Can Adaptive Diffusion Networks Do Better with Less Data?

Daniel G. Tiglea*, Renato Candido†, and Magno T.M. Silva‡
Electronic Systems Engineering Department, Polytechnic School of the University of São Paulo
São Paulo, Brazil
Email: *dtiglea@lps.usp.br, †renatocan.@lps.usp.br, ‡magno@lps.usp.br

***Abstract*—In this paper, we analyze the performance of an algorithm for adaptive diffusion networks that controls the number of nodes sampled per iteration based on the estimation error. The goal of this solution is to keep the nodes sampled while the estimation error is high in magnitude, and to cease their sampling when it is sufficiently low. Our model shows that this approach can preserve the convergence rate in comparison with the case in which every node is sampled permanently, while slightly improving the steady-state performance.**

***Index Terms*—Adaptive diffusion networks, distributed signal processing, sampling, transient analysis, steady-state analysis.**

I. Introduction

Adaptive diffusion networks are comprised of a set of connected nodes, that are able to measure and process data locally, and that can communicate with other nodes in their vicinity. These nodes have a collective objective to estimate a parameter vector of interest in a decentralized manner [1]–[5]. Typically, the distributed learning is carried out in two steps at each time instant: *adaptation* and *combination*. In the former, each node computes its own *local* estimate, whereas in the latter the nodes share their local estimates to form a *global* estimate of the vector of interest [1]–[5].

It is often desirable to restrict the amount of data measured and processed by the nodes, which is known as *sampling* in the literature [6], [7]. Thus, by sampling some of the nodes at each iteration, we can reduce the computational and memory burdens associated with the learning task. However, there may also be a negative impact on the convergence rate. Based on these observations, an adaptive sampling algorithm for diffusion networks was proposed in [8], and later modified in [9], resulting in the "Dynamic-Tuning-and-Resetting Adaptive Sampling" (DTRAS) algorithm. Its goal is to keep every node sampled when the estimation error is high in magnitude, and cease their sampling otherwise. Hence, with a proper selection of their parameters, it can attain a good convergence rate, while presenting a lower computational cost in the steady state. Interestingly, the performance is improved as we sample **less and less** nodes in the steady state.

To investigate this, we recently studied the effects of randomly sampling of the nodes [10]. In this paper, our goal is to extend the theoretical results obtained in [10] to predict the behavior of our adaptive sampling algorithm of [9]. In addition to helping to understand our proposal, we believe that these results may aid in the development of efficient algorithms for adaptive diffusion networks.

This work was supported by CAPES under Grant 88887.512247/2020-00 and Finance Code 001, by CNPq under Grants 303826/2022-3 and 404081/2023-1, and by FAPESP under Grant 2021/02063-6.

This paper is organized as follows. In Sec. II, we present the problem formulation and revisit the DTRAS-dLMS algorithm of [9]. In Sec. III, we present a theoretical analysis to model the behavior of DTRAS-dLMS. Lastly, in Secs. IV and V, we respectively present the simulation results and the main conclusions of our work.

Notation. We use normal font letters for scalars, boldface lowercase letters for vectors, and boldface uppercase letters for matrices. Moreover, $(\cdot)^{\rm T}$ denotes transposition, $\mathrm{E}\{\cdot\}$ the mathematical expectation, $[\cdot]_{\ell k}$ the element of a matrix located at its ℓ-th row and k-th column, $\|\cdot\|$ the Euclidean norm, $|\cdot|$ the cardinality, $\otimes$ the Kronecker product, $\odot$ the Hadamard product, and $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ the ceiling and floor functions, respectively. We denote the $L\times L$ identity matrix by $\mathbf{I}_L$,an $L\times M$ matrix of ones by $\mathbf{1}_{L\times M}$, and L-length column vectors of zeros or ones by $\mathbf{0}_L$ and $\mathbf{1}_L$, respectively. We denote by $\mathrm{diag}\{\cdot\}$ the aggregation of the arguments into a diagonal matrix and by $\mathrm{vec}\{\cdot\}$ the vectorization of a matrix by stacking all of its columns together to form a column vector. To simplify the arguments, we assume real data throughout the paper.

II. Revisiting the DTRAS-dLMS Algorithm

Let us consider a network comprised of V nodes, with labels $k\in\{1,\cdots,V\}$ and a predefined topology. For each node k, we call the set of nodes with which it can communicate, including node k itself, its neighborhood, denoted by $\mathcal{N}_k$. At each time instant n, each node k has access to an input signal $u_k(n)$ and to a desired signal $d_k(n)$, which we model as [1]–[4]

$$d_k(n) = \mathbf{u}_k^{\rm T}(n)\mathbf{w}^{\rm o} + v_k(n), \tag{1}$$

where $\mathbf{w}^{\rm o}$ is an M-length column vector that represents an unknown system [1]–[4], and $\mathbf{u}_k(n) = [u_k(n)\ u_k(n-1)\ \cdots\ u_k(n-M+1)]^{\rm T}$. Lastly, $v_k(n)$ is a zero-mean measurement noise at node k, with variance $\sigma_{v_k}^2$.

