DISTRIBUTED DIFFUSION ADAPTATION OVER GRAPH SIGNALS

Roula Nassif [(1)], *Cédric Richard* [(2)], *Jie Chen* [(3)], *Ali H. Sayed* [(1)]

[(1)]Ecole Polytechnique Fédérale de Lausanne, Switzerland
[(2)]Université Côte d'Azur, France
[(3)]Northwestern Polytechnical University, Xi'an, China

ABSTRACT

Most works on graph signal processing assume static graph signals, which is a limitation even in comparison to traditional DSP techniques where signals are modeled as sequences that evolve over time. For broader applicability, it is necessary to develop techniques that are able to process dynamic or streaming data. Many earlier works on adaptive networks have addressed problems related to this challenge by developing effective strategies that are particularly well-suited to data streaming into graphs. We are thus faced with two paradigms: one where signals are modeled as static and sitting on the graph nodes, and another where signals are modeled as dynamic and streaming into the graph nodes. The objective of this work is to blend these concepts and propose diffusion strategies for adaptively learning from streaming graph signals.

I. INTRODUCTION

The massive deployment of distributed acquisition and signal processing systems, as well as the ubiquity of connected devices, is contributing to the development of graph signal processing. The potential applications are many and include, for example, vehicle networks, communication networks, and energy distribution networks. These connected systems consist of large sets of possibly autonomous agents linked together by a communication network. The formalization of graph signal processing as an extension of classical signal processing creates opportunities for extending and applying traditional techniques to the network domain. Results have been obtained on sampling [1]–[4], spectral analysis [5]–[8], and filtering [9]–[14]. Nevertheless, with few exceptions [4], [12]–[14], most of these results suffer from one serious limitation; they are largely focused on processing *static* graph signals (with respect to time) despite the natural anchoring in dynamic application contexts. Some works consider dynamic variations. However, existing studies are still far from developing a framework that truly parallels the full potential offered by even the simplest of DSP models, such as MA or FIR models and adaptive signal processing. These extensions require the ability to process data that evolve over time. Many earlier works on adaptive networks have actually addressed problems related to these specific challenges by developing strategies that can handle streaming data into graphs (see, e.g., [15]–[17] and the references therein). We are thus faced with two paradigms: one where signals are modeled as static and sitting on the graph nodes, and another where signals are modeled as dynamic and streaming into the graph nodes.

The objective of this paper is to blend these paradigms by combining concepts from adaptive networks and graph signal processing to propose a framework for the adaptation of graph signals. In the first part of the work, we propose an adaptive inference method for streaming graph signals based on the LMS strategy. Since this algorithm is centralized, we show how to distribute it over the graph nodes using the concept of diffusion adaptation [18]. In the second part of the work, we present the performance of the resulting graph diffusion-LMS algorithm in both the mean and mean-square sense, as well as its stability. The third part of the paper illustrates the effectiveness of the method on synthetic and real datasets.

Notation. We use normal font letters to denote scalars, boldface lowercase letters to denote column vectors, and boldface uppercase letters to denote matrices. We use the symbol $\otimes$ to denote Kronecker operation and the symbol $\text{Tr}(\cdot)$ to denote the trace operator. The symbol $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of its matrix argument. The m-th entry of a vector $\boldsymbol{x}$ is denoted by $[\boldsymbol{x}]_m$, the (m,n)-th entry of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{mn}$, the k-th row of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{k,\bullet}$.

II. PROBLEM FORMULATION AND CENTRALIZED LMS

Consider a graph $\mathcal{G}$ consisting of a set $\mathcal{N}$ of N nodes, labeled $k=1,\ldots,N$, and a set $\mathcal{E}$ of edges such that if node k is connected to node ℓ, then $(k,\ell)\in\mathcal{E}$. We are interested in the analysis of signals distributed on the graph $\mathcal{G}$, defined by $\boldsymbol{x}=[x_1,\ldots,x_N]^\top\in\mathbb{R}^N$, where x_k represents the value of the signal at node k. We denote the graph signal available at time i by $\boldsymbol{x}(i)$. We assume that $\mathcal{G}$ is endowed with a graph-shift operator defined by a matrix $\boldsymbol{S}$ of size $N\times N$ whose entry $s_{k\ell}$ is non-zero only if $k=\ell$ or $(k,\ell)\in\mathcal{E}$. Possible choices for the matrix $\boldsymbol{S}$ are the adjacency matrix [10] and the Laplacian matrix [9]. The operation $\boldsymbol{S}\boldsymbol{x}$ is called graph shift and is performed locally at each node k by linearly combining the samples x_ℓ from adjacent nodes, namely, $\sum_{\ell\in\mathcal{N}_k}s_{k\ell}x_\ell$, where $\mathcal{N}_k$ is the set of neighboring nodes of k including k itself. It is known that the non-zero entries of $\boldsymbol{S}^m$ correspond to pairs of nodes that can communicate in m hops [17], [19]. Therefore, the k-th entry of $\boldsymbol{S}^m\boldsymbol{x}$ can be calculated by node k using the signal samples within m-hops.

