An Adaptive Sampling Technique for Graph Diffusion LMS Algorithm

Daniel G. Tiglea, Renato Candido, and Magno T. M. Silva
Escola Politécnica, University of São Paulo, Brazil
{dtiglea, renatocan, magno}@lps.usp.br

***Abstract*—Graph signal processing has attracted attention in the signal processing community, since it is an effective tool to deal with great quantities of interrelated data. Recently, a diffusion algorithm for adaptively learning from streaming graphs signals was proposed. However, it suffers from high computational cost since all nodes in the graph are sampled even in steady state. In this paper, we propose an adaptive sampling method for this solution that allows a reduction in computational cost in steady state, while maintaining convergence rate and presenting a slightly better steady-state performance. We also present an analysis to give insights about proper choices for its adaptation parameters.**

***Index Terms*—Graph signal processing, sampling on graphs, diffusion strategies, graph filtering, convex combination.**

I. Introduction

Over the last years, graph signal processing (GSP) has been a topic of intense research since the observed data can be modelled as graph signals in many network-structured applications, such as sensor networks, smart grids, transportation networks, communication networks, among others [1]–[8]. These applications can be modelled by a graph with a large number of connected nodes and therefore, many techniques for graph signal sampling have been proposed (see, e.g., [6] and its references). These sampling techniques use information from less nodes to make inferences about the whole graph signal, oftentimes by means of random sampling mechanisms [6].

Recently, [7]–[9] took advantages of diffusion strategies [10] to propose new tools for adaptive GSP, which led to distributed solutions based on the least-mean-squares (LMS) algorithm. The distributed algorithm of [8] focuses on the prediction of the graph signals and uses an efficient distributed graph sampling strategy based on a probabilistic approach. On the other hand, [9] proposes a diffusion LMS algorithm for the estimation of the coefficients of graph filters from streaming signals. However, in its current form, the distributed algorithm of [9] requires the use of all of nodes of the graph in the processing. The question that arises is whether that is really necessary, since sampling reduces the computational/memory burden, which is crucial in situations where the measurement and processing of data in every node is very costly.

In this work, we propose a sampling mechanism for the graph diffusion algorithm of [9] that changes adaptively the amount of sampled nodes in the graph based on mean-squared error (MSE) in the neighborhood of each node. Thus, the number of sampled nodes decreases when the MSE is low, allowing for fast convergence in the transient and a significant reduction in the number of sampled nodes in the steady state. We also observed a slightly better steady-state performance when compared to the case in which all nodes are sampled.

This work was supported by FAPESP under Grant 2017/20378-9 and by CNPq under Grants 132586/2018-5 and 304715/2017-4.

The paper is organized as follows. In Section II, we formulate the GSP problem and revisit the distributed algorithm of [9]. In Section III, we propose an adaptive sampling algorithm and in Section IV, we present an analysis to give insights about proper choices for its adaptation parameters. Simulation results are shown in Section V. Finally, Section VI closes the paper with the conclusions.

Notation. Normal font letters denote scalars, boldface lowercase letters denote vectors, and boldface uppercase letters denote matrices. The k-th entry of vector $\mathbf{x}$ is denoted by $[\mathbf{x}]_k$, and if $\mathcal{X}$ is a set, $|\mathcal{X}|$ denotes its cardinality. Furthermore, $(\cdot)^{\mathrm{T}}$ denotes transposition, $\mathrm{E}\{\cdot\}$ the mathematical expectation, $\|\cdot\|$ the Euclidean norm, $\mathrm{col}\{\cdot\}$ the stacking of its arguments to form a column vector, and $\mathrm{diag}\{\cdot\}$ a diagonal matrix with its arguments being the diagonal elements.

II. Diffusion Graph Adaptive Filtering

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph consisting of a set of nodes $\mathcal{V}$ with labels $k = 1, 2, \cdots, V$ and a set of edges $\mathcal{E}$. We represent a signal defined over $\mathcal{G}$ by the column vector $\mathbf{x}(n) = [x_1(n), \cdots, x_V(n)]^{\mathrm{T}} \in \mathbb{R}^V$, where $x_k(n)$ is the value of the signal at node k at time instant n [1]. Furthermore, let $\mathbf{A}$ be a $V \times V$ matrix that denotes the graph shift operator. Possible choices for $\mathbf{A}$ include the adjacency matrix [1] and the graph Laplacian matrix [2]. In particular, the (i, j)-th entry of the adjacency matrix is different from zero only if there is an edge linking nodes i and j, in which case it is equal to the weight of this edge. We then assume that the vector $\mathbf{x}(n)$ is processed to generate the filtered graph vector defined as [9], [11]

