A Graph Diffusion LMS Strategy for Adaptive Graph Signal Processing

Roula Nassif[†], Cédric Richard[†], Jie Chen[‡], Ali H. Sayed[§]
[†]Université Côte d'Azur, France
Email: {roula.nassif,cedric.richard}@oca.eu
[‡] Northwestern Polytechnical University, Xi'an, China
Email: dr.jie.chen@ieee.org
[§] Ecole Polytechnique Fédérale de Lausanne, Switzerland
Email: ali.sayed@epfl.ch

***Abstract*—Graph signal processing allows the generalization of DSP concepts to the graph domain. However, most works assume graph signals that are static with respect to time, which is a limitation even in comparison to classical DSP formulations where signals are generally sequences that evolve over time. Several earlier works on adaptive networks have addressed problems involving streaming data over graphs by developing effective learning strategies that are well-suited to dynamic data scenarios, in a manner that generalizes adaptive signal processing concepts to the graph domain. The objective of this paper is to blend concepts from adaptive networks and graph signal processing to propose new useful tools for adaptive graph signal processing.**

I. Introduction

Signals collected in biology, physics, geology, engineering, and other domains are often characterized by complex structures that can sometimes be captured by graphs or networks. These graphical representations require appropriate signal processing and analysis techniques. Some of these techniques are being developed within the framework of graph signal processing [1], [2]. This formalism extends conventional DSP concepts from structured domains (such as time) to unstructured domains (such as graphs). The extensions create opportunities for applying signal processing techniques to signals represented over graphs. Possible applications include sampling [3]–[5], spectral analysis [6]–[9], and filtering [1], [2], [10]. Unfortunately, most works on graph signal processing deal with *static* signals, i.e., signals that need not evolve with time, despite the dynamic nature of various applications used to motivate this formalism. Only a few recent studies have started to deal with dynamic graph signals [11]–[14]. Several earlier works on adaptive networks have developed learning strategies that deal precisely with problems involving data streaming into graphs over time. These works have extended several concepts from single-agent adaptive signal processing to graphs and have developed effective strategies for filtering, estimation, and detection from dynamic graph signals – see, e.g., the overviews in [15]–[18] and the references therein.

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 supported in part by NSF grant CCF–1524250.

The aim of this paper is to blend concepts from adaptive networks [16]–[18] and graph signal processing [1], [2] to propose new tools for adaptive graph signal processing. In the first part of the work, we propose a *centralized* adaptive method for streaming graph signals based on the LMS strategy. In the second part, we show how to distribute it across the graph nodes using the concept of diffusion adaptation over networks [19]. In the third part of the work, we present the performance of the resulting algorithm in the mean and mean-square error sense, as well as its stability. Finally, we illustrate the effectiveness of the method and the accuracy of the models on synthetic data.

Notation. Normal font letters denote scalars. Boldface lowercase letters denote vectors. Boldface uppercase letters denote matrices. The symbol $\otimes$ denotes the Kronecker product operator. The symbol $\mathrm{Tr}(\cdot)$ denotes the trace operator. The symbol $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of its matrix argument. The $M \times M$ identity matrix is denoted by $\boldsymbol{I}_M$. 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 $[\boldsymbol{X}]_{mn}$, and the k-th row of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{k,\bullet}$.

II. Adaptive Graph filtering

We consider a graph $\mathcal{G}$ consisting of a set of nodes $\mathcal{N}$ of cardinality N, and a set of edges $\mathcal{E}$ such that if nodes k and ℓ are connected, then $(k,\ell) \in \mathcal{E}$. We are interested in the analysis of signals distributed on the graph $\mathcal{G}$, defined by $\boldsymbol{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$. Each element in $\boldsymbol{x}$ corresponds to a signal at a node of $\mathcal{G}$. We denote the graph signal that is available at time i by $\boldsymbol{x}(i)$. We associate with $\mathcal{G}$ a translation (or shift) operator denoted by a matrix $\boldsymbol{S}$ of size $N \times N$ whose component $s_{k\ell}$ can be non-zero only if $k = \ell$ or $(k,\ell) \in \mathcal{E}$. Let $\mathcal{N}_k$ denote the set of neighboring nodes of k, i.e., all nodes connected to k by an edge. The operation $\boldsymbol{S}\boldsymbol{x}$ is called graph shift and is performed locally at each node by linearly combining the samples x_ℓ from neighboring nodes, namely, $[\boldsymbol{S}\boldsymbol{x}]_k = \sum_{\ell \in \mathcal{N}_k} s_{k\ell} x_\ell$. In this paper, we are interested in the estimation problem where, given a desired graph signal $\boldsymbol{y}(i)$ observed at the output of an unknown system at time i, in response to a graph signal denoted by $\boldsymbol{x}(i)$, we would

like to estimate the parameters of a filter that minimize a cost measure. We focus on linear shift-invariant graph filters:

