

A Low-Cost Algorithm for Adaptive Sampling and Censoring in Diffusion Networks

Daniel G. Tiglea, Renato Candido, and Magno T. M. Silva, *Member, IEEE*

Abstract—Distributed signal processing has attracted widespread attention in the scientific community due to its several advantages over centralized approaches. Recently, graph signal processing has risen to prominence, and adaptive distributed solutions have also been proposed in the area. Both in the classical framework and in graph signal processing, sampling and censoring techniques have been topics of intense research, since the cost associated with measuring, processing and/or transmitting data throughout the entire network may be prohibitive in certain applications. In this paper, we propose a low-cost adaptive mechanism for sampling and censoring over diffusion networks that uses information from more nodes when the error in the network is high and from less nodes otherwise. It presents fast convergence during transient and a significant reduction in computational cost and energy consumption in steady state. As a censoring technique, we show that it is able to noticeably outperform other solutions. We also present a theoretical analysis to give insights about its operation, and to help the choice of suitable values for its parameters.

Index Terms—Diffusion strategies, adaptive networks, distributed estimation, graph signal processing, graph filtering, sampling on graphs, energy efficiency, convex combination.

I. INTRODUCTION

OVER the last decade, adaptive diffusion networks have become a consolidated tool for distributed parameter estimation and signal processing. Compared to centralized approaches, which require a central unit to receive and process data from the entire network, this kind of solution presents better scalability, autonomy, and flexibility [1]–[5]. As a result, adaptive diffusion networks are regarded as effective solutions in a handful of applications, such as target localization and tracking [1], spectrum sensing in mobile networks [1], [6], medical applications [7], among others.

These tools consist in a set of connected *agents*, or *nodes*, that are able to collect local data, carry out calculations and communicate with other nearby agents, i.e., its neighbors. The collective goal of the network is to estimate a parameter vector of interest without a central processing unit [1]–[10]. For this purpose, each node usually computes its own *local* estimate in what is called the *adaptation step*. Then, the neighboring nodes cooperate to reach a *global* estimate of the vector of interest. This stage is usually called the *combination step*. The order in which the adaptation and combination stages are performed

leads to two possible schemes: the adapt-then-combine (ATC) and combine-then-adapt (CTA) strategies [1]–[10].

More recently, graph signal processing (GSP) and graph adaptive filtering [11]–[16] have become topics of intense research within the signal processing community, particularly in the field of diffusion networks [9], [10], [17]. In comparison with the original distributed adaptation problem, graph adaptive filters incorporate information from the topology of the network in the adaptation step, which is useful in situations where this topology plays an important role in the dynamics of the signals of interest [9], [10]. This is the case in many network-structured applications that have emerged in recent years, such as smart grids, internet of things, transportation and communication networks, among many others [9]–[17]. In these cases, graphs are convenient modeling tools, since they are well suited to represent irregular structures.

When implementing distributed solutions, it is often desirable to restrict the number of data measurements and the amount of information transmitted across the network. For instance, when these strategies are implemented on wireless sensor networks, where energy consumption is often the most critical constraint [18]–[20]. Consequently, several solutions have been proposed to reduce the energy consumption associated with the communication between nodes. Some seek to reduce the amount of information sent in each transmission [21], [22], whereas others turn links off according to selective communication policies [18], [23]–[25]. Finally, there are the *censoring* techniques, which seek to avoid the transmission of information from certain nodes to any of their neighbors [19], [20], [26]–[28]. Thus, the censored nodes may turn their transmitters off, which saves energy and reduces the amount of information used in the processing [20], [28].

Furthermore, in certain situations, the cost associated with the measurement and processing of the data in every node at every time instant is prohibitively high, and thus some sort of *sampling* mechanism is required [16], [17]. Sampling can greatly reduce the computational cost and memory burden associated with the learning task, but it may also impact the performance of the algorithm. To illustrate this, Fig. 1 shows simulation results obtained in a stationary environment considering a network with 20 nodes, which run the ATC diffuse normalized least-mean-square (dNLMS) algorithm [1]–[3] in conjunction with a sampling technique where V_s nodes are randomly sampled at every iteration. The results are presented for $V_s \in \{5, 10, 15, 20\}$. The simulation scenario is described in detail in Section VI, and we adopt the network mean-square-deviation (NMSD) as a performance indicator [1]. To evaluate the computational cost, we present the average number of

This work was supported by FAPESP under Grant 2017/20378-9, by CNPq under Grants 132586/2018-5 and 304715/2017-4 and by CAPES under Grant 88887.512247/2020-00 and Finance Code 001.

The authors are with the Electronic Systems Engineering Department, Escola Politécnica, University of São Paulo, São Paulo, SP, Brazil, e-mails: {dtiglea, renatocan, magno}@lps.usp.br, ph. +55-11-3091-5134.

multiplications and sums per iteration for each value of V_s . They are presented as percentages of the number of operations performed when all $V_s = 20$ nodes are sampled. We observe that the less nodes are sampled, the lower the computational cost. Nonetheless, there is a clear impact on the convergence rate, which becomes increasingly slower as the number of sampled nodes decreases. Furthermore, we observe that the steady-state performance is not degraded by the sampling. One intuitive explanation for this is that sampling reduces the rate with which information enters the adaptive network, which leads to a slower convergence rate. However, once the algorithm achieves the steady state, the introduction of more information into the network usually does not improve the performance in a stationary environment.

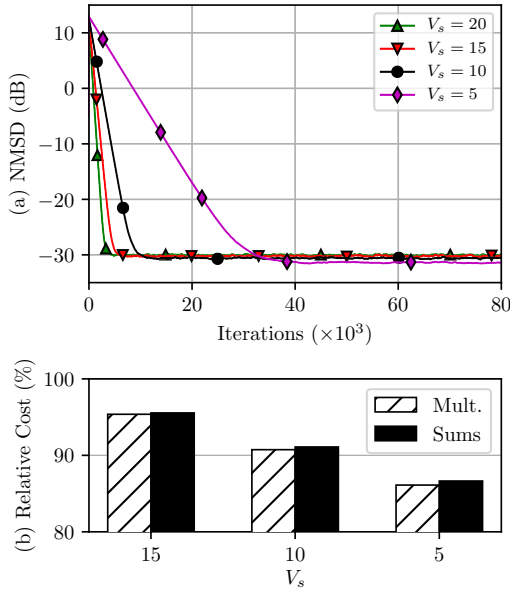


Fig. 1: Simulation results obtained for a 20-node network running ATC dNLMS with V_s nodes sampled per iteration. The simulation scenario is described in Section VI. (a) NMSD curves and (b) Relative number of multiplications and sums in comparison with the case where all $V_s = 20$ nodes are sampled.

The question that arises from this experiment is whether it is possible to design a more “intelligent” sampling strategy, in which more nodes are sampled when the estimation error is high (e.g., during transient) and less nodes otherwise, thus preserving the convergence rate of the algorithm. In our previous works, we have proposed a technique that can greatly reduce the computational cost during steady state while maintaining transient performance [29], [30]. Moreover, with slight modifications it can also be employed as a censoring strategy, allowing the nodes to save energy by transmitting less information to their neighbors. This paper extends these works in numerous ways:

- 1) We consider a unified formulation encompassing graph adaptive filters as well as “classical” adaptive diffusion networks, which have been studied separately in the literature [1], [9], [10], [29], [30].
- 2) Theoretical results obtained in [29], [30] are discussed in more detail and tested in different simulation scenarios.

- 3) We obtain sufficient conditions to ensure that our proposed technique reduces the computational cost in comparison with the original dNLMS algorithm.
- 4) We compare the censoring version of our proposed technique with other state-of-the-art censoring mechanisms [19], [26].
- 5) We test our sampling and censoring algorithms in a non-stationary environment following a random-walk model.

The paper is organized as follows. The unified formulation of diffuse adaptive networks is presented in Section II for both the classical distributed estimation problem and for GSP. In Section III, the adaptive sampling mechanism is introduced, and we analyze its behavior in Section IV. In Section V, the computational cost reduction of the proposed sampling mechanism is analyzed in more detail. Finally, simulation results are presented in Section VI, and Section VII closes the paper with the main conclusions and ideas for future work.

Notation. We use normal font letters to denote scalars, boldface lowercase letters for vectors, and boldface uppercase letters for matrices. Moreover, $[\mathbf{x}]_k$ denotes the k -th entry of the vector \mathbf{x} , and if \mathcal{X} is a set, $|\mathcal{X}|$ denotes its cardinality. Finally, $(\cdot)^T$ denotes transposition, $E\{\cdot\}$ the mathematical expectation, $\text{Tr}[\cdot]$ the trace of a matrix, and $\|\cdot\|$ the Euclidean norm. To simplify the arguments, we assume real data throughout the paper.

II. PROBLEM FORMULATION

Let us consider a network with a predefined topology and V nodes labeled $1, \dots, k, \dots, V$. Two nodes are considered neighbors if they can exchange information, and we denote by \mathcal{N}_k the neighborhood of node k including k itself. Furthermore, as depicted in Fig. 2, each node k has access at each time instant n to an input signal $u_k(n)$ and to a desired signal $d_k(n)$, modeled as [1]–[3], [9], [10]

$$d_k(n) = \mathbf{x}_k^T(n) \mathbf{w}^o + v_k(n), \quad (1)$$

where $v_k(n)$ is the measurement noise at node k , which is assumed to be independent of the other variables and zero-mean with variance $\sigma_{v_k}^2$, and \mathbf{w}^o and $\mathbf{x}_k(n)$ are M -length column vectors that represent respectively the optimal system and a processed version of the input signal $u_k(n)$.

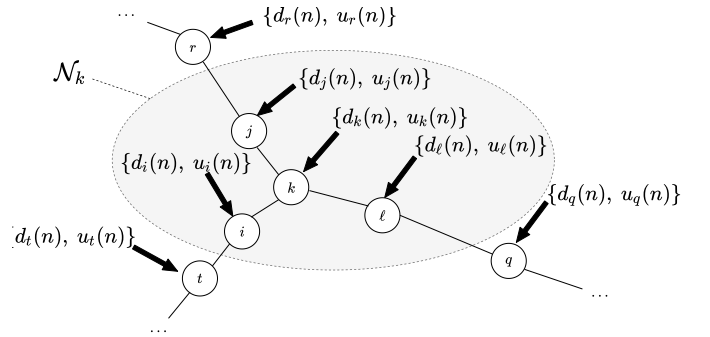


Fig. 2: Example of a diffusion network. In this case, the neighborhood of node k consists of the nodes i, j, k , and ℓ .

In the classical adaptation problem, $\mathbf{x}_k(n)$ is usually considered to be a regressor vector, given by [1]–[3]

$$\mathbf{x}_k(n) = [u_k(n) \ u_k(n-1) \ \dots \ u_k(n-M+1)]^T. \quad (2)$$

Thus, $d_k(n)$ can be seen as a noisy measurement of the output of the finite impulse response (FIR) optimal filter \mathbf{w}^o . In the context of graph adaptive filtering, $\mathbf{x}_k(n)$ is assumed to be related to the topology of the graph through [9], [10]

$$\mathbf{x}_k(n) = [\mathbf{u}(n)]_k [\mathbf{A}\mathbf{u}(n-1)]_k \cdots [\mathbf{A}^{M-1}\mathbf{u}(n-M+1)]_k^T, \quad (3)$$

where $\mathbf{u}(n) = [u_1(n) \ u_2(n) \ \cdots \ u_V(n)]^T$, and \mathbf{A} is the graph shift operator. Possible choices for \mathbf{A} include the adjacency matrix, the graph Laplacian matrix, among others [9], [10]. It is interesting to notice that the graph associated with \mathbf{A} can be different from the communication network [9]. For instance, nodes may be able to communicate with farther peers, whose effects on their signals are negligible. The relation between $\mathbf{x}_k(n)$ and $\mathbf{u}(n)$ in (3) can be interpreted as follows: $\mathbf{u}(n)$ represents the “raw” information available at each node of the network at the iteration n , whereas $\mathbf{x}_k(n)$ models the spreading of that information throughout the graph, which is the result of both a temporal and spacial shift, or “delay”. Moreover, \mathbf{w}^o models how exactly the graph topology and time lag affect the spreading of information, and $d_k(n)$ represents a noisy measurement of the information available at node k as a result of this spreading process [9], [10]. We should notice that there is a clear analogy to the tapped delay line commonly found in discrete-time filters [31].