We consider linear shift-invariant graph filters defined by $\boldsymbol{z}(i)=\boldsymbol{H}\boldsymbol{x}(i)$ where:

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

with $\{h_m^o\}_{m=0}^{M-1}$ denoting the filter coefficients and M its order. One common filtering model for graph signals used in the literature [10], [14] is to assume that the graph vector $\boldsymbol{x}(i)$ is processed by (1) to generate the filtered graph vector $\boldsymbol{y}(i)$ as follows:

$$\boldsymbol{y}(i)=\sum_{m=0}^{M-1}h_m^o\boldsymbol{S}^m\boldsymbol{x}(i)+\boldsymbol{v}(i),\qquad i\geq0, \tag{2}$$

where $\boldsymbol{v}(i)=[v_1(i),\ldots,v_N(i)]^\top$ is an i.i.d. zero-mean noise signal with covariance matrix denoted by $\boldsymbol{R}_v=\text{diag}\{\sigma_{v,k}^2\}_{k=1}^N$ and independent of any other signal. Observe that model (2) assumes the instantaneous diffusion of information. That is, node k can process $y_k(i)$ at each time instant i by collecting samples $\{x_\ell(i)\}$ that are $(M-1)$-hops away from it. This instantaneous assumption

The work of C. Richard was partly supported by the ANR and the DGA, France, (ANR-13-ASTR-0030). The work of A. H. Sayed was also supported in part by NSF grants CCF-1524250 and ECCS-1407712.

is a serious *limitation* of such models. Specifically, the input-output relation (2) assumes that the powers of the shift operator, $\boldsymbol{S}^m$, are applied to the *same* graph vector $\boldsymbol{x}(i)$. This is a static assumption on the graph signal and there is no dynamics or time-evolution embedded into the model, as is customary in standard FIR or MA models in DSP filter analysis.

To enrich the model, and to introduce a temporal dimension into (2), we consider in this work the more general model:

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

In this way, the shift $\boldsymbol{S}^m$ is carried out in m time slots. Observe how input signal $\boldsymbol{x}(i)$ in (2) is now replaced by $\boldsymbol{x}(i-m)$ in (3). By retaining the following samples at each node ℓ:

$$x_\ell(i-1), [\boldsymbol{S}\boldsymbol{x}(i-2)]_\ell, \ldots, [\boldsymbol{S}^{M-2}\boldsymbol{x}(i-M+1)]_\ell, \qquad \forall \ell \in \mathcal{N}$$

only one graph shift is required at each time instant i.

In the following, we assume that the graph signal $\boldsymbol{x}(i)$ is a zero-mean wide-sense stationary process, i.e., $\mathbb{E}\boldsymbol{x}(i) = 0 \; \forall i$ and its auto-correlation sequence $\boldsymbol{R}_x(m) \triangleq \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i-m)\}$ is a function of the time lag m only. To estimate $\boldsymbol{h}^o \triangleq \text{col}\{h_0^o, \ldots, h_{M-1}^o\}$, we consider the mean-square-error criterion

$$J(\boldsymbol{h}) = \mathbb{E}\|\boldsymbol{y}(i) - \boldsymbol{X}_s(i)\boldsymbol{h}\|^2, \quad (4)$$

where $\boldsymbol{X}_s(i)$ is an $N \times M$ matrix given by:

$$\boldsymbol{X}_s(i) \triangleq [\boldsymbol{x}(i), \boldsymbol{S}\boldsymbol{x}(i-1), \ldots, \boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)]. \quad (5)$$

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

$$\boldsymbol{R}_X \boldsymbol{h}^o = \boldsymbol{r}_{Xy}, \quad (6)$$

where the $M \times M$ matrix $\boldsymbol{R}_X$ and the $M \times 1$ vector $\boldsymbol{r}_{Xy}$ are given by:

$$\boldsymbol{R}_X \triangleq \mathbb{E}\{\boldsymbol{X}_s^\top(i)\boldsymbol{X}_s(i)\} \quad \text{and} \quad \boldsymbol{r}_{Xy} \triangleq \mathbb{E}\{\boldsymbol{X}_s^\top(i)\boldsymbol{y}(i)\}. \quad (7)$$

The (m,n)-th entry of $\boldsymbol{R}_X$ is:

$$\begin{aligned}[\boldsymbol{R}_X]_{m,n} &= \mathbb{E}\{\boldsymbol{x}^\top(i-m+1)(\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1}\boldsymbol{x}(i-n+1)\} \\ &= \text{Tr}((\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1}\boldsymbol{R}_x(m-n)). \quad (8)\end{aligned}$$