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

where $\{h_m\}_{m=0}^{M-1}$ are the filter coefficients and M is its order [2]. A graph filter $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ applied to a graph signal $\boldsymbol{x}(i) \in \mathbb{R}^N$ produces an output $\boldsymbol{z}(i) \in \mathbb{R}^N$ which is again a graph signal given by $\boldsymbol{z}(i) = \boldsymbol{H}\boldsymbol{x}(i)$. We assume that, at each time instant i, the desired graph signal $\boldsymbol{y}(i)$ is related to the input graph signal $\boldsymbol{x}(i)$ via the linear model:

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

where $\boldsymbol{v}(i) \triangleq [v_1(i), \ldots, v_N(i)]^\top$ denotes an $N\times 1$ i.i.d. zero-mean measurement noise independent of any other signal and with covariance matrix $\boldsymbol{R}_v = \text{diag}\{\sigma_{v,k}^2\}_{k=1}^N$, while $\boldsymbol{h}^o \triangleq \text{col}\{h_1^o, \ldots, h_{M-1}^o\}$ is the vector of graph filter coefficients to be estimated. The graph signal $\boldsymbol{x}(i)$ is assumed zero-mean with covariance matrix $\boldsymbol{R}_x \triangleq \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i)\} > 0$. Let $\boldsymbol{Z}(i)$ denote the $N \times M$ matrix defined as:

$$\boldsymbol{Z}(i) \triangleq \Big[\boldsymbol{x}(i), \boldsymbol{S}\boldsymbol{x}(i), \boldsymbol{S}^2\boldsymbol{x}(i), \ldots, \boldsymbol{S}^{M-1}\boldsymbol{x}(i)\Big] \tag{3}$$

The optimal filter coefficients $\boldsymbol{h}^o$ can be found by solving the following minimization problem:

$$\boldsymbol{h}^o = \arg\min_{\boldsymbol{h}} \Big\{ J(\boldsymbol{h}) \triangleq \mathbb{E}\|\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}\|^2 \Big\}. \tag{4}$$

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

$$\boldsymbol{R}_Z \boldsymbol{h}^o = \boldsymbol{r}_{Zy}, \tag{5}$$

where $\boldsymbol{R}_Z$ is the $M \times M$ matrix $\boldsymbol{R}_Z \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)\}$ and $\boldsymbol{r}_{Zy}$ is the $M \times 1$ vector $\boldsymbol{r}_{Zy} \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{y}(i)\}$. It can be verified that the (m,n)-th entry of $\boldsymbol{R}_Z$ is given by:

$$\begin{aligned}[\boldsymbol{R}_Z]_{mn} &= \mathbb{E}\{\boldsymbol{x}^\top(i)(\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1}\boldsymbol{x}(i)\} \\ &= \text{Tr}((\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1}\boldsymbol{R}_x),\end{aligned} \tag{6}$$

and that the m-th entry of $\boldsymbol{r}_{Zy}$ is given by:

$$[\boldsymbol{r}_{Zy}]_m = \mathbb{E}\{\boldsymbol{x}^\top(i)(\boldsymbol{S}^{m-1})^\top \boldsymbol{y}(i)\} = \text{Tr}((\boldsymbol{S}^{m-1})^\top \boldsymbol{R}_{xy}), \tag{7}$$

where $\boldsymbol{R}_{xy} \triangleq \mathbb{E}\{\boldsymbol{y}(i)\boldsymbol{x}^\top(i)\}$. It is clear from (5) that when $\boldsymbol{R}_Z$ is positive definite, $\boldsymbol{h}^o$ can be determined uniquely. If, on the other hand, $\boldsymbol{R}_Z$ is singular, then we can use its pseudo-inverse to recover the minimum-norm solution of (4). It is sufficient for the exposition of this work to assume that $\boldsymbol{R}_Z$ is positive definite.