The difference between the classic framework and the graph-based one lies in the role of the spatial aspect of the problem. In the former, the topology of the network does not influence the dynamics of the desired signal. Thus, $d_k(n)$ depends only on the signal $u_k(n)$ and on the measurement noise $v_k(n)$, and is independent of $u_i(n)$ for all $i=1, \dots, V, i \neq k$. This occurs since the information does not “travel” from one node to another. In graph signal processing, if the nodes i and k are immediate neighbors, $d_k(n)$ does depend on $u_i(n-1)$, since the information from one node spreads to its neighbors over time. Moreover, if nodes j and k are two-hop neighbors (i.e., it is possible to travel from node j to node k in two hops), $d_k(n)$ also depends on $u_j(n-2)$, and so forth. Hence, the topology of the network plays a major role in how the desired signal $d_k(n)$ unfolds at each node k . This makes graph adaptive filtering well suited for distributed problems where both time and space must be taken into consideration, e.g., meteorology [9], [10]. Nonetheless, despite the conceptual differences between both applications, in all cases Model (1) is assumed to hold. Thus a common mathematical formulation can be used to describe them to a certain extent.

In both situations, the objective of the network is to obtain an estimate \mathbf{w} of \mathbf{w}^o in a distributed manner by solving [1]–[3], [9], [10]

$$\min_{\mathbf{w}} J_{\text{global}}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{k=1}^V J_k(\mathbf{w}), \quad (4)$$

where $J_k(\mathbf{w})$ are the local costs at each node k , given by

$$J_k(\mathbf{w}) \triangleq \mathbb{E}\{|d_k(n) - \mathbf{x}_k^T(n)\mathbf{w}|^2\}. \quad (5)$$

Thus, at each iteration, every node k calculates a local estimate of \mathbf{w}^o in order to minimize its individual cost function $J_k(\mathbf{w})$. This is done by using only the data available locally, as well

as the information transmitted by neighboring nodes. Then, the nodes cooperate to form the global estimate \mathbf{w} .

Several adaptive solutions have been proposed in the literature to solve (4), one of them being the ATC dNLMS algorithm [1]–[3], [9], [10]. The adaptation and combination steps of this algorithm are respectively given by

$$\begin{cases} \psi_k(n+1) = \mathbf{w}_k(n) + \mu_k(n)\mathbf{x}_k(n)e_k(n) \\ \mathbf{w}_k(n+1) = \sum_{j \in \mathcal{N}_k} c_{jk}(n)\psi_j(n+1), \end{cases} \quad (6a)$$

$$\quad (6b)$$

where ψ_k and \mathbf{w}_k are the local and combined estimates of \mathbf{w}^o at node k . It can be shown that, when the adaptation and combination steps are done properly, \mathbf{w}_k converge to a single common solution for $k = 1, \dots, V$ [1]–[4]. Furthermore,

$$e_k(n) = d_k(n) - \mathbf{x}_k^T(n)\mathbf{w}_k(n) \quad (7)$$

is the estimation error, and

$$\mu_k(n) = \frac{\tilde{\mu}_k}{\delta + \|\mathbf{x}_k(n)\|^2} \quad (8)$$

is a normalized step size with $0 < \tilde{\mu}_k < 2$ and a small regularization factor $\delta > 0$ [1]. Moreover, $\{c_{jk}(n)\}$ are combination weights satisfying [2], [3]

$$c_{jk}(n) \geq 0, \sum_{j \in \mathcal{N}_k} c_{jk}(n) = 1, \text{ and } c_{jk}(n) = 0 \text{ for } j \notin \mathcal{N}_k. \quad (9)$$

Possible choices for $\{c_{jk}\}$ include static combination policies, such as the Uniform, Laplacian, Metropolis, and Relative Degree rules [1], [4], as well as adaptive schemes [8], [32], [33], such as the Adaptive Combination Weights (ACW) algorithm [32], [34]. ACW incorporates information from the noise profile across the network, and is obtained by solving an optimization problem with respect to $\{c_{jk}\}$ [32], [34]. Its equations are given by [34]

$$c_{jk}(n) = \begin{cases} \frac{\hat{\sigma}_{jk}^{-2}(n)}{\sum_{\ell \in \mathcal{N}_k} \hat{\sigma}_{\ell k}^{-2}(n)} & \text{if } j \in \mathcal{N}_k \\ 0, & \text{otherwise} \end{cases}, \quad (10)$$

where $\hat{\sigma}_{jk}^2(n)$ is updated as

$$\hat{\sigma}_{jk}^2(n) = (1 - \nu_k)\hat{\sigma}_{jk}^2(n-1) + \nu_k\|\psi_j(n+1) - \mathbf{w}_k(n)\|^2, \quad (11)$$

with $\nu_k > 0$ for $k = 1, \dots, V$. Hence, greater weights are assigned to the least noisy nodes [34]. We should notice that $\{c_{ik}(n)\}$ given by (10) satisfy (9). To avoid division by zero, in this paper we adopt a regularized version of (10), i.e., we respectively replace $\hat{\sigma}_{jk}^{-2}(n)$ and $\hat{\sigma}_{\ell k}^{-2}(n)$ by $[\delta_c + \hat{\sigma}_{jk}^2(n)]^{-1}$ and $[\delta_c + \hat{\sigma}_{\ell k}^2(n)]^{-1}$ in (10) where $\delta_c > 0$ is a small constant.

Finally, it is worth recalling that we could also employ a CTA strategy [1]–[10] in conjunction with other adaptive solutions [4], [5]. For simplicity, in this paper we will only consider the ATC strategy with the dNLMS algorithm. However, the results can be straightforwardly extended to other approaches.

III. THE SAMPLING ALGORITHM

At each iteration, the ATC dNLMS algorithm estimates the vector \mathbf{w}^o from the data $\{d_k(n), u_k(n)\}$. In our sampling proposal, we define the variable $\bar{s}_k(n)$ that assumes the values zero or one to decide if each node k should be sampled and if (6a) should be computed or not. Thus, we recast (6a) as

$$\psi_k(n+1) = \mathbf{w}_k(n) + \bar{s}_k(n)\mu_k(n)\mathbf{x}_k(n)e_k(n). \quad (12)$$

If $\bar{s}_k(n) = 1$, $d_k(n)$ is sampled, $e_k(n)$ is computed as in (7) and (12) coincides with (6a). In contrast, if $\bar{s}_k(n) = 0$, $d_k(n)$ is not sampled, $\mathbf{x}_k^T(n)\mathbf{w}_k(n)$, $e_k(n)$ and $\mu_k(n)$ are not computed, and $\psi_k(n+1) = \mathbf{w}_k(n)$.

To determine $\bar{s}_k(n)$, we define $s_k(n) \in [0, 1]$ such that

$$\bar{s}_k(n) = \begin{cases} 1, & \text{if } s_k(n) \geq 0.5, \\ 0, & \text{otherwise} \end{cases}. \quad (13)$$

We then minimize

$$J_{s,k}(n) = [s_k(n)]\beta\bar{s}_k(n) + [1-s_k(n)] \sum_{j \in \mathcal{N}_k} c_{jk}(n)e_j^2(n), \quad (14)$$

with respect to $s_k(n)$, where $\beta > 0$ is a parameter introduced to control how much the sampling of the nodes is penalized. When the error is high in magnitude or when node k is not being sampled ($\bar{s}_k = 0$), $J_{s,k}(n)$ is minimized by making $s_k(n)$ closer to one, leading to the sampling of node k . This ensures that the algorithm keeps sampling the nodes while the error is high and resumes the sampling of idle nodes at some point, enabling it to detect changes in the environment. In contrast, when node k is being sampled ($\bar{s}_k = 1$) and the error is small in magnitude in comparison to β , $J_{s,k}(n)$ is minimized by making $s_k(n)$ closer to zero, which leads the algorithm to stop sampling node k . This desirable behavior depends on a proper choice for β , which is addressed in Section IV.

Inspired by convex combination of adaptive filters (see [35], [36] and their references), rather than directly adjusting $s_k(n)$, we update an auxiliary variable $\alpha_k(n)$ related to it via [36]

$$s_k(n) = \phi[\alpha_k(n)] \triangleq \frac{\text{sgm}[\alpha_k(n)] - \text{sgm}[-\alpha^+]}{\text{sgm}[\alpha^+] - \text{sgm}[-\alpha^+]}, \quad (15)$$

where $\text{sgm}[x] = (1 + e^{-x})^{-1}$ is a sigmoidal function and α^+ is the maximum value α_k can assume. The function $\phi[\cdot]$ of (15) is a scaled and shifted version of $\text{sgm}[\cdot]$. It was proposed in [36] to prevent the adaptation process from stopping if $\alpha_k(n)$ becomes too large or too negative. We should notice that $\phi[\alpha^+] = 1$, $\phi[0] = 0.5$, and $\phi[-\alpha^+] = 0$. In the literature, $\alpha^+ = 4$ is usually adopted [36].

By taking the derivative of (14) with respect to $\alpha_k(n)$, we obtain the following stochastic gradient descent rule:

$$\alpha_k(n+1) = \alpha_k(n) + \mu_s \phi'[\alpha_k(n)] \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n)e_i^2(n) - \beta\bar{s}_k(n) \right], \quad (16)$$

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

$$\phi'[\alpha_k(n)] \triangleq \frac{ds_k(n)}{d\alpha_k(n)} = \frac{\text{sgm}[\alpha_k(n)]\{1 - \text{sgm}[\alpha_k(n)]\}}{\text{sgm}[\alpha^+] - \text{sgm}[-\alpha^+]}. \quad (17)$$

Equation (16) cannot be used for sampling since it requires the errors to be computed to decide if the nodes should be

sampled or not, which is contradictory. To address this issue, we replace $e_i(n)$ in (16) by its latest measurement we have access to, which is denoted by $\varepsilon_i(n)$. When the node is sampled, $\varepsilon_i(n) = e_i(n)$. We thus obtain

$$\alpha_k(n+1) = \alpha_k(n) + \mu_s \phi'[\alpha_k(n)] \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n)\varepsilon_i^2(n) - \beta\bar{s}_k(n) \right]. \quad (18)$$

Equation (18) is the foundation of the adaptive sampling mechanism. In conjunction with (12), it leads to an adaptive-sampling version of the dNLMS algorithm, named as adaptive-sampling diffusion NLMS (AS-dNLMS). This algorithm is summarized in Table I. Since (18) depends only on the estimation error at each sampled node, the proposed sampling technique can be extended to any adaptive diffuse algorithm.

It is interesting to notice that although we used $s_k(n)$ in the derivation of the algorithm, it does not have to be calculated explicitly, since it does not arise in (12) or (18). Instead, only $\bar{s}_k(n)$ and $\frac{ds_k(n)}{d\alpha_k(n)}$ appear. The latter can be stored in a look-up table, and the former is related to $\alpha_k(n)$ by

$$\bar{s}_k(n) = \begin{cases} 1, & \text{if } \alpha_k(n) \geq 0, \\ 0, & \text{otherwise} \end{cases}, \quad (19)$$

as can be seen from (13) and (15).

TABLE I: Summary of the AS-dNLMS algorithm

<i>% Initialization</i>
For each node $i = 1, \dots, V$, set $\alpha_i(0) \leftarrow \alpha^+$, $\bar{s}_i(0) \leftarrow 1$, $\varepsilon_i(n) \leftarrow 0$, $\mathbf{x}_i(0) = \mathbf{0}$, $\psi_i(0) \leftarrow \mathbf{0}$, $\mathbf{w}_i(0) \leftarrow \mathbf{0}$.
<i>% Then, repeat the following for every iteration $n \geq 0$ and every node k:</i>
<i>% Adaptation Step</i>
If $\alpha_k(n) \geq 0$, do :
$\bar{s}_k(n) \leftarrow 1$
Else, do :
$\bar{s}_k(n) \leftarrow 0$
End
If $\bar{s}_k(n) = 1$, do :
Update $\mathbf{x}_k(n)$ and $\ \mathbf{x}_k(n)\ ^2$
$\mu_k(n) \leftarrow \hat{\mu}_k / [\delta + \ \mathbf{x}_k(n)\ ^2]$
$e_k(n) \leftarrow d_k(n) - \mathbf{x}_k^T(n)\mathbf{w}_k(n)$
$\varepsilon_k(n) \leftarrow e_k(n)$
$\psi_k(n+1) \leftarrow \mathbf{w}_k(n) + \bar{s}_k(n)\mu_k(n)\mathbf{x}_k(n)e_k(n)$
Else, do :
$\psi_k(n+1) \leftarrow \mathbf{w}_k(n)$
End
<i>% Transmission</i>
Transmit $\psi_k(n+1)$ and $\varepsilon_k^2(n)$ to every node $i \in \mathcal{N}_k$
<i>% Combination Step</i>
$\alpha_k(n+1) \leftarrow \alpha_k(n) + \mu_s \phi'[\alpha_k(n)] \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n)\varepsilon_i^2(n) - \beta\bar{s}_k(n) \right]$
$\mathbf{w}_k(n+1) \leftarrow \sum_{j \in \mathcal{N}_k} c_{jk}(n)\psi_j(n+1)$

The proposed mechanism reduces the number of sampled nodes in steady state, decreasing the computational cost. If β is chosen appropriately, this reduction does not occur in the transient and the adaptive-sampling version of the algorithm maintains the same convergence rate as that of the original with no sampling mechanism. This comes at the expense of a slight increase of the cost during the transient, since the sampling algorithm requires the computation of an additional update equation per node per iteration. This will be explored in more detail in Section V. Furthermore, we should mention that when the node i is sampled, it is required to transmit $\varepsilon_i^2(n) = e_i^2(n)$ to its neighbors. Nonetheless, this information

can be sent bundled with the local estimates ψ_i so as to not increase the number of transmissions.