Similarly, the m-th element of $\boldsymbol{r}_{Xy}$ is:

$$[\boldsymbol{r}_{Xy}]_m = \text{Tr}((\boldsymbol{S}^{m-1})^\top \boldsymbol{R}_{xy}(m-1)), \quad (9)$$

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

It is sufficient for the exposition in this work to assume that $\boldsymbol{R}_X$ is positive definite. In this case, $\boldsymbol{h}^o$ can be determined uniquely by solving (6). Alternatively, $\boldsymbol{h}^o$ can be sought iteratively using gradient descent:

$$\boldsymbol{h}^{\text{cent}}(i+1) = \boldsymbol{h}^{\text{cent}}(i) + \mu\big(\boldsymbol{r}_{Xy} - \boldsymbol{R}_X\boldsymbol{h}^{\text{cent}}(i)\big), \quad (10)$$

where $\mu > 0$ is a small step-size and where the superscript "cent" is used to refer to the centralized solution. Since the second order moments are rarely available beforehand, it is necessary to approximate them via instantaneous approximations, with different constructions leading to different adaptive algorithms. One of the simplest choices is to use the instantaneous approximations:

$$\boldsymbol{R}_X \approx \boldsymbol{X}_s^\top(i)\boldsymbol{X}_s(i) \qquad \boldsymbol{r}_{Xy} \approx \boldsymbol{X}_s^\top(i)\boldsymbol{y}(i). \quad (11)$$

Algorithm (10) then leads to the following LMS graph filter:

$$\boldsymbol{h}^{\text{cent}}(i+1) = \boldsymbol{h}^{\text{cent}}(i) + \mu\boldsymbol{X}_s^\top(i)\big(\boldsymbol{y}(i) - \boldsymbol{X}_s(i)\boldsymbol{h}^{\text{cent}}(i)\big). \quad (12)$$

The stochastic-gradient algorithm (12) is referred to as the *centralized graph-LMS* algorithm. In this centralized solution, each node at each iteration sends its samples $\{y_k(i), x_k(i)\}$ to a fusion center, which in turn updates $\boldsymbol{h}^{\text{cent}}(i)$ according to (12).

III. GRAPH DIFFUSION LMS SOLUTION

From (3), the sample $y_k(i)$ at node k is related to the graph signal $\boldsymbol{x}(i)$ according to:

$$y_k(i) = \sum_{m=0}^{M-1} h_m^o[\boldsymbol{S}^m\boldsymbol{x}(i-m)]_k + v_k(i), \qquad i \geq M-1. \quad (13)$$

Let $\boldsymbol{z}(i-m) \triangleq \boldsymbol{S}^m\boldsymbol{x}(i-m)$, and let $\boldsymbol{z}_k(i)$ be the $M \times 1$ vector:

$$\boldsymbol{z}_k(i) \triangleq \text{col}\,\{[\boldsymbol{z}(i)]_k, [\boldsymbol{z}(i-1)]_k, \ldots, [\boldsymbol{z}(i-M+1)]_k\}\,. \quad (14)$$

This vector aggregates the k-th entries of the vectors $\{\boldsymbol{z}(i-m)|m = 1, \ldots, M-1\}$. As explained in the previous section when introducing the linear model (3), by retaining the samples $\{\boldsymbol{S}^{m-1}\boldsymbol{x}(i-m)|m = 1, \ldots, M-1\}$ at each node $\ell \in \mathcal{N}$ from the previous iteration, this vector can be computed locally at node k and iteration i by collecting samples from the immediate neighborhood. Let $\boldsymbol{R}_{z,k}$ denote the $M \times M$ covariance matrix of $\boldsymbol{z}_k(i)$, namely, $\boldsymbol{R}_{z,k} \triangleq \mathbb{E}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}$. It can be verified that the (m,n)-th element of $\boldsymbol{R}_{z,k}$ is given by:

$$\begin{aligned}[\boldsymbol{R}_{z,k}]_{m,n} &= \mathbb{E}\{[\boldsymbol{z}(i-m+1)]_k[\boldsymbol{z}(i-n+1)]_k\} \\ &= \text{Tr}([\boldsymbol{S}^{m-1}]_{k,\bullet}^\top[\boldsymbol{S}^{n-1}]_{k,\bullet}\boldsymbol{R}_x(m-n)) \quad (15)\end{aligned}$$

where $\boldsymbol{R}_X = \sum_{k=1}^N \boldsymbol{R}_{z,k}$.