Alternatively, $\boldsymbol{h}^o$ can be estimated iteratively using gradient descent:

$$\boldsymbol{h}^{\text{cent}}(i+1) = \boldsymbol{h}^{\text{cent}}(i) + \mu(\boldsymbol{r}_{Xy} - \boldsymbol{R}_Z \boldsymbol{h}^{\text{cent}}(i)) \tag{8}$$

where $\mu > 0$ is a small step-size. This solution requires the centralized processor to have knowledge of the moments $\boldsymbol{R}_Z$ and $\boldsymbol{r}_{Zy}$, which are rarely available beforehand. To address this lack of information, we approximate the unavailable moments based on the realizations $\{\boldsymbol{y}(i), \boldsymbol{x}(i)\}$ as follows:

$$\boldsymbol{R}_Z \approx \boldsymbol{Z}^\top(i)\boldsymbol{Z}(i), \qquad \boldsymbol{r}_{Zy} \approx \boldsymbol{Z}^\top(i)\boldsymbol{y}(i). \tag{9}$$

Algorithm (8) leads to the following graph-LMS algorithm:

$$\boldsymbol{h}^{\text{cent}}(i+1) = \boldsymbol{h}^{\text{cent}}(i) + \mu \boldsymbol{Z}^\top(i)\left(\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}^{\text{cent}}(i)\right). \tag{10}$$

In this centralized solution, at each time instant i, each node in the network sends its data $\{y_k(i), x_k(i)\}$ to the central processor, which in turn updates $\boldsymbol{h}^{\text{cent}}(i)$ according to (10).

III. Graph diffusion LMS

From (2), the signal $y_k(i)$ at vertex k is related to the graph signal $\boldsymbol{x}(i)$ according to:

$$y_k(i) = \sum_{m=0}^{M-1} h_m^o [\boldsymbol{S}^m \boldsymbol{x}(i)]_k + v_k(i), \qquad i \geq 0. \tag{11}$$

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

$$\boldsymbol{z}_k(i) \triangleq \text{col}\Big\{ \big[\boldsymbol{z}^{(0)}(i)\big]_k, \ldots, \big[\boldsymbol{z}^{(M-1)}(i)\big]_k \Big\}. \tag{12}$$

This vector aggregates the k-th entries of all the $\{\boldsymbol{z}^{(m)}(i)\}$. It can be computed locally at node k in $M-1$ hops since the non-zero entries of $\boldsymbol{S}^m$ correspond to pairs of nodes that can communicate in m hops [20]. Let $\boldsymbol{R}_{z,k}$ denote the $M \times M$ covariance matrix of $\boldsymbol{z}_k(i)$, namely, $\boldsymbol{R}_{z,k} \triangleq \mathbb{E}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}$. The (m,n)-th element of $\boldsymbol{R}_{z,k}$ is given by:

$$\begin{aligned}[\boldsymbol{R}_{z,k}]_{mn} &= \mathbb{E}\{\big[\boldsymbol{z}^{(m-1)}(i)\big]_k \big[\boldsymbol{z}^{(n-1)}(i)\big]_k\} \\ &= \text{Tr}([\boldsymbol{S}^{m-1}]_{k,\bullet}^\top [\boldsymbol{S}^{n-1}]_{k,\bullet} \boldsymbol{R}_x).\end{aligned} \tag{13}$$

Relation (11) can be written alternatively as:

$$y_k(i) = \boldsymbol{z}_k^\top(i)\boldsymbol{h}^o + v_k(i), \qquad i \geq 0. \tag{14}$$

The global cost function $J(\boldsymbol{h})$ in (4) becomes:

$$J(\boldsymbol{h}) = \sum_{k=1}^N J_k(\boldsymbol{h}), \tag{15}$$

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

$$J_k(\boldsymbol{h}) \triangleq \mathbb{E}|y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}|^2. \tag{16}$$

We are therefore able to rewrite in (15) the cost function $J(\boldsymbol{h})$ as the aggregate sum of local costs $J_k(\boldsymbol{h})$. This form is amenable to distributed optimization. We now explain how the parameter vector $\boldsymbol{h}$ can be estimated from the data $\{y_k(i), \boldsymbol{z}_k(i)\}$ using distributed adaptive optimization. There are several distributed techniques that can be employed for minimizing (15). We shall use the ATC form of the diffusion LMS algorithm [19] due to its established enhanced performance in adaptive scenarios. We denote by $\boldsymbol{h}_k(i)$ the estimate of $\boldsymbol{h}^o$ at agent k and iteration i. Starting from an initial

condition $\boldsymbol{h}_k(0)$, at every time instant i, the ATC diffusion LMS strategy takes the following form at each agent k:

$$\begin{cases} \boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu\, \boldsymbol{z}_k(i)\left[y_k(i) - \boldsymbol{z}_k^{\top}(i)\boldsymbol{h}_k(i)\right], \\ \boldsymbol{h}_k(i+1) = \sum\limits_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1). \end{cases} \tag{17}$$

where $\mu_k > 0$ is a local step-size parameter and $\{a_{\ell k}\}$ are the non-negative entries of a left-stochastic matrix $\boldsymbol{A}$ satisfying:

$$a_{\ell k} \geq 0, \quad \sum_{\ell\in\mathcal{N}_k} a_{\ell k} = 1, \quad \text{and} \quad a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k. \tag{18}$$