Finally, we remark that the algorithm described in Table I can be implemented in conjunction with any rule for the selection of combination weights. If an adaptive scheme for such selection is employed, the update of $\{c_{ik}(n)\}$ should also be included in Table I. Particularly, if ACW is considered in conjunction with AS-dNLMS and the sampling of node k ceased for a long period of time, the sampling mechanism could potentially harm the update of the combination weights. This occurs since in this case $\hat{\sigma}_{kk}$ could tend towards zero in (11) due to s_k being equal to zero in (12). To avoid this, for $j = k$, we replace $\psi_j(n+1)$ in (11) by $\bar{\psi}_k(n+1) \triangleq \bar{s}_k(n)\psi_k(n+1) + [1 - \bar{s}_k(n)]\bar{\psi}_k(n)$.

The Adaptive Sampling Algorithm as a Censoring Strategy

With a very simple modification, the proposed adaptive sampling mechanism can also be used as a censoring strategy. This alternate version of AS-dNLMS is obtained by not updating ψ_k at all when node k is not sampled. In other words, instead of using (12), we apply

$$\psi_k(n+1) = [1 - \bar{s}_k(n)]\psi_k(n) + \bar{s}_k(n) [\mathbf{w}_k(n) + \mu_k(n)\mathbf{x}_k(n)e_k(n)]. \quad (20)$$

Assuming that the nodes can store past information from their neighbors, this allows us to cut the number of communications between nodes, since in this case ψ_k and ε_k^2 remain static when $\bar{s}_k = 0$ and there is no need for node k to retransmit them. Thus, when node k is not sampled in this version of the algorithm, it only receives data and carries out (6b), and can therefore turn its transmitter off. This version of the proposed algorithm is named as adaptive-sampling-and-censoring diffusion NLMS (ASC-dNLMS), and it features a lower energy consumption as well as a computational cost reduction in comparison with the original dNLMS algorithm. Its pseudocode is identical to that of Table I, except for the “else” part of the second “if” condition, which does not exist in this version.

IV. THEORETICAL ANALYSIS

In the current section, we conduct a theoretical analysis of the proposed sampling mechanism. In IV-A, we show how to choose β so as to ensure that the nodes cease to be sampled during steady state. Then, in IV-B we study how its choice influences the expected number of sampled nodes per iteration. Finally, in IV-C, we analyze how fast the nodes cease to be sampled depending on the choice for μ_s , and how to select this parameter appropriately based on that information.

A. The parameter β and its effects on the algorithm

The parameter β plays a crucial role in the behavior of the AS-dNLMS. It influences the expected number of sampled nodes during steady state, and determines when the sampling mechanism begins to act.

Firstly, we study how to choose β so that we can ensure that every node will cease to be sampled at some point during

steady state. To do so, we examine (18) while node k is being sampled. In this case, $\varepsilon_i^2(n)$ and $\beta\bar{s}_k(n)$ can be replaced by $e_i^2(n)$ and β , respectively. Then, subtracting $\alpha_k(n)$ from both sides in (18) and taking expectations, we get

$$\mathbb{E}\{\Delta\alpha_k(n)\} = \mu_s \mathbb{E}\left\{\phi'[\alpha_k(n)] \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n)e_i^2(n) - \beta \right]\right\}. \quad (21)$$

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 (21) are assumed to be statistically independent. Although this assumption may seem unrealistic, simulation results suggest it is a reasonable approximation. Furthermore, for simplicity, we consider henceforth in our analysis that the combination weights are static and deterministic. Thus, we can write

$$\mathbb{E}\{\Delta\alpha_k(n)\} = \mu_s \mathbb{E}\{\phi'[\alpha_k(n)]\} \times \left[\sum_{i \in \mathcal{N}_k} c_{ik} \mathbb{E}\{e_i^2(n)\} - \beta \right]. \quad (22)$$

In order to stop sampling node k , $\alpha_k(n)$ should decrease along the iterations until it becomes negative. Since $\phi'[\alpha_k(n)]$ is always positive, to enforce $\mathbb{E}\{\Delta\alpha_k(n)\}$ to be negative while node k is sampled, β must satisfy

$$\beta > \sum_{i \in \mathcal{N}_k} c_{ik} \mathbb{E}\{e_i^2(n)\}. \quad (23)$$

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

$$\sigma_{\min}^2 \leq \sum_{i \in \mathcal{N}_k} c_{ik} \mathbb{E}\{e_i^2(n)\} \leq \sigma_{\max}^2, \quad (24)$$

where $\sigma_{\min}^2 \triangleq \min_i \sigma_{v_i}^2$, and $\sigma_{\max}^2 \triangleq \max_i \sigma_{v_i}^2$, $i = 1, \dots, V$. Thus, the condition

$$\beta > \sigma_{\min}^2 \quad (25)$$

is necessary (but not sufficient) if we wish to stop sampling the nodes at some point during steady state. On the other hand,

$$\beta > \sigma_{\max}^2 \quad (26)$$

is a sufficient (although not necessary) condition to ensure a reduction in the number of sampled nodes. Moreover, this ensures that *every node* will cease to be sampled at some iteration during steady state in the mean. When $\sigma_{\min}^2 = \sigma_{\max}^2$, i.e., every node is subject to the same level of noise power, (25) and (26) coincide and form a necessary and sufficient condition.

Moreover, given a certain value of β , we can analyze when the sampling mechanism will begin to act in terms of the mean-squared error (MSE). From (22) we observe that $\mathbb{E}\{\Delta\alpha_k(n)\} \geq 0$ as long as $\sum_{i \in \mathcal{N}_k} c_{ik}(n)\mathbb{E}\{e_i^2(n)\} \geq \beta$. Since we do not allow $\alpha_k(n)$ to become greater than α^+ , we conclude that $\mathbb{E}\{\alpha_k(n)\} = \alpha^+$ for $k = 1, \dots, V$ as long as $\text{MSE}_{\min}(n) > \beta$, where $\text{MSE}_{\min}(n) \triangleq \min_{i=1, \dots, V} \mathbb{E}\{e_i^2(n)\}$. In terms of mean-square deviation, this can be translated as follows. Applying the independence theory [31] to the estimation errors $e_i(n)$, $i = 1, \dots, V$ and assuming that $\mathbf{x}_i(n)$ is wide-sense stationary and statistically

independent from $\mathbf{w}_i(n)$, after some algebraic manipulations we can conclude that $E\{\alpha_k(n)\} = \alpha^+$ as long as

$$\text{MSD}_{\min}(n) > \frac{\beta - \sigma_{\min}^2}{\min_{i=1, \dots, V} E\{\|\mathbf{x}_i(n)\|^2\}},$$

where $\text{MSD}_{\min} \triangleq \min_{i=1, \dots, V} E\{\|\mathbf{w}^o(n) - \mathbf{w}_j(n)\|^2\}$ denotes the lowest MSD in the network. This indicates that, in the mean, the sampling mechanism does not act as long as MSD_{\min} remains greater than a threshold that is directly proportional to β . Consequently, we can be sure that no node will cease to be sampled in the mean during that period. Furthermore, the higher the value of β we adopt, the sooner $E\{\alpha_k(n)\}$ begins to decrease for $k = 1, \dots, V$, and thus the sooner the nodes cease to be sampled for a fixed step size μ_s .

B. The expected number of sampled nodes

Based on the previous section, we can estimate upper and lower bounds for the expected number V_s of sampled nodes in steady state. For this purpose, we consider each $\bar{s}_k(n)$ as an independent Bernoulli random variable during steady state that is equal to one with probability p_{s_k} or to zero with probability $1 - p_{s_k}$ for $k = 1, \dots, V$, with $0 \leq p_{s_k} \leq 1$. Thus,

$$V p_{s_{\min}} \leq E\{V_s\} \leq V p_{s_{\max}}, \quad (27)$$

where $p_{s_{\min}}$ and $p_{s_{\max}}$ are upper and lower bounds for p_{s_k} , $k = 1, \dots, V$.

It is useful to note that the sampling mechanism exhibits a cyclic behavior in steady state. Hence, we could approximate p_{s_k} by the expected “duty cycle” of the mechanism, i.e.,

$$\hat{p}_{s_k} = \frac{\theta_k}{\theta_k + \bar{\theta}_k}, \quad (28)$$

where θ_k denotes the expected number of iterations per cycle in which node k is sampled and $\bar{\theta}_k$ is the expected number of iterations in which it is not. Since we are only interested in estimating $p_{s_{\min}}$ and $p_{s_{\max}}$, we do not need to evaluate (28) for every k . Instead, we only need to estimate upper and lower bounds for θ_k and $\bar{\theta}_k$, which we respectively denote by θ_{\max} , θ_{\min} , $\bar{\theta}_{\max}$ and $\bar{\theta}_{\min}$.

For the sake of brevity, in this section we omit the intermediate calculations and skip to the final results concerning the estimation of these parameters. Nonetheless, a complete demonstration is provided in Appendix A.

Assuming that we can write

$$\sigma_{\min}^2 \leq \sum_{i \in \mathcal{N}_k} c_{ik} E\{\varepsilon_i^2(n)\} \leq \sigma_{\max}^2 \quad (29)$$

for $k = 1, \dots, V$ during steady state, we can estimate θ_{\max} by finding the maximum number of iterations any node can remain sampled in the mean. Considering a worst-case scenario, as well as the fact that every node must be sampled at least once during each cycle, and assuming that (26) is satisfied, we obtain after some approximations

$$\theta_{\max} = \max\{\sigma_{\max}^2/(\beta - \sigma_{\max}^2), 1\}. \quad (30)$$

Following an analogous procedure, the estimated lower bound θ_{\min} of θ_k can be obtained as

$$\theta_{\min} = \max\{\sigma_{\min}^2/(\beta - \sigma_{\min}^2), 1\}. \quad (31)$$

Lastly, for $\bar{\theta}_{\max}$ and $\bar{\theta}_{\min}$, we respectively obtain

$$\bar{\theta}_{\max} = \max\{(\beta - \sigma_{\min}^2)/\sigma_{\min}^2, 1\} \quad (32)$$

and

$$\bar{\theta}_{\min} = \max\{(\beta - \sigma_{\max}^2)/\sigma_{\max}^2, 1\}. \quad (33)$$

Thus, using (28), we can now estimate $p_{s_{\min}}$ and $p_{s_{\max}}$ as

$$\hat{p}_{\min} = \frac{\theta_{\min}}{\theta_{\min} + \bar{\theta}_{\max}} \quad (34)$$

and

$$\hat{p}_{\max} = \frac{\theta_{\max}}{\theta_{\max} + \bar{\theta}_{\min}}. \quad (35)$$

When $\beta < 2\sigma_{\min}^2$, we observe from (31) and (32) that $\theta_{\min} = \sigma_{\min}^2/(\beta - \sigma_{\min}^2)$ and $\bar{\theta}_{\max} = 1$. On the other hand, for $\beta \geq 2\sigma_{\min}^2$, (31) and (32) yield $\theta_{\min} = 1$ and $\bar{\theta}_{\max} = (\beta - \sigma_{\min}^2)/\sigma_{\min}^2$, respectively. In both cases, making these replacements in (34), we get

$$\hat{p}_{\min} = \sigma_{\min}^2/\beta. \quad (36)$$

Analogously, from (30), (33), and (35) we obtain

$$\hat{p}_{\max} = \sigma_{\max}^2/\beta. \quad (37)$$

Thus, replacing (36) and (37) in (27), we finally get

$$V \frac{\sigma_{\min}^2}{\beta} \leq E\{V_s\} \leq V \frac{\sigma_{\max}^2}{\beta}. \quad (38)$$

For $\beta < \sigma_{\max}^2$, (38) yields an upper bound that is greater than the total number V of nodes, which is not convenient. However, we can generalize it for all $\beta > 0$ by recasting it as

$$V \cdot \min \left\{ 1, \frac{\sigma_{\min}^2}{\beta} \right\} \leq E\{V_s\} \leq V \cdot \min \left\{ 1, \frac{\sigma_{\max}^2}{\beta} \right\}. \quad (39)$$

Replacing $\beta < \sigma_{\min}^2$ in (39) implies $E\{V_s\} = V$, which agrees with (25) being a necessary condition to ensure a reduction in the number of sampled nodes. Analogously, replacing $\beta > \sigma_{\max}^2$ we conclude that $E\{V_s\} < V$, which is in accordance with (26) being a sufficient condition. Moreover, the higher the parameter β , the smaller the amount of nodes sampled in the mean during steady state, as expected. Since there is a trade-off between the tracking capability and the gains in terms of computational cost provided by the sampling mechanism, we should care not to choose excessively high values for β , since they can deteriorate the performance in non-stationary environments. Simulation results suggest that if $\beta \leq 5\sigma_{\max}^2$, the good behavior of the algorithm is maintained. Moreover, the upper and lower bounds coincide when $\sigma_{\min}^2 = \sigma_{\max}^2$. Finally, the step size μ_s does not affect the number of sampled nodes.

