Online Distributed Learning over Graphs with Multitask Graph-Filter Models

Fei Hua, Roula Nassif, Cédric Richard, Haiyan Wang, Ali H Sayed

Online Distributed Learning over Graphs with Multitask Graph-Filter Models

Fei Hua, Roula Nassif, *Member, IEEE,* Cédric Richard, *Senior Member, IEEE,* Haiyan Wang and Ali H. Sayed, *Fellow, IEEE*

***Abstract*—In this work, we are interested in adaptive and distributed estimation of graph filters from streaming data. We formulate this problem as a consensus estimation problem over graphs, which can be addressed with diffusion LMS strategies. Most popular graph-shift operators such as those based on the graph Laplacian matrix, or the adjacency matrix, are not energy preserving. This may result in an ill-conditioned estimation problem, and reduce the convergence speed of the distributed algorithms. To address this issue and improve the transient performance, we introduce a preconditioned graph diffusion LMS algorithm. We also propose a computationally efficient version of this algorithm by approximating the Hessian matrix with local information. Performance analyses in the mean and mean-square sense are provided. Finally, we consider a more general problem where the filter coefficients to estimate may vary over the graph. To avoid a large estimation bias, we introduce an unsupervised clustering method for splitting the global estimation problem into local ones. Numerical results show the effectiveness of the proposed algorithms and validate the theoretical results.**

***Index Terms*—Graph signal processing, graph filter, diffusion LMS, node-varying graph filter, clustering.**

I. Introduction

Data generated by network-structured applications often exhibit non-Euclidean structures, which make traditional signal processing techniques inefficient to analyze them. In contrast, graph signal processing (GSP) provides useful tools to analyze and process signals on graphs. They represent them as samples at the vertices of a possibly weighted graph, and use algebraic and spectral properties of the graph to study the signals. These graph representations are useful in applications ranging from social and economic networks to smart grids [3]–[6]. Recent results in the area include sampling [7]–[9], filtering [10], and inference and learning [11]–[14], to cite a few.

The work of C. Richard was funded in part by the PIA program under its IDEX UCAJEDI project (ANR-15-IDEX-0001) and by ANR under grant ANR-19-CE48-0002. The work of F. Hua was supported in part by China Scholarship Council and NSFC grant 61771396. The work of H. Wang was partially supported by NSFC under grants 61571365, 61671386 and National Key R&D Program of China (2016YFC1400200). The work of A. H. Sayed was supported in part by NSF grant CCF-1524250. Part of the results in this paper were presented at EUSIPCO [1] and ASILOMAR [2] conferences.

F. Hua and H. Wang are with School of Marine Science and Technology, Northwestern Polytechnical University, Xian, China. F. Hua is also with Université Côte d'Azur, OCA, CNRS, France. H. Wang is also with School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xian, China (emails: fei.hua@oca.eu; hywang@sust.edu.cn).

C. Richard is with Université Côte d'Azur, OCA, CNRS, France (email: cedric.richard@unice.fr).

R. Nassif and A. H. Sayed are with School of Engineering, Ecole Polytechnique Fédérale de Lausanne, Lausanne, CH-1015, Switzerland (emails: roula.nassif@epfl.ch; ali.sayed@epfl.ch).

In order to cope with graph signals, GSP relies on two ingredients: the graph shift operator (GSO) on one hand, which accounts for the topology of the graph, and the graph Fourier transform (GFT) on the other hand, which allows to represent graph signals in the graph frequency domain. Built upon the definition of the GFT, graph filters play a central role in processing graph signal spectra by selectively amplifying or attenuating frequency components. Various architectures of graph filters have been proposed in the literature, including finite impulse response (FIR) [6], [10] and infinite impulse response (IIR) [15], [16] filters. From a perspective of scalability, and considering energy constraints and band-limited communications that may be encountered in large networks of distributed nodes such as sensor networks, significant efforts have been made recently to derive distributed graph filters. These filtering procedures allow each node to exchange only local information with its neighboring nodes [17]–[21].

Much of the GSP literature has focused on static graph signals, that is, signals that need not evolve with time. However, a wide spectrum of network-structured problems requires adaptation to time-varying dynamics. Sensor networks, social networks, vehicular networks, communication networks, and power grids are some typical examples. Prior to the more recent GSP literature, many earlier works on adaptive networks have addressed problems dealing with this challenge by developing processing strategies that are well-suited to data streaming into graphs; see, e.g., [22]–[24]. Several diffusion strategies have been introduced, and their performance studied in various situations, such as diffusion LMS [25], RLS [26], and APA [27]. By referring to the problem of estimating an optimal parameter vector at a node as a "task", and depending on the relations between the parameter vectors across the entire network, adaptive networks can be divided into single or multitask networks. In single-task networks, all nodes estimate the same parameter vector. Typical works include [22]–[24]. With multitask networks, multiple but related parameter vectors are inferred simultaneously in a cooperative manner, so as to improve the estimation accuracy by using the similarities between tasks [28]–[31].

Related works: In this work, we are interested in online learning of linear graph models for representing streaming graph signals in a distributed manner. We focus on diffusion strategies because they are scalable, robust, and enable network adaptation and learning. Recently, some research works have considered time-varying graph signals. An adaptive graph signal reconstruction algorithm based on the LMS is proposed in [32] but it operates in a centralized manner. In [33], the

authors focus on estimating a network structure capturing the dependencies among time series in the form of a possibly directed, weighted adjacency matrix. A causal autoregressive process is introduced in the time series to capture the intuition that information travels over the network at some fixed speed. In [34], vector autoregressive (VAR) and vector autoregressive moving average (VARMA) models are proposed for predicting time-varying processes on graphs. Joint time-vertex stationarity is introduced for time-varying graph signals in [35], [36], and a joint time-vertex harmonic analysis for graph signals is proposed in [37]. It is shown that joint stationarity facilitates estimation or recovery tasks when compared to purely time or graph methods.

All the aforementioned works focus on centralized solutions whereas distributed algorithms may be more appropriate within the context of big data applications. In [38], [39], the behavior of some distributed graph filters on time-varying graph signals is studied. Considering graph signal sampling and reconstruction, several distributed algorithms have been proposed to track time-varying band-limited graph signals, e.g., LMS-based algorithms in [40], RLS-based methods in [41], Kalman-based methods in [42], and kernel-based algorithms in [43]. In [44], the authors are interested in time-varying graph signals with temporal smoothness prior. They devise distributed gradient descent algorithms to reconstruct the signals. Most of these works assume that the graph signals are band-limited. Another limitation is that the graph Fourier decomposition (eigenvectors) is needed beforehand, which is impractical for large networks.

Contributions: Recently, several works successfully applied adaptive algorithms to graph signals. In [32], [40], [41] for instance, graph Fourier coefficients are learned from streaming graph signals under band-limited assumption to perform adaptive reconstruction and tracking of time-varying graph signals from partial observations. In this work, we are interested in online distributed learning of linear graph models without assumption of band-limited processes. We use graph filter models in the time-vertex domain where there is no need to decompose the graph shift operator. The formulated optimization problem relies on minimizing a global cost consisting of the aggregate sum of individual costs at each vertex. To address this problem, we blend concepts from adaptive networks [45] and graph signal processing to devise graph diffusion LMS strategies. Considering that most popular shift operators are not energy preserving and may result in a slow convergence speed, we introduce a preconditioned optimization strategy to improve the transient performance. As this may lead to an increased computational complexity, we further propose a computationally efficient algorithm. Explicit theoretical performance analyses in the mean and mean-square-error sense are provided. We also give alternative theoretical results that are tractable for large networks. Simulation results show the efficiency of the proposed algorithms and validate the theoretical models. Finally, we extend these node-invariant filter models to more flexible ones where each node in the graph seeks to estimate a local node-varying graph filter. This allows us to exploit more degrees of freedom in the filter coefficients to better model graph signals. We introduce an unsupervised clustering strategy to determine which nodes in the graph share the same graph filter and may collaborate to estimate its parameters. Numerical results on a real-word dataset illustrate the efficiency of the proposed methods.

The rest of the paper is organized as follows. Section II formulates the problem and provides the centralized solution. Section III introduces the distributed algorithms, namely, the graph diffusion LMS strategy and its preconditioned counterparts. Section IV provides their theoretical analyses in the mean and the mean-square sense. A clustering method is devised to estimate local node-varying graph filters in Section V. Numerical results in Section VI show the effectiveness of these algorithms and validate the theoretical models.

Notations: We use normal font letters to denote scalars, boldface lowercase letters to denote column vectors, and boldface uppercase letters to denote matrices. The m-th entry of a vector $\boldsymbol{x}$ is denoted by x_m or $[\boldsymbol{x}]_m$ when needed, the (m,n)-th entry of a matrix $\boldsymbol{X}$ is denoted by x_{mn} or $[\boldsymbol{X}]_{m,n}$ when needed, the k-th row of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{k,\bullet}$. We use the symbol $\otimes$ to denote the Kronecker product and the symbol $\mathrm{Tr}(\cdot)$ to denote the trace operator. The operator $\mathrm{col}\{\cdot\}$ stacks the column vector entries on top of each other. The symbol $\mathrm{vec}(\boldsymbol{X})$ refers to the vectorization operator that stacks the columns of a matrix on top of each other. Operator $\boldsymbol{x}=\mathrm{diag}(\boldsymbol{X})$ stores the diagonal entries of $\boldsymbol{X}$ into vector $\boldsymbol{x}$, and $\boldsymbol{X}=\mathrm{diag}(\boldsymbol{x})$ is a diagonal matrix containing the vector $\boldsymbol{x}$ along its main diagonal. The symbol $\mathrm{bdiag}\{\cdot\}$ forms a matrix from block arguments by placing each block immediately below and to the right of its predecessor. The symbol $\|\cdot\|_{b,\infty}$ denotes the block maximum norm of a matrix. The symbols $\mathbf{1}$ and $\boldsymbol{I}$ are the vector of all ones and the identify matrix of appropriate size, respectively. The symbols $\rho(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the spectral radius and the maximum eigenvalue of its matrix argument, respectively.

II. Problem Formulation and Centralized Solution

We consider an undirected, weighted and connected graph $\mathcal{G}=(\mathcal{N},\mathcal{E},\boldsymbol{W})$ of N nodes, where $\mathcal{N}=\{1,2,\ldots,N\}$ is the set of nodes, and $\mathcal{E}$ is the set of edges such that if node k is connected to node ℓ, then $(k,\ell)\in\mathcal{E}$. We denote by $\mathcal{N}_k$ the neighborhood of node k including itself, namely, $\mathcal{N}_k=\{\ell:\ell=k\vee(\ell,k)\in\mathcal{E}\}$. Matrix $\boldsymbol{W}\in\mathbb{R}^{N\times N}$ is the adjacency matrix whose (k,ℓ)-th entry $w_{k\ell}$ assigns a weight to the relation between vertices k and ℓ. Since the graph is undirected, $\boldsymbol{W}$ is a symmetric matrix. The degree matrix $\boldsymbol{D}\triangleq\mathrm{diag}(\boldsymbol{W}\mathbf{1})$ is a diagonal matrix whose i-th diagonal entry is the degree of node i which is equal to the sum of all the weights of edges incident at node i. The combinatorial Laplacian matrix is defined as $\boldsymbol{L}\triangleq\boldsymbol{D}-\boldsymbol{W}$ which is a real, symmetric, positive semi-definite matrix. We further assume that the graph is endowed with a graph shift operator defined as an $N\times N$ shift matrix $\boldsymbol{S}$ whose entry $s_{k\ell}$ can be non-zero only if $k=\ell$ or $(k,\ell)\in\mathcal{E}$. Although the shift matrix can be any matrix that captures the graph topology for the problem at hand [4], popular choices are the graph Laplacian matrix [6], the adjacency matrix [10], and their normalized counterparts.