Relation (13) can be written alternatively as:

$$y_k(i) = \boldsymbol{z}_k^\top(i)\boldsymbol{h}^o + v_k(i), \qquad i \geq M-1. \quad (16)$$

The global cost (4) becomes:

$$J(\boldsymbol{h}) = \sum_{k=1}^N J_k(\boldsymbol{h}), \quad (17)$$

where $J_k(\boldsymbol{h})$ is the local cost at agent k given by:

$$J_k(\boldsymbol{h}) \triangleq \mathbb{E}|y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}|^2. \quad (18)$$

In the following, we explain how the parameter vector $\boldsymbol{h}^o$ can be estimated by each node k from the data $\{y_k(i), \boldsymbol{z}_k(i)\}$ in a distributed manner through local computations and communications among neighboring agents.

There are several distributed techniques that can be employed to minimize (17) in a fully decentralized manner [15]–[18], [20], [21]. Diffusion strategies [15]–[18] are attractive since they are scalable, robust, and enable continuous learning and adaptation in response to drifts in the location of the minimizer. There are two variations of the adaptive diffusion strategies, namely, the adapt-then-combine (ATC) and the combine-then-adapt (CTA). Let $\boldsymbol{h}_k(i)$ denote the estimate of $\boldsymbol{h}^o$ at node k and iteration i. The ATC diffusion LMS for minimizing (17) takes the following form at each agent k:

$$\text{(ATC)} \begin{cases} \boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\boldsymbol{z}_k(i)(y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)) \\ \boldsymbol{h}_k(i+1) = \displaystyle\sum_{\ell \in \mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1) \end{cases} \quad (19)$$

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

$$a_{\ell k} > 0, \quad \sum_{\ell=1}^N a_{\ell k} = 1, \quad \text{and} \quad a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k. \quad (20)$$

Conditions (20) imply that the matrix $\boldsymbol{A} \triangleq [a_{\ell k}]$ is left-stochastic. The ATC diffusion (19) consists of two steps. The first step is an adaptation step where agent k uses its own data to update its parameter vector $\boldsymbol{h}_k(i)$ to an intermediate value $\boldsymbol{\psi}_k(i+1)$. The second step is a combination step where the intermediate estimates $\{\boldsymbol{\psi}_\ell(i+1)\}$ from the neighborhood of agent k are combined through the combination coefficients $\{a_{\ell k}\}$ to obtain the updated weight estimate $\boldsymbol{h}_k(i+1)$. A similar implementation can be obtained by switching the order of the adaptation and combination steps. In the CTA implementation, agent k first combines the previous estimates of its neighbors to obtain the intermediate estimate $\boldsymbol{\psi}_k(i)$, and then updates this intermediate estimate using its data:

$$\text{(CTA)}\begin{cases}\boldsymbol{\psi}_k(i)=\displaystyle\sum_{\ell\in\mathcal{N}_k}a_{\ell k}\boldsymbol{h}_\ell(i)\\ \boldsymbol{h}_k(i+1)=\boldsymbol{\psi}_k(i)+\mu_k\boldsymbol{z}_k(i)(y_k(i)-\boldsymbol{z}_k^\top(i)\boldsymbol{\psi}_k(i)).\end{cases}\tag{21}$$

In terms of communication steps, the CTA form is advantageous since the exchange of data required to compute $\boldsymbol{z}_k(i)$ in (14) can be done simultaneously with the exchange of past estimates to perform the combination step in (21). The ATC form (19) requires two communication steps: i) exchanging data to compute $\boldsymbol{z}_k(i)$, and ii) exchanging $\{\boldsymbol{\psi}_\ell(i+1)\}$ to perform the combination step. In terms of steady-state performance, it is shown in [15, Sec. 7] that the ATC has a slight performance advantage over CTA.

IV. STOCHASTIC BEHAVIOR

In the following, we briefly present the main results obtained from the analysis in the mean-square-error sense of the adaptive algorithms in (19) and (21). Strategies (19) and (21) can be grouped into a unifying description by considering the following structure in terms of two sets of combination coefficients $\{a_{1,\ell k},a_{2,\ell k}\}$ [16], [18]:

$$\begin{cases}\boldsymbol{\phi}_k(i)=\displaystyle\sum_{\ell\in\mathcal{N}_k}a_{1,\ell k}\boldsymbol{h}_\ell(i)\\ \boldsymbol{\psi}_k(i+1)=\boldsymbol{\phi}_k(i)+\mu_k\boldsymbol{z}_k(i)(y_k(i)-\boldsymbol{z}_k^\top(i)\boldsymbol{\phi}_k(i))\\ \boldsymbol{h}_k(i+1)=\displaystyle\sum_{\ell\in\mathcal{N}_k}a_{2,\ell k}\boldsymbol{\psi}_\ell(i+1).\end{cases}\tag{22}$$

The quantities $\{\boldsymbol{\phi}_k(i),\boldsymbol{\psi}_k(i)\}$ are intermediate variables, while the nonnegative entries of the $N\times N$ matrices $\boldsymbol{A}_1\triangleq[a_{1,\ell k}]$ and $\boldsymbol{A}_2\triangleq[a_{2,\ell k}]$ are assumed to satisfy the same conditions (20). The choice $\boldsymbol{A}_1=\boldsymbol{I}_N$ and $\boldsymbol{A}_2=\boldsymbol{A}$ corresponds to the ATC form (19), while the choice $\boldsymbol{A}_1=\boldsymbol{A}$ and $\boldsymbol{A}_2=\boldsymbol{I}_N$ corresponds to the CTA form (21).

