Expander Graph and Communication-Efficient Decentralized Optimization," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 1715–1720, IEEE. [4]

Damiani, A., Vallati, M., Gatta, R., Dinapoli, N., Jochems, A., Deist, T., van Soest, J., Dekker, A., and Valentini, V. (2015), "Distributed Learning to Protect Privacy in Multi-Centric Clinical Studies," in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 65–75, Springer. [1]

Deng, W., and Yin, W. (2016), "On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers," *Journal of Scientific Computing*, 66, 889–916. [6]

Eisen, M., Mokhtari, A., and Ribeiro, A. (2017), "Decentralized Quasi-Newton Methods," *IEEE Transactions on Signal Processing*, 65, 2613–2628. [3]

Fan, J., and Li, R. (2001), "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties," *Journal of the American Statistical Association*, 96, 1348–1360. [2]

Gallager, R. G., Humblet, P. A., and Spira, P. M. (1983), "A Distributed Algorithm for Minimum-Weight Spanning Trees," *ACM Transactions on Programming Languages and systems (TOPLAS)*, 5, 66–77. [4]

Guo, F., Yu, F. R., Zhang, H., Ji, H., Leung, V. C., and Li, X. (2020), "An Adaptive Wireless Virtual Reality Framework in Future Wireless Networks: A Distributed Learning Approach," *IEEE Transactions on Vehicular Technology*, 69, 8514–8528. [1]

Hallac, D., Leskovec, J., and Boyd, S. (2015), "Network Lasso: Clustering and Optimization in Large Graphs, in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 387–396, ACM. [2,3,4]

Hastie, T., and Tibshirani, R. (1996), "Discriminant Analysis by Gaussian Mixtures," *Journal of the Royal Statistical Society*, Series B, 58, 155–176. [2]

Jiang, Z., Mukherjee, K., and Sarkar, S., (2018), "On Consensus-Disagreement Tradeoff in Distributed Optimization," in *2018 Annual American Control Conference (ACC)*, pp. 571–576, IEEE. [2,3]

Jochems, A., Deist, T. M., Van Soest, J., Eble, M., Bulens, P., Coucke, P., Dries, W., Lambin, P., and Dekker, A. (2016), "Distributed Learning: Developing a Predictive Model Based on Data from Multiple Hospitals Without Data Leaving the Hospital–a Real Life Proof of Concept," *Radiotherapy and Oncology*, 121, 459–467. [1]

Ke, Z. T., Fan, J., and Wu, Y. (2015), "Homogeneity Pursuit," *Journal of the American Statistical Association*, 110, 175–194. [4,6]

Knuth, D. E. (1976), "Big Omicron and Big Omega and Big Theta," *ACM Sigact News*, 8, 18–24. [2]

Konecny, J., McMahan, B., and Ramage, D. (2015), "Federated Optimization: Distributed Optimization Beyond the Datacenter," arXiv preprint arXiv:1511.03575. [2]

Lee, D.-J., Zhu, Z., and Toscas, P. (2015), "Spatio-Temporal Functional Data Analysis for Wireless Sensor Networks Data," *Environmetrics*, 26, 354–362. [2,9]

Li, F., and Sang, H. (2018), "Spatial Homogeneity Pursuit of Regression Coefficients for Large Datasets," *Journal of the American Statistical Association*, 114, 1050–1062. [2,4,5,6,7]

Li, X., Zhou, Y., Asrar, G. R., and Zhu, Z. (2018a), "Creating a Seamless 1 km Resolution Daily Land Surface Temperature Dataset for Urban and Surrounding Areas in the Conterminous United States," *Remote Sensing of Environment*, 206, 84–97. [11]

——— (2018b), "Developing a 1 km Resolution Daily Air Temperature Dataset for Urban and Surrounding Areas in the Conterminous United States," *Remote Sensing of Environment*, 215, 74–84. [9,11]

Lin, S.-B., Wang, D., and Zhou, D.-X. (2020), "Distributed Kernel Ridge Regression with Communications," *Journal of Machine Learning Research*, 21, 1–38. [1]

Ma, S., and Huang, J. (2017), "A Concave Pairwise Fusion Approach to Subgroup Analysis, *Journal of the American Statistical Association*, 112, 410–423. [2,4,5,6]

Ma, S., Huang, J., Zhang, Z., and Liu, M. (2020), "Exploration of Heterogeneous Treatment Effects via Concave Fusion," *The International Journal of Biostatistics*, 16. *https://doi.org/10.1515/ijb-2018-0026*. [2,4,5,6]

Mateos, G., Bazerque, J. A., and Giannakis, G. B. (2010), "Distributed Sparse Linear Regression," *IEEE Transactions on Signal Processing*, 58, 5262–5276. [1]

Menne, M. J., Durre, I., Korzeniewski, B., McNeal, S., Thomas, K., Yin, X., Anthony, S., Ray, R., Vose, R. S., Gleason, B. E., and Houston, T. G., (2012), "Global Historical Climatology Network-Daily (ghcn-daily), version 3," NOAA National Climatic Data Center, 10, p. V5D21VHZ. [11]

Nedic, A., and Ozdaglar, A. (2009), "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Transactions on Automatic Control*, 54, 48–61. [3]

Rohe, K., Chatterjee, S., Yu, B. (2011), "Spectral Clustering and the High-Dimensional Stochastic Blockmodel," *The Annals of Statistics*, 39, 1878–1915. [2]

Shen, J., and He, X. (2015), "Inference for Subgroup Analysis with a Structured Logistic-Normal Mixture Model," *Journal of the American Statistical Association*, 110, 303–312. [2]

Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W. (2014), "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," *IEEE Transactions on Signal Processing*, 62, 1750–1761. [3]

Tang, L., and Song, P. X. (2016), "Fused Lasso Approach in Regression Coefficients Clustering: Learning Parameter Heterogeneity in Data Integration," *The Journal of Machine Learning Research*, 17, 3915–3937. [2,4,6]

Wahlberg, B., Boyd, S., Annergren, M., and Wang, Y. (2012), "An ADMM Algorithm for a Class of Total Variation Regularized Estimation Problems, in *IFAC Proceedings* (Vol. 45), pp. 83–88. [5]

Wang, W., Wang, J., Kolar, M., and Srebro, N. (2018), "Distributed Stochastic Multi-Task Learning with Graph Regularization," arXiv preprint arXiv:1802.03830. [2,3,7,11]

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014), "Distributed Bayesian Posterior Sampling via Moment Sharing," in *Advances in Neural Information Processing Systems* (Vol. 27), pp. 3356–3364. [1]

Xu, P., Wang, Y., Chen, X., and Tian, Z. (2021), "Coke: Communication-Censored Decentralized Kernel Learning," *Journal of Machine Learning Research*, 22, 1–35. [1]

Yuan, K., Ling, Q., and Yin, W., "On the Convergence of Decentralized Gradient Descent," *SIAM Journal on Optimization*, 26, 1835–1854. [3]

Zhu, Y. (2017), "An Augmented ADMM Algorithm with Application to the Generalized Lasso Problem," *Journal of Computational and Graphical Statistics*, 26, 195–204. [5]

Zou, H. (2006), "The Adaptive Lasso and its Oracle Properties," *Journal of the American Statistical Association*, 101, 1418–1429. [4,11]