A graph signal is defined as $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample associated with node k. Let $\boldsymbol{x}(i)$ denote the graph signal at time i. Operation $\boldsymbol{S}\boldsymbol{x}$ is called graph shift. It can be performed locally at each node k by linearly combining the samples x_ℓ from its neighboring nodes $\ell \in \mathcal{N}_k$.

A. Graph filter and data model

In this paper, we focus on linear shift-invariant FIR graph filters $\boldsymbol{H} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ of order M, which are polynomials of the graph-shift operator [10]:

$$\boldsymbol{H} \triangleq \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m, \tag{1}$$

where $\boldsymbol{h}^o = \{h_m^o\}_{m=0}^{M-1}$ are the scalar filter coefficients. With the definitions of graph signal and graph shift operator, one common filtering model assumes that the filtered graph signal $\boldsymbol{y}(i)$ is generated from the input graph signal $\boldsymbol{x}(i)$ as follows [10], [46]:

$$\boldsymbol{y}(i) = \boldsymbol{H}\boldsymbol{x}(i) + \boldsymbol{v}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i) + \boldsymbol{v}(i), \tag{2}$$

where $\boldsymbol{v}(i) = [v_1(i), \ldots, v_N(i)]^\top \in \mathbb{R}^N$ denotes an i.i.d. zero-mean noise independent of any other signal and with covariance matrix $\boldsymbol{R}_v = \text{diag}\{\sigma_{v,k}^2\}_{k=1}^N$. For each node k, the filtered signal $y_k(i)$ can be computed by linearly combining the input signals at nodes located in an $(M-1)$-hop neighborhood [6]. This model however assumes the instantaneous diffusion of information over the graph since $\boldsymbol{S}^m\boldsymbol{x}(i)$ translates $\boldsymbol{x}(i)$ without time delay. As this assumption may appear as a serious limitation, we consider the more general model embedding the temporal dimension as follows [38], [47]:

$$\boldsymbol{y}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i-m) + \boldsymbol{v}(i). \tag{3}$$

Observe that the input signal $\boldsymbol{x}(i)$ in (2) has been replaced by $\boldsymbol{x}(i-m)$ in (3), i.e., the m-hop spatial shift $\boldsymbol{S}^m$ is now carried out in m time slots. This model implements an FIR filter in both graph domain and temporal domain. By retaining the following shifted signals that form the $N \times M-1$ matrix:

$$\boldsymbol{X}_r = \Big[\boldsymbol{x}(i-1), \boldsymbol{S}\boldsymbol{x}(i-2), \ldots, \boldsymbol{S}^{M-2}\boldsymbol{x}(i-M+1)\Big], \tag{4}$$

note that only one shift is required at time instant i to produce the filtered signal $\boldsymbol{y}(i)$. This means that $y_k(i)$ can be computed using only local information available within the one-hop neighborhood of node k. Let $\boldsymbol{Z}(i)$ denote the $N \times M$ matrix given by:

$$\boldsymbol{Z}(i) \triangleq \Big[\boldsymbol{x}(i),\, \boldsymbol{S}\boldsymbol{x}(i-1), \ldots, \boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)\Big], \tag{5}$$

then model (3) can be written alternatively as:

$$\boldsymbol{y}(i) = \boldsymbol{Z}(i)\boldsymbol{h}^o + \boldsymbol{v}(i) \tag{6}$$

From model (6), sample $y_k(i)$ at node k can be written as:

$$y_k(i) = \boldsymbol{z}_k^\top(i)\boldsymbol{h}^o + v_k(i), \tag{7}$$

where $\boldsymbol{z}_k^\top(i)$ is the k-th row of $\boldsymbol{Z}(i)$ given by:

$$\boldsymbol{z}_k(i) \triangleq \text{col}\big\{[\boldsymbol{x}(i)]_k,\, [\boldsymbol{S}\boldsymbol{x}(i-1)]_k, \ldots, [\boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)]_k\big\}. \tag{8}$$

Observe in (7) that each node shares the same filter coefficient vector $\boldsymbol{h}^o$. The corresponding graph filter (1) is referred to as *node-invariant* graph filter. A more flexible model was introduced in [19], and called a *node-variant* graph filter. It allows the filter coefficients to vary across nodes as follows:

$$\boldsymbol{H} \triangleq \sum_{m=0}^{M-1} \text{diag}(\boldsymbol{h}^{(m)})\boldsymbol{S}^m, \tag{9}$$

with $\boldsymbol{h}^{(m)} \in \mathbb{R}^N$. If $\boldsymbol{h}^{(m)} = h_m\mathbf{1}$ for all m, model (9) reduces to the node-invariant model (1). Otherwise, each node applies different weights to the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$. Then $y_k(i)$ in (7) can be re-written as:

$$y_k(i) = \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k^o + v_k(i), \tag{10}$$

where $\boldsymbol{h}_k^o \in \mathbb{R}^M$ is the filter coefficient vector at node k collected into $\boldsymbol{h}^{(m)}$, i.e., $[\boldsymbol{h}_k^o]_m = [\boldsymbol{h}^{(m)}]_k$. In this work, we seek to estimate $\boldsymbol{h}_k^o$ from the filtered graph signal $y_k(i)$ and inputs $\boldsymbol{z}_k(i)$, in a distributed, collaborative and adaptive manner. Distributed algorithms such as the diffusion LMS exist in the literature to address single-task and multitask inference problems with similar data models as (7) or (10). In this work, however, regressors $\boldsymbol{z}_k(i)$ in (8) are raised from graph shifted signals. This paper aims to exploit the graph shift structure in the regression data and incorporate it into the formulation of the distributed algorithm – see expression (22) further ahead. In the sequel, first, we shall study the case where the filter coefficients are common for all nodes, i.e. $\boldsymbol{h}_k^o = \boldsymbol{h}^o, \forall k \in \mathcal{N}$. We shall show how to estimate $\boldsymbol{h}^o$ from streaming data $\{\boldsymbol{y}(i), \boldsymbol{x}(i)\}$ in a centralized way and then, in a distributed way. Next, we shall assume that there are clusters of nodes within the graph, and each node in the same cluster uses the same filter. This model is called a *hybrid node-varying* graph filter [48]. We shall introduce an unsupervised clustering method to allow each node to identify which neighboring nodes it should collaborate with.

B. Centralized solution

Before introducing the distributed method, we first introduce the centralized solution. Consider the data model (6) and assume that $\{\boldsymbol{y}(i), \boldsymbol{x}(i), \boldsymbol{v}(i)\}$ are zero-mean jointly wide-sense stationary random processes. Estimating $\boldsymbol{h}^o$ from $\{\boldsymbol{y}(i), \boldsymbol{Z}(i)\}$ can be performed by solving the following problem:

$$\boldsymbol{h}^o = \arg\min_{\boldsymbol{h}} J(\boldsymbol{h}), \tag{11}$$

where $J(\boldsymbol{h})$ denotes the mean-square-error criterion:

$$\begin{aligned} J(\boldsymbol{h}) &= \mathbb{E}\|\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}\|^2 \\ &= \mathbb{E}\{\boldsymbol{y}^\top(i)\boldsymbol{y}(i)\} - 2\boldsymbol{h}^\top\boldsymbol{r}_{Zy} + \boldsymbol{h}^\top\boldsymbol{R}_Z\boldsymbol{h}, \end{aligned} \tag{12}$$

and the $M \times M$ matrix $\boldsymbol{R}_Z$ and the $M \times 1$ vector $\boldsymbol{r}_{Zy}$ are given by:

$$\boldsymbol{R}_Z \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)\}, \quad \boldsymbol{r}_{Zy} \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{y}(i)\}. \tag{13}$$

By setting the gradient vector of $J(\boldsymbol{h})$ to zero, the optimal parameter vector $\boldsymbol{h}^o$ can be found by solving:

$$\boldsymbol{R}_Z \boldsymbol{h}^o = \boldsymbol{r}_{Zy}. \tag{14}$$

It can be verified that the (m,n)-th entry of $\boldsymbol{R}_Z$ is given by:

$$[\boldsymbol{R}_Z]_{m,n} = \text{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1} \boldsymbol{R}_x(m-n)\right) \tag{15}$$

where $\boldsymbol{R}_x(m) \triangleq \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i-m)\}$. The m-th entry of the vector $\boldsymbol{r}_{Zy}$ is given by:

$$[\boldsymbol{r}_{Zy}]_m = \text{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{R}_{xy}(m)\right), \tag{16}$$

with $\boldsymbol{R}_{xy}(m) \triangleq \mathbb{E}\{\boldsymbol{y}(i)\boldsymbol{x}^\top(i-m)\}$ denoting cross correlation function, which is assumed independent of time i.

Instead of solving (14), $\boldsymbol{h}^o$ can be sought iteratively by using the gradient-descent method:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu\big[\boldsymbol{r}_{Zy} - \boldsymbol{R}_Z \boldsymbol{h}(i)\big], \tag{17}$$

with $\mu > 0$ a small step-size. Since the statistical moments are usually unavailable beforehand, one way is to replace them by the instantaneous approximations $\boldsymbol{R}_Z \approx \boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)$ and $\boldsymbol{r}_{Zy} \approx \boldsymbol{Z}^\top(i)\boldsymbol{y}(i)$. This yields the LMS graph filter:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu \boldsymbol{Z}^\top(i)\big[\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}(i)\big]. \tag{18}$$

This stochastic-gradient algorithm is referred to as the *centralized graph-LMS* algorithm. In this centralized setting, each node k at each time instant i sends its data $\{x_k(i), y_k(i)\}$ to a fusion center which will update $\boldsymbol{h}(i)$ according to (18). Note that the step-size μ in (18) must satisfy $0 < \mu < \frac{2}{\lambda_{\max}(\boldsymbol{R}_Z)}$ in order to guarantee stability in the mean under certain independence conditions on the data [49].

III. Diffusion LMS Strategies over Graph Signals

In this section, we seek to estimate the graph filter coefficients in a distributed fashion. First, we review the graph diffusion LMS strategy [47]. Then, a preconditioned algorithm is proposed to improve the transient performance. We also devise a computationally efficient counterpart of this algorithm.