The ATC diffusion (17) consists of two steps. The first step is an adaptation step where agent k uses the data $y_k(i), \boldsymbol{z}_k(i)$ to update its parameter vector $\boldsymbol{h}_k(i)$ to an intermediate value $\boldsymbol{\psi}_k(i+1)$ where $\boldsymbol{z}_k(i)$ is computed by node k in $M-1$ hops. The second step is a combination step where the intermediate estimate $\{\boldsymbol{\psi}_\ell(i+1)\}$ from the neighborhood of agent k are combined through the combination coefficients $\{a_{\ell k}\}$ to obtain the updated weight estimate $\boldsymbol{h}_k(i+1)$.

IV. Stochastic behavior

In this section, we briefly present the performance in the mean and mean-square-error sense of algorithm (17). Let $\widetilde{\boldsymbol{h}}(i)$ denote the network error vector defined as:

$$\widetilde{\boldsymbol{h}}(i) \triangleq \operatorname{col}\left\{\boldsymbol{h}^o - \boldsymbol{h}_k(i)\right\}_{k=1}^{N}. \tag{19}$$

Using the data model (14) and following the same line of reasoning as in [19, Sec. 6], we find that the network error vector evolves according to:

$$\widetilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}(i)\widetilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{g}(i), \tag{20}$$

where $\boldsymbol{\mathcal{A}} \triangleq \boldsymbol{A}\otimes\boldsymbol{I}_M$, $\boldsymbol{\mathcal{M}} \triangleq \operatorname{diag}\{\mu_1\boldsymbol{I}_M, \ldots, \mu_N\boldsymbol{I}_M\}$

$$\boldsymbol{\mathcal{B}}(i) \triangleq \boldsymbol{\mathcal{A}}^{\top}(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_z(i)), \tag{21}$$

$$\boldsymbol{g}(i) \triangleq \operatorname{col}\{\boldsymbol{z}_1(i)v_1(i), \ldots, \boldsymbol{z}_N(i)v_N(i)\}, \tag{22}$$

$$\boldsymbol{\mathcal{R}}_z(i) \triangleq \operatorname{diag}\{\boldsymbol{z}_1(i)\boldsymbol{z}_1^{\top}(i), \ldots, \boldsymbol{z}_N(i)\boldsymbol{z}_N^{\top}(i)\}. \tag{23}$$

To perform the analysis, we introduce the following assumption on the vectors $\{\boldsymbol{z}_k(i)\}$.

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

This assumption means that $\boldsymbol{z}_k(i)$ is independent of $\widetilde{\boldsymbol{h}}_\ell(j)$ for all ℓ and $j \leq i$. This assumption is commonly used when analyzing adaptive constructions since it allows to simplify the derivations without constraining the conclusions [21]. It is worth noting that several works studying the performance of diffusion strategies under partial observation scenario and singular covariance matrices exist in the literature [17], [18]. In this work, it is sufficient to focus on the case of non-singular covariance matrices.

Taking expectation of both sides of (20), and using Assumption 1 and the fact that $\mathbb{E}\boldsymbol{g}(i) = 0$, we obtain:

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

where

$$\boldsymbol{\mathcal{B}} \triangleq \boldsymbol{\mathcal{A}}^{\top}(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_z), \tag{25}$$

$$\boldsymbol{\mathcal{R}}_z \triangleq \operatorname{diag}\{\boldsymbol{R}_{z,1}, \ldots, \boldsymbol{R}_{z,N}\}. \tag{26}$$

Thus, the estimates $\{\boldsymbol{h}_k(i)\}$ generated by algorithm (17) converge in the mean to the optimal solution $\boldsymbol{h}^o$ if $\boldsymbol{\mathcal{B}}$ is stable, namely, $\rho(\boldsymbol{\mathcal{B}}) < 1$. The stability of $\boldsymbol{\mathcal{B}}$ is ensured if the step-sizes are chosen to satisfy [19]:

$$0 < \mu_k < \frac{2}{\lambda_{\max}\left(\boldsymbol{R}_{z,k}\right)}, \quad k = 1, \ldots, N. \tag{27}$$