C. Choosing the step size μ_s

In this section, we show how to choose a proper value for the parameter μ_s . To do so, we study how fast the nodes cease to be sampled (i.e., how fast we arrive at $E\{\alpha_k(n)\} \leq 0$) after the algorithm's initialization with $\alpha_k(0) = \alpha^+$ for $k = 1, \dots, V$. From (22) and (24), we can write

$$E\{\Delta\alpha_k(n)\} \leq \mu_s E\{\phi'[\alpha_k(n)]\}(\sigma_{\max}^2 - \beta). \quad (40)$$

Since in this case we consider $\alpha_k(n) \in [0, \alpha^+]$, approximating $\phi'[\alpha_k(n)]$ by its first-order Taylor expansion around $\alpha_k(n) = 0$ is not a suitable approach. Instead, we now approximate $\phi'[\alpha_k(n)]$ in that interval by a straight line that crosses the points $(0, \phi'_0)$ and $(\alpha^+, \phi'_{\alpha^+})$, in which ϕ'_0 and ϕ'_{α^+} respectively denote the value of $\phi'[\alpha_k(n)]$ evaluated at $\alpha_k(n) = 0$ and $\alpha_k(n) = \alpha^+$. This approximation is given by

$$\phi'[\alpha_k(n)] \approx \zeta \alpha_k(n) + \phi'_0, \quad (41)$$

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 (41) in (40), we obtain

$$E\{\alpha_k(n+1)\} \lesssim E\{\alpha_k(n)\}(1 + \zeta\rho) + \phi'_0\rho, \quad (42)$$

where $\rho = \mu_s(\sigma_{\max}^2 - \beta)$. Since we assumed $E\{\alpha_k(n)\} \approx \alpha^+$ 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 $E\{\alpha_k(n_0)\} \approx \alpha^+$ in (42) and applying it recursively, we obtain

$$E\{\alpha_k(n_0 + \Delta n)\} \lesssim \alpha^+(1 + \zeta\rho)^{\Delta n} + \phi'_0\rho \sum_{\eta=0}^{\Delta n-1} (1 + \zeta\rho)^\eta. \quad (43)$$

After some algebraic manipulations, we arrive at

$$E\{\alpha_k(n_0 + \Delta n)\} \lesssim [(\zeta\alpha^+ + \phi'_0)(1 + \zeta\rho)^{\Delta n} - \phi'_0]/\zeta. \quad (44)$$

Since we are interested in studying how fast we arrive at $E\{\alpha_k(n)\} \leq 0$ depending on our choice of μ_s , we set $E\{\alpha_k(n_0 + \Delta n)\}$ to zero in (44). Thus, for a desired value of Δn and $\beta > \sigma_{\max}^2$, we should choose

$$\mu_s > \frac{\alpha^+}{(\beta - \sigma_{\max}^2)(\phi'_0 - \phi'_{\alpha^+})} \left[\left(\frac{\phi'_0}{\phi'_{\alpha^+}} \right)^{\frac{1}{\Delta n}} - 1 \right]. \quad (45)$$

From (45), we observe that the smaller the Δn , the larger the value of μ_s , which is reasonable. Moreover, as β approaches σ_{\max}^2 , (45) yields increasingly large values for μ_s . Since (26) is a sufficient condition, the nodes may cease to be sampled even for $\beta \leq \sigma_{\max}^2$. When $\beta \approx \sigma_{\max}^2$ and $\sigma_{\min}^2 < \sigma_{\max}^2$, (45) may overestimate the value of μ_s required to cease the sampling of the nodes within Δn iterations. Nonetheless, this does not invalidate (45), since we are only interested in ensuring that the sampling will cease in at most Δn iterations.

V. COMPUTATIONAL COST ANALYSIS

If (26) is satisfied, the proposed mechanism leads to a reduction in the expected number of sampled nodes. However, this does not guarantee an advantage in terms of computational cost, since the sampling algorithm also requires a certain number of operations. Analyzing Table I, we see that the sampling mechanism requires $|\mathcal{N}_k| + 1 + \bar{s}_k(n)$ sums, $|\mathcal{N}_k| + 1$ multiplications and two comparisons per iteration for each sampled node k of the network. However, when node k is not sampled, AS-dNLMS does not have to calculate $\mathbf{x}_k^T(n)\mathbf{w}_k(n)$, $e_k(n)$, and $\mu_k(n)$, thus requiring $2M - \sum_{i \in \mathcal{N}_k} \bar{s}_i(n)$ less multiplications, $2M - |\mathcal{N}_k| + 1$ less sums, and one less division than dNLMS.

These results are summarized in Table II for both algorithms with ACW applied to classical distributed signal processing. We consider an implementation of $\phi'[\alpha_k(n)]$ through a look-up table, which is not taken into account in Table II.

In this section, we analyze which conditions have to be satisfied in order to ensure that the computational cost of AS-dNLMS is lower than that of dNLMS. In our analysis, we focus on the number of multiplications (\otimes). Analogous results can be obtained for the number of sums, but since they are less restrictive for AS-dNLMS, they are omitted here.

Firstly, we subtract the second row of Table II from the first one, obtaining

$$\Delta \otimes_k = 2M - 2(M+1)\bar{s}_k(n) - |\mathcal{N}_k|, \quad (46)$$

where $\Delta \otimes_k$ represents the difference in the number of multiplications between dNLMS and AS-dNLMS.

Summing $\Delta \otimes_k$ for $k = 1, \dots, V$ and taking expectations, we obtain for the whole network

$$E\{\Delta \otimes\} = 2VM - \sum_{k=1}^V [2(M+1)E\{\bar{s}_k(n)\} + |\mathcal{N}_k|], \quad (47)$$

where we have defined $\Delta \otimes \triangleq \sum_{k=1}^V \Delta \otimes_k$.

AS-dNLMS is advantageous over dNLMS in terms of computational cost when $E\{\Delta \otimes\} > 0$. Assuming again that \bar{s}_k can be seen as a Bernoulli random variable in steady state, we have $E\{\bar{s}_k(n)\} = p_{s_k}$. In this case, the worst-case scenario occurs if we consider $E\{\bar{s}_k(n)\} = p_{s_{\max}}$, since this minimizes $E\{\Delta \otimes\}$, leading to

$$E\{\Delta \otimes_{\min}\} = 2VM - \left[2(M+1)Vp_{s_{\max}} + \sum_{k=1}^V |\mathcal{N}_k| \right]. \quad (48)$$

Enforcing $E\{\Delta \otimes_{\min}\} > 0$, we conclude from (48) that, in order to ensure that AS-dNLMS requires less multiplications than dNLMS, we must have

$$p_{s_{\max}} < \frac{2VM - \sum_{k=1}^V |\mathcal{N}_k|}{2V(M+1)}. \quad (49)$$

Replacing $p_{s_{\max}}$ by $\hat{p}_{s_{\max}}$ from (37) in (49), we finally get

$$\beta > \frac{2V(M+1)}{2VM - \sum_{k=1}^V |\mathcal{N}_k|} \cdot \sigma_{\max}^2. \quad (50)$$

We remark that (50) is a sufficient (but not necessary) condition to ensure that AS-dNLMS presents a lower computational cost than dNLMS. The factor that multiplies σ_{\max}^2 in (50) is always greater than one, which is in accordance with our expectations. Moreover, the right-hand side of (50) approaches σ_{\max}^2 as M grows. Thus, the higher the order of the filter, the greater the computational cost reduction of AS-dNLMS in comparison with dNLMS for a fixed β . Furthermore, we can only ensure a decrease in the computational cost if

$$M > \frac{1}{2} \left(\frac{1}{V} \sum_{k=1}^V |\mathcal{N}_k| \right). \quad (51)$$

If (51) is not met, there is no finite value for $\beta > 0$ that satisfies (50), since this would imply $p_{s_{\max}} \leq 0$ in (49). Thus, the filter length should be greater than half the average

TABLE II: Computational cost comparison between dNLMS and AS-dNLMS with ACW for classical distributed signal processing: number of operations per iteration for each node k .

Algorithm	Multiplications (\otimes)	Sums (\oplus)	Divisions	Comparisons
dNLMS	$M(3 + \mathcal{N}_k) + 4$	$M(3 + \mathcal{N}_k) + 3$	$ \mathcal{N}_k $	0
AS-dNLMS	$\bar{s}_k(n)(2M + 2) + M(1 + \mathcal{N}_k) + \mathcal{N}_k + 4$	$\bar{s}_k(n)(2M + 2) + M(\mathcal{N}_k + 1) + \mathcal{N}_k + 2$	$ \mathcal{N}_k + \bar{s}_k(n) - 1$	2

neighborhood size in the network. This is a reasonable condition, since sparse and cluster topologies are more common in most applications [13]–[17]. Finally, we remark that we would obtain a different expression for β if we considered other diffuse algorithms [4], [5] and other rules for the selection of the combination weights [1], [8], [33].

VI. SIMULATION RESULTS

In this section, we present simulation results to illustrate the behavior of the proposed sampling mechanism and to validate the results of Sections IV and V. The results presented were obtained over an average of 100 independent realizations. For better visualization, we filtered the curves by a moving-average filter with 64 coefficients, unless otherwise stated.

We consider the ATC dNLMS algorithm and a heterogeneous network with 20 nodes. Half of the them use $\tilde{\mu}_k = 0.1$, while the other half uses $\tilde{\mu}_k = 1$, as depicted in Fig. 3(a). The network was generated randomly according to an Erdős-Renyi model [9], and the average degree of the nodes is 10. Furthermore, each node k is subject to a different noise variance $\sigma_{v_k}^2$, as shown in Fig. 3(b). For the optimal system \mathbf{w}^o , we consider a random vector with $M = 50$ coefficients uniformly distributed in $[-1, 1]$.

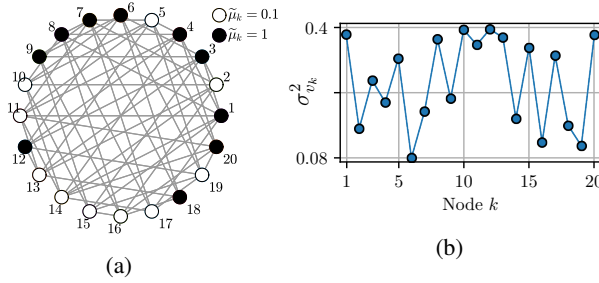


Fig. 3: (a) Network used in the simulations of Sections VI-A to VI-D. (b) Noise variance $\sigma_{v_k}^2$ for $k = 1, \dots, V$.

The combination weights are updated using the ACW algorithm with $\nu_k = 0.2$ for $k = 1, \dots, V$ [34], and we use $\delta = 10^{-5}$ and $\delta_c = 10^{-8}$ as regularization factors. As a performance indicator, we adopt the network mean-square-deviation (NMSD), given by

$$\text{NMSD}(n) = \frac{1}{V} \sum_{k=1}^V \mathbb{E} \{ \|\mathbf{w}^o(n) - \mathbf{w}_k(n)\|^2 \}. \quad (52)$$

Moreover, in some situations we also analyze the network mean-square-error (NMSE), given by

$$\text{NMSE}(n) = \frac{1}{V} \sum_{k=1}^V \mathbb{E} \{ e_k^2(n) \}. \quad (53)$$

This section is divided as follows. In VI-A, we compare AS-dNLMS with the random sampling technique of Fig. 1.