A. Graph diffusion LMS

Consider the local data model (7) at node k. It is worth noting that, by retaining the past shifted signals $\{[\boldsymbol{S}^{m-1}\boldsymbol{x}(i-m)]_\ell : m = 1,\ldots,M-1\}$ at each node ℓ in the network from previous iterations, $\boldsymbol{z}_k(i)$ can be computed locally at node k from its one-hop neighbors at each iteration i. Let $\boldsymbol{R}_{z,k} \triangleq \mathbb{E}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}$ denote the $M \times M$ covariance matrix with (m,n)-th entry given by [47]:

$$[\boldsymbol{R}_{z,k}]_{m,n} = \text{Tr}\left([\boldsymbol{S}^{m-1})]_{k,\bullet}^\top [\boldsymbol{S}^{n-1}]_{k,\bullet} \boldsymbol{R}_x(m-n)\right). \tag{19}$$

Considering the local cost $J_k(\boldsymbol{h})$ at node k:

$$J_k(\boldsymbol{h}) = \mathbb{E}|y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}|^2, \tag{20}$$

the global cost (12) is now the aggregation of the local costs over the graph:

$$J(\boldsymbol{h}) = \sum_{k=1}^N J_k(\boldsymbol{h}). \tag{21}$$

In order to minimize (12) in a decentralized fashion, there are several useful techniques, e.g., incremental strategy [50], consensus strategy [51] and diffusion strategy [24]. Diffusion strategies are attractive since they are scalable, robust, and enable continuous learning and adaptation. In particular, the adapt-then-combine (ATC) diffusion LMS takes the following form at node k [47]:

$$\boldsymbol{z}_k^\top(i) = \Big[x_k(i), \sum_{\ell\in\mathcal{N}_k} s_{k\ell}\ [\boldsymbol{z}_\ell(i-1)]_1, \ldots, \sum_{\ell\in\mathcal{N}_k} s_{k\ell}\ [\boldsymbol{z}_\ell(i-1)]_{M-1}\Big], \tag{22a}$$

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{z}_k(i)\big[y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)\big], \tag{22b}$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1), \tag{22c}$$

where $\mu_k > 0$ is a local step-size parameter and $\{a_{\ell k}\}$ are non-negative combination coefficients chosen to satisfy:

$$a_{\ell k} > 0, \quad \sum_{\ell=1}^N a_{\ell k} = 1, \ \text{ and } \ a_{\ell k} = 0 \ \text{ if } \ \ell \notin \mathcal{N}_k. \tag{23}$$

This implies that the matrix $\boldsymbol{A}$ with (ℓ,k)-th entry $a_{\ell k}$ is a left-stochastic matrix, which means that the sum of each of its columns is equal to 1. In the first step (22a), each node k uses the first $M-1$ entries of $\boldsymbol{z}_\ell(i-1)$ from its one-hop neighbors and its own input sample $x_k(i)$ to compute $\boldsymbol{z}_k(i)$. Note that the first $M-1$ entries of $\boldsymbol{z}_k(i)$ then need to be retained for the next iteration. In the adaptation step (22b), each node k updates its local estimate $\boldsymbol{h}_k(i)$ to an intermediate estimate $\boldsymbol{\psi}_k(i+1)$. In the combination step (22c), node k aggregates all the intermediate estimates $\boldsymbol{\psi}_\ell(i+1)$ from its neighbors to obtain the updated estimate $\boldsymbol{h}_k(i+1)$.

B. Graph diffusion preconditioned LMS

The regressor $\boldsymbol{z}_k(i)$ used in the adaptation step (22b) results from shifted graph signals while the shift matrix $\boldsymbol{S}$ is not energy preserving in general [52]. This is due to the fact that the magnitude of the eigenvalues of the shift operator $\boldsymbol{S}$ are not uniformly equal to 1; the energy of the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$ changes exponentially with m. Thus, the eigenvalue spread of $\boldsymbol{R}_{z,k}$ may be large and the LMS update may suffer from slow convergence speed in this case [49]. To address this issue, albeit at an increased computational cost, we resort to a form of Newton's method. Focusing on the adaptation step, we have:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) - \mu_k[\nabla^2_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))]^{-1}[\nabla_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))], \tag{24}$$

where $\nabla^2_{\boldsymbol{h}} J_k(\cdot)$ denotes the Hessian matrix for $J_k(\cdot)$ and $\nabla_{\boldsymbol{h}} J_k(\cdot)$ is its gradient vector, if available. For the quadratic cost function (20), expression (24) would lead to:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{R}_{z,k}^{-1}\big[\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)\big], \tag{25}$$

where $\boldsymbol{r}_{zy,k} = \mathbb{E}\{\boldsymbol{z}_k(i)y_k(i)\}$. Note that the second term on the RHS of (25) requires second-order moments. Since they are rarely available beforehand, we can replace $\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)$ by the instantaneous approximation:

$$\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i) \approx \boldsymbol{z}_k(i)e_k(i) \tag{26}$$

with $e_k(i) = y_k(i) - \boldsymbol{z}_k^{\top}(i)\boldsymbol{h}_k(i)$. The adaptation step (25) becomes:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \widehat{\boldsymbol{R}}_{z,k}^{-1}(i)\boldsymbol{z}_k(i)e_k(i), \qquad (27)$$

where $\widehat{\boldsymbol{R}}_{z,k}(i)$ is an estimate for $\boldsymbol{R}_{z,k}(i)$ which can possibly be obtained recursively:

$$\widehat{\boldsymbol{R}}_{z,k}(i) = (1-\mu)\,\widehat{\boldsymbol{R}}_{z,k}(i-1) + \mu\big[\boldsymbol{z}_k(i)\boldsymbol{z}_k^{\top}(i)\big], \quad i \geq 1, \ (28)$$

where μ is a small factor that can be chosen in $(0, 0.1]$ in practice. It can be verified that $\mathbb{E}\{\widehat{\boldsymbol{R}}_{z,k}(i)\} = \boldsymbol{R}_{z,k}$ is an unbiased estimate when $i \to \infty$. As discussed before, $\boldsymbol{S}$ may not be energy preserving and results in a large eigenvalue spread of $\boldsymbol{R}_{z,k}$, which may even be close to singular. The inverse $\boldsymbol{R}_{z,k}^{-1}$ would then be ill-conditioned and lead to undesirable effects. To address this problem, it is common to use regularization [49]. We obtain the diffusion LMS-Newton algorithm:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\big[\epsilon\boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1}\boldsymbol{z}_k(i)e_k(i), \quad (29a)$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1), \qquad (29b)$$

with $\epsilon \geq 0$ a small regularization parameter. Compared with the diffusion LMS algorithm (22), algorithm (29) requires first to recursively estimate the Hessian matrix according to (28) and then calculate $\big[\epsilon\boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1}$. This algorithm can lead to improved performance as shown in the sequel, but at the expense of additional computation cost.

In order to reduce the computational complexity of the LMS-Newton algorithm, we propose to use a preconditioning matrix $\boldsymbol{P}_k$ does not depend on the graph signal $\boldsymbol{x}(i)$ in the adaptation step, instead of the Hessian matrix $\boldsymbol{R}_{z,k}$ or its estimate $\widehat{\boldsymbol{R}}_{z,k}$. Since the large eigenvalue spread of the input covariance matrix $\boldsymbol{R}_{z,k}$ results mainly from the shift matrix $\boldsymbol{S}$ and the filter order M, we construct an $M\times M$ preconditioning matrix $\boldsymbol{P}_k$ as follows:

$$\boldsymbol{P}_k \triangleq \text{diag}\{\|[\boldsymbol{S}^{(m-1)}]_{k,\bullet}\|^2\}_{m=1}^{M}. \qquad (30)$$

The rationale behind (30) is that, in the case where $\boldsymbol{x}(i)$ is i.i.d. with variance σ^2, it follows from (19) that $\boldsymbol{R}_{z,k} = \sigma^2\boldsymbol{P}_k$. According to (30), matrix $\boldsymbol{P}_k$ does not depend on $\boldsymbol{x}(i)$ and can be evaluated beforehand at each node k during an initial step. Each node k only requires to know the edge weights in its M-hop neighborhood, which can be performed in a decentralized manner. Interestingly, $\boldsymbol{P}_k$ is a diagonal matrix, which means that the matrix product in the adaptation step does not require expensive matrix inversion. Following the same line of reasoning as for the Newton algorithm (29), a regularization term $\epsilon\boldsymbol{I}_M$ can be added to $\boldsymbol{P}_k$. This leads to:

$$\boldsymbol{D}_k = (\epsilon\boldsymbol{I}_M + \boldsymbol{P}_k)^{-1}. \qquad (31)$$

We arrive at the following preconditioned graph diffusion LMS strategy:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\boldsymbol{D}_k\boldsymbol{z}_k(i)e_k(i), \qquad (32a)$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1). \qquad (32b)$$

At each iteration i, node k uses the local information to update the intermediate estimate $\boldsymbol{\psi}_k(i+1)$ in the adaptation step (32a). Then, in the combination step (32b), the intermediate estimates $\boldsymbol{\psi}_\ell(i+1)$ from the neighborhood of node k are combined to get $\boldsymbol{h}_k(i+1)$. Although the preconditioning matrix $\boldsymbol{D}_k$ is not the true Hessian matrix, we prove in Section IV that the algorithm converges to the optimal solution $\boldsymbol{h}^o$ provided that it is stable.

C. Comparison with the graph diffusion LMS

We explain how preconditioning with (30) improves performance. For comparison purposes, let us first focus on the adaptation step of diffusion LMS. At each node k, the m-th entry of $\boldsymbol{h}_k(i)$ is updated as follows:

$$[\boldsymbol{\psi}_k(i+1)]_m = [\boldsymbol{h}_k(i)]_m + \mu_k\big[\boldsymbol{z}_k(i)e_k(i)\big]_m. \qquad (33)$$

During the transient phase, the m-th entry $[\boldsymbol{h}_k(i)]_m$ exponentially converges to its optimal value with a time constant [49]:

$$\tilde{\tau}_m \approx \frac{1}{2\mu_k\lambda_m} \qquad (34)$$

where λ_m denotes the m-th eigenvalue of $\boldsymbol{R}_{z,k}$. Given μ_k, the convergence rate of each entry of $\boldsymbol{h}_k(i)$ then depends on the corresponding eigenvalue. Disparity between entries increases as the eigenvalue spread defined as $\lambda_{\max}/\lambda_{\min}$ increases.

The preconditioning matrix $\boldsymbol{D}_k$ is diagonal at each node k, which means that the m-th entry of $\boldsymbol{h}_k(i)$ in (32a) converges to its optimal value with a time constant:

$$\tilde{\tau}_m \approx \frac{1}{2\mu_k d_{k,m}\lambda_m} \qquad (35)$$

where $d_{k,m}$ denotes the m-th diagonal entry of $\boldsymbol{D}_k$. The convergence speed now depends on $d_{k,m}\lambda_m$. Considering the case where $\boldsymbol{P}_k$ is proportional to $\boldsymbol{R}_{z,k}$, then $d_{k,m}$ is inversely proportional to λ_m, which mitigates the effects of the eigenvalues spread. We shall analyze and illustrate in Section IV and Section VI, respectively, how this preconditioning improves convergence speed in more general cases.

IV. Performance analysis

We shall now analyze the stochastic behavior of the diffusion preconditioned LMS (PLMS) algorithm (32) in the sense of mean and mean-square error. We introduce the following weight error vectors at each node k:

$$\tilde{\boldsymbol{h}}_k(i) = \boldsymbol{h}^o - \boldsymbol{h}_k(i), \qquad \tilde{\boldsymbol{\psi}}_k(i) = \boldsymbol{h}^o - \boldsymbol{\psi}_k(i), \qquad (36)$$

and we collect them across the nodes into the network weight error vectors :

$$\tilde{\boldsymbol{h}}(i) \triangleq \text{col}\{\tilde{\boldsymbol{h}}_1(i), \tilde{\boldsymbol{h}}_2(i), \ldots, \tilde{\boldsymbol{h}}_N(i)\}, \qquad (37)$$

$$\tilde{\boldsymbol{\psi}}(i) \triangleq \text{col}\{\tilde{\boldsymbol{\psi}}_1(i), \tilde{\boldsymbol{\psi}}_2(i), \ldots, \tilde{\boldsymbol{\psi}}_N(i)\}. \qquad (38)$$

We refer to *mean stability* of the error vector $\tilde{\boldsymbol{h}}(i)$ if the limit superior of $\|\mathbb{E}\tilde{\boldsymbol{h}}(i)\|$ is bounded. Furthermore, we will claim that the algorithm converges in the mean to the optimum if $\mathbb{E}\tilde{\boldsymbol{h}}(i)$ converges to zero as i tends to $+\infty$ regardless of the starting point. *Mean-square stability* refers to the case where the superior limit of $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2$ is bounded.