The mean-square-error behavior of algorithm (17) can be characterized by studying the evolution of $\mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\Sigma}}^2$ for any positive semi-definite matrix $\boldsymbol{\Sigma}$ that we are free to choose. From recursion (20) and using the independence Assumption 1, the weighted variance evolves according to:

$$\mathbb{E}\,\|\widetilde{\boldsymbol{h}}(i+1)\|_{\boldsymbol{\Sigma}}^2 = \mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\Sigma}'}^2 + \operatorname{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}) \tag{28}$$

where

$$\boldsymbol{\Sigma}' \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^{\top}(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\}, \tag{29}$$

$$\boldsymbol{\mathcal{G}} \triangleq \mathbb{E}\{\boldsymbol{g}(i)\boldsymbol{g}^{\top}(i)\} = \operatorname{diag}\{\sigma_{v,k}^2\boldsymbol{R}_{z,k}\}_{k=1}^{N}, \tag{30}$$

where we used the spatially independence assumption of the zero-mean noise $v_k(i)$. Let $\boldsymbol{\sigma} \triangleq \operatorname{vec}(\boldsymbol{\Sigma})$ denote the vector representation of $\boldsymbol{\Sigma}$ obtained by stacking the columns entries of $\boldsymbol{\Sigma}$ on top of each other. Let $\boldsymbol{\sigma} \triangleq \operatorname{vec}(\boldsymbol{\Sigma}')$. The vector $\boldsymbol{\sigma}'$ can be related to $\boldsymbol{\sigma}$ according to $\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma}$ where $\boldsymbol{\mathcal{F}} = \mathbb{E}\{\boldsymbol{\mathcal{B}}^{\top}(i)\otimes\boldsymbol{\mathcal{B}}^{\top}(i)\}$ [19]. 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 where terms involving higher-order powers of the step-sizes are ignored [19]. Under this approximation, the stability of $\boldsymbol{\mathcal{F}}$ is ensured if $\rho(\boldsymbol{\mathcal{B}}) < 1$, i.e., if the step-sizes are chosen according to (27). The variance relation (28) can be written alternatively as:

$$\mathbb{E}\,\|\widetilde{\boldsymbol{h}}(i+1)\|_{\boldsymbol{\sigma}}^2 = \mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}^2 + [\operatorname{vec}(\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}})]^{\top}\boldsymbol{\sigma} \tag{31}$$

where the notation $\|\boldsymbol{x}\|_{\boldsymbol{\sigma}}^2$ is used to denote the same quantity $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}}^2 = \boldsymbol{x}^{\top}\boldsymbol{\Sigma}\boldsymbol{X}$. If $\boldsymbol{\mathcal{F}}$ is stable, the variance recursion converges [19]:

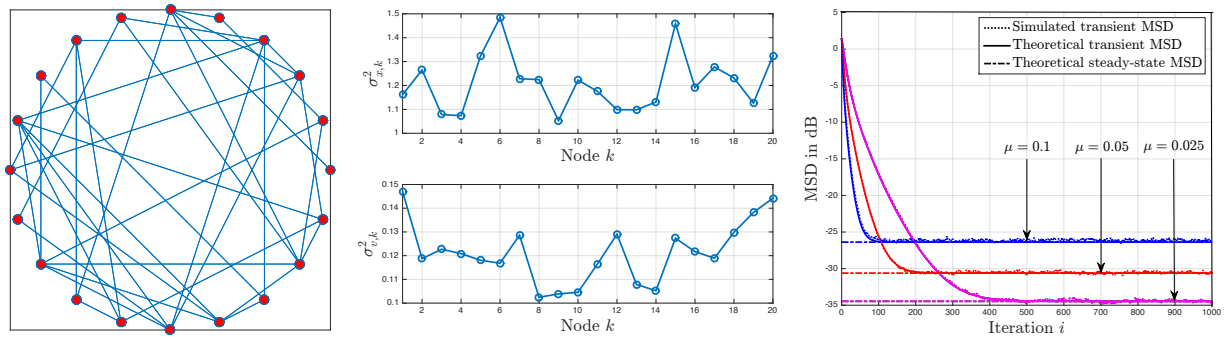
$$\lim_{i\to\infty} \mathbb{E}\,\|\widetilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\sigma}}^2 = [\operatorname{vec}(\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}})]^{\top}(\boldsymbol{I} - \boldsymbol{\mathcal{F}})^{-1}\boldsymbol{\sigma}. \tag{32}$$

