Efficient Estimation of Graph Signals With Adaptive Sampling

Mohammad Javad Ahmadi , Reza Arablouei, and Reza Abdolee

Abstract—We propose two new least mean squares (LMS)-based algorithms for adaptive estimation of graph signals that improve the convergence speed of the LMS algorithm while preserving its low computational complexity. The first algorithm, named extended least mean squares (ELMS), extends the LMS algorithm by virtue of reusing the signal vectors of previous iterations alongside the signal available at the current iteration. Utilizing the previous signal vectors accelerates the convergence of the ELMS algorithm at the expense of higher steady-state error compared to the LMS algorithm. To further improve the performance, we propose the fast ELMS (FELMS) algorithm in which the influence of the signal vectors of previous iterations is controlled by optimizing the gradient of the mean-square deviation (GMSD). The FELMS algorithm converges faster than the ELMS algorithm and has steady-state errors comparable to that of the LMS algorithm. We analyze the mean-square performance of ELMS and FELMS algorithms theoretically and derive the respective convergence conditions as well as the predicted MSD values. In addition, we present an adaptive sampling strategy in which the sampling probability of each node is changed according to the GMSD of the node. Computer simulations using both synthetic and real data validate the theoretical results and demonstrate the merits of the proposed algorithms.

Index Terms—Adaptive learning, graph signal processing, least mean squares, mean-square analysis, adaptive sampling.

I. INTRODUCTION

SIGNAL processing over networks has received a great amount of attention in various areas related to distributed learning, optimization, and control [1]–[11]. Many involving applications in social networks, sensor networks, vehicular networks or biological networks use graph vertices to model the signals of interest over networks [12]–[15]. The goal of graph signal processing (GSP) is to apply different concepts of classical discrete signal processing to signals defined over an irregular discrete domain in which different units (vertices) are connected within a graph. Each element of a graph signal is attributed

to a node of the graph and the edges of the graph specify connections between the elements. Two main approaches are commonly adopted in the GSP framework, which are based on utilizing either the graph Laplacian matrix [15] or the graph adjacency matrix [14], [16]. Although these approaches lead to different processing and analysis of the graph signals due to dissimilar foundations, both are similarly useful in various signal processing applications. Spectral analysis is an important processing tool in graph signal processing, which introduces the concept of graph Fourier transform (GFT). The GFT can be realized by projecting the graph signal onto the eigenbasis of either the adjacency matrix [15], [17], [18] or the Laplacian matrix [14], [19].

Sampling interpolation is one of the most crucial tasks in the GSP framework. The theory of sampling on graphs is studied in [17], which is extended in [20] and, recently, in [19], [21]–[24]. There are two major categories of reconstruction methods for graph signals: iterative [22], [25]–[28] and single shot [19], [21], [29], [30]. Some studies have also presented frame-based approaches for reconstructing signals from subsets of samples [17], [21], [22].

Graph signals have a time-varying nature in many applications such as communication networks, biological neural networks, and transportation networks. Therefore, the relevant GSP algorithms should be able to learn and track time-varying graph signals using a designed sampling set. Recently, the GSP framework is applied to solve specific learning tasks such as semi-supervised classification on graphs [31]–[33] and graph dictionary learning [34], [35]. A least mean squares (LMS) algorithm for estimating graph signals is proposed in [36], which is applied to the distributed setting in [37]. Reference [38] presents a kernel-based method for reconstructing time-varying signals over graphs with changing topologies. A distributed method is introduced in [26] for tracking of band-limited graph signals. The work of [39] extends the classical adaptive estimation algorithms, the recursive least squares (RLS) and LMS, to estimate graph signals. The RLS algorithm has the benefit of high convergence speed but has relatively high computational complexity. On the contrary, the LMS algorithm for graph signals imposes a lower computational load at the expense of slower convergence.

In this paper, we propose two new adaptive graph signal processing algorithms for improving the performance of the adaptive algorithms presented in [39]. The proposed algorithms inherit the low computational complexity of the LMS algorithm while enhancing its convergence speed. The first algorithm, called extended least mean squares (ELMS), utilizes the signal

Manuscript received September 20, 2019; revised March 29, 2020 and June 10, 2020; accepted June 10, 2020. Date of publication June 17, 2020; date of current version June 29, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Vincent Y. F. Tan. (Corresponding author: Mohammad Javad Ahmadi.)

Mohammad Javad Ahmadi is with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: ahmadi@ee.bilkent.edu.tr).

Reza Arablouei is with the Commonwealth Scientific and Industrial Research Organisation, Pullenvale, QLD 4069, Australia (e-mail: reza.arablouei@csiro.au).

Reza Abdolee is with the Department of Computer Science, California State University, Channel Islands, Camarillo, CA 93012 USA (e-mail: rabdolee@csu.edu).

Digital Object Identifier 10.1109/TSP.2020.3002607

vectors of a few previous iterations as well as that of the current iteration. Although this algorithm converges faster than the LMS algorithm, its steady-state error is higher. The second proposed algorithm, called fast ELMS (FELMS), further improves the convergence speed of the ELMS algorithm by minimizing the instantaneous gradient of the mean square deviation (GMSD) at each iteration. In addition, unlike the ELMS algorithm which has higher steady-state error than LMS, the steady-state error of FELMS is on par with that of LMS. We analyze the mean-square performance of the proposed algorithms and derive theoretical equations for transient and steady-state mean square deviation (MSD). Moreover, we propose an adaptive sampling strategy that minimizes the sampling frequency without incurring any degradation in performance.

This paper is organized as follows. In Section II, we present some GSP tools and, in Section III, some background on estimation algorithms for graph signals. In Section IV, we describe our proposed ELMS and FELMS algorithms. We discuss the computational complexity of the proposed algorithms in Section IV-C and analyze their performance theoretically in Section V. In Section VI, we propose an efficient strategy for adaptive sampling to be used in conjunction with the proposed algorithms. We provide some simulation results in Section VII and conclude the paper in Section VIII.

II. GRAPH SIGNAL PROCESSING TOOLS

Let us consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising N nodes indexed in $\mathcal{V} = \{1, 2, \dots, N\}$ and connected to each other according to the set of weighted edges $\mathcal{E} = \{a_{ij}\}_{i,j \in \mathcal{V}}$ where $a_{ij} > 0$ if node j and node i are connected and $a_{ij} = 0$ otherwise. The adjacency matrix $\mathbf{A}$ is the collection of all edge weights such that a_{ij} is its (i, j) th entry. The Laplacian matrix is defined as $\mathbf{L} = \text{diag}(\mathbf{1}^T \mathbf{A}) - \mathbf{A}$ where $\mathbf{1}$ denotes the all-ones column vector and the diagonal matrix $\text{diag}(\mathbf{1}^T \mathbf{A})$ is the degree matrix with the node degrees on its diagonal. If the graph is undirected, the Laplacian matrix is symmetric and positive semi-definite and admits the eigendecomposition $\mathbf{L} = \mathbf{V} \mathbf{\Gamma} \mathbf{V}^H$ where $\mathbf{V}$ has the eigenvectors of $\mathbf{L}$ as its columns and $\mathbf{\Gamma}$ is a diagonal matrix with the eigenvalues of $\mathbf{L}$ being its entries.

In $\mathcal{G}$, the signal $\mathbf{x}$ maps the vertex set to the set of complex numbers, i.e., $\mathcal{V} \rightarrow \mathbb{C}$. The signal $\mathbf{s}$ is obtained by applying GFT to $\mathbf{x}$. It is then projected onto the eigenbasis $\mathbf{V}$, i.e.,

$$\mathbf{s} = \mathbf{V}^H \mathbf{x}. \quad (1)$$

The support of $\mathbf{s}$ is denoted as

$$\mathcal{F} = \{i \in \mathcal{V} | s_i \neq 0\} \quad (2)$$

and the cardinality of $\mathcal{F}$, i.e., $|\mathcal{F}|$, defines the bandwidth of the graph signal $\mathbf{x}$.

Considering a subset of frequency indices $\mathcal{F} \subseteq \mathcal{V}$, an ideal bandpass filtering operator can be introduced as

$$\mathbf{H}_f = \mathbf{V}_f \mathbf{V}_f^H, \quad (3)$$

where the columns of $\mathbf{V}_f$ are those of $\mathbf{V}$ whose positional indexes are in $\mathcal{F}$.

III. ADAPTIVE ESTIMATION OF GRAPH SIGNALS

The purpose of this section is to review the LMS, RLS, best linear unbiased estimator (BLUE), and least squares (LS) algorithms proposed in the literature for estimating graph signals.

A. Least Mean Squares Estimation of Graph Signals

Let us consider a signal $\mathbf{x}^o = [x_1^o, \dots, x_N^o]^T \in \mathcal{R}^{N \times 1}$ over the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ modeled as

$$\mathbf{x}^o = \mathbf{V}_f \mathbf{s}^o \quad (4)$$

where $\mathbf{s}^o \in \mathcal{R}^{|\mathcal{F}| \times 1}$ denotes the vector of GFT coefficients of the frequency support of the graph signal $\mathbf{x}^o$. In addition, we consider a linear model relating an observable perturbed signal at time instance n , denoted by $\mathbf{d}(n)$, to $\mathbf{x}^o$ as

$$\mathbf{d}(n) = \mathbf{G}_s(n)(\mathbf{x}^o + \mathbf{v}(n)) = \mathbf{G}_s(n)(\mathbf{V}_f \mathbf{s}^o + \mathbf{v}(n)) \quad (5)$$

where $\mathbf{G}_s(n) = \text{diag}\{g_1(n), \dots, g_N(n)\} \in \mathcal{R}^{N \times N}$. The variable $g_i(n)$ is a binary sampling coefficient, which is equal to 1 if $i \in S(n)$ and 0 otherwise where $S(n)$ represents the random sampling set at time n . In addition, $\mathbf{v}(n) \in \mathcal{R}^{N \times 1}$ is the zero-mean spatially and temporally independent observation noise at time instance n with the covariance matrix $\mathbf{C}_v = \text{diag}\{\sigma_1^2, \dots, \sigma_N^2\}$. As $\mathbf{v}(n)$ is a wide-sense stationary random process, an optimal least-mean-square-error estimate of $\mathbf{s}^o$ can be obtained by solving the following optimization problem at any time instance n :

$$\min_{\mathbf{s}} E\{\|\mathbf{G}_s(n)(\mathbf{d}(n) - \mathbf{V}_f \mathbf{s})\|^2\} \quad (6)$$

where $E\{\cdot\}$ denotes the expectation operator. Approximating the expected value (mean square error) in (6) with its instantaneous value at time instance n and using the stochastic gradient-descent method while taking $\mathbf{s}(n)$ as the most recent estimate, the LMS algorithm calculates an estimate of $\mathbf{s}^o$ in an iterative manner as

$$\mathbf{s}(n+1) = \mathbf{s}(n) + \mu \mathbf{V}_f^H \mathbf{G}_s(n)(\mathbf{d}(n) - \mathbf{V}_f \mathbf{s}(n)) \quad (7)$$

where μ is the step-size parameter. Since $\mathbf{V}_f$ has orthonormal columns, estimations of $\mathbf{s}^o$ and $\mathbf{x}^o$ are equivalent from the mean-square-error perspective. Using (4) and (3), (7) can be rewritten with respect to the estimation of the graph signal $\mathbf{x}^o$ as

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \mu \mathbf{H}_f \mathbf{G}_s(n)(\mathbf{d}(n) - \mathbf{x}(n)) \quad (8)$$

where $\mathbf{x}(n) = \mathbf{V}_f \mathbf{s}(n)$ is the estimate of $\mathbf{x}^o$ at time instance (iteration) n .