$$\mathbf{y}(n) = \sum_{\ell=0}^{M-1} h_\ell^{\mathrm{o}} \mathbf{A}^\ell \mathbf{x}(n-\ell) + \mathbf{v}(n), \tag{1}$$

where $h_0^{\mathrm{o}}, \cdots, h_{M-1}^{\mathrm{o}}$ are the M filter coefficients, $\mathbf{v}(n) = [v_1(n), \cdots, v_V(n)]^{\mathrm{T}} \in \mathbb{R}^V$ is an independent and identically distributed (i.i.d.) zero-mean noise vector assumed independent of any other signal and with covariance matrix $\mathbf{R}_v = \mathrm{diag}\{\sigma_{v_k}^2\}_{k=1}^V$.

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

The length-M column vector $\mathbf{h}$ that estimates $\mathbf{h}^{\mathrm{o}} = [h_0^{\mathrm{o}}, \cdots, h_{M-1}^{\mathrm{o}}]^{\mathrm{T}}$ can be obtained by minimizing [9]

$$J(\mathbf{h}) = \sum_{k=1}^{V} J_k(\mathbf{h}), \quad \text{with} \quad J_k(\mathbf{h}) \triangleq \mathrm{E}\{|y_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\mathbf{h}|^2\}, \quad (2)$$

where $y_k(n) = [\mathbf{y}(n)]_k$ and the vector

$$\mathbf{z}_k(n) \triangleq \mathrm{col}\{[\mathbf{A}^0\mathbf{x}(n)]_k, \cdots, [\mathbf{A}^{M-1}\mathbf{x}(n-M+1)]_k\} \quad (3)$$

aggregates the k-th entries of all $\{\mathbf{A}^{\ell}\mathbf{x}(n-\ell)\}_{\ell=0}^{M-1}$ and can be computed locally as pointed out in [9]. Following a diffusion adaptation strategy to minimize (2), [9] proposed two variations of a diffusion LMS algorithm: one based on the adapt-then-combine (ATC) strategy and another based on the combine-then-adapt (CTA) one. In this paper, we only consider the ATC strategy since the results can be extended straightforwardly to the CTA one. Furthermore, the ATC diffusion LMS algorithm of [9] is extended here to a normalized version:

$$\begin{cases} \boldsymbol{\psi}_k(n+1) = \mathbf{h}_k(n) + \mu_k(n)\mathbf{z}_k(n)e_k(n) \\ \mathbf{h}_k(n+1) = \sum_{j\in\mathcal{N}_k} w_{kj}\boldsymbol{\psi}_j(n+1) \end{cases}, \quad (4)$$

where

$$e_k(n) = y_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\mathbf{h}_k(n), \quad (5)$$

is the estimation error at node k, the M-length column vectors $\boldsymbol{\psi}_k(n)$ and $\mathbf{h}_k(n)$ represent respectively local and combined estimates of $\mathbf{h}^{\mathrm{o}}$ at node k, $\mathcal{N}_k$ denotes the neighborhood of node k (including k itself), and $\{w_{k,j}\}$ are non-negative weights that satisfy

$$w_{kj} \geqslant 0, \quad \sum_{j\in\mathcal{N}_k} w_{kj} = 1, \quad \text{and} \quad w_{kj} = 0 \ \text{ for } \ j \notin \mathcal{N}_k.$$

We also consider the normalized local step-size

$$\mu_k(n) = \widetilde{\mu}_k/[\delta + \gamma_k(n)], \quad (6)$$

where $0 < \widetilde{\mu}_k < 2$, $\gamma_k(n) = \lambda\gamma_k(n-1) + (1-\lambda)\|\mathbf{z}_k(n)\|^2$ is a low-pass filtered estimate for $\|\mathbf{z}_k(n)\|^2$ with forgetting factor $0 \ll \lambda < 1$ and initialization $\gamma_k(-1) = 0$, and δ is a small positive constant used to avoid large step sizes when $\gamma_k(n)$ becomes small. The choice of step size in this normalization version does not depend on the power of the input signal, which is particularly important for nonstationary signals [12].