Let us introduce the following $N \times N$ block matrices with individual entries of size $M \times M$:

$$\boldsymbol{\mathcal{A}} \triangleq \boldsymbol{A} \otimes \boldsymbol{I}_M, \tag{39}$$

$$\boldsymbol{\mathcal{M}} \triangleq \text{bdiag}\{\mu_k \boldsymbol{I}_M\}_{k=1}^N, \tag{40}$$

$$\boldsymbol{\mathcal{D}} \triangleq \text{bdiag}\{\boldsymbol{D}_k\}_{k=1}^N. \tag{41}$$

The estimation error in (32a) can be written as:

$$e_k(i) = y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i) = \boldsymbol{z}_k^\top(i)\tilde{\boldsymbol{h}}_k(i) + v_k(i). \tag{42}$$

Subtracting $\boldsymbol{h}^o$ from both sides of (32a) and using the above relation, then stacking $\tilde{\boldsymbol{\psi}}_k(i)$ across the nodes, we obtain

$$\tilde{\boldsymbol{\psi}}(i+1) = (\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z(i))\,\tilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i), \tag{43}$$

where $\boldsymbol{\mathcal{R}}_z(i)$ is an $N \times N$ block matrix with entries of size $M \times M$ define as:

$$\boldsymbol{\mathcal{R}}_z(i) \triangleq \text{bdiag}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}_{k=1}^N, \tag{44}$$

and $\boldsymbol{p}_{zv}(i)$ is an $N \times 1$ block column vector with entries of size $M \times 1$ given by:

$$\boldsymbol{p}_{zv}(i) \triangleq \text{col}\{\boldsymbol{z}_k(i)v_k(i)\}_{k=1}^N. \tag{45}$$

Subtracting $\boldsymbol{h}^o$ from both sides of (32b), we obtain the block weight error vector:

$$\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{A}}^\top\tilde{\boldsymbol{\psi}}(i+1). \tag{46}$$

Finally, combing (43) and (46), the network weight error vector $\tilde{\boldsymbol{h}}(i)$ of algorithm (32) evolves according to:

$$\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}(i)\tilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i), \tag{47}$$

with

$$\boldsymbol{\mathcal{B}}(i) = \boldsymbol{\mathcal{A}}^\top\Big(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z(i)\Big). \tag{48}$$

To proceed with the analysis, we introduce the following assumption.

Assumption 1 (independent inputs)**.** *The inputs $\boldsymbol{z}_k(i)$ arise from a zero-mean random process that is temporally white with $\boldsymbol{R}_{z,k} > 0$.*

A consequence of Assumption 1 is that $\boldsymbol{z}_k(i)$ is independent of $\boldsymbol{h}_\ell(j)$ for all ℓ and $j \leq i$. This independence assumption is not true in the current work. Two successive regressors $\boldsymbol{z}_k$ involve common entries that cannot be statistically independent as in a conventional FIR implementation. However, when the step-size is sufficiently small, conclusions derived under this assumption tend to be realistic. For more details and discussions, see [49, Section 16.4]. Since this assumption helps to simplify the derivations without constraining the conclusions, it is widely used in the literature of adaptive filters and adaptive networks [24], [49]. We shall see in Section VI that the resulting expressions match well the simulation results for sufficiently small step-sizes.

A. Mean-error behavior analysis

Taking expectations of both sides of (47), using the fact that $\mathbb{E}\boldsymbol{p}_{zv}(i) = 0$, and applying Assumption 1, we find that the network mean error vector evolves according to:

$$\mathbb{E}\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}\,\mathbb{E}\tilde{\boldsymbol{h}}(i), \tag{49}$$

where:

$$\boldsymbol{\mathcal{B}} \triangleq \mathbb{E}\boldsymbol{\mathcal{B}}(i) = \boldsymbol{\mathcal{A}}^\top(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z), \tag{50}$$

$$\boldsymbol{\mathcal{R}}_z \triangleq \mathbb{E}\boldsymbol{\mathcal{R}}_z(i) = \text{bdiag}\{\boldsymbol{R}_{z,k}\}_{k=1}^N. \tag{51}$$

Theorem 1 (Convergence in the mean)**.** *Assume that data model* (7) *and Assumption 1 hold. Then, for any initial condition, algorithm* (32) *converges asymptotically in the mean toward the optimal vector $\boldsymbol{h}^o$ if, and only if, the step-sizes in $\boldsymbol{\mathcal{M}}$ are chosen to satisfy:*

$$\rho\left(\boldsymbol{\mathcal{A}}^\top(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z)\right) < 1, \tag{52}$$

where $\rho(\cdot)$ denotes the spectral radius of its matrix argument. In the case where the signal $\boldsymbol{x}(i)$ is i.i.d, a sufficient condition for (52) *to hold is to choose μ_k such that:*

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{D}_k\boldsymbol{R}_{z,k})}, \qquad k = 1, \ldots, N. \tag{53}$$

Proof. The weight error vector $\tilde{\boldsymbol{h}}(i)$ converges to zero if, and only if, the coefficient matrix $\boldsymbol{\mathcal{B}}$ in (49) is a stable matrix, namely, $\rho(\boldsymbol{\mathcal{B}}) < 1$. Since any induced matrix norm is lower bounded by the spectral radius, we have the following relation in terms of block maximum norm [24]:

$$\begin{aligned}\rho(\boldsymbol{\mathcal{B}}) &\leq \|\boldsymbol{\mathcal{A}}^\top(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z)\|_{b,\infty} \\ &\leq \|\boldsymbol{\mathcal{A}}^\top\|_{b,\infty} \cdot \|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty} \\ &= \|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty},\end{aligned} \tag{54}$$

where the last equality follows from the fact that $\boldsymbol{\mathcal{A}}$ is left stochastic, which implies that $\|\boldsymbol{\mathcal{A}}^\top\|_{b,\infty} = 1$ from Lemma D.4 of [24]. Matrix $\boldsymbol{\mathcal{R}}_z$ is block diagonal if $\boldsymbol{x}(i)$ is i.i.d. Since $\boldsymbol{\mathcal{D}}$ is also diagonal, their product is symmetric, and $\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z$ is a block diagonal symmetric matrix. Then, following Lemma D.5 of [24], its block maximum norm agrees with its spectral radius:

$$\|\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z\|_{b,\infty} = \rho(\boldsymbol{I}_{NM} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z). \tag{55}$$

Combining (54) and (55), we verify that condition (53) ensures the stability of $\boldsymbol{\mathcal{B}}$. □

B. Mean-square-error behavior analysis

We shall now study the mean-square-error behavior of algorithm (32). Let $\boldsymbol{\Sigma}$ be any $NM \times NM$ positive semi-definite matrix that we are free to choose. The freedom in selecting $\boldsymbol{\Sigma}$ will allow us to derive different performance measures about the network and the nodes. We consider the weighted mean-square error vector, i.e., $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\Sigma}}^2$, where $\|\tilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\Sigma}}^2 \triangleq \tilde{\boldsymbol{h}}^\top(i)\boldsymbol{\Sigma}\tilde{\boldsymbol{h}}(i)$. From Assumption 1 and $\mathbb{E}\boldsymbol{p}_{zv}(i) = \boldsymbol{0}$, using (47), we obtain the following variance relation:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|_{\boldsymbol{\Sigma}}^2 = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\Sigma}'}^2 + \mathbb{E}\|\boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|_{\boldsymbol{\Sigma}}^2, \tag{56}$$

where $\boldsymbol{\Sigma}' \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\}$. Let $\boldsymbol{\sigma}$ denote the $(NM)^2 \times 1$ vector obtained by vectorizing matrix $\boldsymbol{\Sigma}$, namely, $\boldsymbol{\sigma} = \text{vec}(\boldsymbol{\Sigma})$. With some abuse of notation, we shall use $\|\cdot\|^2_{\boldsymbol{\sigma}}$ to also refer to the quantity $\|\cdot\|^2_{\boldsymbol{\Sigma}}$ when it is more convenient. Let $\boldsymbol{\sigma}' = \text{vec}(\boldsymbol{\Sigma}')$. Considering that $\text{vec}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}) = (\boldsymbol{W}^\top \otimes \boldsymbol{U})\boldsymbol{\sigma}$, it can be verified that:

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma}, \tag{57}$$

where $\boldsymbol{\mathcal{F}}$ is the $(NM)^2 \times (NM)^2$ matrix given by

$$\begin{aligned}\boldsymbol{\mathcal{F}} \triangleq& \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i) \otimes \boldsymbol{\mathcal{B}}^\top(i)\} \\ =& \Big(\boldsymbol{I}_{(NM)^2} - \boldsymbol{I}_{NM} \otimes \boldsymbol{\mathcal{R}}_z^\top \boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{R}}_z^\top \boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{NM} \\ &+ \mathcal{O}(\boldsymbol{\mathcal{M}}^2)\Big)(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}),\end{aligned} \tag{58}$$

where $\mathcal{O}(\boldsymbol{\mathcal{M}}^2)$ denotes $\mathbb{E}\{\boldsymbol{\mathcal{R}}_z^\top(i)\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{\mathcal{R}}_z^\top(i)\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}}\}$, which depends on the square of the step-sizes, $\{\mu_k^2\}$. While we can continue the analysis by taking this factor into account as was done in other studies [53], it is sufficient for the exposition to focus on the case of sufficiently small step-sizes where terms involving higher powers of the step-sizes $\{\mu_k\}$ can be ignored. Following the same line of reasoning, for sufficiently small step-sizes $\{\mu_k\}$, $\boldsymbol{\mathcal{F}}$ can be approximated by:

$$\boldsymbol{\mathcal{F}} \approx \boldsymbol{\mathcal{B}}^\top \otimes \boldsymbol{\mathcal{B}}^\top. \tag{59}$$

The second term on the RHS of (56) can be written as:

$$\mathbb{E}\|\boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}} = \text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}), \tag{60}$$

where

$$\boldsymbol{\mathcal{G}} \triangleq \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}, \tag{61}$$

$$\boldsymbol{\mathcal{S}} \triangleq \mathbb{E}\{\boldsymbol{p}_{zv}(i)\boldsymbol{p}_{zv}^\top(i)\} = \text{bdiag}\{\boldsymbol{\sigma}_{v,k}^2 \boldsymbol{R}_{z,k}\}_{k=1}^N. \tag{62}$$

Using the property $\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{W}) = [\text{vec}(\boldsymbol{W}^\top)]^\top\boldsymbol{\sigma}$, combining (57) and (60), the variance relation (56) can be re-written as:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\sigma}. \tag{63}$$

Theorem 2 (Stability in the mean-square)**.** *Assume that data model* (7) *and Assumption 1 hold. Algorithm* (32) *converges in the mean-square sense if the matrix* $\boldsymbol{\mathcal{F}}$ *in* (58) *is stable. Assume further that the step-sizes are sufficiently small such that* (59) *is a reasonable approximation. In that case, the stability of* $\boldsymbol{\mathcal{F}}$ *is ensured if* $\boldsymbol{\mathcal{B}}$ *is stable.*