B. Recursive Least Squares Estimation of Graph Signals

In comparison with the LMS algorithm, the RLS algorithm generally converges faster but at the expense of higher computational complexity as it exploits the information of all past and present time instances [40]. To compute the RLS estimate for graph signals, the following optimization problem is solved:

$$\min_{\mathbf{s}} \sum_{l=1}^n \beta^{n-l} \|\mathbf{G}_s(l)(\mathbf{d}(l) - \mathbf{V}_f \mathbf{s})\|_{\mathbf{C}_v^{-1}}^2 + \beta^n \|\mathbf{s}\|_{\mathbf{\Gamma}}^2 \quad (9)$$

where $0 \ll \beta \leq 1$ is the exponential forgetting factor and $\mathbf{\Pi} = \epsilon \mathbf{I} \succeq \mathbf{0}$ is the regularization matrix with $\epsilon > 0$ being a small positive number. Considering (4), the solution of (9) yields the estimate of $\mathbf{x}^o$ at time instance n as

$$\hat{\mathbf{x}}(n) = \mathbf{V}_f \mathbf{s}(n) = \mathbf{V}_f \mathbf{\Omega}^{-1}(n) \boldsymbol{\psi}(n) \quad (10)$$

where

$$\mathbf{\Omega}(n) = \sum_{l=1}^n \beta^{n-l} \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{C}_v^{-1} \mathbf{V}_f + \beta^n \mathbf{\Pi} \quad (11)$$

$$\boldsymbol{\psi}(n) = \sum_{l=1}^n \beta^{n-l} \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{C}_v^{-1} \mathbf{d}(l). \quad (12)$$

To make the computations more efficient, (11) and (12) can be rewritten in the following recursive forms

$$\mathbf{\Omega}(n) = \beta \mathbf{\Omega}(n-1) + \mathbf{V}_f^H \mathbf{G}_s(n) \mathbf{C}_v^{-1} \mathbf{V}_f \quad (13)$$

$$\boldsymbol{\psi}(n) = \beta \boldsymbol{\psi}(n-1) + \mathbf{V}_f^H \mathbf{G}_s(n) \mathbf{C}_v^{-1} \mathbf{d}(n) \quad (14)$$

where $\mathbf{\Omega}(0) = \mathbf{\Pi}$ and $\boldsymbol{\psi}(0) = \mathbf{0}$.

C. Least Squares and Best Linear Unbiased Estimators

In the previous subsections, we described two iterative algorithms, which require a number of iterations to reach their desired estimation accuracy. Here, we describe two non-iterative algorithms used for graph signal estimation. Considering the noisy signal model of (5), the LS and BLUE estimators are obtained by solving the following optimization problem:

$$\min_{\mathbf{s}} \sum_{l=1}^n \|\mathbf{G}_s(l)(\mathbf{d}(l) - \mathbf{V}_f \mathbf{s})\|_{\Theta}^2. \quad (15)$$

Setting $\Theta = \mathbf{I}$, gives the LS estimator of graph signal as [29], [30]:

$$\hat{\mathbf{x}}(n) = \mathbf{V}_f \left(\sum_{l=1}^n \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{V}_f \right)^{-1} \sum_{l=1}^n \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{d}(l). \quad (16)$$

By setting $\Theta = \mathbf{C}_v^{-1}$, the BLUE, which is in fact a weighted LS estimator, is also expressed as [23]

$$\hat{\mathbf{x}}(n) = \mathbf{V}_f \left(\sum_{l=1}^n \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{C}_v^{-1} \mathbf{V}_f \right)^{-1} \sum_{l=1}^n \mathbf{V}_f^H \mathbf{G}_s(l) \mathbf{C}_v^{-1} \mathbf{d}(l). \quad (17)$$

Note that these estimators require to access the entire available data at once and unlike the LMS and RLS algorithms cannot be implemented in an online fashion to process the streams of data becoming available over time.

IV. PROPOSED ADAPTIVE ALGORITHMS FOR ESTIMATING GRAPH SIGNALS

In this section, we present two algorithms that improve the performance of the LMS algorithm. Both algorithms use the signal samples of previous moments as well as the current sample to enhance the convergence speed of the LMS. The proposed algorithms are studied in the following sub-sections in detail.

A. Extended LMS for Graph Signals

Let us recall the noisy observed signal $\mathbf{d}(n)$ defined in (5) and express it as

$$\mathbf{d}(n) = \mathbf{G}_s(n) \mathbf{V}_f \mathbf{s}^o + \mathbf{G}_s(n) \mathbf{v}(n). \quad (18)$$

where $\mathbf{G}_s(n) = \text{diag}\{g_1(n), \dots, g_N(n)\}$ and $g_i(n)$ is a binary sampling coefficient being 1 if i is in the sampling set $S(n)$ and 0 otherwise. From (18), we observe that the entries of $\mathbf{d}(n)$ with corresponding zero sampling coefficients are equal to zero. Thus, we have $\mathbf{d}(n) = \mathbf{G}_s(n) \mathbf{d}(n)$. Consequently, (18) can be rearranged as

$$\mathbf{G}_s(n) \mathbf{v}(n) = \mathbf{G}_s(n) (\mathbf{d}(n) - \mathbf{V}_f \mathbf{s}^o). \quad (19)$$

The above equation is for time instant n . It can also be written for $K-1$ previous time instances as

$$\mathbf{G}_s(n-j) \mathbf{v}(n-j) = \mathbf{G}_s(n-j) (\mathbf{d}(n-j) - \mathbf{V}_f \mathbf{s}^o) \quad \forall j = 1, \dots, K-1. \quad (20)$$

In the same vein as the LMS algorithm, the optimal signal can be estimated by solving any of the following optimization problems corresponding to K different instances:

$$\min_{\mathbf{s}} E \{ \|\mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{V}_f \mathbf{s})\|^2 \} \quad \forall j = 0, \dots, K-1. \quad (21)$$

However, to enhance the estimation performance, we propose to solve the following optimization problem that is constructed by summing the cost functions in (21)

$$\min_{\mathbf{s}} E \left\{ \sum_{j=0}^{K-1} \|\mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{V}_f \mathbf{s})\|^2 \right\}. \quad (22)$$

Our proposed ELMS algorithm solves (22) via the gradient-descent method, i.e., using the following update equation

$$\begin{aligned} \mathbf{x}(n+1) &= \mathbf{x}(n) \\ &+ \mu \mathbf{H}_f \sum_{j=0}^{K-1} \mathbf{G}_s(n-j) (\mathbf{d}(n-j) - \mathbf{x}(n)). \end{aligned} \quad (23)$$

Although the ELMS algorithm requires more memory to store the reused past data, it does not need any additional multiplication compared to the LMS algorithm. This will be discussed in Section IV-C.

B. Fast ELMS for Graph Signals

The proposed ELMS algorithm suffers from higher values of steady-state error in comparison with the LMS algorithm (see Appendix A). To tackle this problem and further increase the convergence speed, we develop the fast ELMS (FELMS) algorithm by generalizing (23) as

$$\begin{aligned} \mathbf{x}(n+1) &= \mathbf{x}(n) + \mu \mathbf{H}_f \mathbf{G}_s(n) (\mathbf{d}(n) - \mathbf{x}(n)) \\ &+ \alpha(n) \mu \mathbf{H}_f \sum_{j=1}^{K-1} \mathbf{G}_s(n-j) (\mathbf{d}(n-j) - \mathbf{x}(n)) \end{aligned} \quad (24)$$

where $\alpha(n)$ is a scalar through which we can control the influence of the previous signal vectors in each update. If $\alpha(n)$ is set to zero, FELMS reduces to the LMS algorithm. Setting $\alpha(n)$ to one also converts FELMS to ELMS. Therefore, FELMS can get the best of both worlds by varying $\alpha(n)$ appropriately. The parameter $\alpha(n)$ should be tuned according to the occasion. In the transient state where high convergence speed is needed, $\alpha(n)$ can take a non-zero value so that the FELMS algorithm takes advantage of the good convergence speed of the ELMS algorithm. In the steady state when the algorithm has converged, $\alpha(n)$ should be set to zero so the FELMS algorithm benefits from the low steady-state error of the LMS algorithm.

Instead of setting $\alpha(n) = 1$ in the transient state, we aim at finding an optimal value to further improve the convergence speed. For this purpose, we calculate the gradient mean-square deviation (GMSD) and compute the optimal $\alpha(n)$ by minimizing the GMSD at time n . The GMSD is defined as

$$\Delta_\alpha(n) = E\|\tilde{\mathbf{x}}(n+1)\|_{\Psi(n)}^2 - E\|\tilde{\mathbf{x}}(n)\|_{\Psi(n)}^2 \quad (25)$$

where $\tilde{\mathbf{x}}(n) = \mathbf{x}^o - \mathbf{x}(n)$ denotes the weight error vector, $\|\mathbf{t}\|_{\mathbf{A}}^2 = \mathbf{t}^H \mathbf{A} \mathbf{t}$, and $\Psi(n) = \mathbf{G}_s^H(n) \mathbf{G}_s(n)$ is a diagonal matrix that eliminates the effect of unselected nodes on the GMSD. The GMSD can be a good indicator of the convergence status of the adaptive algorithm at each iteration. If it has a small negative value, $\mathbf{x}(n+1)$ is a better estimate of $\mathbf{x}^o$ than $\mathbf{x}(n)$. Therefore, minimizing GMSD at each iteration can enhance the convergence speed of the algorithm.