The theoretical results of Section IV are validated in VI-B, and in VI-C we compare ASC-dNLMS to other censoring techniques. Next, in VI-D, we study the tracking capability of the proposed techniques. Finally, in VI-C, we employ AS-dNLMS in the context of graph distributed adaptive filtering.

A. Comparison with Random Sampling

Firstly, we return to the simulation of Fig. 1 and compare the behavior of AS-dNLMS to that of the original dNLMS with the random sampling technique and different numbers of sampled nodes V_s . Nonetheless, here we simulate a change in the environment by flipping the parameter vector \mathbf{w}^o in the middle of each realization. For the network of Fig. 3, (51) yields $M > 5.4$, which is thus satisfied. For $M = 50$, (50) in its turn yields $\beta > 1.0610\sigma_{\max}^2$. We adjusted AS-dNLMS to obtain approximately the same computational cost as that of dNLMS with $V_s = 5$ nodes sampled. For this purpose, we adopted $\beta = 1.6\sigma_{\max}^2$ and $\mu_s = 0.06$. Figs. 4(a) and 4(b) present respectively the NMSD performance and the average number of multiplications per iteration. As seen in Fig. 1, the more nodes are sampled during the transient, the faster the convergence rate. Moreover, we observe that AS-dNLMS is able to detect the change in the optimal system and, since all nodes are sampled during the transients, it converges as fast as the dNLMS algorithm with all nodes sampled. It is interesting to note that the sampling of less nodes per iteration leads to a slight reduction of the steady-state NMSD in comparison with the original algorithm. The dNLMS algorithm with $V_s = 5$ nodes sampled achieves a steady-state NMSD that is approximately 1.3 dB lower than the one presented by the version with all nodes sampled. In its turn, AS-dNLMS reaches a steady-state NMSD that is 0.3 dB higher than that of the algorithm with five nodes sampled, but with a much faster convergence rate. One possible interpretation for this is that although the adaptation step is important for the convergence during the transient and for detecting changes in the environment, in steady state it introduces noise into the network, which the combination step tends to remove [29], [37]. Thus, by reducing the sampling rate during steady state, there may be a slight reduction in the NMSD. However, we remark that this reduction can be considered marginal in the case of Fig. 4(a). From Fig. 4(b) we observe that during the transients the computational cost of AS-dNLMS is slightly higher than that of the dNLMS algorithm with all nodes sampled, but decreases significantly in steady-state. A similar behavior is observed for the number of sums.

B. Validation of the Theoretical Analysis

In order to validate (39), we also tested the AS-dNLMS algorithm in a stationary environment with different values

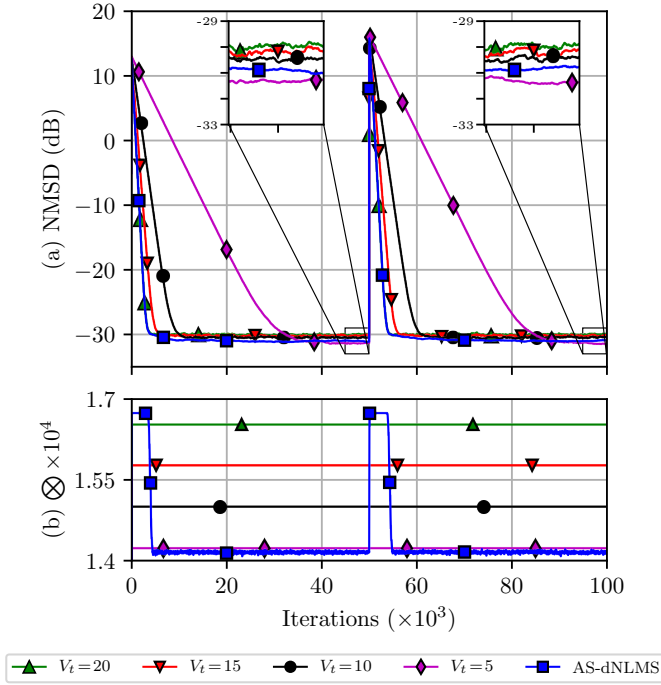


Fig. 4: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 1.6\sigma_{\max}^2$, $\mu_s = 0.06$). (a) NMSD curves and (b) Multiplications per iteration.

of $\beta \geq \sigma_{\min}^2$ and three methods for the selection of the combination weights: the Uniform and Metropolis rules [1], and the ACW algorithm [34]. Two scenarios were considered: one with the noise power in the network distributed as in Fig. 3(b), and another where $\sigma_{v_k}^2 = 0.4$ for $k = 1, \dots, V$. The results are shown in Fig. 5(a) and 5(b), respectively. For the ease of visualization, they are presented in terms of β_r , defined as $\beta_r \triangleq \beta/\sigma_{\max}^2$. Along with the experimental data, the predicted upper and lower bounds $\hat{V}_{s_{\max}}$ and $\hat{V}_{s_{\min}}$ are presented for each β_r using dashed lines. We should notice that these bounds coincide in Fig. 5(b), since $\sigma_{\min}^2 = \sigma_{\max}^2$ in this case. Moreover, in Fig. 5(a), the upper bound remains fixed at $V_{s_{\max}} = V = 20$ for $\beta \leq \sigma_{\max}^2$. We also observe from Fig. 5 that the higher β is, the less nodes are sampled in both scenarios, as expected. Furthermore, the experimental data lie between the theoretical bounds for all combination rules and for all values of β_r in Fig 5(a). On the other hand, from 5(b) we notice that the theoretical model slightly overestimates the number of sampled nodes for $1 < \beta_r \leq 20$. One possible explanation for this resides in the fact that the Assumption (29) translates to $E\{\varepsilon_k^2(n)\} = \sigma_{\max}^2$ for $k = 1, \dots, V$ in the case of Fig. 5(b). However, since $\bar{s}_k(n)$ and $\varepsilon_k(n)$ are not independent, $E\{\varepsilon_k^2(n)\} \neq \sigma_{v_k}^2 = \sigma_{\max}^2$. In general, we observe that $E\{\varepsilon_k^2(n)\} \leq \sigma_{v_k}^2$, which can be attributed to ε_k remaining at a fixed value for possibly long periods of time, which tends to reduce its variance. Furthermore, the difference between the theoretical and analytical results in Fig. 5(b) is especially noticeable for the ACW algorithm. This can be attributed to the fact that the analysis presented in Section IV was derived considering static combination weights, which is not the case

of ACW. Nonetheless, the scenario considered in Fig. 5 is not realistic, since some level of noise power discrepancy across the network is expected in most environments [1]–[5]. Lastly, in both Fig. 5(a) and Fig. 5(b), the adoption of the ACW algorithm led to a smaller number of sampled nodes in comparison with the Uniform and Metropolis rules.

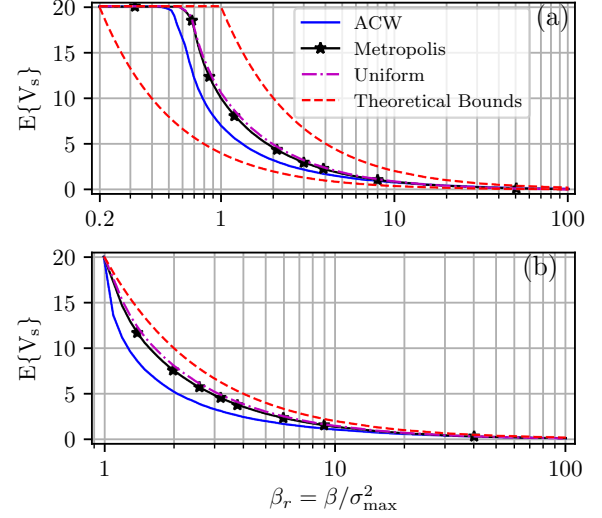


Fig. 5: Theoretical bounds and average number of nodes sampled by AS-dNLMS with three combination rules as a function of $\beta \geq \sigma_{\min}^2$. (a) $\sigma_{v_k}^2$ as in Fig. 3. (b) $\sigma_{v_k}^2 = 0.4$ for $k = 1, \dots, V$.

In Fig. 6, we test (45) by using it to set the step size μ_s for different values of β with $\Delta n = 3000$. In Fig. 6(a) we show the NMSD curves, in Fig. 6(b) the number of sampled nodes per iteration, and in Fig. 6(c) the NMSE.

From Figs. 6(b) and 6(c) we observe that, before the abrupt change in the optimal system, the number of sampled nodes stabilizes at approximately the same time for all $\beta_r > 1.1$. For $\beta_r = 1.1$, we can notice that (45) slightly overestimates μ_s . This is expected for $\beta_r \approx 1$, as discussed in Section IV-C. In this case, AS-dNLMS ceased to sample the nodes before reaching the steady state in terms of NMSD, which compromised the convergence rate. This illustrates the importance of a proper choice for μ_s as well as β . Nonetheless, since the sampling of the nodes ceased in less than Δn iterations after the beginning of the steady state in terms of NMSE, the results obtained support the validity of (45). However, this shows that some care must be taken when using (45) for $\beta \gtrsim \sigma_{\max}^2$. Lastly, we observe that the sampling of less nodes leads to a slight reduction in the steady-state NMSD, as in Fig. 4(a).

In Fig. 7 we repeated the experiments of Fig. 6 with higher values of β_r . We observe that the number of sampled nodes stabilizes almost simultaneously for all values of β_r before the abrupt change and that the performance of AS-dNLMS is maintained before the change in the optimal system. Nonetheless, after the change occurs, the NMSD is affected for $\beta_r \geq 8$. The higher the parameter β , the more intense the deterioration in performance. The difference in the behavior of the algorithm before and after the change in the optimal system can be explained by the initialization with $\alpha_k(0) = \alpha^+$

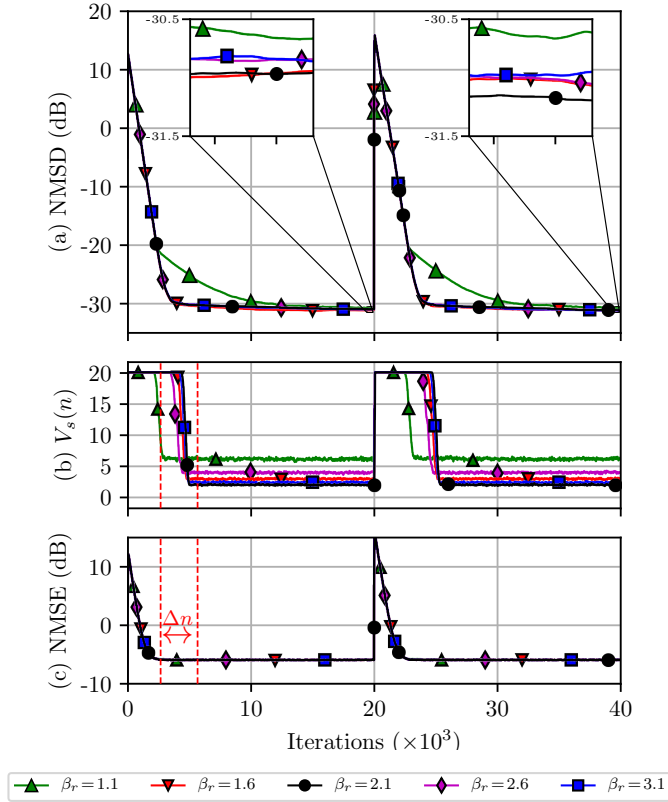


Fig. 6: Simulation results obtained with $1.1\sigma_{\max}^2 \leq \beta \leq 3.1\sigma_{\max}^2$ and μ_s adjusted by (45) for each case. (a) NMSD curves, (b) Number of sampled nodes per iteration, and (c) NMSE curves.

for $k = 1, \dots, V$. In contrast, right before the abrupt change, we have $\alpha_k(n) \ll \alpha^+$. Thus, the algorithm ceases to sample the nodes earlier in this case, as can be seen in Fig. 7(b). We recall that $\beta \leq 5\sigma_{\max}^2$ seems to be a safe interval for the choice of β , according to various simulations results.

C. Application as a Censoring Technique

In this section, we test the ASC-dNLMS algorithm and compare it to other techniques found in the literature, namely, the ACW-Selective (ACW-S) algorithm of [19] and the energy-aware diffusion algorithm (EA-dNLMS) of [26]. Assuming that the nodes can broadcast their data to all of their neighbors at once, we present in Fig. 8(a) the NMSD curves, in Fig. 8(b), the number $V_t(n)$ of transmitting nodes per iteration, i.e. the amount of broadcasts in the network, and in Fig. 8(c) the average number of multiplications per iteration.