Proof. Iterating (63) starting from $i = 0$, we obtain

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(0)\|^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top \sum_{j=0}^{i} \boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}, \tag{64}$$

with initial condition $\tilde{\boldsymbol{h}}(0) = \boldsymbol{h}^o - \boldsymbol{h}(0)$. Provided $\boldsymbol{\mathcal{F}}$ is stable, $\boldsymbol{\mathcal{F}}^i \to 0$ as $i \to \infty$, then the first item on the RHS of (64) converges to zero and the second item converges to a finite value. The weighted mean-square error converges to a finite value as $i \to \infty$ which implies that the algorithm (32) will converge in the mean-square sense if $\boldsymbol{\mathcal{F}}$ is stable. Under the sufficiently small step-sizes assumption where the higher-order terms of $\boldsymbol{\mathcal{F}}$ in (58) can be neglected, approximation (59) is reasonable. The eigenvalues of $\boldsymbol{\mathcal{F}}$ are all the products of the eigenvalues of $\boldsymbol{\mathcal{B}}$, which means that $\rho(\boldsymbol{\mathcal{F}}) = [\rho(\boldsymbol{\mathcal{B}})]^2$. It follows that $\boldsymbol{\mathcal{F}}$ is stable if $\boldsymbol{\mathcal{B}}$ is stable. Therefore, when the graph signal $\boldsymbol{x}(i)$ is i.i.d, according to Theorem 1, condition (53) ensures mean-square stability of the algorithm under the assumed approximation (59). □

Theorem 3 (Network transient MSD)**.** *Assume sufficiently small step-sizes that ensure mean and mean-square stability. The network transient mean-square deviation (MSD) defined as* $\zeta(i) = \frac{1}{N}\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2$ *evolves according to the following recursion for* $i \geq 0$*:*

$$\begin{aligned}\zeta(i+1) = \zeta(i) + \frac{1}{N}\Big([vec(\mathbb{E}\{\tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0)\})]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(NM)^2}) \\ + [vec(\boldsymbol{\mathcal{G}}^\top)]^\top\Big)\boldsymbol{\mathcal{F}}^i vec(\boldsymbol{I}_{NM}).\end{aligned} \tag{65}$$

Proof. Comparing (64) at time $i+1$ and i, $\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}}$ is related to $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}}$ as follows:

$$\begin{aligned}\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma} + \\ [\text{vec}(\mathbb{E}\{\tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0)\})]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I}_{(NM)^2})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}.\end{aligned} \tag{66}$$

Substituting $\boldsymbol{\sigma}$ by $\frac{1}{N}\text{vec}(\boldsymbol{I}_{NM})$ leads to (65). □

Although expression (65) gives a compact form of the transient MSD model, it may not be practical to use since $\boldsymbol{\mathcal{F}}$ is of size $(NM)^2 \times (NM)^2$ and may become huge for large networks or high order filters. For example, in the simulation Section VI, we considered a network consisting of $N = 60$ nodes and a graph filter of degree $M = 5$. Matrix $\boldsymbol{\mathcal{F}}$ is of size 90000×90000 and requires a prohibitive amount of computational time and memory space (about 60GB in that case). To tackle this issue, we make use of the following properties of the Kronecker product:

$$\text{vec}(\boldsymbol{X}\boldsymbol{Y}\boldsymbol{Z}) = (\boldsymbol{Z}^\top \otimes \boldsymbol{X})\text{vec}(\boldsymbol{Y}) \tag{67}$$

$$\text{Tr}(\boldsymbol{X}\boldsymbol{Y}) = \big(\text{vec}(\boldsymbol{Y}^\top)\big)^\top\text{vec}(\boldsymbol{X}). \tag{68}$$

This leads to:

Corollary 1 (Alternative network transient MSD expression)**.**

$$\begin{aligned}\zeta(i+1) = \zeta(i) + \frac{1}{N}Tr\Big(\boldsymbol{\mathcal{B}}^i\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{B}}^i)^\top \\ + \tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0)\left((\boldsymbol{\mathcal{B}}^{i+1})^\top\boldsymbol{\mathcal{B}}^{i+1} - (\boldsymbol{\mathcal{B}}^i)^\top\boldsymbol{\mathcal{B}}^i\right)\Big).\end{aligned} \tag{69}$$

While the update with $\boldsymbol{\mathcal{F}}$ has a computation complexity of order $\mathcal{O}((NM)^2)$, using $\boldsymbol{\mathcal{B}}$ only requires matrix manipulations of order $\mathcal{O}(NM)$.

Corollary 2 (Network steady-state MSD)**.** *Consider sufficiently small step-sizes to ensure mean and mean-square convergence. The network steady-state MSD is given by*

$$\zeta^\star = \frac{1}{N}[vec(\boldsymbol{\mathcal{G}}^\top)]^\top(\boldsymbol{I}_{(NM)^2} - \boldsymbol{\mathcal{F}})^{-1}\ vec(\boldsymbol{I}_{NM}). \tag{70}$$

Proof. The network steady-state MSD is defined as:

$$\zeta^\star = \lim_{i\to\infty} \frac{1}{N}\mathbb{E}\{\|\tilde{\boldsymbol{h}}(i)\|^2\}. \tag{71}$$

If $\boldsymbol{\mathcal{F}}$ is stable, we obtain from (63) as $i \to \infty$:

$$\lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{(\boldsymbol{I}_{(NM)^2} - \boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\sigma}. \tag{72}$$

We obtain (70) by substituting $\boldsymbol{\sigma}$ in (72) by $\frac{1}{N}(\boldsymbol{I}_{(NM)^2} - \boldsymbol{\mathcal{F}})^{-1}\ \mathrm{vec}(\boldsymbol{I}_{NM})$. □

Following the same line of reasoning as for the transient MSD model, the steady-state MSD given by (70) can be equivalently expressed as:

Corollary 3 (Alternative steady-state MSD expression)**.**

$$\zeta^{\star} = \frac{1}{N}\sum_{i=0}^{\infty} Tr\Big(\boldsymbol{\mathcal{B}}^i \boldsymbol{\mathcal{G}} (\boldsymbol{\mathcal{B}}^i)^{\top}\Big) \tag{73}$$

This expression is obtained by a series expansion of (70). In practice, a limited number of iterations can be used instead of the upper limit index $i \to \infty$ to obtain an accurate result.

V. Unsupervised Clustering for hybrid node-varying graph filter

In Section III, we investigated the scenario where the nodes in a graph share a common filter coefficient vector. Now we extend this model to the more flexible case (10), which allows the filter coefficients to vary across the graph. We further assume that the graph is decomposed into Q clusters of nodes $\mathcal{C}_q$ and, within each cluster $\mathcal{C}_q$, there is a common filter coefficient vector $\boldsymbol{h}_q^o$ to estimate, namely,

$$\boldsymbol{h}_k^o = \boldsymbol{h}_q^o, \qquad \text{if } k \in \mathcal{C}_q. \tag{74}$$

We assume that there is no prior information on the clusters composition and that the nodes do not know which other nodes share the same estimation task. Applying the algorithm (32) within this context may result a bias due to aggregate intermediate estimates from different data models. To address this issue, automatic network clustering strategies may be used [28], [54]–[56] in order to inhibit cooperation between nodes from different clusters. These methods are based on local stand-alone estimation strategies that may not be efficient for the current context. Basically, the polynomial form (1) of graph filters does not make the estimation of the filter coefficients reliable enough for the higher degrees. In the following, we tackle this problem by devising a clustering strategy based on the PLMS.

First, we introduce the $N \times N$ instantaneous clustering matrix $\boldsymbol{E}_i$, whose (ℓ, k)-th entry shows if node k believes at time i that its neighboring node ℓ belongs to the same cluster or not, namely,

$$[\boldsymbol{E}_i]_{\ell k} = \begin{cases} 1, \ \text{if } \ell \in \mathcal{N}_k \text{ and } k \text{ believes that } \boldsymbol{h}_k^o = \boldsymbol{h}_\ell^o, \\ 0, \ \text{otherwise.} \end{cases} \tag{75}$$

At each time instant i, node k infers which neighbors belong to its cluster based on the non-zeros entries of the k-th column of $\boldsymbol{E}_i$. We collect these entries into a set $\mathcal{N}_{k,i}$, so that node k only combines the intermediate estimates from its neighbors in $\mathcal{N}_{k,i}$. Condition (23) on the combination coefficients becomes:

$$a_{\ell k} > 0, \quad \sum_{\ell=1}^{N} a_{\ell k} = 1, \ \text{ and } \ a_{\ell k} = 0 \ \text{ if } \ \ell \notin \mathcal{N}_{k,i}, \tag{76}$$

Since the clustering information is unknown beforehand, we propose to learn $\boldsymbol{E}_i$ in an online way by computing a Boolean variable $b_{\ell k}(i)$ defined as follows:

$$b_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{\psi}_\ell(i+1) - \boldsymbol{h}_k(i)\|^2 \le \beta, \\ 0, & \text{otherwise,} \end{cases} \tag{77}$$

with $\beta > 0$ a preset threshold. Variable $b_{\ell k}(i)$ is defined from the ℓ_2-norm distance between the estimates at two neighboring nodes. If this distance is smaller than the threshold β, the two nodes are then assigned to the same cluster. Note that the distance between $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ is used in (77), instead of the distance between $\boldsymbol{h}_\ell(i+1)$ and $\boldsymbol{h}_k(i+1)$, in order to merge the learning and the clustering processes. Consider the learning process at any node k defined in (32). Information about clusters is used in the combination step (32b), where $\mathcal{N}_k$ now denotes the neighboring nodes of k that share the same estimation task as node k. This information should be available as soon as possible in order to avoid estimation bias. Considering the distance between $\boldsymbol{h}_\ell(i+1)$ and $\boldsymbol{h}_k(i+1)$ to decide if nodes k and ℓ are in the same cluster would allow to update the composition of sets $\mathcal{N}_k$ and $\mathcal{N}_\ell$ in the combination step (32b) used to calculate parameter vectors $\boldsymbol{h}_\ell(i+2)$ and $\boldsymbol{h}_k(i+2)$. This latency time can be shortened by considering the distance between $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ right after the adaptation step (32a), and using this information to define $\mathcal{N}_k$ in the combination step.

This strategy usually fails if left as is, because the estimation of higher-order coefficients is not reliable enough and results in bad clustering performance. We propose to estimate this distance from M_k principal components of the estimates, to be defined, of the estimates. Because $\boldsymbol{R}_{z,k}$ cannot be reasonably used to perform a Principal Component Analysis (PCA) of the input data, as it is rarely available beforehand and would involve significant additional computational effort, we suggest instead using matrix $\boldsymbol{P}_k$. As for the PLMS, the rationale behind this is that $\boldsymbol{R}_{z,k} = \sigma^2 \boldsymbol{P}_k$ when $\boldsymbol{x}(i)$ is i.i.d. with variance σ^2. Another interest lies in that $\boldsymbol{P}_k$ is a diagonal matrix, which greatly simplifies calculations. Without loss of generality, consider that the diagonal entries of $\boldsymbol{P}_k$ are in decreasing order. Projecting data onto the first M_k principal axes then reduces to selecting their first M_k entries and set the other entries to zero. Dimension M_k can be determined by setting the ratio of explained variance to total variance to some desired level τ as follows:

$$\begin{aligned} &\text{minimize} \quad M_k \\ &\text{subject to} \quad \sum_{m=1}^{M_k} \hat{\pi}_{k,m} \ge \tau \end{aligned} \tag{78}$$

with $\hat{\pi}_{k,m} = [\boldsymbol{p}_k]_m / \mathrm{Tr}(\boldsymbol{P}_k)$ an approximation of the *proportion of total variance* [57]. In practice, we can use a predefined threshold $\tau \in [0.9, 1)$ to decide how many entries should be retained. The Boolean variable (77) becomes:

$$b_{\ell k}(i) = \begin{cases} 1, & \text{if } \frac{\|\boldsymbol{\psi}'_\ell(i+1) - \boldsymbol{h}'_k(i)\|^2}{\|\boldsymbol{h}'_k(i)\|^2} \le \beta, \\ 0, & \text{otherwise,} \end{cases} \tag{79}$$

with $\boldsymbol{\psi}'_\ell(i+1)$ and $\boldsymbol{h}'_k(i)$ the first M_k entries of $\boldsymbol{\psi}_\ell(i+1)$ and $\boldsymbol{h}_k(i)$ respectively. Compared to (77), note that the distance

in (79) that accounts for the similarity between $\boldsymbol{\psi}'_\ell(i+1)$ and $\boldsymbol{h}'_k(i)$ has been normalized. We suggest to choose $\beta \in (0, 0.01]$ to lower the false detection rate. To reduce noise effects, we further introduce a smoothing step:

$$t_{\ell k}(i) = \nu t_{\ell k}(i-1) + (1-\nu) b_{\ell k}(i), \tag{80}$$

where $t_{\ell k}(i)$ is a trust level, and ν is a forgetting factor in $(0, 1)$ to balance the past and present cluster assignments. Once the trust level $t_{\ell k}(i)$ exceeds a preset threshold θ, which can be chosen in $[0.5, 1)$, node k concludes that node ℓ belongs to its cluster, that is,

$$[\boldsymbol{E}_i]_{\ell k} = \begin{cases} 1, & \text{if } t_{\ell k}(i) \geq \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{81}$$

Based on $[\boldsymbol{E}_i]_{\ell k}$, each node k determines at each time instant i those nodes ℓ that it believes they belong to the same cluster, updates the combination coefficients according to (76), and finally combines the estimates from its neighbors with (32b).