III. The proposed sampling algorithm

At each iteration, the ATC diffusion algorithm (4) estimates the parameter vector $\mathbf{h}^{\mathrm{o}}$ from the data $\{y_k(n), \mathbf{z}_k(n)\}$. In our sampling proposal, we define the variable $\overline{s}_k(n)$ that assumes the values zero or one to decide if $e_k(n)$ should be computed at each iteration or not. Thus, we recast the adaptation step in (4) as

$$\boxed{\boldsymbol{\psi}_k(n+1) = \mathbf{h}_k(n) + \overline{s}_k(n)\mu_k(n)\mathbf{z}_k(n)e_k(n).} \quad (7)$$

If $\overline{s}_k(n) = 1$, $e_k(n)$ is computed as (5) and (7) coincides with the adaptation step of (4). On the other hand, if $\overline{s}_k(n) = 0$, $e_k(n)$ is not computed, $y_k(n)$ is not sampled, $\mathbf{z}_k(n)$, $\mu_k(n)$, and $\gamma_k(n)$ are not computed, and $\boldsymbol{\psi}_k(n+1) = \mathbf{h}_k(n)$.

To obtain the binary variable $\overline{s}_k(n) \in \{0, 1\}$ and select the nodes that should be sampled, we consider the variable $s_k(n) \in [0, 1]$ such that $\overline{s}_k(n) = 0$ for $s_k(n) < 0.5$ and $\overline{s}_k(n) = 1$ otherwise. We then minimize the following cost function with respect to $s_k(n)$:

$$J_{s,k}(n) = [s_k(n)]\beta s_k(n) + \big[1 - s_k(n)\big]\,\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k} e_i^2(n), \quad (8)$$

where $\beta > 0$ is a parameter introduced to control how much the sampling of the nodes is penalized. This cost function is a convex combination in $s_k(n)$ of the ℓ_1 norm $\|s_k(n)\|_1 = s_k(n)$, weighted by β, and the MSE in the neighborhood of node k. Thus, when the MSE is high (e.g., during transient), $J_{s,k}(n)$ is minimized by making $s_k(n)$ close to one so that the second term in (8) becomes small, which enforces node k to be sampled. On the other hand, when $\sum_{i\in\mathcal{N}_k} e_i^2(n)$ is small in comparison to $\beta s_k(n)$, the cost function is minimized by making $s_k(n)$ closer to zero, which means that node k should not be sampled. If β is chosen properly, this should happen in steady state. Since the expected value of $\sum_{i\in\mathcal{N}_k} e_i^2(n)$ in steady state may change from one application to another, β should be chosen accordingly.

Inspired by convex combination of adaptive filters (see, e.g., [13], [14] and their references), rather than directly adjusting $s_k(n)$, we update an auxiliary parameter $\alpha_k(n)$ which is deterministically related to $s_k(n)$ via [14]

$$s_k(n) = \phi_{\alpha_k}(n) \triangleq \frac{\mathrm{sgm}[\alpha_k(n)] - \mathrm{sgm}[-\alpha^+]}{\mathrm{sgm}[\alpha^+] - \mathrm{sgm}[-\alpha^+]}, \quad (9)$$

where $\mathrm{sgm}[x] = (1 + e^{-x})^{-1}$ is the sigmoidal function, and α^+ is the positive maximum value that $\alpha_k(n)$ can assume. We should notice that $s_k(n)$ attains values 1 and 0 for $\alpha_k(n) = \alpha^+$ and $\alpha_k(n) = -\alpha^+$, respectively. A common value adopted in the literature is $\alpha^+ = 4$.

By taking the derivative of (8) with respect to $\alpha_k(n)$, we obtain the following stochastic gradient descendent rule:

$$\alpha_k(n+1) = \alpha_k(n) + \mu_s\phi'_{\alpha_k}(n)\left[\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k} e_i^2(n) - \beta s_k(n)\right], \quad (10)$$

where $\mu_s > 0$ is a step size and

$$\phi'_{\alpha_k}(n) \triangleq \frac{ds_k(n)}{d\alpha_k(n)} = \frac{\mathrm{sgm}[\alpha_k(n)]\{1 - \mathrm{sgm}[\alpha_k(n)]\}}{\mathrm{sgm}[\alpha^+] - \mathrm{sgm}[-\alpha^+]}. \quad (11)$$