To compute the GMSD defined in (25), we state the update equation associate with the weight error vector by subtracting both sides of (24) from $\mathbf{x}^o$ as

$$\tilde{\mathbf{x}}(n+1) = \tilde{\mathbf{x}}(n) - \mathbf{m}(n) \quad (26)$$

where

$$\mathbf{m}(n) = \mathbf{m}_1(n) + \alpha(n) \mathbf{m}_2(n) \quad (27)$$

$$\mathbf{m}_1(n) = \mu \mathbf{H}_f \mathbf{G}_s(n) (\mathbf{d}(n) - \mathbf{x}(n)) \quad (28)$$

$$\mathbf{m}_2(n) = \mu \mathbf{H}_f \sum_{j=1}^{K-1} \mathbf{G}_s(n-j) (\mathbf{d}(n-j) - \mathbf{x}(n)). \quad (29)$$

Calculating the $\Psi(n)$ -weighted norm followed by applying the expectation to both sides of (26) gives

$$E\|\tilde{\mathbf{x}}(n+1)\|_{\Psi(n)}^2 = E\|\tilde{\mathbf{x}}(n)\|_{\Psi(n)}^2 + f_2(n) - f_1(n) - f_1^*(n) \quad (30)$$

where

$$f_1(n) = E\{\tilde{\mathbf{x}}^H(n) \Psi(n) \mathbf{m}(n)\}, \quad (31)$$

$$f_2(n) = E\{\mathbf{m}^H(n) \Psi(n) \mathbf{m}(n)\} = \alpha^2(n) f_{21} + \alpha(n) (f_{22} + f_{22}^*) + f_{23}, \quad (32)$$

$$f_{21} = E\{\mathbf{m}_2^H(n) \Psi(n) \mathbf{m}_2(n)\}, \quad (33)$$

$$f_{22} = E\{\mathbf{m}_1^H(n) \Psi(n) \mathbf{m}_2(n)\}, \quad (34)$$

$$f_{23} = E\{\mathbf{m}_1^H(n) \Psi(n) \mathbf{m}_1(n)\}. \quad (35)$$

Obviously, the GMSD can be calculated if $f_1(n)$ and $f_2(n)$ in (30) are given. However, because the weight error vector $\tilde{\mathbf{x}}$ is unknown and the expected values are not available, $f_1(n)$ and $f_2(n)$ cannot be calculated directly. To eliminate $\tilde{\mathbf{x}}$ from $f_1(n)$, we use $\mathbf{G}_s(n) \mathbf{x}^o = \mathbf{d}(n) - \mathbf{G}_s(n) \mathbf{v}(n)$ as in (5) to rewrite (31) as

$$\begin{aligned} f_1(n) &= E\{\tilde{\mathbf{x}}^H(n) \Psi(n) \mathbf{m}(n)\} \\ &= E\{(\mathbf{d}(n) - \mathbf{G}_s(n) \mathbf{x}(n) - \mathbf{G}_s(n) \mathbf{v}(n))^H \mathbf{G}_s(n) \mathbf{m}(n)\} \\ &= E\{(\mathbf{d}(n) - \mathbf{G}_s(n) \mathbf{x}(n))^H \mathbf{G}_s(n) \mathbf{m}(n)\} \\ &\quad - E\{\mathbf{v}^H(n) \mathbf{G}_s^H(n) \mathbf{G}_s(n) \mathbf{m}(n)\}. \end{aligned} \quad (36)$$

Considering that the noise, $\mathbf{v}(n)$, is independent of all other stochastic processes, (36) can be expressed as

$$f_1(n) = f_{11} + \alpha(n) f_{12} \quad (37)$$

where

$$f_{11} = E\{(\mathbf{d}(n) - \mathbf{G}_s(n) \mathbf{x}(n))^H \mathbf{G}_s(n) \mathbf{m}_1(n)\} - \mu E\{\text{tr}\{\mathbf{C}_v \mathbf{G}_s(n) \mathbf{H}_f \mathbf{G}_s(n)\}\} \quad (38)$$

$$f_{12} = E\{(\mathbf{d}(n) - \mathbf{G}_s(n) \mathbf{x}(n))^H \mathbf{G}_s(n) \mathbf{m}_2(n)\}. \quad (39)$$

From (30), (32), and (37), the GMSD is calculated as

$$\begin{aligned} \Delta_\alpha(n) &= E\|\tilde{\mathbf{x}}(n+1)\|_{\Psi(n)}^2 - E\|\tilde{\mathbf{x}}(n)\|_{\Psi(n)}^2 \\ &= \alpha^2(n) f_{21} + \alpha(n) f_3 + f_4 \end{aligned} \quad (40)$$

where

$$f_3 = f_{22} + f_{22}^* - f_{12}^* - f_{12}$$

$$f_4 = f_{23} - f_{11} - f_{11}^*.$$

Although GMSD in (40) does not depend on the unknown $\tilde{\mathbf{x}}$, we cannot calculate it at each iteration as some expected values are needed for computing f_{21} , f_3 , and f_4 . To circumvent this problem, we use the instantaneous values instead [41], [42]. Now, since the GMSD in (40) is a function of $\alpha(n)$, the optimal $\alpha(n)$ can be found by solving the following optimization problem:

$$\alpha_o(n) = \min_{\alpha(n)} \Delta_\alpha(n). \quad (41)$$

To obtain the optimum $\alpha(n)$ in time instance n , we set the derivative of $\Delta_\alpha(n)$ with respect to $\alpha(n)$ to zero and obtain

$$\alpha_o(n) = -\frac{f_3}{2f_{21}}. \quad (42)$$

Substituting $\alpha_o(n)$ into (40) gives the optimal value of GMSD, $\Delta_{\alpha_o}(n)$.

It will be shown in Section V-A [see (65)] that the MSD comprises two terms: the variance term that depends on the noise covariance matrix and the bias term that consists of the initial MSD. The bias term has a large initial value (depending on the initial weight vector) and converges to zero when the algorithm converges to the steady state. However, the variance term can be nonzero at the steady state. At the transient state when the bias term still has a large value, using $\alpha(n)$ leads to

the maximal convergence speed due to the maximum reduction made in the bias term. At the steady state when the bias term is zero, using $\alpha_o(n)$ may deteriorate the performance as the algorithm tends to learn the patterns within the noise. Therefore, not only choosing optimum $\alpha_o(n)$ in the steady-state may not decrease the MSD, but it may even increase it. Therefore, we set $\alpha(n) = 0$ at the steady state to reach the low steady-state MSD of the LMS algorithm, i.e., we set

$$\alpha(n) = \begin{cases} \alpha_o(n) & \Delta_{\alpha_o}(n) < \theta \\ 0 & \text{otherwise} \end{cases} \quad (43)$$

where $\theta < 0$ is a threshold parameter for which we propose a suitable value in Section V-B. However, in order to curtail possible fluctuations in the learning curve due to using the instantaneous values in place of the expected ones for calculating $\alpha_o(n)$, we propose to update $\alpha(n)$ as

$$\alpha(n) = \gamma(n)\alpha_o(n) \quad (44)$$

where $\gamma(n)$ is updated as

$$\gamma(n) = \begin{cases} 1 & \Delta_{\alpha_o}(n) < \theta \\ \eta\gamma(n-1) & \text{otherwise} \end{cases} \quad (45)$$

where $\gamma(0) = 1$ and $\eta < 1$ is a damping coefficient.

C. Computational Complexity

Considering the update equations of the LMS, ELMS, and FELMS algorithms in (8), (23), and (24), we can write a general update equation for all these algorithms as

$$\tilde{\mathbf{x}}(n+1) = \tilde{\mathbf{x}}(n) - \mu \mathbf{V}_f \mathbf{V}_f^H (\mathbf{b}_1(n) + \mathbf{b}_2(n)) \quad (46)$$

where

$$\mathbf{b}_1(n) = \mathbf{G}_s(n)(\mathbf{d}(n) - \mathbf{x}(n)) \quad (47)$$

$$\mathbf{b}_2(n) = \alpha(n) \sum_{j=1}^{K-1} \mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{x}(n)) \quad (48)$$

$$\alpha(n) = \begin{cases} 0 & \text{LMS} \\ 1 & \text{ELMS} \\ \alpha(n) \text{ in (44)} & \text{FELMS} \end{cases} \quad (49)$$

In (46), $\mathbf{b}_1(n) + \mathbf{b}_2(n) \in \mathcal{R}^{N \times 1}$ and $\mathbf{V}_f(n) \in \mathcal{R}^{N \times |\mathcal{F}|}$. Therefore, the order of complexity for calculating $\mu \mathbf{V}_f \mathbf{V}_f^H (\mathbf{b}_1(n) + \mathbf{b}_2(n))$ is $\mathcal{O}(N|\mathcal{F}|)$. Note that $\mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{x}(n)) \quad \forall j = 0, 1, \dots, K-1$ does not need any multiplications because $\mathbf{G}_s(n-j)$ is a diagonal matrix with elements 0 and 1. Therefore, the LMS and ELMS algorithms do not need any multiplications for calculating $\mathbf{b}_1(n) + \mathbf{b}_2(n)$. However, the FELMS algorithm needs some multiplications for calculating $\alpha(n)$ and multiplying it to the vector $\sum_{j=1}^{K-1} \mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{x}(n))$. To calculate $\alpha(n)$, we need to compute $\Delta_{\alpha_o}(n)$ and $\alpha_o(n)$ through the scalars f_{21} , f_{22} , f_{23} , f_{11} , and f_{12} defined in (33)–(35), (38), and (39). Since the number of multiplications required for all these scalars are the same, we can count the number of multiplications required to calculate $f_{23} = \mathbf{m}_1^H(n) \Psi(n) \mathbf{m}_1(n)$ then generalize the

result. As $\Psi(n) = \mathbf{G}_s^H(n) \mathbf{G}_s(n)$ is a diagonal matrix with 0 or 1 entries, multiplying it to $\mathbf{m}_1(n)$ and $\mathbf{m}_1^H(n)$ does not need any multiplication operation. Therefore, the order of multiplications required for computing $\alpha(n)$ in (44) and multiplying it to the vector $\sum_{j=1}^{K-1} \mathbf{G}_s(n-j)(\mathbf{d}(n-j) - \mathbf{x}(n))$ is $\mathcal{O}(N)$. Consequently, in total, we need $\mathcal{O}(N|\mathcal{F}|)$ multiplications for the LMS, ELMS, and FELMS algorithms at each iteration. Moreover, in view of (10) and (13)–(17), the per-iteration computational complexity of the RLS, LS, and BLUE algorithms is $\mathcal{O}(|\mathcal{F}|^3 + M|\mathcal{F}|^2 + N|\mathcal{F}|)$ where M is the average number of nodes selected at each iteration. We can conclude that the proposed algorithms have substantially lower computational complexity than the existing BLUE, LS, and RLS algorithms.