Using the data model (16) and following the same line of reasoning as in [18, Sec. 6], we find that the network error vector defined as $\widetilde{\boldsymbol{h}}(i)\triangleq\text{col}\{\boldsymbol{h}^o-\boldsymbol{h}_k(i)\}_{k=1}^N$ evolves according to:

$$\widetilde{\boldsymbol{h}}(i+1)=\boldsymbol{\mathcal{B}}(i)\widetilde{\boldsymbol{h}}(i)-\boldsymbol{\mathcal{A}}_2^\top\boldsymbol{\mathcal{M}}\boldsymbol{p}_{zv}(i),\tag{23}$$

where $\boldsymbol{\mathcal{B}}(i)\triangleq\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{MN}-\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_z(i))\boldsymbol{\mathcal{A}}_1^\top$, $\boldsymbol{\mathcal{A}}_1\triangleq\boldsymbol{A}_1\otimes\boldsymbol{I}_M$, $\boldsymbol{\mathcal{A}}_2\triangleq\boldsymbol{A}_2\otimes\boldsymbol{I}_M$, $\boldsymbol{\mathcal{M}}\triangleq\text{diag}\{\mu_k\}_{k=1}^N$, $\boldsymbol{p}_{zv}(i)\triangleq\text{col}\{\boldsymbol{z}_k(i)v_k(i)\}_{k=1}^N$, and $\boldsymbol{\mathcal{R}}_z(i)\triangleq\text{diag}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}_{k=1}^N$.

To perform the analysis, we introduce the following assumption.

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

This independence assumption is commonly used when analyzing the performance of diffusion LMS strategies [18, Sec. 6]. Although this assumption is not true in the current work, it is commonly used to analyze adaptive constructions since it allows to simplify the derivations without constraining the conclusions. There are several results in the adaptation literature that show that the performance results obtained under this independence assumption match well the actual performance when the step-sizes are *sufficiently small* [22, App. 24.A]. It is worth noting that several works studying the diffusion strategies under partial observation scenario and singular covariance matrices exist in the literature [16]. It is sufficient for the exposition in this article to present the results in the case of non-singular covariance matrices $\boldsymbol{R}_{z,k}$.

Taking expectations of both sides of (23), we obtain:

$$\mathbb{E}\widetilde{\boldsymbol{h}}(i+1)=\boldsymbol{\mathcal{B}}\,\mathbb{E}\widetilde{\boldsymbol{h}}(i),\tag{24}$$

where $\boldsymbol{\mathcal{B}}\triangleq\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{MN}-\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_z)\boldsymbol{\mathcal{A}}_1^\top$ and $\boldsymbol{\mathcal{R}}_z\triangleq\text{diag}\{\boldsymbol{R}_{z,k}\}_{k=1}^N$. We used the fact that $\mathbb{E}\boldsymbol{p}_{zv}(i)=0$, and $\widetilde{\boldsymbol{h}}(i)$ and $\boldsymbol{\mathcal{R}}_z(i)$ are independent of each other in view of Assumption 1. It turns out that all the estimates $\{\boldsymbol{h}_k(i)\}$ across the network converge asymptotically in the mean to the optimal solution $\boldsymbol{h}^o$ if $\boldsymbol{\mathcal{B}}$ is stable, i.e., $\rho(\boldsymbol{\mathcal{B}})<1$. The stability of $\boldsymbol{\mathcal{B}}$ is ensured if $\{\mu_k\}$ satisfy [18]:

$$0<\mu_k<2/\lambda_{\max}(\boldsymbol{R}_{z,k}).\tag{25}$$

We complete the performance analysis by characterizing the evolution of the mean-square-error quantity $\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}}$, where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix that we are free to choose. Using the independence Assumption 1, we obtain from (23):

$$\mathbb{E}\|\widetilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\Sigma}}=\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}'}+\mathbb{E}\|\boldsymbol{\mathcal{A}}_2^\top\boldsymbol{\mathcal{M}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}},\tag{26}$$

where

$$\boldsymbol{\Sigma}'\triangleq\mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\}.$$