The form of the algorithm (10) does not allow to use it for sampling since the error signals in the neighborhood of node k (including k itself) must be computed to decide if node k should be sampled or not, which is clearly contradictory. To address this issue, we consider a heuristic modification by replacing $e_i(n)$ in (10) by $\varepsilon_i(n)$, which in turn denotes the latest measurement of $e_i(n)$ that we have access to. However, this modification may lead the algorithm to stop sampling all nodes permanently, which deteriorates its tracking capability. This can be avoided by replacing $\beta s_k(n)$ in (10) by $\beta\overline{s}_k(n)$, as

explained in the sequel. Incorporating these changes into (10), we arrive at

$$\boxed{\alpha_k(n+1)=\alpha_k(n)+\mu_s\phi'_{\alpha_k}(n)\left[\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k}\varepsilon_i^2(n)-\beta\bar{s}_k(n)\right]}. \quad (12)$$

This algorithm is named as adaptive sampling diffusion LMS (AS-dLMS). We should notice that $\mu_s\phi'_{\alpha_k}(n)$, the sum of $\varepsilon_i^2(n)$, and $\beta\bar{s}_k$ in (12) are always non-negative. Thus, supposing that in a given iteration node k is sampled ($\bar{s}_k = 1$) and that $\alpha_k = \alpha^+$, the algorithm (7) updates $\boldsymbol{\psi}_k$ to minimize the graph MSE. After a number of iterations, the MSE in the neighborhood of node k in (12) (first term inside the brackets) becomes smaller than $\beta\bar{s}_k = \beta$. Therefore, in a stationary environment, the correction term becomes negative, which enforces α_k to decrease until it also becomes negative. When $\alpha_k < 0$, $\bar{s}_k = 0$ and node k is no longer sampled. In this case, since $\beta\bar{s}_k = 0$, α_k increases, becoming positive again after a number of iterations and node k is sampled again. Thus, the algorithm does not stop sampling any node permanently, which is essential to detect changes in the environment. Furthermore, the greater the MSE in the neighborhood of node k, the sooner the sampling restarts.

This mechanism leads to a reduction of sampled nodes in steady state, decreasing the computational cost. If β is chosen appropriately, this reduction does not occur in the transient and AS-dLMS maintains the same convergence rate of the ATC algorithm with no sampling mechanism. However, we should mention that there is a slight increase of the cost during the transient, as shown in the simulations. Finally, different from the ATC algorithm with no sampling mechanism, AS-dLMS requires each sampled node to transmit $e_i^2(n)$ to its neighbors. This information can be sent bundled with the local estimates $\boldsymbol{\psi}_i$ so that the number of transmissions remains unaltered.

IV. Choosing the parameters β and μ_s

The good behavior of AS-dLMS depends on a proper choice of β and μ_s. In order to choose these parameters in a suitable manner, we analyze $\alpha_k(n)$ while node k is sampled. In this case, we can replace $\varepsilon_i^2(n)$ and $\beta\bar{s}_k(n)$ in (12) by $e_i^2(n)$ and β, respectively. Making these replacements, subtracting $\alpha_k(n)$ from both sides, and taking expectations, we get

$$\mathrm{E}\{\Delta\alpha_k(n)\}=\mu_s\mathrm{E}\left\{\phi'_{\alpha_k}(n)\left[\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k}e_i^2(n)-\beta\right]\right\}, \quad (13)$$

where $\Delta\alpha_k(n) \triangleq \alpha_k(n+1)-\alpha_k(n)$. To make the analysis more tractable, $\phi'_{\alpha_k}(n)$ and the term between brackets in (13) are assumed statistically independent. Although this assumption may seem unrealistic, simulation results suggest it is a reasonable approximation. Thus, Equation (13) can be recast as

$$\mathrm{E}\{\Delta\alpha_k(n)\}=\mu_s\mathrm{E}\{\phi'_{\alpha_k}(n)\}\left[\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k}\mathrm{E}\{e_i^2(n)\}-\beta\right]. \quad (14)$$

In order to stop sampling node k, α_k should decrease along the iterations until it becomes negative. Since $\mu_s\mathrm{E}\{\phi'_{\alpha_k}(n)\}$ is always positive, to enforce $\Delta\alpha_k(n)$ to be negative in the mean, β must satisfy

$$\beta > \frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k}\mathrm{E}\{e_i^2(n)\}. \quad (15)$$