V. MEAN-SQUARE PERFORMANCE ANALYSIS

In this section, we analyze the performance of the LMS, ELMS, and FELMS algorithms and derive a theoretical formula for their mean-square deviation (MSD). Note that the steady-state performance of the LMS algorithm has been analyzed in [39]. However, its transient behavior has not been analyzed before. To compute the MSD at time n , i.e., $E\|\tilde{\mathbf{x}}(n)\|^2$, we state the update equation of the weight error vector in (26) in a different format as

$$\begin{aligned} \tilde{\mathbf{x}}(n+1) &= (\mathbf{Q}_1(n) + \alpha(n)\mathbf{Q}_2(n)) \tilde{\mathbf{x}}(n) \\ &\quad - \bar{\mathbf{v}}_1(n) - \alpha(n)\bar{\mathbf{v}}_2(n) \end{aligned} \quad (50)$$

where

$$\mathbf{Q}_3(n) = \mu \mathbf{H}_f \mathbf{G}_s(n) \quad (51a)$$

$$\mathbf{Q}_1(n) = \mathbf{I} - \mathbf{Q}_3(n) \quad (51b)$$

$$\mathbf{Q}_2(n) = - \sum_{j=1}^{K-1} \mathbf{Q}_3(n-j) \quad (51c)$$

$$\bar{\mathbf{v}}_1(n) = \mathbf{Q}_3(n) \mathbf{v}(n) \quad (51d)$$

$$\bar{\mathbf{v}}_2(n) = \sum_{j=1}^{K-1} \mathbf{Q}_3(n-j) \mathbf{v}(n-j). \quad (51e)$$

Given an arbitrary matrix $\Phi \in \mathcal{R}^{N \times N}$, taking Φ -weighted norm followed by expectation from both sides of (50) results in

$$E\|\tilde{\mathbf{x}}(n+1)\|_{\Phi}^2 = E\|\tilde{\mathbf{x}}(n)\|_{\Phi(n)}^2 + q_1(n) \quad (52)$$

where

$$\begin{aligned} \dot{\Phi}(n) &= \mathbf{Q}_1^H(n) \Phi \mathbf{Q}_1(n) + \alpha^2(n) \mathbf{Q}_2^H(n) \Phi \mathbf{Q}_2(n) \\ &\quad + \alpha(n) (\mathbf{Q}_2^H(n) \Phi \mathbf{Q}_1(n) + \mathbf{Q}_1^H(n) \Phi \mathbf{Q}_2(n)) \end{aligned} \quad (53)$$

$$q_1(n) = \bar{\mathbf{v}}_1^H(n) \Phi \bar{\mathbf{v}}_1(n) + \bar{\mathbf{v}}_2^H(n) \Phi \bar{\mathbf{v}}_2(n) \alpha^2(n). \quad (54)$$

Applying the vectorization operator to both sides of (52), (53), and (54) and using the property $\text{vec}(\mathbf{A}\Sigma\mathbf{C}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\Sigma)$, we have

$$E\{\mathbf{z}(n+1)\phi\} = E\{\mathbf{z}(n)\text{vec}(\dot{\Phi}(n))\} + E\{\text{vec}(q_1(n))\} \quad (55)$$

where $\otimes$ denotes the Kronecker product, $\mathbf{z}(n) = \tilde{\mathbf{x}}^T(n) \otimes \tilde{\mathbf{x}}^H(n)$, and $\phi = \text{vec}(\Phi)$. Assuming that $\mathbf{z}(n)$ is independent of $\text{vec}(\Phi(n))$ and since ϕ is a deterministic vector, (55) can be restated as

$$E\{\mathbf{z}(n+1)\}\phi = E\{\mathbf{z}(n)\}\mathbf{F}_\alpha(n)\phi + \mathbf{f}_\alpha(n)\phi \quad (56)$$

where

$$\mathbf{F}_\alpha(n) = \mathbf{Q}_6 + (\mathbf{Q}_7 + \mathbf{Q}_7^H)\alpha(n) + \alpha^2(n)\mathbf{Q}_8 \quad (57)$$

$$\mathbf{f}_\alpha(n) = \text{vec}(\mathbf{C}_v)^T(\alpha^2(n)\mathbf{Q}_{10} + \mathbf{Q}_9) \quad (58)$$

and

$$\mathbf{Q}_6 = E\{\mathbf{Q}_1^T(n) \otimes \mathbf{Q}_1^H(n)\} \quad (59a)$$

$$\mathbf{Q}_7 = E\{\mathbf{Q}_2^T(n) \otimes \mathbf{Q}_1^H(n)\} \quad (59b)$$

$$\mathbf{Q}_8 = E\{\mathbf{Q}_2^T(n) \otimes \mathbf{Q}_2^H(n)\} \quad (59c)$$

$$\mathbf{Q}_9 = E\{\mathbf{Q}_3^T(n) \otimes \mathbf{Q}_3^H(n)\} \quad (59d)$$

$$\mathbf{Q}_{10} = E\left\{\sum_{j=1}^{K-1} \mathbf{Q}_3^T(n-j) \otimes \mathbf{Q}_3^H(n-j)\right\}. \quad (59e)$$

A. Transient and Steady-State Performance Analysis of LMS and ELMS

Since $\alpha(n)$ is constant in the LMS and ELMS algorithms (0 or 1, respectively), $\mathbf{F}_\alpha(n)$ and $\mathbf{f}_\alpha(n)$ in (56) are fixed for time-invariant sampling strategies and are denoted by $\mathbf{F}_\alpha$ and $\mathbf{f}_\alpha$. Therefore, (56) can be unfolded over the iterations resulting in

$$E\{\mathbf{z}(n)\}\phi = E\{\mathbf{z}(0)\}\mathbf{F}_\alpha^n\phi + \mathbf{f}_\alpha \sum_{l=0}^{n-1} \mathbf{F}_\alpha^l\phi \quad (60)$$

or equivalently

$$E\|\tilde{\mathbf{x}}(n)\|_\Phi^2 = E\{\mathbf{z}(0)\}\mathbf{F}_\alpha^n\phi + \mathbf{f}_\alpha \sum_{l=0}^{n-1} \mathbf{F}_\alpha^l\phi. \quad (61)$$

The theoretical value of the instantaneous MSD can be obtained by setting $\Phi = \mathbf{I}$ in (61) as

$$\begin{aligned} \text{MSD}(n) &= E\|\tilde{\mathbf{x}}(n)\|^2 \\ &= E\{\mathbf{z}(0)\}\mathbf{F}_\alpha^n\mathbf{r} + \mathbf{f}_\alpha \sum_{l=0}^{n-1} \mathbf{F}_\alpha^l\mathbf{r} \end{aligned} \quad (62)$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{r} = \text{vec}(\mathbf{I})$.

Taking the limit $n \rightarrow \infty$ in (56) gives the steady-state MSD of LMS [with $\alpha(n) = 0$] and ELMS [with $\alpha(n) = 1$] as

$$\lim_{n \rightarrow \infty} \text{MSD}(n) = \mathbf{f}_\alpha(\mathbf{I} - \mathbf{F}_\alpha)^{-1}\mathbf{r}. \quad (63)$$

B. Transient Performance Analysis of FELMS

In the FELMS algorithm, $\mathbf{F}_\alpha(n)$ and $\mathbf{f}_\alpha(n)$ change over time since $\alpha(n)$ is time-varying. Thus, (56) can be unfolded as

$$E\{\mathbf{z}(n)\}\phi = E\{\mathbf{z}(0)\}\mathbf{H}(n)\phi + h(n)\phi \quad (64)$$

where

$$\begin{aligned} \mathbf{H}(n) &= \prod_{l=0}^{n-1} \mathbf{F}_\alpha(l) \\ h(n) &= \sum_{l=0}^{n-1} \left\{ \mathbf{f}_\alpha(l) \prod_{k=l+1}^{n-1} \mathbf{F}_\alpha(k) \right\}. \end{aligned}$$

Similar to (62), the theoretical value of MSD for FELMS is given by

$$\begin{aligned} \text{MSD}(n) &= E\|\tilde{\mathbf{x}}(n)\|^2 \\ &= E\{\mathbf{z}(0)\}\mathbf{H}(n)\mathbf{r} + h(n)\mathbf{r}. \end{aligned} \quad (65)$$

The theoretical value of $\Delta_\alpha(n)$ can be obtained from (56) by setting $\Phi = \mathbf{G}_s^H(n)\mathbf{G}_s(n)$ as

$$\begin{aligned} \Delta_\alpha(n) &= E\{\|\tilde{\mathbf{x}}(n+1)\|_\Phi^2 - E\|\tilde{\mathbf{x}}(n)\|_\Phi^2\} \\ &= E\{\mathbf{z}(n+1)\}\phi - E\{\mathbf{z}(n)\}\phi \\ &= E\{\mathbf{z}(n)\}(\mathbf{F}_\alpha(n) - \mathbf{I})\phi + \mathbf{f}_\alpha(n)\phi. \end{aligned} \quad (66)$$

To calculate the optimal $\alpha(n)$ based on the theoretical $\Delta_\alpha(n)$, we set the derivative of $\Delta_\alpha(n)$ in (66) with respect to $\alpha(n)$ to zero and attain

$$\alpha_o(n) = \frac{-E\{\mathbf{z}(n)\}(\mathbf{Q}_7 + \mathbf{Q}_7^*)\phi}{2(E\{\mathbf{z}(n)\}\mathbf{Q}_8 + \text{vec}(\mathbf{C}_v)^T\mathbf{Q}_{10})\phi}. \quad (67)$$

As in (44), we calculate $\alpha(n)$ using the following equation

$$\alpha(n) = \gamma(n)\alpha_o(n) \quad (68)$$

where

$$\gamma(n) = \begin{cases} 1 & \Delta_{\alpha_o}(n) < \theta \\ \eta\gamma(n-1) & \text{otherwise} \end{cases} \quad (69)$$

where $\gamma(0) = 1$. An appropriate value of θ should distinguish the transient and steady states of $\Delta_{\alpha_o}(n)$, i.e., $\Delta_{\alpha_o}(n)$ enters its steady state when $\Delta_{\alpha_o}(n) > \theta$ is satisfied. Hence, a suitable value of θ can be chosen as a lower bound for the steady-state $\Delta_{\alpha_o}(n)$. In Appendix B, we derive one such lower bound as

$$\lim_{n \rightarrow \infty} \Delta_{\alpha_o}(n) \geq -(2\mu K + 1)\text{tr}\{\mathbf{C}_v\} \quad (70)$$

assuming that the steady-state $\alpha_o(n)$ is 1. Therefore, we have

$$\theta = -(2\mu K + 1)\text{tr}\{\mathbf{C}_v\}. \quad (71)$$

Since we have $\lim_{n \rightarrow \infty} \alpha(n) = 0$ in FELMS, the steady-state MSD of this algorithm is equal to that of the LMS algorithm as in (63).