The second term on the RHS of (26) can be written as [18]:

$$\mathbb{E}\|\boldsymbol{\mathcal{A}}_2^\top\boldsymbol{\mathcal{M}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}}=\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}})\tag{27}$$

where $\boldsymbol{\mathcal{G}}\triangleq\boldsymbol{\mathcal{A}}_2^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}_2$ and $\boldsymbol{\mathcal{S}}\triangleq\mathbb{E}\{\boldsymbol{p}_{zv}(i)\boldsymbol{p}_{zv}^\top(i)\}=\text{diag}\{\sigma^2_{v,k}\boldsymbol{R}_{z,k}\}_{k=1}^N$. Let $\boldsymbol{\sigma}\triangleq\text{vec}(\boldsymbol{\Sigma})$ denote the vector representation of $\boldsymbol{\Sigma}$ obtained by stacking the column entries of $\boldsymbol{\Sigma}$ on top of each other. Let $\boldsymbol{\sigma}'\triangleq\text{vec}(\boldsymbol{\Sigma}')$. The vector $\boldsymbol{\sigma}'$ can be related to $\boldsymbol{\sigma}$ according to $\boldsymbol{\sigma}'=\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}$ [18] where $\boldsymbol{\mathcal{F}}\triangleq\mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i)\otimes\boldsymbol{\mathcal{B}}^\top(i)\}$. The evaluation of $\boldsymbol{\mathcal{F}}$ requires knowledge of the fourth order moments of the graph signals, which are not available under the current assumptions. A common alternative is to use the approximation $\boldsymbol{\mathcal{F}}\approx\boldsymbol{\mathcal{B}}^\top\otimes\boldsymbol{\mathcal{B}}^\top$ for sufficiently small step-sizes [18]. As long as this approximation is reasonable, the stability of $\boldsymbol{\mathcal{F}}$ is ensured if $\rho(\boldsymbol{\mathcal{B}})<1$, i.e., if the step-sizes are chosen according to (25). The variance relation (26) can be written as:

$$\mathbb{E}\|\widetilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}}=\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}+[\text{vec}(\boldsymbol{\mathcal{G}})]^\top\boldsymbol{\sigma},\tag{28}$$

where the notation $\|\boldsymbol{x}\|^2_{\boldsymbol{\sigma}}$ is used to denote the same quantity $\|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}}=\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}$. The variance $\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}}$ converges if $\boldsymbol{\mathcal{F}}$ is stable:

$$\lim_{i\to\infty}\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}}=[\text{vec}(\boldsymbol{\mathcal{G}})]^\top(\boldsymbol{I}-\boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}.\tag{29}$$

Finally, the learning curve $\zeta(i)\triangleq\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}}$ can be obtained from recursion (28) [18]. Iterating (28) starting from $i=0$, we obtain:

$$\zeta(i+1)=\mathbb{E}\|\widetilde{\boldsymbol{h}}(0)\|^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}}+[\text{vec}(\boldsymbol{\mathcal{G}})]^\top\sum_{j=0}^{i}\boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}.\tag{30}$$

Comparing (30) at time instant $i+1$ and i, we obtain:

$$\zeta(i+1)=\zeta(i)+\mathbb{E}\|\widetilde{\boldsymbol{h}}(0)\|^2_{(\boldsymbol{\mathcal{F}}-\boldsymbol{I})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}}+[\text{vec}(\boldsymbol{\mathcal{G}})]^\top\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}.\tag{31}$$

V. EXPERIMENTAL RESULTS

V-A. Illustrative example

We tested the graph diffusion LMS strategies (19) and (21) on a random connected graph of $N=20$ nodes generated with an Erdös-Renyi topology shown in Fig. 1 (left). Using a similar construction as in [13], this graph was obtained by generating an $N\times N$ symmetric matrix $\boldsymbol{S}$ whose entries are governed by the Gaussian distribution $\mathcal{N}(0,1)$ and then thresholding edges to be between 1.2 and 1.8 in absolute value to yield an effective probability of an edge $p\approx0.07$. 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. We assumed that the graph signal

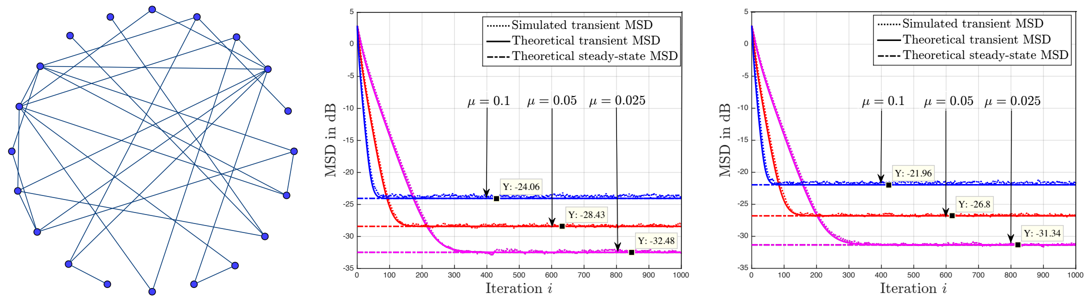

Fig. 1. *(Left)* Erdös-Renyi graph. *(Middle)* Network MSD: ATC algorithm (19). *(Right)* Network MSD: CTA algorithm (21).