Assuming that the order of the adaptive filter is sufficient and that $\tilde{\mu}_k$, $k=1,2,\cdots,V$, are chosen properly so that the gradient noise can be disregarded, it is reasonable to assume in steady state that $\mathrm{E}\{e_i^2(n)\} \approx \sigma_{v_i}^2$, which leads to

$$\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k}\mathrm{E}\{e_i^2(n)\} \leqslant \sigma_{\max}^2 \triangleq \max_{i\in\mathcal{V}}\sigma_{v_i}^2. \quad (16)$$

Thus, the condition

$$\boxed{\beta > \sigma_{\max}^2} \quad (17)$$

is sufficient to ensure that in the mean every node ceases to be sampled at some iteration in steady state. The iteration in which this occurs is different for each node since it depends on the neighborhood MSE. Furthermore, as explained in Section III, the absence of sampling of the nodes is not permanent. We should notice that (17) does not guarantee good performance in terms of MSE. In particular, high values of β can lead to $\mathrm{E}\{\Delta\alpha_k(n)\} < 0$ during transient, thus affecting the convergence rate and tracking capability of AS-dLMS. Simulation results suggest that if β is chosen in the interval $]\sigma_{\max}^2, 10\sigma_{\max}^2]$, the good behavior of AS-dLMS is preserved.

By choosing β properly, $\mathrm{E}\{\alpha_k(n)\} \approx \alpha^+$ during transient and $\mathrm{E}\{\Delta\alpha_k(n)\} \leqslant 0$ when $\bar{s}_k(n) = 1$ in steady state. In this case, we can find a proper value for μ_s by studying how fast we arrive at $\mathrm{E}\{\alpha_k(n)\} \leqslant 0$. Using (14) and (16), we get

$$\mathrm{E}\{\Delta\alpha_k(n)\} \leqslant \mu_s\mathrm{E}\{\phi'_{\alpha_k}(n)\}(\sigma_{\max}^2-\beta). \quad (18)$$

To simplify (18), we approximate $\phi'_{\alpha_k}(n)$ in the interval $[0,\alpha^+]$ by a straight line that crosses the points $(0,\phi'_0)$ and $(\alpha^+,\phi'_{\alpha^+})$, in which ϕ'_0 and ϕ'_{α^+} denote the values of ϕ'_{α_k} evaluated at $\alpha_k = 0$ and $\alpha_k = \alpha^+$, respectively. This approximation is given by

$$\phi'_{\alpha_k}(n) \approx \zeta\alpha_k(n) + \phi'_0, \quad (19)$$

where $\zeta = [\phi'_{\alpha^+} - \phi'_0]/\alpha^+$. For $\alpha^+ = 4$, this is a good approximation since its mean-squared error in $[0,\alpha^+]$ is of the order of 5×10^{-4}. Replacing (19) in (18), we obtain

$$\mathrm{E}\{\alpha_k(n+1)\} \lessapprox \mathrm{E}\{\alpha_k(n)\}(1+\zeta\theta) + \phi'_0\theta, \quad (20)$$

where $\theta = \mu_s(\sigma_{\max}^2-\beta)$. Since we assumed $\mathrm{E}\{\alpha_k(n)\}\approx\alpha^+$ during transient, we denote the first iteration of the steady state by n_0 and define $n_0 + \Delta n \triangleq n+1$. Then, considering $\mathrm{E}\{\alpha_k(n_0)\}\approx\alpha^+$ in (20) and applying it recursively, we obtain

$$\mathrm{E}\{\alpha_k(n_0+\Delta n)\} \lessapprox \alpha^+(1+\zeta\theta)^{\Delta n} + \phi'_0\theta\sum_{\eta=0}^{\Delta n-1}(1+\zeta\theta)^{\eta}. \quad (21)$$

After some algebraic manipulations, we arrive at

$$\mathrm{E}\{\alpha_k(n_0+\Delta n)\} \lessapprox [(\zeta\alpha^+ + \phi'_0)(1+\zeta\theta)^{\Delta n} - \phi'_0]/\zeta. \quad (22)$$

Since we are interested in studying how fast we arrive at $\mathrm{E}\{\alpha_k(n)\}\leqslant 0$ depending on our choice of μ_s, we set

$\mathrm{E}\{\alpha_k(n_0+\Delta n)\}$ to zero in (22). Thus, for a desired value of Δn, we should choose

$$\mu_s \gtrapprox \frac{\alpha^+}{(\beta-\sigma^2_{\max})(\phi'_0-\phi'_{\alpha^+})}\left[\left(\frac{\phi'_o}{\phi'_{\alpha^+}}\right)^{\frac{1}{\Delta n}}-1\right]. \quad (23)$$

From (23), we can observe that the smaller Δn, the larger the value of μ_s. We should notice that the larger β, the worse the approximation (23), since larger values of β may lead to $\mathrm{E}\{\alpha_k(n_0)\}$ significantly lower than α^+, contradicting our assumption. However, for $\beta \in]\sigma^2_{\max}, 10\sigma^2_{\max}]$, (23) agrees with the simulation results and enables a more suitable choice for μ_s, as shown next.