C. Steady-State Performance

Taking the expectation of both sides of (50) yields

$$E\{\tilde{\mathbf{x}}(n+1)\} = (\mathbf{I} - \mu\mathbf{H}_f\mathbf{P}t_\alpha) E\{\tilde{\mathbf{x}}(n)\} \quad (72)$$

where

$$\mathbf{P} = E\{\mathbf{G}_s(n)\} = \text{diag}\{p_1, \dots, p_N\} \quad (73)$$

represents the sampling probability matrix and $t_\alpha = 1 + \alpha(n)(K - 1)$. To guarantee the convergence in the mean sense, the condition $\|\mathbf{I} - \mu \mathbf{H}_f \mathbf{P} t_\alpha\| < 1$ should be satisfied, which implies

$$\mu < \frac{2}{t_\alpha \lambda_{\max}(\mathbf{H}_f \mathbf{P})}. \quad (74)$$

To guarantee the mean-square-sense convergence (stability) in (60) and (64), $\mathbf{F}_\alpha(n)$ [re (57)] should be stable [43]. With $\Phi = \mathbf{I}$ and $\mathbf{U}_\Phi = \mathbf{I}$, we can rewrite (57) as

$$\mathbf{F}_\alpha(n) = \mathbf{I} - \mu \mathbf{V} + \mu^2 \mathbf{W} \quad (75)$$

where

$$\begin{aligned} \mathbf{Z} &= t_\alpha ((\mathbf{P} \mathbf{H}_f^T) \otimes \mathbf{I} + \mathbf{I} \otimes (\mathbf{P} \mathbf{H}_f^*)) \\ \mathbf{W} &= (\mathbf{G}_t \otimes \mathbf{G}_t)(\mathbf{H}_f^T \otimes \mathbf{H}_f^*) \\ \mathbf{G}_t &= E \left\{ \mathbf{G}_s(n) + \alpha(n) \sum_{j=1}^{K-1} \mathbf{G}_s(n-j) \right\}. \end{aligned}$$

Consequently, the condition on μ that guarantees the convergence in the mean-square sense is [44]

$$0 < \mu < \min \left\{ \frac{1}{\lambda_{\max}(\mathbf{W}^{-1} \mathbf{Z})}, \frac{1}{\max(\lambda(\mathbf{O}) \in \mathbb{R}^+)} \right\} \quad (76)$$

where $\mathbf{O} = \begin{bmatrix} \frac{1}{2} \mathbf{W} & -\frac{1}{2} \mathbf{Z} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$.

VI. SAMPLING

As shown in previous studies [30], [36], [39], the performance of adaptive algorithms for estimating graph signals strongly depends on the strategy by which samples are selected. In this section, we describe several existing sampling strategies as well as our proposed adaptive sampling strategy. Subsequently, we compare them to investigate the performance of the proposed strategy.

A. Background on Sampling Strategies

Greedy sampling strategies select the sampling set by solving different optimization problems [19], [21], [36], [45], [46]. The Min-MSD sampling strategy of [36] aims at minimizing the steady-state MSD. The Min-Pinv strategy selects the columns of $\mathbf{V}_f$ that maximize $\mathbf{V}_f^H \mathbf{D} \mathbf{V}_f$ where $\mathbf{D}$ is a diagonal matrix with entries 0 or 1 [21]. The Max-SingMin strategy of [19] selects nodes that maximize the smallest singular value of the matrix $\mathbf{V}_f^H \mathbf{D} \mathbf{V}_f$. These greedy sampling strategies are shown in Table I in detail where $\delta_{\min}(\mathbf{A})$ and $\lambda_i(\mathbf{A})$ denote the minimum singular value and the i^{th} eigenvalue of the matrix $\mathbf{A}$, respectively, and $J = \min(|\mathcal{S}|, |\mathcal{F}|)$.

Unlike the deterministic and time-invariant greedy sampling strategies that select a certain sampling set, the probabilistic sampling strategy in [39] assigns a selection probability to each

TABLE I
GREEDY SAMPLING STRATEGIES PROPOSED IN [19], [21], AND [36]

Input Data : M , the number of samples.	
Output Data : $\mathcal{S}$, the sampling set.	
Function: initialize $\mathcal{S} \equiv \emptyset$	
While $\mathcal{S} < M$.	
$s = \begin{cases} \arg \min_j \mathbf{f}_\alpha(\mathcal{S} \cup \{j\})(\mathbf{I} - \mathbf{F}_\alpha(\mathcal{S} \cup \{j\}))^{-1} \phi & \text{Min - MSD} \\ \arg \max_j \prod_{l=1}^J \lambda_l(\mathbf{V}_f^H \mathbf{G}_s(\mathcal{S} \cup \{j\}) \mathbf{V}_f) & \text{Min - Pinv} \\ \arg \max_j \delta_{\min}(\mathbf{V}_f^H \mathbf{G}_s(\mathcal{S} \cup \{j\}) \mathbf{V}_f) & \text{Max - SingMin} \end{cases}$	
$\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$	
End	

node. This sampling strategy finds the optimal sampling probability vector $\mathbf{p}$ by solving the following optimization problem

$$\mathbf{p} = \arg \min_{\mathbf{p}} \frac{\text{Tr}(\mathbf{V}_f^H \text{diag}(\mathbf{p}) \mathbf{C}_v \mathbf{V}_f)}{\text{Tr}(\mathbf{V}_f^H \text{diag}(\mathbf{p}) \mathbf{V}_f)}. \quad (77)$$

Although (77) is a convex problem and has a global optimum, it requires numerical methods for its solution. Solving an optimization problem at each iteration is a significant drawback of this sampling strategy compared with the greedy ones.

B. Adaptive Sampling Strategy

In this section, we propose a probabilistic sampling strategy that updates the sampling probability vector adaptively based on the convergence state of the algorithm at each iteration. Not only this strategy does not become fixated on a certain deterministic sampling set [the advantage of the probabilistic strategy of (77)], but also it does not need the solution of any optimization problem at every iteration (the advantage of the greedy strategies).

To achieve an efficient sampling strategy, we select the sampling set in each node based on the GMSD of the node. In each iteration, if the GMSD of a node is negative, the node is considered worthy of being selected and its sampling probability is kept unchanged. Otherwise, the sampling probability in a node with a positive GMSD is decreased.

To calculate the GMSD of the l^{th} node at time n , (26) can be rewritten as

$$[\tilde{\mathbf{x}}(n+1)]_l = [\tilde{\mathbf{x}}(n)]_l - [\mathbf{m}(n)]_l, \quad (78)$$

where $[\mathbf{t}]_l$ denotes the l^{th} element of vector $\mathbf{t}$. By squaring and taking the expectation of both sides of (78) and defining GMSD of the l^{th} node at time n as $\text{GMSD}_l(n)$, we have

$$\begin{aligned} \text{GMSD}_l(n) &= E\{([\tilde{\mathbf{x}}(n+1)]_l)^2 - ([\tilde{\mathbf{x}}(n)]_l)^2\} \\ &= -2E\{[\tilde{\mathbf{x}}(n)]_l [\mathbf{m}(n)]_l\} + E\{([\mathbf{m}(n)]_l)^2\}. \end{aligned} \quad (79)$$

Considering (5), the following equation can be obtained for the l^{th} node, where $l \in \mathcal{S}(n)$:

$$[\mathbf{d}(n)]_l = [\mathbf{x}^o]_l + [\mathbf{v}(n)]_l. \quad (80)$$

Recalling that the noise at each node is zero-mean and spatially and temporally independent, we have

$$\begin{aligned} [\tilde{\mathbf{x}}(n)]_l [\mathbf{m}(n)]_l &= E\{[\mathbf{d}(n) - \mathbf{x}(n)]_l [\mathbf{m}(n)]_l\} \\ &= \mu[\mathbf{H}_f]_{l,l} \sigma_l^2 \end{aligned} \quad (81)$$

where $[\mathbf{A}]_{l,l}$ is the l^{th} diagonal element of $\mathbf{A}$. Substituting (81) into (79), we have

$$\begin{aligned} \text{GMSD}_l(n) &= ([\mathbf{m}(n)]_l)^2 + 2\mu[\mathbf{H}_f]_{l,l} \sigma_l^2 \\ &\quad - 2[\mathbf{d}(n) - \mathbf{x}(n)]_l [\mathbf{m}(n)]_l. \end{aligned} \quad (82)$$

Note that since the exact expected values in (82) are not available, the instantaneous values are used.