VI. Numerical results

A. Experiment with i.i.d. input data

We first considered a zero-mean i.i.d. Gaussian graph signal $\boldsymbol{x}(i)$ with covariance $\boldsymbol{R}_x = \text{diag}\{\sigma^2_{x,k}\}_{k=1}^N$. Variances $\sigma^2_{x,k}$ were randomly generated from the uniform distribution $\mathcal{U}(1, 1.5)$. In this setting, the graph signal sample $x_k(i)$ was independent of $x_\ell(j)$ for all ℓ and $j \leq i$. We assumed the linear data model (7). The graph filter order was set to $M = 5$ and the coefficients h^o_m were randomly generated from the uniform distribution $\mathcal{U}(0, 1)$. Noise $\boldsymbol{v}(i)$ was zero-mean Gaussian with covariance $\boldsymbol{R}_v = \text{diag}\{\sigma^2_{v,k}\}_{k=1}^N$. Variances $\sigma^2_{v,k}$ were randomly generated from the uniform distribution $\mathcal{U}(0.1, 0.15)$. We considered this data model with an Erdős-Rényi random graph and on a sensor network graph. Both consisted of $N = 60$ nodes. The Erdős-Rényi random graph was generated in a similar construction as in [33]. Namely, it was obtained by generating an $N \times N$ symmetric matrix $\boldsymbol{S}$ whose entries were governed by the Gaussian distribution $\mathcal{N}(0, 1)$, and then thresholding edges to be between 1.2 and 1.8 in absolute value. Then, the edges were soft thresholded by 1.1 to be between 0.1 and 0.7 in magnitude. The shift matrix $\boldsymbol{S}$ was normalized by 1.1 times its largest eigenvalue. The sensor network was generated by using GSPBOX [58]. Each node was connected to its 5 nearest neighbors. The shift matrix was the normalized adjacency matrix, that is, $\boldsymbol{S} = \frac{\boldsymbol{W}}{1.1\lambda_{\max}(\boldsymbol{W})}$. In this case, all the eigenvalues of $\boldsymbol{S}$ are smaller than 1 and the energy of the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$ diminishes for large m. The smallest eigenvalue $\lambda_{\min}(\boldsymbol{R}_{z,k})$ was very small, and, for some node, it was close to 0.

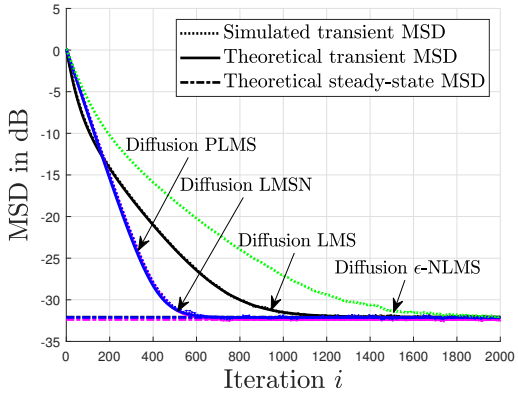


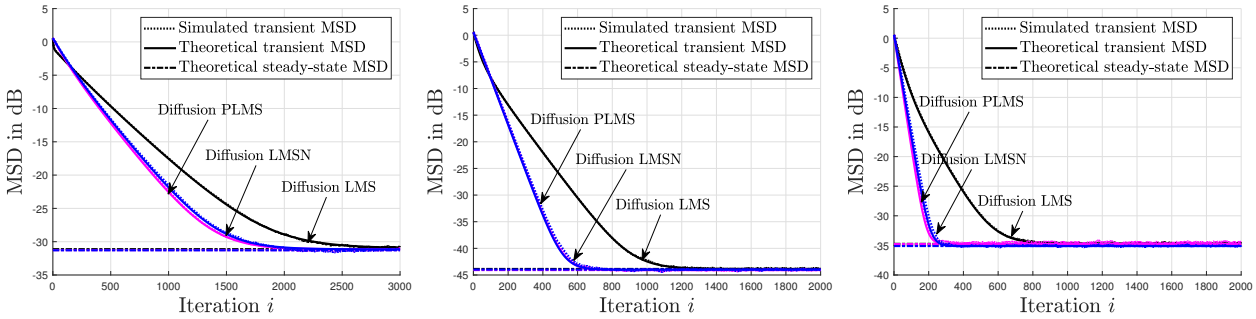
Fig. 1: Network MSD performance with the Erdős-Rényi graph.

With this setting, we compared the diffusion LMS algorithm (22), the diffusion LMS-Newton (LMSN) algorithm (29), and the diffusion preconditioned LMS (PLMS) algorithm (32). Simulated results were averaged over 500 Monte-Carlo runs. For the LMSN and PLMS algorithms, we set the regularization parameter as $\epsilon = 0.01$. We ran algorithms (22), (29) and (32) by setting $a_{\ell,k} = \frac{1}{|\mathcal{N}_k|}$ for $\ell \in \mathcal{N}_k$. We used a uniform step-size for all nodes, i.e., $\mu_k = \mu$ for all k. We also considered the ϵ-normalized LMS (ϵ-NLMS) method for comparison purposes. In this case, the adaptation step (32a) is substituted by:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \frac{\mu_k}{\|\boldsymbol{z}_k(i)\|^2 + \epsilon} \boldsymbol{z}_k(i) e_k(i). \tag{82}$$

With the Erdős-Rényi graph, we compared the performance of the LMS, PLMS, LMSN and ϵ-NLMS algorithms. We set $\mu = \{0.08, 0.008, 0.01, 0.05\}$, respectively. The network MSD performance of each algorithm is reported in Fig. 1. The theoretical transient and steady-state MSD are also reported. With the sensor network graph, we compared the performance of the LMS, PLMS and LMSN algorithms. We set $\mu = \{0.08, 0.005, 0.0055\}$, respectively. The performance of each algorithm is reported in Fig. 2(a). In Fig. 1, we observe that the diffusion ϵ-NLMS converged slower than all other algorithms. Observe that the diffusion LMSN and PLMS algorithms converged faster than the LMS algorithm for both graphs, and the diffusion PLMS performed similarly compared with the LMSN in terms of convergence rate. Also, note that the theoretical results match well the simulated curves.

In a second experiment, we considered the normalized graph Laplacian matrix $\boldsymbol{S} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$, and the adjacency matrix $\boldsymbol{W}$, as shift operators. For the normalized graph Laplacian, $\lambda_{\max}(\boldsymbol{R}_{z,k})$ was large for all nodes. Therefore, for the diffusion LMS algorithm, the step-size was chosen relatively small to guarantee convergence. We set $\mu = \{0.004, 0.01, 0.008\}$ for the LMS, LMSN and PLMS. The results are reported in Fig. 2(b). For the adjacency matrix, we used uniform step-sizes $\mu = \{0.02, 0.018\}$ for the LMSN and PLMS, respectively. The step-size was set to $\mu_k = 0.05 \cdot \frac{2}{\lambda_{\max}(\boldsymbol{R}_{z,k})}$ for each node k for the LMS update in order to achieve the same steady-state MSD. The results are reported in Fig. 2(c). We observe in Fig. 2 that the diffusion LMSN and PLMS algorithms converged faster than the LMS algorithm with the three graph shift operators. The PLMS algorithm achieved the same performance as the LMSN algorithm with a lower computational complexity.



(a) Normalized Adjacency Matrix (b) Normalized Laplacian Matrix (c) Adjacency Matrix

Fig. 2: Network MSD performance for different types of shift operators with the sensor network.

B. Experiment with correlated input data

We tested the algorithms over the sensor network graph with correlated graph signals. We first considered a zero-mean i.i.d. Gaussian graph signal driven by a non-diagonal covariance matrix $\boldsymbol{R}_x$. This means that the input data were correlated over the vertex domain, but uncorrelated over time. Matrix $\boldsymbol{R}_x$ was generated as $\boldsymbol{R}_x = \boldsymbol{V}\text{diag}\{\sigma_{x,k}^2\}_{k=1}^N \boldsymbol{V}^\top$, with $\sigma_{x,k}^2$ randomly chosen from the uniform distribution $\mathcal{U}(1, 1.5)$ and $\boldsymbol{V}$ is the graph Fourier transform matrix. The graph shift operator was defined by the normalized adjacency matrix. The filter degree was set as $M = 3$. We observe in Fig. 3 that the proposed diffusion PLMS algorithm performed as well as the LMSN algorithm, and converged faster than the diffusion LMS algorithm.

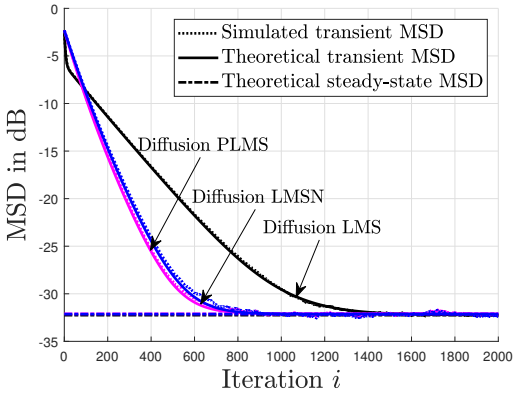


Fig. 3: Network MSD performance with a vertex domain correlated input signal.