V. Simulation Results

In this section, we present simulation results obtained over an average of 100 realizations. For the sake of better visualization, we also filtered the curves by a moving-average filter with 64 coefficients. In each realization, a random graph with $V = 20$ nodes is generated. Each node is assigned X and Y coordinates following uniform distributions in the interval $[-1,\ 1]$. If the Euclidean distance between two nodes is less than or equal to 0.9, they are considered connected and an edge with unitary weight is created. An example is shown in Fig. 1(a). Graphs with isolated nodes are discarded and we consider the graph adjacency matrix as the shift operator. Furthermore, we assume that $x_k(n)$ and $v_k(n)$ are generated from i.i.d. Gaussian random processes with variances $\sigma^2_{x_k}=1$ and $\sigma^2_{v_k}$ as shown in Fig. 1(b) for $k=1,\cdots,V$. To simulate an abrupt change in the optimal system, we consider

$$\mathbf{h}^{\mathrm{o}}(n) = \begin{cases} [1.00 \quad 0.50 \quad 0.25]^{\mathrm{T}}, & \text{if } n \leqslant N/2, \\ [0.25 \quad 0.50 \quad 1.00]^{\mathrm{T}}, & \text{otherwise} \end{cases},$$

where N is the number of iterations. The combination weights are given by $w_{kj}=1/|\mathcal{N}_k|$ if $j\in\mathcal{N}_k$ and $w_{kj}=0$ otherwise. A different value of $\tilde{\mu}_k$ is considered for each node as shown in Fig. 1(c), and we set $\delta=10^{-5}$ and $\lambda=0.9$ in all simulations. As a performance indicator, we adopt the mean-square deviation (MSD) given by $\frac{1}{V}\sum_{k=1}^{V}\mathrm{E}\{\|\mathbf{h}^{\mathrm{o}}(n)-\mathbf{h}_k(n)\|^2\}$.

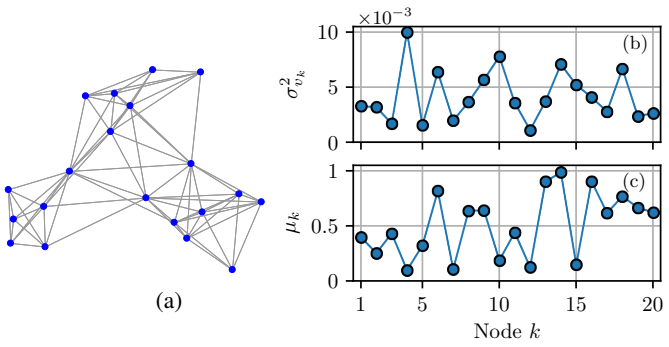

Fig. 1: (a) An example of a graph generated in one realization using GSPBOX [15]. (b) $\sigma^2_{v_k}$ and (c) $\tilde{\mu}_k$ across the graph.

Firstly, we compare the behavior of the AS-dLMS algorithm with the diffusion LMS algorithm that uses a random sampling technique, where V_s nodes are randomly sampled at every iteration [8]. For each algorithm, the MSD curves, the average amount of sampled nodes, and the number of multiplications are shown along the iterations in Figs. 2(a), 2(b), and 2(c), respectively. We can observe that the more nodes are sampled, the faster the convergence. AS-dLMS was able to detect the abrupt change in the optimal system since the sampling of the nodes does not cease permanently. Furthermore, all nodes were sampled during the transients, which led it to converge as fast as the diffusion LMS algorithm with all sampled nodes. Interestingly, when it achieves the steady state, only two to three nodes are sampled in the mean and there is a slight reduction in the MSD. This reduction also occurs when the random sampling technique is used at the expense of slower convergence. Furthermore, the lower the number of sampled nodes, the lower the achieved MSD. We may interpret this result as follows: during the transient, measuring y_k in each node k provides useful information to improve the estimates of the algorithm, but during the steady state it does not lead to further improvement. Moreover, the local estimate $\boldsymbol{\psi}_k$ from the observed data y_k tends to be noisier than $\mathbf{h}_k$ obtained by taking the average of the estimates in its neighborhood. Finally, we can also observe that during the transients, the computational cost of AS-dLMS is slightly higher than that of dLMS algorithm with all sampled nodes, which is widely compensated by the cost savings in steady state.