The algorithms were adjusted to achieve approximately the same level of steady-state NMSD. Table III shows the adopted values for the parameters of each solution. In this regard, it is worth noting that EA-dNLMS presents a high number of parameters, which may be difficult to adjust. We consider the version of EA-dNLMS that allows node k to receive and combine the estimates from its neighbors even when it is not transmitting [26], and we adopt a normalized step size following (8). For comparison, we also present results obtained with the original dNLMS and with the non-cooperative case.

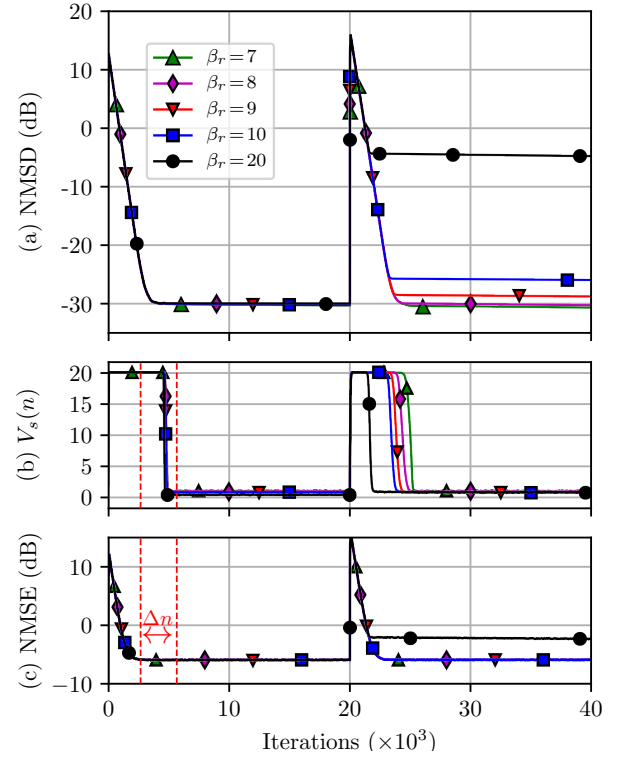


Fig. 7: Simulation results obtained with $7\sigma_{\max}^2 \leq \beta \leq 20\sigma_{\max}^2$ and μ_s adjusted by (45) for each case. (a) NMSD curves, (b) Number of sampled nodes per iteration, and (c) NMSE curves.

Unlike AS-dNLMS, which led to a slight reduction in the steady-state NMSD in comparison to dNLMS with all nodes sampled, ASC-dNLMS achieves a slightly higher level of NMSD in steady state in comparison with the original algorithm. The same occurs for the ACW-S and EA-dNLMS algorithms, as can be seen in Fig. 8(a). We observe that EA-dNLMS presents a notably slower convergence rate in comparison with ACW-S and ASC-dNLMS, which converge at a rate similar to that of dNLMS. On the other hand, from Fig. 8 we see that ACW-S utilizes a comparatively high number of broadcasts, thus saving less energy. During steady state, both ACW-S and EA-dNLMS transmit more than the proposed ASC-dNLMS, which maintains all transmissions during the transient but drastically reduces the number of broadcasts after converging. Thus, the proposed technique saves more energy in steady state while preserving the convergence rate.

Lastly, it should be noted from Fig. 8(c) that ASC-dNLMS requires approximately the same number of multiplications per iteration as EA-dNLMS, while leading to 50% less broadcasts. In comparison with ACW-S, ASC-dNLMS requires 21% more multiplications, but leads to 81% less broadcasts. This gap in the number of multiplications is due to the fact that ASC-dNLMS considers the local estimates in the combination step even if they were not updated in the current iteration. In contrast, ACW-S considers that the neighborhood of each node only includes its neighbors that have broadcast their estimate in the current iteration. Despite this, we recall that in censoring applications we are mostly concerned with the number of transmissions, since they are usually the main responsible for

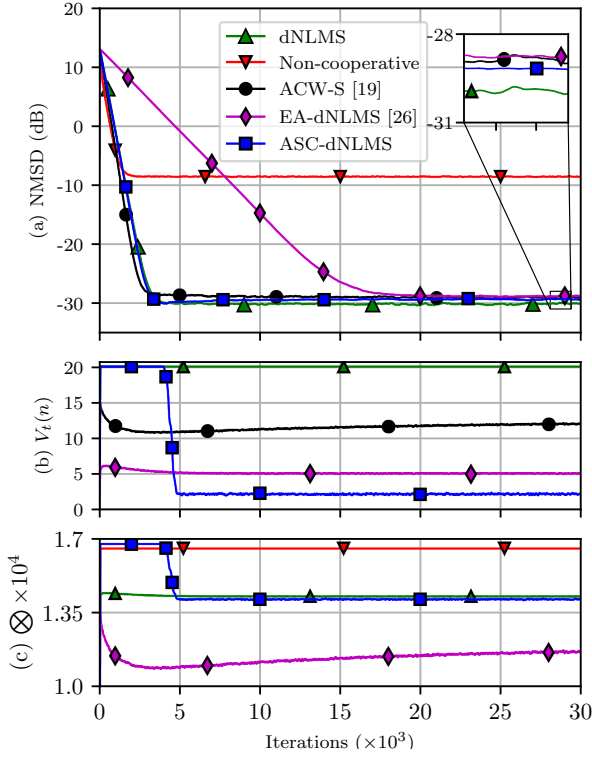


Fig. 8: Comparison between the ASC-dNLMS, ACW-S and EA-dNLMS algorithms. The parameters adopted are shown in Table III. (a) Steady-state NMSD curves. (b) Number of broadcasts and (b) multiplications per iteration.

energy consumption in the network [19], [20], [26]. Moreover, given the percentage differences, ASC-dNLMS can be deemed an efficient solution for censoring.

TABLE III: Parameters used in the simulations of Fig. 8

ACW-S [19]	$E_T = 1, E_T = 2$
EA-dNLMS [26]	$E_{Act} = 33.5966 \cdot 10^{-3}, E_{Tx} = 15.16 \cdot 10^{-3}, K_{\ell,1} = 2, K_{\ell,2} = 0.5, K_g = 2, \gamma_g = 2, \gamma_\ell = 2, \delta = 0.5, \rho = 0.01, r = 2$
ASC-dNLMS	$\beta = 2.1\sigma_{\max}^2, \mu_s = 0.0333$

D. Random-Walk Tracking

As can be observed from Fig. 7, increased values of β may hinder the tracking capability of AS-dNLMS. Thus, in this section, we investigate the behavior of the algorithm in nonstationary environments following a random-walk model, in which the optimal solution $\mathbf{w}^o(n)$ varies according to

$$\mathbf{w}^o(n) = \mathbf{w}^o(n-1) + \mathbf{q}(n), \quad (54)$$

where $\mathbf{q}(n)$ is a zero-mean i.i.d. column vector with length M and autocovariance matrix $\mathbf{Q} = E\{\mathbf{q}(n)\mathbf{q}^T(n)\}$ independent of any other signal [8], [31]. In our experiments, we consider a Gaussian distribution for $\mathbf{q}(n)$ with $\mathbf{Q} = \sigma_q^2 \mathbf{I}$, where \mathbf{I} denotes the identity matrix. In Fig. 9, we present the results obtained with the AS-dNLMS algorithm and different values of β as a function of $\text{Tr}[\mathbf{Q}]$. For each β_r , we maintained the corresponding step size μ_s used in the simulations of Fig. 6.

For comparison, we also show the results obtained with the dNLMS algorithm with all nodes sampled. In Fig. 9(a), we present the steady-state levels of NMSD, in Fig. 9(b) the average number of sampled nodes per iteration and in Fig. 9(c) the steady-state NMSE. The results presented were obtained by averaging the data over the last 600 iterations of each realization, after all the algorithms achieved steady state.

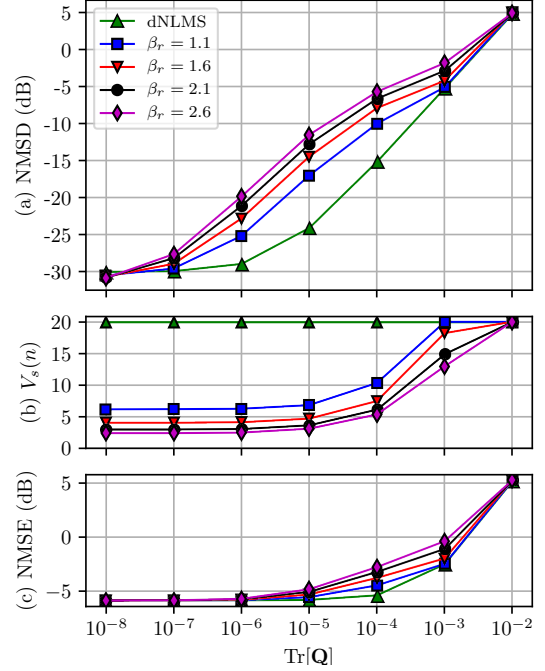


Fig. 9: Simulation results in a nonstationary environment following Model (54). (a) Steady-state NMSD, (b) Number of nodes sampled per iteration, and (c) Steady-state NMSE.

From Fig. 9(a) we can observe that, in slowly-varying environments ($\text{Tr}[\mathbf{Q}] = 10^{-8}$), the performance of AS-dNLMS is similar to that of dNLMS with all nodes sampled. However, for $10^{-7} \leq \text{Tr}[\mathbf{Q}] \leq 10^{-3}$, there is a degradation in performance in comparison with dNLMS. The higher the parameter β , the more intense this deterioration becomes for a fixed value of $\text{Tr}[\mathbf{Q}]$. For $\text{Tr}[\mathbf{Q}] \leq 10^{-5}$ and a fixed β , this deterioration in comparison with dNLMS intensifies with the increase of $\text{Tr}[\mathbf{Q}]$. On the other hand, for $\text{Tr}[\mathbf{Q}] > 10^{-5}$, the difference in performance begins to decrease as the variations in the optimal system become faster. This can be explained by analyzing Figs. 9(b) and 9(c). We observe that, when the environment varies slowly or moderately, the number of nodes sampled by the AS-dNLMS is not significantly affected by the increase of $\text{Tr}[\mathbf{Q}]$. This occurs since the effects of the changes in the optimal system are small in comparison with those of the measurement noise for $\text{Tr}[\mathbf{Q}] < 10^{-5}$, and thus the NMSE does not increase noticeably, as seen in Fig. 9(c). However, as these variations become faster, they begin to affect the estimation error more intensely, and the NMSE starts to increase for $\text{Tr}[\mathbf{Q}] \geq 10^{-5}$, leading to a gradual rise in the number of sampled nodes in Fig. 9(b). For $\text{Tr}[\mathbf{Q}] = 10^{-2}$, the algorithm does not cease to sample any of the nodes for $\beta_r \leq 2.6$, and thus its performance matches that of dNLMS.

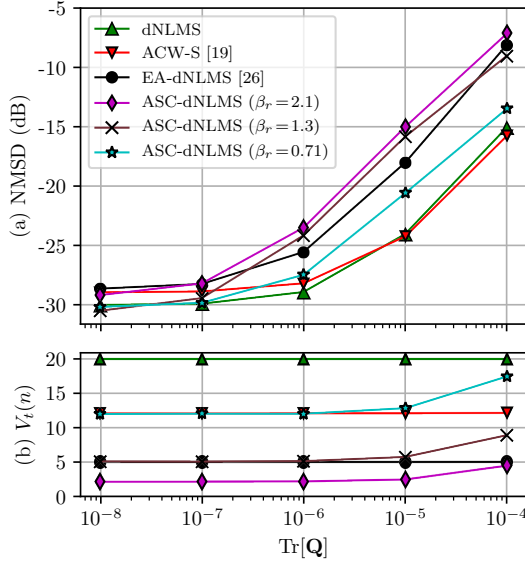


Fig. 10: Simulation results in a nonstationary environment following Model (54) with the algorithms listed in Table III. (a) Steady-state NMSD, and (b) Broadcasts per iteration.