Next, we considered a graph signal $\boldsymbol{x}(i)$ that was correlated over both vertex and time domains. We assumed that $\boldsymbol{x}(i)$ is a Gaussian process with zero mean and covariance matrix $\boldsymbol{R}_x$ satisfying the discrete Lyapunov equation:

$$\boldsymbol{S}\boldsymbol{R}_x\boldsymbol{S}^\top - \boldsymbol{R}_x + \boldsymbol{I} = 0. \tag{83}$$

Graph signal sample $\boldsymbol{x}(i)$ was related to $\boldsymbol{x}(i-1)$ as follows:

$$\boldsymbol{x}(i) = \boldsymbol{S}\boldsymbol{x}(i-1) + \boldsymbol{w}(i) \tag{84}$$

with $\boldsymbol{S}$ the normalized adjacency matrix and $\boldsymbol{w}(i)$ a zero-mean i.i.d. Gaussian noise with covariance $\boldsymbol{I}_N$. It can be checked that $\boldsymbol{x}(i)$ is wide-sense stationary with $\mathbb{E}\{\boldsymbol{x}(i)\} = 0$, and $\boldsymbol{R}_x(\tau) = \boldsymbol{S}^\tau \boldsymbol{R}_x(0)$ for all $\tau > 0$, where $\boldsymbol{R}_x(0)$ satisfies the Lyapunov equation (83). The graph filter order was set as $M = 3$. The step-sizes were set to $\mu = \{0.1, 0.038, 0.03\}$ for the LMS, LMSN and PLMS, respectively. The regularization parameter ϵ was set to $\epsilon = 0.1$. Fig. 4 depicts the simulated and theoretical MSD performance. We observe that, due to correlation over time, the diffusion LMSN method converged faster than the PLMS. However, the proposed PLMS algorithm still performed better than the diffusion LMS method.

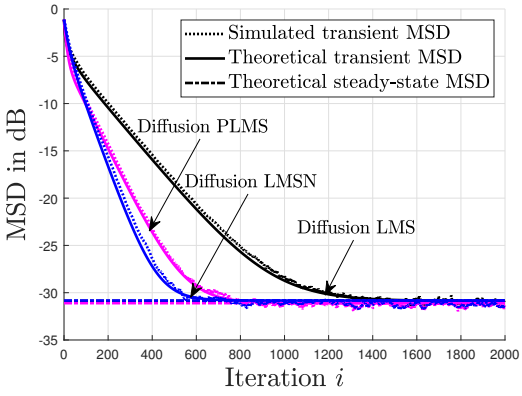


Fig. 4: Network MSD performance with input graph signal correlated over both vertex and time domains.

C. Clustering method for node-varying graph filter

Finally, we considered a scenario where nodes do not share the same filter coefficients. We assumed the linear data model (10). The graph shift operator was defined by the normalized adjacency matrix, and the graph filter order was set as $M = 3$. The nodes were grouped into three clusters: $\mathcal{C}_1 = \{1, \ldots, 20\}$, $\mathcal{C}_2 = \{21, \ldots, 40\}$, and $\mathcal{C}_3 = \{41, \ldots, 60\}$. The optimal graph filter coefficients $\boldsymbol{h}_k^o$ were set according to $[0.5\ 0.4\ 0.9]^\top$ if $k \in \mathcal{C}_1$, $[0.3\ 0.1\ 0.4]^\top$ if $k \in \mathcal{C}_2$, and $[0.9\ 0.3\ 0.7]^\top$ if $k \in \mathcal{C}_3$. We considered for comparison purpose the PLMS algorithm with clustering mechanism (78)-(81)

, with basic clustering mechanism (77) and $M_k = M$ for all k, the oracle PLMS algorithm where the clusters are assumed to be known a priori, the PLMS algorithm without clustering mechanism, and the non-cooperative algorithm where $a_{\ell k} = 1$ if $k = \ell$ and zero otherwise. All algorithms used the adaptation step (32a) with the same step-size $\mu_k = 0.01$ for all k. Parameters $\{\tau, \beta, \theta, \nu\}$ were set to $\{0.9, 0.01, 0.5, 0.98\}$, respectively. As shown in Fig. 5, the non-cooperative method did not achieve acceptable MSD level. The main reason is that, with the normalized adjacency matrix as graph shift operator $\boldsymbol{S}$, the entries of $\boldsymbol{z}_k(i)$ in (8) corresponding to higher powers of $\boldsymbol{S}$ are significantly diminished, resulting in poor estimation performance of filter coefficients when nodes cannot cooperate. The PLMS algorithm without clustering mechanism did not achieve good performance too because it has been designed to converge toward a consensual solution $\boldsymbol{h}^o$, which does not make sense for this scenario. The PMLS with clustering mechanism (71) and $M_k = M$ achieved slightly improved performance because the estimation of higher-order coefficients in $\boldsymbol{h}_k$ was not reliable enough, leading to incorrect clustering. The proposed PLMS algorithm with clustering mechanism (78)-(81) performed as well as the oracle algorithm. Fig. 6 (a) shows the topology of the graph given by the adjacency matrix $\boldsymbol{A}$ (and the shift matrix $\boldsymbol{S}$). Fig. 6 (b) presents the clusters inferred by the proposed method. These clusters perfectly match the ground truth clusters $\mathcal{C}_1$ to $\mathcal{C}_3$.

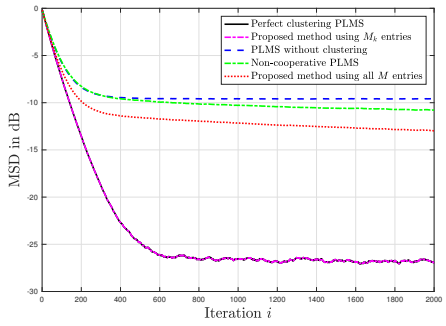
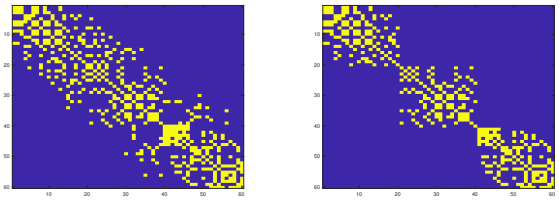


Fig. 5: Network MSD performance for different clustering algorithms.



(a) Adjacency Matrix (b) Inferred clusters

Fig. 6: Graph topology and clusters.

Next, we considered that optimal parameter vectors $\boldsymbol{h}_k^o$ change over time while clusters remain unchanged. Nodes were grouped into two clusters $\mathcal{C}_1 = \{1, \ldots, 30\}$ and $\mathcal{C}_2 = \{31, \ldots, 60\}$. The optimal parameter vectors changed for both clusters at time instant $i = 1000$. Simulation results in Fig. 7 show that the proposed clustering method was able to track well this change. Finally, we considered the scenario where clusters and models change simultaneously. At Stage 1, the nodes were grouped into two clusters defined as $\mathcal{C}_1 = \{1, \ldots, 30\}$ and $\mathcal{C}_2 = \{31, \ldots, 60\}$. Stage 2 started at time instant $i = 1000$ with three clusters $\mathcal{C}_1 = \{1, \ldots, 20\}$, $\mathcal{C}_2 = \{21, \ldots, 40\}$, and $\mathcal{C}_3 = \{41, \ldots, 60\}$. Stage 3 started at time instant $i = 2000$ with two clusters $\mathcal{C}_1 = \{1, \ldots, 25\}$, $\mathcal{C}_2 = \{26, \ldots, 60\}$. At each stage, the optimal parameter vectors $\boldsymbol{h}_k^o$ changed accordingly. Ground truth clusters for the three stages are depicted in Fig. 9 (Top). Parameters $\{\mu, \tau, \beta, \theta, \nu\}$ were set to $\{0.01, 0.9, 0.01, 0.5, 0.4\}$, respectively. Fig. 8 shows the simulated transient MSD of the proposed PLMS algorithm with clustering mechanism. It is compared with the oracle PLMS algorithm where the clusters are assumed to be known a priori. Fig. 9 (Bottom) depicts the inferred clusters at $i = 1000, 2000, 3000$ during one Monte Carlo run. The proposed algorithm was able to track changes in both clusters and models.

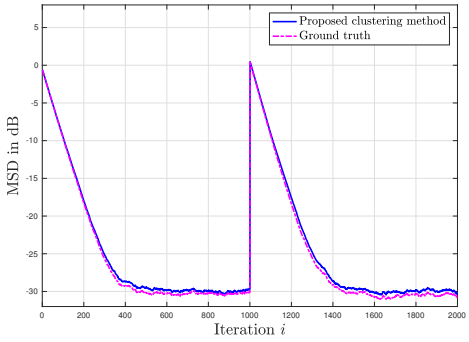


Fig. 7: Network MSD performance with model change.

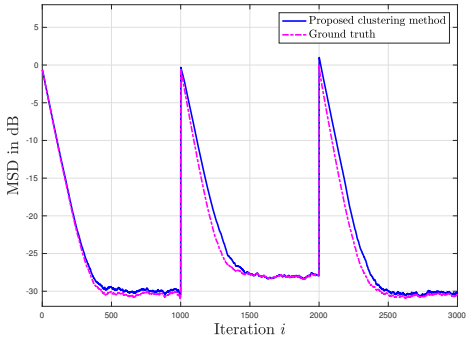


Fig. 8: Network MSD performance with model and clusters change.

D. Reconstruction on U.S. temperature dataset

We considered a dataset that collects hourly temperature measurements at $N = 109$ stations for $T = 8759$ hours across the United States in 2010 [59]. An undirected graph, illustrated in Fig. 10, was constructed according to the nodes coordinates by using the k-NN approach ($k = 7$) of GSPBOX.

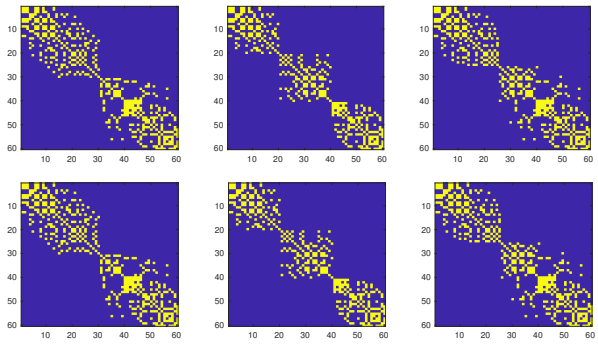
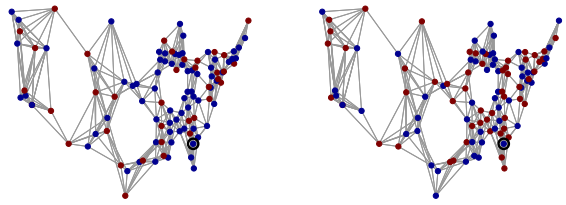


Fig. 9: *(Top)* Ground truth cluster. *(Bottom)* Inferred clusters at steady-state of a single Monte Carlo run. From left to right: Stage 1, Stage 2, Stage 3.



(a) (b)

Fig. 10: Graph topology for the U.S. temperatures dataset. Temperatures were sampled at the red nodes in red. Data at the blue nodes were unobserved. (a) 37 sampled nodes. (b) 54 sampled nodes.