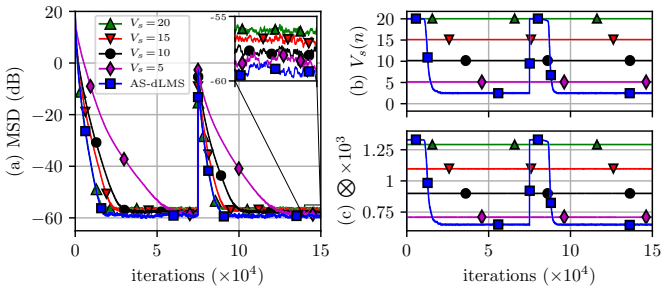

Fig. 2: Simulation results obtained with the diffusion LMS algorithm of [9] with a random sampling technique considering different amounts of sampled nodes and with AS-dLMS ($\beta=0.03$, $\mu_s=0.22$): (a) MSD curves, (b) Amount of sampled nodes per iteration, and (c) Number of multiplications per iteration.

In Fig. 3, we present the MSD curves and amount of sampled nodes along the iterations for AS-dLMS, by considering different values of β and a fixed step size ($\mu_s=0.22$). In Figs. 3(a) and 3(c), we consider the same scenario of Fig. 2, whereas in Figs. 3(b) and 3(d) the noise power was set to $\sigma^2_{v_k}=\sigma^2_{\max}=0.01$ for $k=1,\cdots,V$. In the first case, we note that (17) provides a conservative estimate for the minimum value of β. In the second case, $\beta=\sigma^2_{\max}$ maintains the sampling of all nodes, whereas $\beta>\sigma^2_{\max}$ ensures that some nodes cease to be sampled. Furthermore, we notice that the greater β, the lower the amount of sampled nodes in steady state, and the faster the nodes cease to be sampled, which is beneficial in terms of computational cost. However, high values of β may also compromise the performance of the algorithm, since in both cases $\beta=1$ led to low convergence rates.

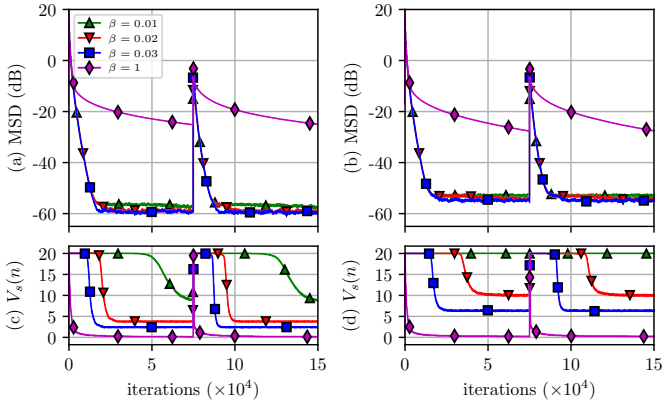

Fig. 3: Simulation results obtained with AS-dLMS ($\mu_s = 0.22$ and different values of β): (a) MSD curves and (c) amount of sampled nodes per iteration for $\sigma^2_{v_k}$ as in Figure 1(b); (b) and (d) Results for fixed $\sigma^2_{v_k} = 0.01$ for all k.

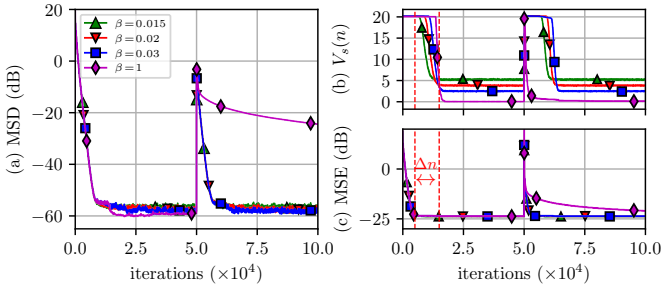

Fig. 4: Simulation results obtained with AS-dLMS (different values of β and μ_s adjusted by (23) for each case): (a) MSD curves, (b) Amount of sampled nodes per iteration, and (c) MSE Curves.