Next, we repeated the experiment of Fig. 9 for ASC-dNLMS, ACW-S and EA-dNLMS with the parameters of Table III. The results are shown in Fig. 10. We also present the results obtained with ASC-dNLMS with $\beta_r = 1.3$ and $\beta_r = 0.71$, which were respectively adjusted to lead to the same number of broadcasts as those of EA-dNLMS and ACW-S for $\text{Tr}[\mathbf{Q}] \leq 10^{-6}$. Finally, we also show results obtained with the dNLMS algorithm. We observe from Fig. 10(a) that ASC-dNLMS with $\beta_r = 2.1$ achieves a performance similar to that of the other solutions for $\text{Tr}[\mathbf{Q}] = 10^{-8}$ and $\text{Tr}[\mathbf{Q}] = 10^{-7}$. However, it is outperformed for $\text{Tr}[\mathbf{Q}] \geq 10^{-6}$. It also employs less transmissions than any other solution in these scenarios. With $\beta_r = 1.3$, ASC-dNLMS outperforms EA-dNLMS for $\text{Tr}[\mathbf{Q}] \leq 10^{-7}$ and $\text{Tr}[\mathbf{Q}] = 10^{-4}$, although its NMSD is higher for $\text{Tr}[\mathbf{Q}] = 10^{-6}$ and $\text{Tr}[\mathbf{Q}] = 10^{-5}$. With $\beta_r = 0.71$, ASC-dNLMS outperforms ACW-S for $\text{Tr}[\mathbf{Q}] \leq 10^{-7}$, while the opposite occurs for other values of $\text{Tr}[\mathbf{Q}]$. The results suggest that ASC-dNLMS generally outperforms ACW-S and EA-dNLMS in stationary or slowly-varying environments while utilizing the same number of transmissions. Moreover, in these cases it can achieve a comparatively similar performance while transmitting less. However, ASC-dNLMS must be employed with caution in scenarios in which the optimal system changes rapidly. Finally, we can control the trade-off between energy saving and performance by adjusting β .

E. Application in Graph Adaptive Filtering

In this section, we test the proposed sampling algorithm on a graph diffuse adaptive filter. For this experiment, we consider a temperature dataset of daily average measurements from 12/25/2001 to 12/21/2012 at $V = 100$ weather stations across Brazil, as depicted in Fig. 11 [38]. Each station corresponds to a node of a directed weighted graph in which each node k is

connected to the six nearest stations. Denoting this set by $\mathcal{N}_{\mathbf{A}_k}$, each element \mathbf{A}_{kj} of the adjacency matrix \mathbf{A} is given by [9]

$$\mathbf{A}_{kj} = \begin{cases} \frac{e^{-g_{kj}^2}}{\sum_{\ell \in \mathcal{N}_{\mathbf{A}_k}} e^{-g_{\ell k}^2} \sum_{i \in \mathcal{N}_{\mathbf{A}_j}} e^{-g_{ji}^2}}, & \text{if } j \in \mathcal{N}_{\mathbf{A}_k} \\ 0, & \text{otherwise} \end{cases}, \quad (55)$$

where g_{kj} is the geodesical distance between nodes k and j .

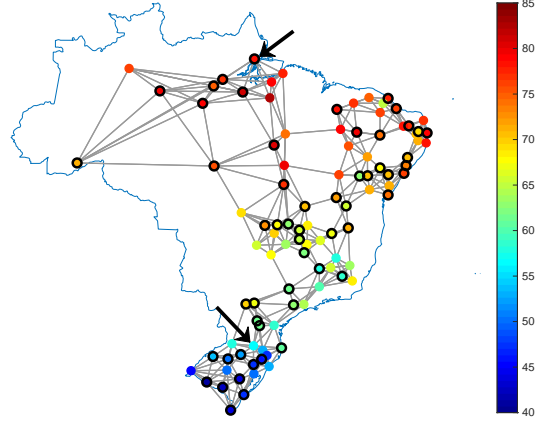


Fig. 11: Daily average temperature measured by 100 weather stations on 06/21/2002 ($^{\circ}\text{F}$). Circled nodes use $\tilde{\mu}_k = 1$, whereas the others use $\tilde{\mu}_k = 0.1$. Each edge is a communication link. The arrows point to the stations whose data are used in Fig. 13.

We consider the dNLMS and AS-dNLMS algorithms with $M = 5$ and $\tilde{\mu}_k = 1$ for half of the nodes, while the other half utilizes $\tilde{\mu}_k = 0.1$. Furthermore, we use the ACW algorithm with $\nu_k = 0.2$ and $\delta_c = 10^{-5}$, and we consider that nodes i and j can communicate if $i \in \mathcal{N}_{\mathbf{A}_j}$ or $j \in \mathcal{N}_{\mathbf{A}_i}$.

We divided our dataset into training and testing sets. The former consists of $N_{\text{tr}} = 3650$ measurements from 12/25/2001 to 12/22/2011, which were periodically replicated to form 20 training epochs. During this period, we consider that $d_k(n) = u_k(n+1)$, where $u_k(n)$ denotes the temperature measurement at node k and time instant n . The vector $\mathbf{x}_k(n)$ is formed as in (3). The testing set consists of the measurements from 12/23/2011 to 12/21/2012. In this case, we do not have access to the temperatures registered on the following day. To keep the adaptation going, we use the last estimate of the algorithm as the desired signal. Thus, we set $d_k(n) = \mathbf{x}_k^T(n-1)\mathbf{w}_k(n-1)$. As a performance indicator, we adopt the squared relative reconstruction error (SSRE), given by [9]

$$\text{SRRE} = \frac{1}{V} \sum_{k=1}^V \frac{(u_k(n+1) - \mathbf{x}_k^T(n)\mathbf{w}_k(n))^2}{u_k^2(n+1)}. \quad (56)$$

We converted the temperature to degrees Fahrenheit in our experiments to avoid division by zero in (56).

It should be noted that, in this scenario, σ_{\max}^2 is not known *a priori*, making it hard to choose β beforehand. To set β , we ran AS-dNLMS like the original dNLMS in the first epoch, calculated the average NMSE during the last 730 days, and multiplied the result by 3. Then, we used (45) with $\Delta_n = 9N_{\text{tr}}$ to set μ_s , and ran the algorithm normally as in Table I.

In Figs. 12(a) and (b) we present the SSRE obtained in the training and testing periods, respectively. Similarly, Figs. 12(c)

and (d) respectively show the number of multiplications per iteration during training and testing. For the ease of visualization, and due to the noisy nature of the data, the curves of Figs. 12(a) and (c) were filtered by a moving-average filter with 1024 coefficients. Nonetheless, in Fig. 12(a) we also show the envelope of the unfiltered curves as dashed lines.

We can observe that AS-dNLMS and the original dNLMS algorithm achieved similar performances during both periods. From the envelope displayed in Fig. 12(a) and the curves of Fig. 12(b), we can see that the SRRE of both algorithms during the test phase is slightly higher than the one observed during training, as expected. Furthermore, from Fig. 12(c) we see that the computational cost of AS-dNLMS remains slightly higher than that of dNLMS during transient, but falls significantly after converging. In the test phase, AS-dNLMS sampled 20 nodes on average per iteration, and performed 45% less multiplications than the original dNLMS while preserving the performance. Considering both the training and test phases, AS-dNLMS reduced the number of multiplications by 17%.

Lastly, as an illustrative example, in Fig. 13 we present the estimates provided by AS-dNLMS for the temperature at the stations of Fig. 11 indicated by the arrows, along with the measured data at these locations. We observe that the estimates of the algorithm follow closely the patterns of the real data even with a reduced number of sampled nodes, as desired.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed adaptive mechanisms for sampling and censoring over distributed solutions. The resulting algorithms use the information from more nodes when the error in the network is high and from less nodes otherwise. They feature fast convergence rates while significantly reducing the computational cost and the consumption of energy associated with the communication between nodes. Furthermore, we derived analytical expressions that help understand the roles of the parameters β and μ_s and their effects in terms of performance, computational cost reduction, and energy saving. These theoretical results allow to choose proper values for β and μ_s and were validated by the simulation results. It was shown that AS-dNLMS maintains the performance of the original diffuse NLMS algorithm, while noticeably reducing the computational burden. Moreover, it can be employed in graph adaptive filtering as well as in classical distributed signal processing. It was also shown that ASC-dNLMS is capable of saving more energy than other state-of-the-art techniques while achieving a similar steady-state performance and preserving the convergence rate of dNLMS. However, the proposed techniques must be employed with some caution in rapidly-varying environments, as their performance may deteriorate in comparison with dNLMS or other techniques. In their current form, this seems to be the main limitation of the proposed algorithms. For future work, we intend to improve their tracking capability in such scenarios. Nevertheless, encouraging results for slowly-varying environments indicate AS-dNLMS and ASC-dNLMS as the recommended solutions for such cases, in which they outperform similar techniques. Furthermore, we aim to obtain an upper bound for the step

size μ_s in order to ensure that the sampling of the nodes does not cease until the steady state is achieved in terms of NMSE or NMSD. Finally, the proposed mechanisms could also be used on other distributed solutions, such as diffuse recursive least-squares [4] or the diffuse affine projection algorithm [5], which is another suggestion for future research.

APPENDIX A

DERIVING EQUATIONS (30) TO (33)

In order to estimate upper or lower bounds for θ_k , we must understand under which circumstances node k remains sampled for the greatest (or lowest) number of iterations in the mean. This can be achieved by estimating the maximum and minimum values $E\{\alpha_k(n)\}$ and $E\{\Delta\alpha_k(n)\}$ can assume in the mean during steady state when node k is sampled (i.e., $\bar{s}_k = 1$). Performing the same analysis for $\bar{s}_k = 0$, we can determine upper and lower bounds for $\bar{\theta}_k$. For simplicity, we assume in our calculations that (26) is satisfied, although the final result is generalized in Section IV-B for all $\beta > 0$.

Firstly, let us assume that at the iteration n , $\alpha_k(n)$ is negative but approximately zero, which we denote by $\alpha_k(n) = 0^-$. In this case, taking expectations from both sides of (18) yields

$$E\{\alpha_k(n+1)|\alpha_k(n) = 0^-\} = \mu_s \phi'_0 \sum_{i \in \mathcal{N}_k} c_{ik} E\{\varepsilon_i^2(n)\}, \quad (57)$$

Thus, at $n+1$ the sampling of node k resumes and, recalling (26), $E\{\Delta\alpha_k(n+1)|\alpha_k(n) = 0^-\} < 0$. Therefore, from iteration $n+1$ onward, α_k decreases until it becomes negative again, meaning that (57) yields the maximum value α_k can assume in the mean in steady state. Moreover, assuming (29), (57) yields a different value for each node k that lies in

$$\mu_s \phi'_0 \sigma_{\min}^2 \leq E\{\alpha_{k_{\max}}^{s.s.}\} \leq \mu_s \phi'_0 \sigma_{\max}^2, \quad (58)$$

where $E\{\alpha_{k_{\max}}^{s.s.}\}$ denotes the maximum value $\alpha_k(n)$ can assume in the mean in steady state. Analogously, we now assume that at a certain iteration n , $\alpha_k(n)$ is positive but approximately zero, which we denote by $\alpha_k(n) = 0^+$. Making this replacement in (18) and taking expectations, we obtain

$$E\{\alpha_k(n+1)|\alpha_k(n) = 0^+\} = \mu_s \phi'_0 E\left\{\sum_{i \in \mathcal{N}_k} c_{ik} \varepsilon_i^2(n) - \beta\right\}. \quad (59)$$

Considering (26), we conclude from (59) that $E\{\alpha_k(n+1)|\alpha_k(n) = 0^+\} < 0$, meaning that node k ceases to be sampled at iteration $n+1$ and, therefore, $E\{\Delta\alpha_k(n)|\alpha_k(n) = 0^+\} > 0$. Thus, (59) provides the minimum value α_k can assume in the mean during steady state, which lies in the range

$$\mu_s \phi'_0 (\sigma_{\min}^2 - \beta) \leq E\{\alpha_{k_{\min}}^{s.s.}\} \leq \mu_s \phi'_0 (\sigma_{\max}^2 - \beta), \quad (60)$$

where $E\{\alpha_{k_{\min}}^{s.s.}\}$ denotes the minimum value $\alpha_k(n)$ can assume in the mean in steady state, $k = 1, \dots, V$.