In the first experiment, the dataset was divided into a training set containing $T_{train} = 6570$ hours data (about 75% of total). The remaining data were assigned to the test set. The goal of this experiment was to learn a graph filter that minimizes the reconstruction error over the training set, i.e.,

$$\min \sum_{i=1}^{T_{train}} \sum_{k=1}^{N} |y_k(i) - \sum_{m=1}^{M} h_{m,k}[\boldsymbol{S}^m \boldsymbol{x}(i-m+1)]_k|^2, \quad (85)$$

where $\boldsymbol{y}(i)$ is the ground truth temperature at time i, and $\boldsymbol{x}(i)$ is the partial observation given by $\boldsymbol{x}(i) = \text{diag}(\mathbf{1}_{\mathcal{S}})\boldsymbol{y}(i)$. Here $\mathbf{1}_{\mathcal{S}}$ denotes the set indicator vector, whose k-th entry is equal to one if node k is sampled, and zero otherwise. The sampling set, illustrated in Fig. 10 (a) was fixed over time in the first experiment. The normalized adjacency matrix was set as graph shift operator. Graph filter degree was set to $M = 4$. Note that if $h_{m,k} = h_m$ for all k, problem (85) refers to the single-task problem where all the nodes seek to find common graph filter coefficients; see model (1). Otherwise, problem (85) refers to the multitask problem; see model (9). We ran different models and algorithms on the training set to learn graph filter coefficients. In Fig. 11, we provide the true temperature and the reconstructed ones obtained by the different algorithms at an unobserved node, black circled in Fig. 10, over the last 120 hours samples of the test set. For comparison purposes, reconstruction results of the Kernel Kalman Filter (KKF) and the Kernel Ridge Regression (KRR) in [43] are also reported in Fig. 11. We observe that the single-task diffusion LMSN and multitask diffusion LMS were not able to reconstruct the true temperature, whereas the multitask diffusion PLMS and the multitask diffusion LMSN showed a good reconstruction performance, and performed better than the KKF and KRR at the selected unobserved node. To evaluate the performance over all unobserved nodes on the test set, we considered the normalized mean square error (NMSE) defined as:

$$\text{NMSE} = \frac{\sum_{i=T_{train}+1}^{T} \|\text{diag}(\mathbf{1}_{\bar{\mathcal{S}}})\,(\boldsymbol{y}(i) - \hat{\boldsymbol{y}}(i))\,\|_2^2}{\sum_{i=T_{train}+1}^{T} \|\text{diag}(\mathbf{1}_{\bar{\mathcal{S}}})\boldsymbol{y}(i)\|_2^2} \quad (86)$$

where $\hat{\boldsymbol{y}}(i)$ denotes the reconstructed estimate at time i, $\mathbf{1}_{\bar{\mathcal{S}}}$ is the set indicator vector whose k-th entry is equal to one if node k has not been sampled, and zero otherwise. The results are reported in Table I. We observe that the multitask diffusion PLMS performed as well as the LMSN at a lower computational cost, and both performed better than the KFF and KRR. Finally, Figure 12 reports the original topology and the clusters learned by the multitask diffusion PLMS.

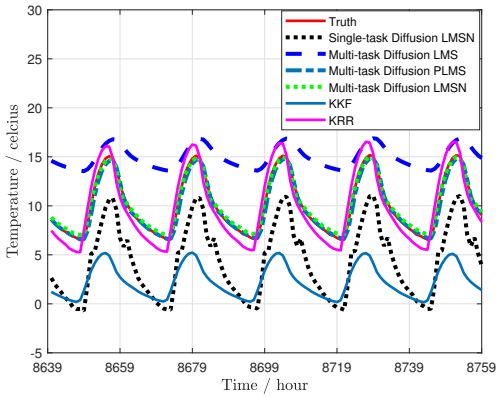
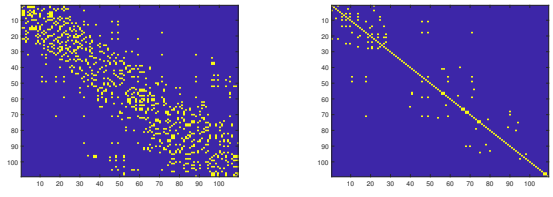


Fig. 11: True temperatures and reconstructed ones at an unobserved node. $\mu_{\text{LMS}} = 10^{-5}$, $\mu_{\text{PLMS}} = \mu_{\text{LMSN}} = 10^{-4}$.

TABLE I: NMSE of different algorithms.

Algorithm	NMSE
KKF	0.1093
KRR	0.0479
Multitask diffusion LMS	0.1152
Multitask diffusion PLMS	0.0090
Multitask diffusion LMSN	0.0031

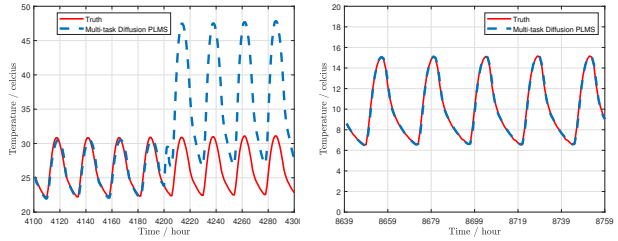
In the second experiment, we divided the dataset into two parts. The first part contained the first 4200 hours sampled at the nodes showed in Fig. 10 (a), and the second part contained the remaining hours sampled at the nodes showed in Fig. 10 (b). This means that the sampling set abruptly changed at time $t = 4201$ (the black circled node was unobserved in both case). We applied the multitask diffusion PLMS method over the entire dataset. In Fig. 13 (a), the reconstructed temperature at the unobserved black circled node is reported from time $t = 4100$ to $t = 4300$ by using the filter coefficients learned up to time $t = 4000$. As expected, we can notice that the



(a) Adjacency Matrix (b) Inferred clusters

Fig. 12: U.S. temperature graph topology and learned clusters.

reconstruction performance was successful from $t = 4100$ to $t = 4200$, and dramatically deteriorated after $t = 4201$. This is due to the fact that the sampling set changed, which led to a drift in the filter coefficients to estimate. Figure 13 (b) depicts the reconstruction behavior over last 120 hours using the filter coefficients learned up to time $t = 8600$. It can be observed that the proposed method was able to track the drift in the filter coefficients.



(a) (b)

Fig. 13: True temperatures and reconstructed ones at an unobserved node. For clarity purposes, focus on the intervals (a) $[4100, 4300]$ and (b) $[8640, 8759]$.

VII. Conclusion

Diffusion LMS strategies were considered to estimate graph filter coefficients in an adaptive and distributed manner. A diffusion LMS with Newton-like descent procedure was first proposed to achieve improved convergence rate, since usual algorithms may suffer from ill conditioning effects due to the use of non-energy preserving graph shift operators. A preconditioned diffusion LMS strategy, which does not require computationally intensive matrix inversion and only uses local information, was then devised to reduce the computational burden. Its convergence behavior was analyzed in the mean and mean-square-error sense. Finally, for hybrid node-varying graph filters, a clustering mechanism to be used with the preconditioned diffusion LMS was proposed. Simulation results validated the theoretical models and showed the efficiency of the proposed algorithms. In this work, we assumed that the graph shift operator is known and time invariant. In future works, we will consider ways to estimate the graph shift operator and the filter coefficients simultaneously. Because of their improved flexibility, which allows to address a variety of non-linear identification problems, we will also focus on non-parametric methods such as [60], [61] and see how they can be incorporated into GSP framework.

References

[1] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *Proc. 2018 26th Eur. Signal Process. Conf. (EUSIPCO)*, Rome, Italy, Sep. 2018, pp. 111–115.

[2] ——, "Decentralized clustering for node-variant graph filtering with graph diffusion LMS," in *Proc. 2018 52nd Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, Pacific Grove, CA, USA, Oct. 2018, pp. 1418–1422.

[3] P. Djuric and C. Richard, *Cooperative and Graph Signal Processing: Principles and Applications*. Academic Press, Elsevier, 2018.

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

[5] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[7] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.

[8] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, Jul. 2016.

[9] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, Sep. 2016.

[10] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[11] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, "A regularization framework for learning over multitask graphs," *IEEE Signal Process. Lett.*, vol. 26, no. 2, pp. 297–301, Feb. 2019.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2016, pp. 3844–3852.

[13] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2018.

[14] A. Anis, A. El Gamal, A. S. Avestimehr, and A. Ortega, "A sampling theory perspective of graph-based semi-supervised learning," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2322–2342, Apr. 2019.

[15] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite impulse response graph filters in wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1113–1117, Aug. 2015.

[16] J. Liu, E. Isufi, and G. Leus, "Filter design for autoregressive moving average graph filters," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 1, pp. 47–60, Mar. 2019.

[17] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1931–1935, Nov. 2015.

[18] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, Jan. 2017.

[19] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, Aug. 2017.

[20] M. A. Coutino Minguez, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2320–2333, May 2019.

[21] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 736–751, Dec. 2018.

[22] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, May. 2013.

[23] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[24] ——, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, S. Theodoridis and R. Chellappa, Eds. Academic Press, Elsevier, 2014, vol. 3, pp. 322–454.

[25] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[26] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[27] L. Li and J. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *Proc. 2009 IEEE Statist. Signal Process. Workshop (SSP)*, Cardiff, UK, 2009, pp. 757–760.

[28] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, Jun. 2015.

[29] J. Chen, C. Richard, A. O. Hero, and A. H. Sayed, "Diffusion LMS for multitask problems with overlapping hypothesis subspaces," in *Proc. 2014 IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Reims, France, 2014, pp. 1–6.

[30] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Multitask diffusion adaptation over asynchronous networks," *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2835–2850, Jun. 2016.

[31] ——, "Proximal multitask learning over networks with sparsity-inducing coregularization," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6329–6344, Dec. 2016.

[32] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, "Adaptive least mean squares estimation of graph signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 555–568, Dec. 2016.

[33] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, Apr. 2017.

[34] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, Sep. 2019.

[35] A. Loukas and N. Perraudin, "Stationary time-vertex signal processing," *arXiv preprint arXiv:1611.00255*, 2016.

[36] A. Loukas, E. Isufi, and N. Perraudin, "Predicting the evolution of stationary graph signals," in *Proc. 2017 51st Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, Pacific Grove, CA, USA, Oct. 2017, pp. 60–64.

[37] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 817–829, Feb. 2018.

[38] E. Isufi, G. Leus, and P. Banelli, "2-dimensional finite impulse response graph-temporal filters," in *Proc. 2016 IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Greater Washington, D.C., USA, 2016, pp. 405–409.

[39] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4406–4421, Aug. 2017.

[40] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4193–4208, Aug. 2017.

[41] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Trans. Signal Process.*, vol. 66, no. 13, pp. 3584–3598, Jul. 2018.

[42] E. Isufi, P. Banelli, P. Di Lorenzo, and G. Leus, "Observing and tracking bandlimited graph processes," *arXiv preprint arXiv:1712.00404*, 2017.

[43] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[44] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, "Time-varying graph signal reconstruction," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 870–883, Sep. 2017.

[45] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4-5, pp. 311–801, 2014.

[46] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Proc. 2017 51st Asilomar Conf. Signals, Syst., and Computers (ASILOMAR)*, Pacific Grove, CA, USA, Oct. 2017, pp. 1973–1976.

[47] ——, "Distributed diffusion adaptation over graph signals," in *Proc. 2018 IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, Calgary, AB, Canada, Apr. 2018, pp. 4129–4133.

[48] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," in *Proc. IEEE Data Science Workshop (DSW)*, Lausanne, Switzerland, Jun. 2018, pp. 1–5.

[49] A. H. Sayed, *Adaptive Filters*. John Wiley & Sons, 2008.

[50] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.

[51] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 2005 4th Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, Boise, ID, USA, 2005, pp. 63–70.

[52] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, no. 23, pp. 6303–6318, Dec. 2017.

[53] A. Sayed, *Fundamentals of adaptive filtering*. New York: Wiley, 2003.

[54] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3285–3300, Jul. 2015.

[55] J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, "Unsupervised diffusion-based LMS for node-specific parameter estimation over wireless sensor networks," in *Proc. 2016 IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, Shanghai, China, 2016, pp. 4159–4163.

[56] S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized clustering and linking by networked agents," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3526–3537, Jul. 2017.

[57] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, pp. 1–16, 2016.

[58] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *arXiv preprint arXiv:1408.5781*, 2014.

[59] "1981-2010 U.S. climate normals," [Online]. Available: https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/climate-normals/1981-2010-normals-data.

[60] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized online learning with kernels," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3240–3255, Jun. 2018.

[61] H. Pradhan, A. S. Bedi, A. Koppel, and K. Rajawat, "Exact nonparametric decentralized online optimization," in *Proc. 2018 IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Anaheim, CA, USA, 2018, pp. 643–647.