To reduce the complexity, especially in large-scale graphs [47], the sampling probability of the l^{th} node should be minimized. Now, we can devise a sampling strategy based on the value of $\text{GMSD}_l(n)$ as follows. Since any node with a positive GMSD increases the total MSD of the graph, we can set the sampling probability of the l^{th} node to 0 when $\text{GMSD}_l(n) > 0$ and keep it unchanged otherwise, i.e.,

$$p_l(n) = \begin{cases} p_l(n-1) & \text{GMSD}_l(n) < 0 \\ 0 & \text{GMSD}_l(n) > 0 \end{cases} \quad (83)$$

To prevent $p_l(n)$ from having sudden changes, we rewrite (83) as

$$\bar{p}_l(n) = \begin{cases} 1 & \text{GMSD}_l(n) < \theta' \\ p_l(n-1) & \theta' < \text{GMSD}_l(n) < 0 \\ \tau p_l(n-1) & \text{GMSD}_l(n) > 0 \end{cases} \quad (84)$$

$$p_l(n) = \max(\bar{p}_l(n), \zeta) \quad (85)$$

where θ' is a threshold parameter, $\tau < 1$ is to induce smooth changes in $p_l(n)$, and $\zeta > 0$ is to prevent $p_l(n)$ from becoming very small.

Similar to the threshold parameter θ used for the calculation of $\alpha(n)$ in (45), the parameter θ' in (84) ought to be a lower bound for the steady-state $\text{GMSD}_l(n)$ to distinguish the steady state of $\text{GMSD}_l(n)$ from its transient state. Let $\mathbf{I}_l \in \mathcal{R}^{N \times N}$ be an all-zero matrix with only a single 1 in its l^{th} diagonal entry. Clearly, $\text{GMSD}_l(n)$ is equal to $\Delta_\alpha(n)$ when $\Phi = \mathbf{I}_l$. Hence, a lower bound on $\text{GMSD}_l(n)$ can be obtained from (101) as

$$\theta' = -(2\mu K \alpha(n) + 1) \text{tr}\{\mathbf{C}_v\} \quad (86)$$

where at the steady state, $\alpha(n) = 0$ for the FELMS and LMS algorithms and $\alpha(n) = 1$ for the ELMS algorithm.

C. Comparison Between Sampling Strategies

The probabilistic sampling strategy of [39] has a high computational complexity as it requires to solve an optimization problem at every iteration. As shown in Table I, the greedy Min-MSD sampling strategy needs to calculate MSD as in (63) several times for different sampling sets requiring matrix inversions or solution of multiple systems of linear equations. The Min-Pinv and Max-SingMin sampling strategies need to calculate $\mathbf{V}_f^H \mathbf{G}_s(\mathcal{S} \cup \{j\}) \mathbf{V}_f$ and subsequently compute its determinant and singular value decomposition. The proposed

Fig. 1. Graph topology.

adaptive sampling strategy calculates $\text{GMSD}_l(n)$ for each node at each iteration. As $\mathbf{m}(n)$ is calculated by the adaptive algorithm, the required number of multiplications in (82) and (84) corresponding to N nodes at each iteration is $\mathcal{O}(N)$. Hence, the total computational complexity of the proposed adaptive sampling strategy is $\mathcal{O}(N)$. The overall computational complexity of our adaptive sampling strategy depends on the number of iterations of the algorithm. Therefore, it cannot be directly compared with that of the greedy and the probabilistic strategies, which do not have any extra computation between the iterations. However, since the order of computational complexity of the LMS, ELMS, and FELMS algorithms is linear in N , the proposed sampling strategy does not at least augment the order of complexity of these algorithms.

One issue with the greedy sampling strategies is about determining the proper number of selected samples at each iteration (denoted by M in Table I). Clearly, larger M leads to better performance but at the expense of higher computational complexity. Therefore, the greedy sampling strategies have to make a trade-off between computational complexity and performance through the choice of the parameter M . Unlike the greedy sampling strategies, the proposed sampling strategy adaptively selects the most suitable nodes at each iteration and is not bound by any preset number of nodes to select.

VII. SIMULATION RESULTS

In this section, we demonstrate the performance of the proposed algorithms via several computer simulations.

A. Performance of the Algorithms

We consider a graph with $N = 30$ nodes as shown in Fig. 1. The spectral content of the graph is limited to the first ten eigenvectors of its Laplacian matrix, i.e. $|\mathcal{F}| = 10$. The observation noise in (5) is zero-mean Gaussian with a diagonal covariance matrix where each entry is drawn from a uniform distribution between 0 and 0.01. In all simulations, we show the normalized mean square deviation (NMSD), $E\{\frac{\|\mathbf{x}^o - \mathbf{x}(n)\|^2}{\|\mathbf{x}^o\|^2}\}$, which is

Fig. 2. Random sampling probabilities.

Fig. 3. The NMSD learning curves of ELMS and FELMS for $\mu = 0.01$ and different values of K .

evaluated by ensemble averaging over 50 independent trials. For FELMS algorithm, the parameters θ and η are set to -1 and 0.9 , respectively. Note that in the simulations where the adaptive sampling strategy is not used, the sampling probability in each node is set randomly between 0.4 and 1 . For example, for the graph with $N = 30$ nodes, sampling probabilities of nodes are shown in Fig. 2.

In Fig. 3, the NMSD learning curves of ELMS and FELMS are shown for $\mu = 0.01$ and different values of K . It can be seen that the higher the value of parameter K is, the higher the convergence speed and steady-state NMSD of ELMS are. Conversely, the steady-state NMSD of FELMS is almost the same for different values of K while its convergence is accelerated by increasing K .

In Fig. 4, we show the NMSD learning curves of RLS ($\beta = 0.9, 1$) [39], LMS [39], LS [29], [30], BLUE [23], ELMS, and FELMS for $\mu = 0.01$ and $K = 16$. As seen, both ELMS and FELMS converge faster than LMS. Furthermore, FELMS outperforms ELMS and has a steady-state error comparable with that of LMS. The RLS algorithm with $\beta = 1$, LS, and

Fig. 4. The NMSD learning curves of different algorithms for $\mu = 0.01$, $K = 16$, and $\beta = 1$ and 0.9 . Fig. 5. The NMSD learning curves of LMS, ELMS, and FELMS with different values of K in the same-steady-state scenario.

BLUE utilize the information available at the current and all past instances. This leads to their lower steady-state error compared with the other algorithms. However, the RLS algorithm with $\beta = 1$, LS, and BLUE can experience significant performance degradation with time-varying graphs. It is also seen in Fig. 4 that the FELMS algorithm has a lower steady-state error than RLS with $\beta = 0.9$ while being computationally less complex. Considering its low computational complexity and improved performance compared to the LMS and RLS ($\beta = 0.9$) algorithms, the FELMS algorithm is evidently competent.

In Fig. 5, the NMSD learning curves of the LMS, ELMS, and FELMS algorithms are plotted in a same-steady-state scenario to compare the convergence speed of the algorithms while their steady-state errors are the same. Since the FELMS and LMS algorithms have the same steady-state error, we only tune the step-size parameter μ for the ELMS algorithm to make its steady-state error equal to that of others. As seen in Fig. 5, the

Fig. 6. The NMSD learning curves of different algorithms for $\mu = 0.01$, $K = 16$, and different sampling strategies.

Fig. 7. Average number of selected nodes in different algorithms with different sampling strategies.

ELMS and FELMS algorithms converge faster than the LMS algorithm.

To examine the performance of the proposed adaptive sampling strategy in (84)–(85), the NMSD learning curves of different algorithms are shown in Fig. 6 where the proposed adaptive sampling strategy with $\tau = 0.995$ and $\theta = 0.01$ and random sampling with the probabilities shown in Fig. 2 are used. In addition, the average number of selected nodes at each iteration is plotted in Fig. 7. The results demonstrate that although the proposed adaptive sampling strategy entails fewer selected nodes compared to random sampling, it does not incur any degradation in convergence rate or steady-state error.

Fig. 8 shows the NMSD learning curves of the LMS algorithm when using the considered existing sampling strategies, i.e., random, greedy Min-MSD [36], greedy Max-SingMin [19], and greedy Min-Pinv [21], with $\mu = 1$, $N = 20$, $|\mathcal{F}| = 7$, and $M = 12$. To compare our proposed adaptive sampling strategy with the previous ones without cluttering the figures, we compare our strategy with the aforementioned strategies with $M = 18$

Fig. 8. The NMSD learning curves of LMS for random and greedy sampling strategies with $M = 12$.

Fig. 9. The NMSD learning curves of LMS for adaptive, random, and greedy sampling strategies with $M = 18$.

in Fig. 9. We also show the average number of selected nodes in each iteration by the considered algorithms in Fig. 10. We can observe from Figs. 8–10 that the convergence speed when using the random and greedy strategies degrades substantially as M decreases from 18 to 12. The proposed adaptive strategy leads to a higher convergence rate and steady-state accuracy compared with the random and greedy strategies with $M = 12$ and 18. This is mainly due to the adaptive selection of the nodes at each iteration by the proposed strategy. It is also clear from Fig. 10 that the proposed adaptive sampling strategy selects a small number of nodes at the steady state.

To compare the time complexity of the proposed sampling strategy with the greedy sampling strategies, we measured the associated processing times in the scenario leading to Fig. 9. The greedy Min-MSD sampling strategy took 4 seconds to run while the Min-Pinv and Max-SingMin sampling strategies took about 4 milliseconds. The processing time for the proposed adaptive strategy was 0.1 millisecond per iteration on average. We can

Fig. 10. The average number of selected nodes in the LMS algorithms for adaptive, random, and greedy sampling strategies with $M = 12$ and 18.

conclude from the above runtimes that the time complexity of the proposed strategy is reasonably low while that of Min-MSD is significantly higher. We implemented the simulations using MATLAB R2018b on a PC with a 3.40GHz CPU and 32 GB of RAM.

B. Parameter Values and Performance

In this section, we examine the impact of the values of the user-defined parameters in the proposed algorithms on their performance.

Recall (45) where the threshold parameter θ is used in the FELMS algorithm. If $\Delta_{\alpha_o}(n)$ is near zero, the algorithm is in its steady state. Therefore, $\alpha(n)$ should be decreased until it converges to zero. The parameter θ is a negative number that is used as a threshold to determine the convergence state of the algorithm so that when $\Delta_{\alpha_o}(n) < \theta$ the algorithm is considered to be in the transient state. Thus, θ can be seen as a lower bound for the steady-state $\Delta_{\alpha_o}(n)$. We propose a suitable value for θ in Section V-B [see (71)]. In Fig. 11, we plot the instantaneous values of $\Delta_{\alpha_o}(n)$ and its lower bound θ given by (71) for different values of μ and K . Note that as the dynamic range of $\Delta_{\alpha_o}(n)$ is very high, we plot a limited range of it for clarity. As seen, for different values of μ and K , the calculated lower bound is a good value for the threshold θ .