Since $E\{\alpha_k(n)\}$ keeps oscillating around the point $E\{\alpha_k(n)\} = 0$ during steady state, we replace $\phi'[\alpha_k(n)]$ in (22) by its first-order Taylor expansion around $\alpha_k(n) = 0$, which is simply equal to the constant ϕ'_0 . Thus, when node k is being sampled ($\bar{s}_k = 1$), subtracting $\alpha_k(n)$ from both sides of (22) and taking expectations yields

$$-\mu_s \phi'_0 (\beta - \sigma_{\min}^2) \leq E\{\Delta\alpha_k(n)\} \leq -\mu_s \phi'_0 (\beta - \sigma_{\max}^2) < 0. \quad (61)$$

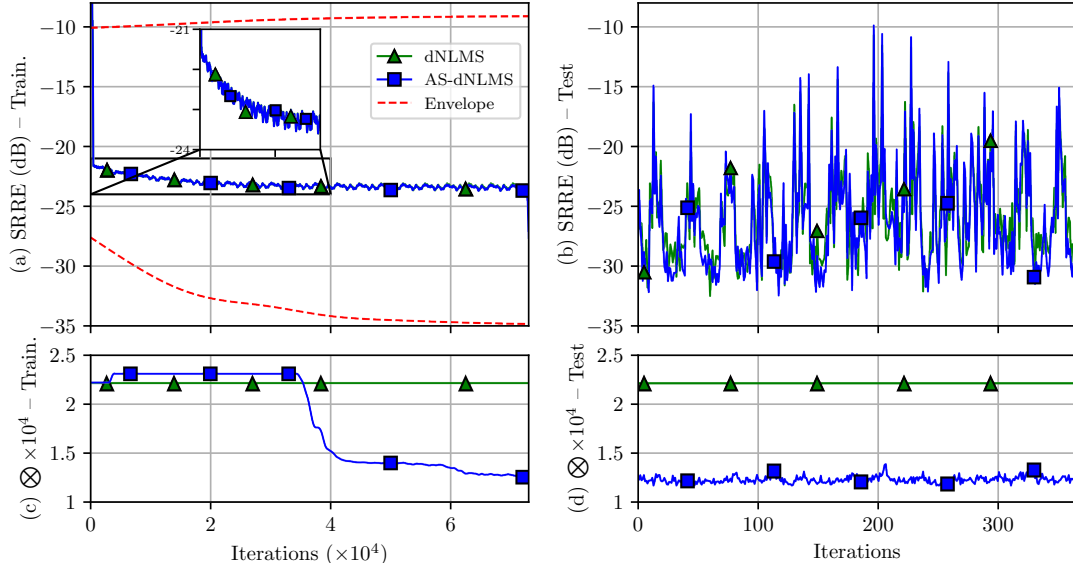


Fig. 12: Comparison between dNLMS and AS-dNLMS ($\beta = 80.49$, $\mu_s = 2.5 \cdot 10^{-5}$). (a) and (b): SRRE in the training and testing periods, respectively. (c) and (d): Multiplications per iteration during training and testing, respectively.

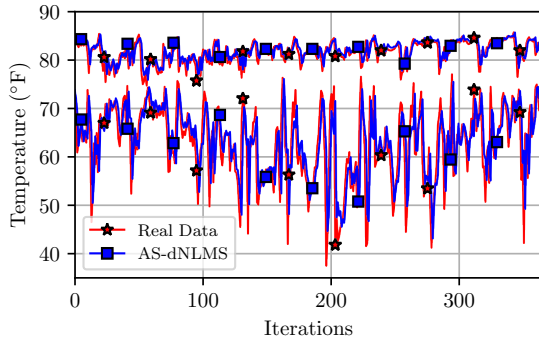


Fig. 13: Comparison between the temperature measured at two stations and the estimates provided by AS-dNLMS for them.

Analogously, when the node is not sampled ($\bar{s}_k = 0$),

$$\mu_s \phi'_0 \sigma_{\min}^2 \leq E\{\Delta\alpha_k(n)\} \leq \mu_s \phi'_0 \sigma_{\max}^2. \quad (62)$$

From a certain iteration n_0 onward, we consider the model

$$E\{\alpha_k(n_0 + \theta_k)\} = E\{\alpha_k(n_0)\} + \theta_k E\{\Delta\alpha_k(n)\}. \quad (63)$$

In order to estimate an upper bound θ_{\max} for θ_k , we assume that $E\{\alpha_k(n_0)\} = E\{\alpha_{k_{\max}}^{s.s.}\}$ and calculate the expected number of iterations required for $E\{\alpha_k(n)\}$ to fall below zero in the scenario where the node is sampled for the maximum number of iterations. This occurs if $E\{\alpha_k(n_0)\} = \mu_s \phi'_0 \sigma_{\max}^2$, which is the upper bound for $E\{\alpha_{k_{\max}}^{s.s.}\}$, and $E\{\Delta\alpha_k(n)\} = -\mu_s \phi'_0 (\beta - \sigma_{\max}^2)$, which is the least negative variation for $E\{\Delta\alpha_k(n)\}$ according to (61). Making $\theta_k = \theta_{\max}$, setting $E\{\alpha_k(n_0 + \theta_{\max})\} = 0$ in (63), and taking into account the fact that the node must be sampled at least once during each cycle, after some algebra we obtain (30). Analogously, using (63) for the lower bound $\theta_k = \theta_{\min}$, we get (31).

For $\bar{\theta}_k$, we replace θ_k in (63) by $\bar{\theta}_k$ and consider that at the iteration n_0 , $E\{\alpha_k(n_0)\} = E\{\alpha_{k_{\min}}^{s.s.}\}$. Thus, the upper bound $\bar{\theta}_{\max}$ for $\bar{\theta}_k$ can be obtained by setting $E\{\alpha_k(n_0)\} =$

$\mu_s \phi'_0 \sigma_{\min}^2$, which is the lower bound for $E\{\alpha_{k_{\min}}^{s.s.}\}$, and $E\{\Delta\alpha_k(n)\} = \mu_s \phi'_0 \sigma_{\min}^2$, which is the minimum value for $E\{\Delta\alpha_k(n)\}$ according to (62). Thus, (32) is obtained. Finally, as an estimate for the lower bound $\bar{\theta}_{\min}$ of $\bar{\theta}_k$, we get (33).

REFERENCES

- [1] A.H. Sayed, *Adaptation, Learning, and Optimization over Networks*, vol. 7, Foundations and Trends in Machine Learning, now Publishers Inc., Hanover, MA, 2014.
- [2] C.G. Lopes and A.H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, pp. 3122–3136, Jul. 2008.
- [3] F.S. Cattivelli and A.H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, pp. 1035–1048, Mar. 2009.
- [4] F.S. Cattivelli, C.G. Lopes, and A.H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 1865–1877, May 2008.
- [5] L. Li and J. A. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *Proc. IEEE SSP*, 2009, pp. 757–760.
- [6] P. Di Lorenzo, S. Barbarossa, and A.H. Sayed, "Bio-inspired decentralized radio access based on swarming mechanisms over adaptive networks," *IEEE Trans. Signal Process.*, vol. 61, pp. 3183–3197, Jun. 2013.
- [7] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, pp. 102–114, Aug. 2002.
- [8] J. Fernandez-Bes, J. Arenas-García, M.T.M. Silva, and L.A. Azpicueta-Ruiz, "Adaptive diffusion schemes for heterogeneous networks," *IEEE Trans. Signal Process.*, vol. 65, pp. 5661–5674, Nov. 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] F. Hua, R. Nassif, C. Richard, H. Wang, and A.H. Sayed, "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *Proc. EUSIPCO*, 2018, pp. 111–115.
- [11] A. Sandryhaila and J.M.F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.
- [12] 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.

- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] N. Takahashi and I. Yamada, "Link probability control for probabilistic diffusion least-mean squares over resource-constrained networks," in *Proc. IEEE ICASSP*, 2010, pp. 3518–3521.
- [19] R. Arroyo-Valles, S. Maleki, and G. Leus, "A censoring strategy for decentralized estimation in energy-constrained adaptive diffusion networks," in *Proc. IEEE SPAWC*, 2013, pp. 155–159.
- [20] J. Fernandez-Bes, R. Arroyo-Valles, J. Arenas-García, and J. Cid-Sueiro, "Censoring diffusion for harvesting WSNs," in *Proc. IEEE CAMSAP*, 2015, pp. 237–240.
- [21] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, pp. 472–484, Jan. 2014.
- [22] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Trading off complexity with communication costs in distributed adaptive learning via Krylov subspaces for dimensionality reduction," *IEEE J. Sel. Topics in Signal Process.*, vol. 7, pp. 257–273, Apr. 2013.
- [23] C.G. Lopes and A.H. Sayed, "Diffusion adaptive networks with changing topologies," in *Proc. IEEE ICASSP*, 2008, pp. 3285–3288.
- [24] X. Zhao and A.H. Sayed, "Single-link diffusion strategies over adaptive networks," in *Proc. IEEE ICASSP*, 2012, pp. 3749–3752.
- [25] S. Xu, R.C. de Lamare, and H.V. Poor, "Adaptive link selection algorithms for distributed estimation," *EURASIP J. Adv. in Signal Process.*, vol. 2015, pp. 86, 2015.
- [26] O.N. Gharehshiran, V. Krishnamurthy, and G. Yin, "Distributed energy-aware diffusion least mean squares: Game-theoretic learning," *IEEE J. Sel. Topics in Signal Process.*, vol. 7, pp. 821–836, Oct. 2013.
- [27] L. Yang, H. Zhu, K. Kang, X. Luo, H. Qian, and Y. Yang, "Distributed censoring with energy constraint in wireless sensor networks," in *Proc. IEEE ICASSP*, 2018, pp. 6428–6432.
- [28] D.K. Berberidis, V. Kekatos, G. Wang, and G.B. Giannakis, "Adaptive censoring for large-scale regressions," in *Proc. IEEE ICASSP*, 2015, pp. 5475–5479.
- [29] D.G. Tiglea, R. Candido, and M.T.M. Silva, "An adaptive sampling technique for graph diffusion LMS algorithm," in *Proc. EUSIPCO*, 2019, pp. 1364–1368.
- [30] D.G. Tiglea, R. Candido, and M.T.M. Silva, "A sampling algorithm for diffusion networks," To appear in *Proc. EUSIPCO*, 2020.
- [31] A.H. Sayed, *Adaptive Filters*, John Wiley & Sons, NJ, 2008.
- [32] N. Takahashi, I. Yamada, and A.H. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 58, pp. 4795–4810, Sep. 2010.
- [33] C.-K. Yu and A.H. Sayed, "A strategy for adjusting combination weights over adaptive networks," in *Proc. IEEE ICASSP*, 2013, pp. 4579–4583.
- [34] S.-Y. Tu and A.H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proc. Int. Workshop on Comput. Adv. in Multi-Sensor Adaptive Process. (CAMSAP)*, 2011, pp. 317–320.
- [35] J. Arenas-García, L.A. Azpicueta-Ruiz, M.T.M. Silva, V.H. Nascimento, and A.H. Sayed, "Combinations of adaptive filters: performance and convergence properties," *Signal Process. Mag.*, vol. 33, pp. 120–140, Jan. 2016.
- [36] M. Lázaro-Gredilla, L.A. Azpicueta-Ruiz, A.R. Figueiras-Vidal, and J. Arenas-García, "Adaptively biasing the weights of adaptive filters," *IEEE Trans. Signal Process.*, vol. 58, pp. 3890–3895, Jul. 2010.
- [37] J.-W. Lee, J.-T. Kong, W.-J. Song, and S.-E. Kim, "Data-Reserved Periodic Diffusion LMS With Low Communication Cost Over Networks," *IEEE Access.*, vol. 6, pp. 54636–54650, Sep. 2018.
- [38] Instituto Nacional de Meteorologia – INMET (National Institute of Meteorology). "Historical Temperature Dataset," Sep. 2020. [Online]. Available: <https://portal.inmet.gov.br/> (in Portuguese). Data used in the simulations also available at: <https://github.com/dgtiglea/Daily-Average-Temperature-Brazilian-Stations>.



Daniel G. Tiglea received the B.S. degree in 2018 and the M.S. degree in 2020, both in Electrical Engineering from Escola Politécnica, Universidade de São Paulo, Brazil. Since August 2020, he has been pursuing his PhD at the same institution. His research interests include linear and nonlinear adaptive filtering, and distributed signal processing.



learning.

Renato Candido received the B.S. degree in 2006 from Universidade Presbiteriana Mackenzie, São Paulo, Brazil and the M.S. and Ph.D. degrees in 2009 and 2014 from Escola Politécnica, Universidade de São Paulo, Brazil, all in Electrical Engineering. From 2015 to 2017, he worked as a Postdoctoral Researcher at the Department of Electronic Systems Engineering, Escola Politécnica, Universidade de São Paulo and currently he collaborates as a researcher at the same university. His research interests include signal processing, adaptive filtering, and machine



Associate Editor for the IEEE Signal Processing Letters. His research interests include linear and nonlinear adaptive filtering, and machine learning for signal processing.

Magno T. M. Silva (M'05) received the B.S. degree in 1999, the M.S. degree in 2001, and the Ph.D. degree in 2005, all in Electrical Engineering from Escola Politécnica, Universidade de São Paulo, Brazil. Since August 2006, he has been with the Department of Electronic Systems Engineering at Escola Politécnica, Universidade de São Paulo, where he is currently an Associate Professor. From January to July 2012, he worked as a Postdoctoral Researcher at Universidad Carlos III de Madrid, Leganés, Spain. From 2015 to 2018, he served as