In Fig. 4, we use (23) to set values of μ_s for different values of β with $\Delta n = 10^4$. Besides the MSD and the amount of sampled nodes along the iterations, Fig. 4(c) also shows the graph MSE given by $\frac{1}{V}\sum_{k=1}^{V} \mathrm{E}\{e_k^2(n)\}$. We can observe from Figs. 4(b) and 4(c) that, before the abrupt change in the optimal system, the number of sampled nodes begins to fall approximately at the same time for all values of β and that the number of iterations between the beginning of the MSE steady-state regime and the end of this process is fairly close to Δn, which supports the validity of the approximation (23). In this regard, it should be noted that the start of steady-state regime in terms of MSE and MSD does not necessarily coincide, as can be seen by comparing Figs 4(a) and 4(c), which is an argument in favor of choosing high values for Δn. Comparing Figs. 3(a) and 4(a), we notice that choosing $\mu_s = 0.0044$ for $\beta = 1$ led to a better performance in terms of MSD before the abrupt change. However, after this change, the algorithm stops sampling the nodes while MSD is still high, leading again to a slow convergence. This occurs since we have $\mathrm{E}\{\alpha_k(n)\} \ll \alpha^+$ right before the change, and the high value for β prevents the algorithm from sampling enough nodes to improve its estimate again. This does not occur for $\beta \leqslant 0.03$ since the algorithm resumes the sampling of every node in the graph and the steady state of MSD is quickly achieved again. These results indicate that, even with a proper step size μ_s, high values of β (e.g., $\beta > 10\sigma^2_{\max}$) should be avoided as they can affect the performance of AS-dLMS.

VI. Conclusions

In this paper, we proposed modifications to the graph distributed LMS algorithm of [9] in order to incorporate a sampling technique. The proposed adaptive sampling mechanism uses the information from more nodes when the error in the network is high and less nodes otherwise. From the simulations, AS-dLMS maintains the convergence rate of the original algorithm of [9] during transient while displaying a lower computational cost in steady state. Furthermore, it was shown experimentally that the adoption of sampling techniques may lead to a slight reduction in steady-state MSD. We have also obtained theoretical results to help the choice of the adaptation parameters β and μ_s, which were also supported by the simulation results. At the same time, it was shown that poor choices of β and μ_s may compromise the convergence rate and tracking capabilities of the algorithm. For future work, we intend to obtain a performance analysis of AS-dLMS and test it in other scenarios using real-world data.

References

[1] A. Sandryhaila and J.M.F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.
[2] 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, pp. 83–98, May 2013.
[3] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, 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 Trans. Signal Process.*, vol. 64, pp. 3775–3789, Jul. 2016.
[5] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, pp. 4845–4860, Sep. 2016.
[6] 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, pp. 3584–3598, Jul. 2018.
[7] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *Proc. 26th EUSIPCO*, 2018, pp. 111–115.
[8] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Trans. Signal Process.*, vol. 65, pp. 4193–4208, Aug. 2017.
[9] R. Nassif, C. Richard, J. Chen, and A.H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE ICASSP*, 2018, pp. 4129-4133.
[10] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, vol. 7, Foundations and Trends in Machine Learning, now Publishers Inc., Hanover, MA, 2014.
[11] E. Isufi, G. Leus, and P. Banelli, "2-dimensional finite impulse response graph-temporal filters," in *Proc. IEEE GlobalSIP*, 2016, pp. 405–409.
[12] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, NJ, 2008.
[13] J. Arenas-Garcia, L. A. Azpicueta-Ruiz, M. T. M. Silva, V. H. Nascimento, and A. H. Sayed, "Combinations of adaptive filters: Performance and convergence properties," *IEEE Signal Process. Mag.*, vol. 33, pp. 120–140, Jan. 2016.
[14] M. Lázaro-Gredilla, L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, and J. Arenas-Garcia, "Adaptively biasing the weights of adaptive filters," *IEEE Trans. Signal Process.*, vol. 58, pp. 3890–3895, Jul. 2010.
[15] N. Perraudin, J. Paratte, D. I. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *arXiv*, vol. preprint:1408.5781, 2014.