The threshold parameter θ' is used in the proposed adaptive sampling strategy [see (84)] to distinguish the steady state of $\text{GMSD}_l(n)$ from its transient state. A suitable value for this parameter is given in Section VI-B, i.e., (86). In Fig. 12, we see that for different values of μ and K , θ' is a good distinguisher of the transient and steady states of $\text{GMSD}_l(n)$.

Recalling (44) and (45), the parameter η specifies the damping rate applied to $\alpha(n)$. The effect of η on the performance of the FELMS algorithm is shown in Figs. 13 and 14. As seen in these figures, if $\eta = 1$, then $\alpha(n) = \alpha_o(n) \forall n$ and the FELMS algorithm does not switch to the LMS algorithm [$\alpha(n) = 0$] at the steady state. Hence, it does not achieve the low steady-state error of the LMS algorithm. Smaller values of η make $\alpha(n)$

Fig. 11. The evolution of $\Delta_{\alpha_o}(n)$ over iterations and its steady-state lower bound, θ , for different values of μ and K .

Fig. 12. The evolution of $\text{GMSD}_l(n)$ over iterations and its steady-state lower bound, θ' , for different values of μ and K .

Fig. 13. The NMSD learning curves of FELMS with $\mu = 0.01$, $K = 16$, and different values of η .

Fig. 14. The NMSD learning curves of FELMS with $\mu = 0.5$, $K = 16$, and different values of η .

Fig. 15. The NMSD learning curves of FELMS with the adaptive sampling strategy for $\mu = 0.01$, $K = 16$, and different values of τ .

Fig. 16. Average number of selected nodes in FELMS with the adaptive sampling strategy for different values of τ .

Fig. 17. The NMSD learning curves of LMS for different values of μ .

converge to zero earlier leading to a slower transition to the steady state. As seen in Fig. 14, for higher values of μ , the FELMS algorithm's performance does not change substantially with variations in η . For smaller values of μ as in Fig. 13, the algorithm's performance deteriorates only when η is very small or very close to one. Evidently, a value around 0.9 is a good choice for η notwithstanding that, for large values of μ , the FELMS algorithm has little sensitivity to the value of η .

The damping coefficient τ is used in the proposed adaptive sampling strategy [recall (84)]. Larger values of this parameter cause the sampling probabilities decrease slower. The effect of this parameter on the performance of the FELMS algorithm is shown in Figs. 15 and 16. It can be seen from Fig. 15 that by increasing τ , the steady-state error of the algorithm increases. In the extreme case when $\tau = 1$, the sampling probability of all nodes is fixed at $p_i = 1$. It means that all nodes are selected in each iteration. Therefore, the algorithm shows its best performance. Fig. 16 shows that the number of selected nodes decreases as τ increases. Visibly, there is a trade-off between

the complexity of the FELMS algorithm (the number of selected nodes) and its performance that is governed by the value of τ . We conclude from Figs. 15 and 16 that a value of τ close to 1 (for example $\tau = 0.99$) is a good choice as the number of selected nodes is substantially decreased while the performance is nearly similar to when all nodes are selected ($\tau = 1$).

The step-size parameter μ is used in the gradient descent iterations. If it is larger than a threshold, the iterations will diverge. In (74) and (76), we provide the bounds for μ that guarantee the convergence of the FELMS algorithm. Within the stable range, larger values of μ lead to higher convergence speed but also to higher steady-state error, and vice versa. The effect of the value of μ on the performance of the LMS, ELMS, and FELMS is shown in Figs. 17–19. Notice that the convergence speed of the FELMS algorithm does not degrade significantly when μ decreases. That is because the FELMS algorithm enjoys fast convergence due to using appropriate values of $\alpha(n)$ in the transient state.

Fig. 18. The NMSD learning curves of ELMS for $K = 8$ and different values of μ .

Fig. 19. The NMSD learning curves of FELMS for $K = 8$ and different values of μ .

Fig. 20. Graph topology.

C. Theoretical Performance

To verify our theoretical results, we compare the simulated and theoretical NMSD values of LMS, ELMS, and FELMS. To this aim, we consider a graph topology with 10 nodes as shown in Fig. 20. The theoretical and simulated learning curves of LMS,

Fig. 21. The simulated and theoretical steady-state NMSD learning curves of different algorithms for $\mu = 0.01$ and $K = 8$.

Fig. 22. The simulated and theoretical NMSD of FELMS and ELMS for different values of μ and K .

ELMS, and FELMS are plotted in Fig. 21 for $\mu = .01$, $\gamma = 0.9$, and $K = 8$. This figure shows a good agreement between simulation and theory. In Fig. 22, the theoretical and simulated steady-state NMSD for FELMS and ELMS with $K = 2, 4$, and 8 are depicted for different values of μ . A good agreement between theoretical and simulated steady-state NMSD values can be seen. We can also infer from this figure that by increasing K the stability bound of ELMS decreases and its steady-state NMSD increases. The FELMS outperforms ELMS in terms of both stability region extent and steady-state NMSD.

D. Application to Real Data

First, we consider mean snowfall measurements at $N = 503$ National Weather Service (NWS) stations across the United States for the 1971-2000 period [48]. The graph signal at each vertex shows mean snowfall at its corresponding station. In Fig. 23, we show the true and estimated values of the mean snowfall measured at an unobserved randomly-chosen station.

Fig. 23. True snowfall and its estimates at a randomly chosen unobserved station, for different algorithms.

Fig. 24. True snowfall and its estimates versus time at a randomly chosen unobserved station, for different algorithms.

We estimate the graph signal using different algorithms such as LMS, ELMS, FELMS, and RLS with $\mu = 0.01$, $\gamma = 0.9$, $K = 16$, and $|\mathcal{F}| = 100$. The noise covariance matrix is diagonal with each element randomly selected in the interval $[0, 0.01]$. It can be interpreted from Fig. 23 that the proposed ELMS and FELMS algorithms estimate the true value of snowfall much faster than LMS. Moreover, in comparison with ELMS, the FELMS is more accurate and has less fluctuations around the true value. The FELMS shows comparable performance to RLS despite its much lower computational complexity.

We also study the tracking performance of the proposed algorithms in comparison with LMS and RLS. Fig. 24 illustrates the true behavior of the snowfall measured at a randomly-chosen unobserved station. The data is collected over the first 120 hours of 2010. As seen, the proposed FELMS estimates the snowfall faster than ELMS and LMS. Moreover, as graph signal is not constant over time, RLS performs worse than FELMS and ELMS. The good performance of FELMS is also evident in

Fig. 25. The NMSD learning curves of different algorithms with $\mu = 0.01$ and $K = 16$ at a randomly chosen unobserved station.

Fig. 25 where the NMSD learning curves of different algorithms are plotted for the same unobserved node considered earlier.

VIII. CONCLUSION

We proposed two algorithms for adaptive learning of graph signals by reusing the past data and sampling the nodes in a smart and efficient manner. The proposed strategies have a higher convergence speed than the LMS algorithm while preserving its low computational complexity. The first algorithm is an extension of the LMS algorithm named ELMS. This algorithm uses signal vectors of previous iterations as well as the vector at the current iteration. Employing the previous signal vectors increases the convergence speed of the ELMS algorithm at the expense of higher steady-state error compared with the LMS algorithm. The second algorithm called FELMS improves the convergence speed of ELMS and decreases its steady-state error. We studied the mean-square performance of ELMS and FELMS algorithms. Furthermore, we proposed an adaptive sampling strategy where the sampling probability of each node is adapted at each iteration according to the GMSD associated with the node. Using this strategy, the number of selected nodes decreases over time as the algorithm converges without incurring any substantial degradation in the estimation performance. Unlike the existing greedy sampling strategies, the proposed adaptive sampling strategy does not trade estimation accuracy for computational complexity. Rather, it optimizes the estimation accuracy jointly with the sampling (node selection) performance to yield fast convergence. More importantly, it achieves this efficiently with a little extra computational burden for sampling.

APPENDIX

A. Comparing the Steady-State Errors of LMS and ELMS Algorithms

To compare the steady-state error of the LMS and ELMS algorithms, we use (63) while setting $\alpha(n) = 1$ for ELMS and $\alpha(n) = 0$ for LMS. To provide a simple example, we consider a special case when $\mathbf{H}_f = \mathbf{G}_s(n) = \mathbf{I}$. Using these values in (57)

and (58), $\mathbf{F}_\alpha$ and $\mathbf{f}_\alpha$ are computed as

$$\text{LMS} : \begin{cases} \mathbf{f}_0 = \mu^2 \text{vec}(\mathbf{C}_v)^T \mathbf{I} \\ \mathbf{F}_0 = (1 - \mu)^2 \mathbf{I} \end{cases} \quad (87)$$

$$\text{ELMS} : \begin{cases} \mathbf{f}_1 = K\mu^2 \text{vec}(\mathbf{C}_v)^T \mathbf{I} \\ \mathbf{F}_1 = (1 - K\mu)^2 \mathbf{I} \end{cases}. \quad (88)$$

Using these values in (63), the steady-state MSD for LMS and ELMS is computed as

$$\text{MSD}_{\text{LMS}}(n) = \frac{\mu^2 \text{tr}\{\mathbf{C}_v\}}{1 - (1 - \mu)^2} \quad (89)$$

$$\text{MSD}_{\text{ELMS}}(n) = \frac{K\mu^2 \text{tr}\{\mathbf{C}_v\}}{1 - (1 - K\mu)^2}. \quad (90)$$

Let us consider the ratio of these values as follows

$$\frac{\text{MSD}_{\text{LMS}}(n)}{\text{MSD}_{\text{ELMS}}(n)} = \frac{2 - K\mu}{2 - \mu}. \quad (91)$$

Since $K > 1$, then $2 - K\mu < 2 - \mu$. Therefore,

$$\frac{\text{MSD}_{\text{LMS}}(n)}{\text{MSD}_{\text{ELMS}}(n)} = \frac{2 - K\mu}{2 - \mu} < 1. \quad (92)$$

Therefore, for the special case of $\mathbf{H}_f = \mathbf{G}_s(n) = \mathbf{I}$, the steady-state MSD of the ELMS algorithm is always larger than that of the LMS algorithm.