The learning curve $\zeta(i) \triangleq \mathbb{E}\|\widetilde{\boldsymbol{h}}(i)\|_{\boldsymbol{\sigma}}^2$ is obtained from (31) as follows. Iterating (31) starting from $i = 0$ leads to:

$$\zeta(i+1) = \mathbb{E}\|\widetilde{\boldsymbol{h}}(0)\|_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}}^2 + [\operatorname{vec}(\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}})]^{\top}\sum_{j=0}^{i}\boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}. \tag{33}$$

Comparing (33) at time instant i and $i+1$, we obtain:

$$\begin{aligned}\zeta(i+1) = \zeta(i) + \mathbb{E}\|\widetilde{\boldsymbol{h}}(0)\|_{(\boldsymbol{\mathcal{F}}-\boldsymbol{I})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}}^2 + \\ [\operatorname{vec}(\boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}})]^{\top}\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}.\end{aligned} \tag{34}$$



(a) Erdös-Renyi graph (b) Signal and noise variances (c) Network MSD

Fig. 1. Simulation results

V. Simulation Results

We tested the ATC graph diffusion LMS algorithm (17) on a random connected graph of $N = 20$ nodes generated with an Erdös-Renyi topology shown in Fig. 1(a). Using a similar construction as in [11], 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. 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 its largest eigenvalue. The graph signal $\boldsymbol{x}(i) = [x_1(i), \ldots, x_N(i)]^\top$ was generated according to a Gaussian distribution with zero-mean and diagonal covariance matrix $\boldsymbol{R}_x = \text{diag}\{\sigma^2_{x,k}\}_{k=1}^N$. The noise $\boldsymbol{v}(i) = [v_1(i), \ldots, v_N(i)]$ was zero-mean Gaussian with covariance $\boldsymbol{R}_v = \text{diag}\{\sigma^2_{v,k}\}_{k=1}^N$. The variances $\sigma^2_{x,k}$ and $\sigma^2_{v,k}$ are shown in Fig. 1(b). The filter order M was set to 3 and the filter coefficients h^o_m were randomly generated from the uniform distribution $\mathcal{U}(0, 1)$. We ran algorithm (17) by setting $a_{\ell k} = \frac{1}{|\mathcal{N}_k|}$ 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. Figure 1(c) shows the network mean-square-deviation (MSD) performance of algorithm (17), given by $\frac{1}{N}\sum_{k=1}^N \mathbb{E}\|\boldsymbol{h}^o - \boldsymbol{h}_k(i)\|^2$, for three different values of μ. For each case, we report the theoretical transient MSD, the theoretical steady-state MSD, and the simulated MSD. We observe that the theoretical curves match well the actual performance.

VI. Conclusion

This work deals with adaptive graph signal processing. Based on the diffusion formalism, we presented a distributed adaptive algorithm for the estimation of the coefficients of linear shift-invariant graph filters from streaming signals. We analyzed the performance of the algorithm and validated the theoretical models by experiments.

References

[1] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, April 2013.

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

[4] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, July 2016.

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

[6] B. Girault, P. Gonçalves, and E. Fleury, "Translation on graphs: an isometric shift operator," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2416–2420, December 2015.

[7] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *CoRR*, vol. abs/1601.02522, 2016.

[8] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Stationary graph processes: Nonparametric spectral estimation," in *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, July 2016, pp. 1–5.

[9] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *CoRR*, vol. abs/1603.04667, 2016.

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

[11] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, April 2017.

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

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

[14] R. Nassif, C. Richard, J. Chen, R. Couillet, and P. Borgnat, "Filtrage LMS sur graphe. Algorithme et analyse.," in *Actes du 26e Colloque GRETSI sur le Traitement du Signal et des Images*, Nice, France, Sep. 2017.

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

[16] 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 Processing Magazine*, vol. 30, no. 3, pp. 155–171, May 2013.

[17] A. H. Sayed, "Adaptation, learning, and optimization over networks," in *Foundations and Trends in Machine Learning*, vol. 7, pp. 311–801. NOW Publishers, Boston-Delft, Jul. 2014.

[18] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.

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

[20] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129 – 150, 2011.

[21] A. H. Sayed, *Adaptive Filters*, Wiley, NY, 2008.