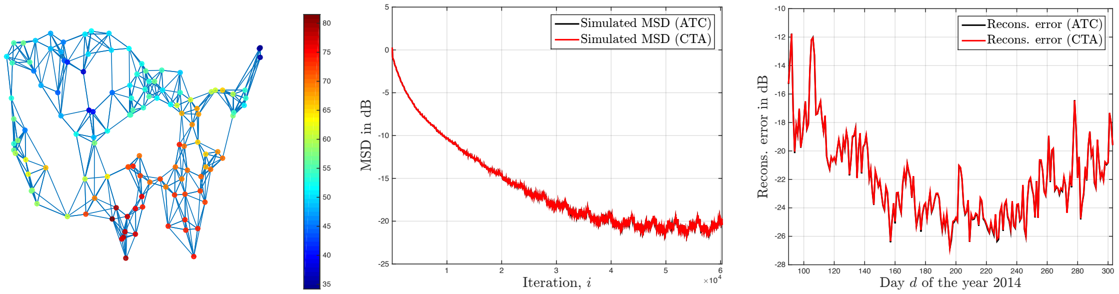

Fig. 2. *(Left)* Temperature measured by 125 weather stations across the US on May 1, 2003. *(Middle)* Network MSD w.r.t. the minimizer of (33) using data measured from 2003 to 2013. *(Right)* Reconstruction error over data measured from April 1, 2014 to October 31, 2014.

process $\boldsymbol{x}(i) = [x_1(i), \ldots, x_N(i)]^\top$ is i.i.d. Gaussian with zero mean and covariance matrix $\boldsymbol{R}_x$ where $\boldsymbol{R}_x$ was chosen as the solution of the Lyapunov equation $\boldsymbol{S}\boldsymbol{R}_x\boldsymbol{S}^\top - \boldsymbol{R}_x + \boldsymbol{I} = 0$. The noise $\boldsymbol{v}(i) = [v_1(i), \ldots, v_N(i)]^\top$ was zero-mean Gaussian with covariance $\boldsymbol{R}_v = \text{diag}\{\sigma^2_{v,k}\}_{k=1}^N$ where the variances $\sigma^2_{v,k}$ were randomly generated from the uniform distribution $\mathcal{U}(0.1, 0.15)$. The filter degree M was set to 3 and the coefficients h^o_m of this filter were randomly generated from the uniform distribution $\mathcal{U}(0, 1)$.

We ran algorithms (19) and (21) by setting $a_{\ell k} = 1/|\mathcal{N}_k|$ for $\ell \in \mathcal{N}_k$, where $|\cdot|$ denotes the cardinality of its entry. We used a constant step-size μ for all agents. The results were averaged over 200 Monte-Carlo runs. The network MSD performance of the ATC algorithm (19) (Fig. 1 (middle)) and the CTA algorithm (21) (Fig. 1 (right)), given by $\frac{1}{N}\sum_{k=1}^N \mathbb{E}\|\boldsymbol{h}^o - \boldsymbol{h}_k(i)\|^2$, is shown for three different values of μ. For each case, we report the theoretical transient MSD (31), the theoretical steady-state MSD (29), and the simulated MSD. We observe that the theoretical curves match well the actual performance and that the ATC algorithm (19) outperforms the CTA strategy (21).

V-B. Prediction in temperature sensor networks

We tested algorithms (19) and (21) on a temperature dataset corresponding to a collection of daily average temperature measurements taken from 2003 to 2013 (a total of $D = 4018$ days) at $N = 125$ weather stations located around the continental United States [23]. These stations were represented with a directed 6-nearest neighbor distance graph, in which every sensor corresponds to a vertex and is connected to 6 closest sensors [24] by directed edges weighted by:

$$s_{k\ell} = \frac{e^{-d^2_{k\ell}}}{\sqrt{\sum_{m\in\mathcal{N}_k} e^{-d^2_{km}} \sum_{n\in\mathcal{N}_\ell} e^{-d^2_{\ell n}}}}, \qquad \ell \in \mathcal{N}_k, \qquad (32)$$

where $d_{k\ell}$ denotes the geodesical distance between the k-th and the ℓ-th sensors – See Fig. 2 (left). Let $\boldsymbol{y}_d$ denote the graph signal at day d, $d = 1, \ldots, D$. Given the graph signal $\boldsymbol{y}_{d+1}$ at day $d+1$ and the graph signals at previous days $\{\boldsymbol{y}_{d-m}, m = 0, \ldots, M-1\}$, the objective is to estimate the filter coefficients $\{h^o_m\}$ such that the empirical cost [13]:

$$J_e(\boldsymbol{h}) = \frac{1}{D-M}\sum_{d=M}^{D-1}\left\|\boldsymbol{y}_{d+1} - \sum_{m=0}^{M-1} h_m \boldsymbol{S}^m \boldsymbol{y}_{d-m}\right\|^2, \qquad (33)$$

is minimized. The filter degree M was set to 5. We ran algorithms (19) and (21) by setting $\mu_k = 10^{-5}$ for all k. We assumed symmetric combination coefficients when performing the combination step. We set $a_{\ell k} = 1/\max\{|\bar{\mathcal{N}}_k|, |\bar{\mathcal{N}}_\ell|\}$ if $k \neq \ell$ and $\ell \in \bar{\mathcal{N}}_k^-$, $a_{kk} = 1 - \sum_{\ell\in\bar{\mathcal{N}}_k^-} a_{\ell k}$, and $a_{\ell k} = 0$ if $\ell \notin \bar{\mathcal{N}}_k$, where $\bar{\mathcal{N}}_k \triangleq \{m, s_{km} \neq 0$ and $s_{mk} \neq 0\}$ and $\bar{\mathcal{N}}_k^- \triangleq \bar{\mathcal{N}}_k \setminus k$. We ran algorithms (19) and (21) for $i \geq M$ over $n_e = 15$ epochs where we set in the model (3) $\boldsymbol{y}(i) = \boldsymbol{y}_{d+1}$ for $i = 0, \ldots, n_e D - 1$, where $d = i \mod D$, with mod the modulo operator, and $\boldsymbol{x}(i-m) = \boldsymbol{y}(i-m-1)$. Note that, the implementation in this section is the so-called *empirical* stochastic gradient descent implementation where the number of samples is finite [25]. In Fig. 2 (middle) we illustrate the network MSD performance with respect to the minimizer of the cost (33) and in Fig. 2 (right) we illustrate the reconstruction error defined as

$$\frac{1}{N}\sum_{k=1}^N \frac{|[\boldsymbol{y}_{d+1} - \sum_{m=0}^{M-1} h_{k,m}(\infty)\boldsymbol{S}^m\boldsymbol{y}_{d-m}]_k|^2}{|[\boldsymbol{y}_{d+1}]_k|^2} \qquad (34)$$

where $h_{k,m}(\infty)$ is the steady-state (last) estimate of h^o_m reached by node k, applied to a new set of daily temperature data recorded from April 1, 2014 to October 31, 2014 (a total of $N = 214$ days). The same performance is observed for the ATC (19) and CTA (21).

VI. CONCLUSION

Most prior literature on graph signal processing is concerned with graph signals that are static with respect to time. In the current work, we developed diffusion strategies for adaptive graph signal processing, which allow the model to handle dynamic scenarios. We presented briefly the performance of the strategies and validated the theoretical models by experiments.

VII. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] B. Girault, P. Gonçalves, and E. Fleury, “Translation on graphs: An isometric shift operator,” *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2416–2420, Dec. 2015.

[6] N. Perraudin and P. Vandergheynst, “Stationary signal processing on graphs,” *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3462–3477, Jul. 2017.

[7] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, “Stationary graph processes: Nonparametric spectral estimation,” in *Proc. IEEE Sens. Array Multichannel Signal Process. Workshop (SAM)*, Jul. 2016, pp. 1–5.

[8] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Stationary graph processes and spectral estimation,” *Available as arXiv:1603.04667*, Mar. 2016.

[9] 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.

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

[11] 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.

[12] 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.

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

[14] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, “A graph diffusion LMS strategy for adaptive graph signal processing,” in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Oct. 2017.

[15] 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.

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

[17] A. H. Sayed, “Adaptation, learning, and optimization over networks,” *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[18] A. H. Sayed, “Diffusion adaptation over networks,” in *Academic Press Library in Signal Processing*, Elsevier, Ed., vol. 3, pp. 322–454. 2014.

[19] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129 – 150, 2011.

[20] 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.

[21] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Distributed stochastic subgradient projection algorithms for convex optimization,” *J. Optim. Theory Appl.*, vol. 147, no. 3, pp. 516–545, 2010.

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

[23] J. H. Lawrimore, M. J. Menne, B. E. Gleason, C. N. Williams, D. B. Wuertz, and R. S. Vose, Global Historical Climatology Network–Monthly (GHCN-M). NOAA National Climatic Data Center, Available: ftp:/ftp.ncdc.noaa.gov/pub/data/gsod.

[24] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Frequency analysis,” *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.

[25] K. Yuan, B. Ying, S. Vlaski, and A. H. Sayed, “Stochastic gradient descent with finite samples sizes,” in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, Salerno, Italy, Sept. 2016, pp. 1–6.