B. Lower Bound for the Steady-State $\Delta_\alpha(n)$

Here, we derive a lower bound for $\Delta_\alpha(n) = E\|\tilde{\mathbf{x}}(n+1)\|_{\Phi}^2 - E\|\tilde{\mathbf{x}}(n)\|_{\Phi}^2$. From (66), the theoretical value of $\Delta_\alpha(n)$ is

$$\Delta_\alpha(n) = E\{\mathbf{z}(n)\}(\mathbf{F}_\alpha(n) - \mathbf{I})\phi + \mathbf{f}_\alpha(n)\phi \quad (93)$$

$$\geq E\{\mathbf{z}(n)\}(\mathbf{F}_\alpha(n) - \mathbf{I})\phi \quad (94)$$

$$\geq E\{\mathbf{z}(n)\}((\mathbf{Q}_7 + \mathbf{Q}_7^H)\alpha(n) - \mathbf{I})\phi. \quad (95)$$

Note that the first and second inequalities are based on the facts that $\mathbf{f}_\alpha(n)$ is non-negative and $E\{\mathbf{z}(n)\}(\mathbf{Q}_6 + \alpha^2(n)\mathbf{Q}_8)\phi \geq 0$, respectively [see (58) and (59)]. Using (59b) and the properties of the $\text{vec}(\cdot)$ operator, (95) can be restated as

$$\Delta_\alpha(n) \geq E\{\tilde{\mathbf{x}}^H(n)[2\alpha(n)\Upsilon - \Phi]\tilde{\mathbf{x}}(n)\} \quad (96)$$

where $\Upsilon = E\{\mathbf{Q}_1^H(n)\Phi\mathbf{Q}_2(n)\}$. Assuming that, at the steady state, the signal is recovered exactly, (96) becomes

$$\Delta_\alpha(n) \geq 2\alpha(n)E\{\text{tr}\{\mathbf{C}_v\Upsilon\}\} - E\{\text{tr}\{\mathbf{C}_v\Phi\}\}. \quad (97)$$

In view of (51), (97) can be stated as

$$\Delta_\alpha(n) \geq -2\mu\alpha(n)h_1 + 2\alpha(n)\mu^2h_2 - E\{\text{tr}\{\mathbf{C}_v\Phi\}\}. \quad (98)$$

where

$$h_1 = E\left\{\text{tr}\left\{\mathbf{C}_v\Phi\mathbf{H}_f\sum_{j=1}^{K-1}\mathbf{G}_s(n-j)\right\}\right\}, \quad (99)$$

$$h_2 = E\left\{\text{tr}\left\{\mathbf{C}_v\mathbf{H}_f\mathbf{G}_s(n)\Phi\mathbf{H}_f\sum_{j=1}^{K-1}\mathbf{G}_s(n-j)\right\}\right\}. \quad (100)$$

It is easy to show that $h_2 \geq 0$, $h_1 \leq K\text{tr}\{\mathbf{C}_v\}$, and $\text{tr}\{\mathbf{C}_v\Phi\} \leq \text{tr}\{\mathbf{C}_v\}$ when $\Phi = \mathbf{G}_s(n)$. Using these inequalities, (98) leads to

$$\Delta_\alpha(n) \geq -(2\mu K\alpha(n) + 1)\text{tr}\{\mathbf{C}_v\}. \quad (101)$$

REFERENCES

- [1] 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.
- [2] X. Zhao, S.-Y. Tu, and A. H. Sayed, "Diffusion adaptation over networks under imperfect information exchange and non-stationary data," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3460–3475, Jul. 2012.
- [3] J. Chen and A. H. Sayed, "On the benefits of diffusion cooperation for distributed optimization and learning," in *Proc. 21st Eur. Signal Process. Conf.*, IEEE, 2013, pp. 1–5.
- [4] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning-Part I: Algorithm development," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 708–723, Feb. 2018.
- [5] J. Chen and A. H. Sayed, "Distributed pareto optimization via diffusion strategies," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 205–220, Apr. 2013.
- [6] Z. J. Towfic, J. Chen, and A. H. Sayed, "Excess-risk of distributed stochastic learners," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5753–5785, Oct. 2016.
- [7] Z. J. Towfic and A. H. Sayed, "Stability and performance limits of adaptive primal-dual networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2888–2903, Jun. 2015.
- [8] F. S. Cattivelli and A. H. Sayed, "Distributed detection over adaptive networks using diffusion adaptation," *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 1917–1932, May 2011.
- [9] F. Cattivelli, "Distributed collaborative processing over adaptive networks," Ph.D. dissertation, UCLA, Los Angeles, CA, USA, 2010.
- [10] A. E. Feitosa, V. H. Nascimento, and C. G. Lopes, "Adaptive detection in distributed networks using maximum likelihood detector," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 974–978, Jul. 2018.
- [11] C. G. Lopes, L. F. Chamon, and V. H. Nascimento, "Towards spatially universal adaptive diffusion networks," in *Proc. IEEE Global Conf. Signal Inf. Process.*, IEEE, 2014, pp. 803–807.
- [12] K. He, L. Stankovic, J. Liao, and V. Stankovic, "Non-intrusive load disaggregation using graph signal processing," *IEEE Trans. Smart Grid*, vol. 9, no. 3, pp. 1739–1747, 2016.
- [13] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.
- [14] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.
- [15] 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.
- [16] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Graph Fourier transform," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, IEEE, 2013, pp. 6167–6170.
- [17] I. Pesenson, "Sampling in Paley-Wiener spaces on combinatorial graphs," *Trans. Amer. Math. Soc.*, vol. 360, no. 10, pp. 5603–5627, 2008.
- [18] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 3921–3924.
- [19] 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.
- [20] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 5445–5449.
- [21] 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.
- [22] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, May 2015.

- [23] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.
- [24] F. Gama, A. G. Marques, G. Mateos, and A. Ribeiro, "Rethinking sketching as sampling: A graph signal processing approach," in *Proc. 50th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, Nov. 2016, pp. 522–526.
- [25] S. K. Narang, A. Gadde, E. Sanou, and A. Ortega, "Localized iterative methods for interpolation in graph structured data," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 491–494.
- [26] X. Wang, M. Wang, and Y. Gu, "A distributed tracking algorithm for reconstruction of graph signals," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 728–740, Jun. 2015.
- [27] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Reconstruction of graph signals through percolation from seeding nodes," *IEEE Trans. Signal Process.*, vol. 64, no. 16, pp. 4363–4378, Aug. 2016.
- [28] J. Miettinen, S. Vorobyov, and E. Ollila, "Robust least squares estimation of graph signals," *IEEE Int. Conf. Acoust., Speech Signal Process.* 2019, pp. 5416–5420.
- [29] 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.
- [30] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 2, no. 4, pp. 539–554, 2016.
- [31] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, "Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring," *IEEE Trans. Signal Process.*, vol. 62, no. 11, pp. 2879–2893, Jun. 2014.
- [32] A. Sandryhaila and J. M. Moura, "Classification via regularization on graphs," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 495–498.
- [33] V. N. Ekambaram, G. Fanti, B. Ayazifar, and K. Ramchandran, "Wavelet-regularized graph semi-supervised learning," in *Proc. IEEE Global Conf. Signal Inf. Process.*, IEEE, 2013, pp. 423–426.
- [34] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, IEEE, 2012, pp. 3373–3376.
- [35] D. Thanou, D. I. Shuman, and P. Frossard, "Parametric dictionary learning for graph signals," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 487–490.
- [36] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, "Adaptive least mean squares estimation of graph signals," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 2, no. 4, pp. 555–568, 2016.
- [37] 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, 2017.
- [38] 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, 2017.
- [39] 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.
- [40] C. F. Cowan and P. M. Grant, *Adaptive Filters*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985, vol. 152.
- [41] M. S. E. Abadi and M. J. Ahmadi, "Diffusion improved multiband-structured subband adaptive filter algorithm with dynamic selection of nodes over distributed networks," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 66, no. 3, pp. 507–511, Mar. 2019.
- [42] M. Shams Esfand Abadi and M. J. Ahmadi, "Weighted improved multiband-structured sub-band adaptive filter algorithms," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 66, no. 12, pp. 2077–2081, Dec. 2019.
- [43] H.-C. Shin and A. H. Sayed, "Mean-square performance of a family of affine projection algorithms," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 90–102, Jan. 2004.
- [44] M. S. E. Abadi, J. H. Husøy, and M. J. Ahmadi, "Two improved multiband structured subband adaptive filter algorithms with reduced computational complexity," *Signal Process.*, vol. 154, pp. 15–29, 2019.
- [45] F. Wang, G. Cheung, and Y. Wang, "Low-complexity graph sampling with noise and signal reconstruction via neumann series," *IEEE Trans. Signal Process.*, vol. 67, no. 21, pp. 5511–5526, Nov. 2019.
- [46] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Trans. Signal Process.*, vol. 66, no. 1, pp. 34–47, Jan. 2017.
- [47] 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, 2018.
- [48] "1971–2000 u.s. snow normals (clim20-02)," 2018. [Online]. Available: <https://www.ncdc.noaa.gov/cgi-bin/climatenormals/climatenormals.pl>. Accessed on: Sep. 20, 2019.

Mohammad Javad Ahmadi received the B.S. degree (with honors) in electrical engineering from Babol Noshirvani University of Technology, Babol, Iran, in 2013 and the M.Sc. degree in communications from Sharif University of Technology, Tehran, Iran, in 2015. He is currently working toward the Ph.D. degree with the Department of EE, Bilkent University, Ankara, Turkey. His research interests include adaptive algorithms, estimation theory, and machine learning.

Reza Arablouei received the Ph.D. degree in telecommunications engineering from the Institute for Telecommunications Research, University of South Australia, Mawson Lakes, SA, Australia, in 2013. He was a Research Fellow with the University of South Australia from 2013 to 2015. He is currently a Research Scientist with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Pullenvale, QLD, Australia. His research interests are signal processing and machine learning for embedded systems.

Reza Abdoolee received the Ph.D. degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada, in 2014. He is currently an Assistant Professor with the Department of Computer Science, California State University Channel Islands, Camarillo, CA, USA, where he manages the Cybersecurity and Wireless Systems Lab. In the past few years, he has held different academic and industry positions in several universities and high-tech companies, including University of California,

Los Angeles (UCLA), University of California, Santa Barbara, California State University, Bakersfield (CSUB), Bell Labs and Qualcomm. His research has resulted in several inventions in addition to many interesting peer-reviewed journal publications and conference proceedings. He is currently conducting research in the area of cybersecurity and wireless communications with application to Internet of Things (IoT) and the next generation of wireless communication systems.