The objective of the network is to obtain an estimate $\mathbf{w}$ of $\mathbf{w}^{\rm o}$ in a distributed manner by solving [1]–[4]

$$\min_{\mathbf{w}} J_{\rm global}(\mathbf{w}) = \min_{\mathbf{w}} \textstyle\sum_{k=1}^{V} J_k(\mathbf{w}), \tag{2}$$

where $J_k(\mathbf{w})$ is a local cost function at node k. One of the most common choices for the $J_k(\mathbf{w})$, $k=1,\cdots,V$ is the mean squared error (MSE) [11], [12]. Hence, one obtains $J_k(\mathbf{w}) = \text{MSE}_k(n) \triangleq \text{E}\{[d_k(n) - \mathbf{u}_k^{\text{T}}(n)\mathbf{w}]^2\}$, where MSE_k denotes the MSE at node k [1]–[4]. It is worth noting that the vector $\mathbf{w}_{\text{o}}$ is the optimal solution to (2), and for this reason it is often referred to as the "optimal solution" in the literature [11], [12].

At each iteration, every node k calculates a local estimate of $\mathbf{w}^{\text{o}}$ in order to minimize $J_k(\mathbf{w})$. To this end, it uses the data available locally, as well as the estimates produced by neighboring nodes in what is known as the adaptation step. Then, each node k cooperates with its neighbors to assemble a combined estimate. This process is known as the the combination step. The adoption of various strategies for the adaptation step leads to different algorithms, such as the dLMS algorithm [1]–[4], the diffusion recursive least squares (dRLS) [5], diffusion Normalized LMS (dNLMS) [8], [9], [13], among others [14], [15].

In particular, the DTRAS-dLMS algorithm is obtained by incorporating the sampling in dLMS. Its adaptation and combination steps are respectively given by [9]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \mu_k \zeta_k(n) \mathbf{u}_k(n) e_k(n) & \text{(3a)} \\ \mathbf{w}_k(n) = \sum_{i \in \mathcal{N}_k} c_{ik} \boldsymbol{\psi}_i(n). & \text{(3b)} \end{cases}$$

where $\boldsymbol{\psi}_k$ and $\mathbf{w}_k$ are the local and combined estimates of $\mathbf{w}_{\text{o}}$ at node k, respectively, $\mu_k > 0$ is a step size,

$$e_k(n) = d_k(n) - \mathbf{u}_k^{\text{T}}(n)\mathbf{w}_k(n-1) \tag{4}$$

is the estimation error at node k, and $\zeta_k(n) \in \{0, 1\}$ is a binary variable. If $\zeta_k(n) = 1$, in which case we say that node k is *sampled*, $\boldsymbol{\psi}_k(n)$ is updated as usual. On the other hand, when $\zeta_k(n) = 0$, (3a) becomes simply $\boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1)$. In this case, we say that node k is not sampled, and $\mathbf{u}_k^{\text{T}}(n)\mathbf{w}_k(n-1)$ and $e_k(n)$ do not need to be calculated. Finally, c_{ik} are combination weights satisfying [1]–[4]

$$c_{ik} \geq 0, \ \sum_{i \in \mathcal{N}_k} c_{ik} = 1, \ \text{and} \ c_{ik} = 0 \ \text{for} \ i \notin \mathcal{N}_k. \tag{5}$$

There are many possible rules for the selection of the combination weights. For instance, adopting $c_{ik} = 1$ if $i = k$ and $c_{ik} = 0$ otherwise, leads to a setup in which the nodes do not exchange their local estimates, which is referred to as the non-cooperative strategy [1]–[4]. On the other hand, cooperative strategies include the Uniform, Metropolis, and Hastings rules, among others [1]–[3], as well as adaptive schemes [16]–[18]. For simplicity, in this paper we focus on static combination rules. In the simulations, we use Metropolis weights, given by

$$c_{ik} = \begin{cases} \dfrac{1}{\max\{|\mathcal{N}_k|, |\mathcal{N}_i|\}}, & \text{if } i \neq k \text{ and } i \in \mathcal{N}_k \\ 1 - \sum_{i \in \mathcal{N}_k} c_{ik}, & \text{if } i = k \\ 0, & \text{otherwise.} \end{cases}$$

Inspired by convex combinations of adaptive filters [19], instead of directly adapting $\zeta_k(n)$, we use an auxiliary variable $\alpha_k(n) \in [-\alpha^+, \alpha^+]$ such that $\zeta_k(n) = 0$ for $\phi[\alpha_k(n)] < 0.5$ and $\zeta_k(n) = 1$ otherwise, with $\phi[\cdot]$ given by [19]

$$\phi[\alpha_k(n)] \triangleq \frac{\text{sgm}[\alpha_k(n)] - \text{sgm}[-\alpha^+]}{\text{sgm}[\alpha^+] - \text{sgm}[-\alpha^+]}, \tag{6}$$

where $\text{sgm}[x] = [1 + \exp(-x)]^{-1}$ is a sigmoidal function. In the literature, $\alpha^+ = 4$ is usually adopted [19]. For the compactness of notation, we henceforth write $\phi[\alpha_k(n)]$ as $\phi_k(n)$. Thus, $\zeta_k(n)$ is related to $\alpha_k(n)$ by

$$\zeta_k(n) = \begin{cases} 1, & \text{if } \alpha_k(n) \geq 0, \\ 0, & \text{otherwise} \end{cases} . \tag{7}$$

We then introduce the following cost function [8], [9]:

$$J_{\alpha_k}(n) = \phi_k(n)\rho_k(n)\zeta_k(n) + [1 - \phi_k(n)] \sum_{i \in \mathcal{N}_k} c_{ik} e_i^2(n), \tag{8}$$

where the parameter $\rho_k(n) > 0$ is introduced to control the penalty of sampling node k. When the error is high in magnitude, $J_{\alpha_k}(n)$ is minimized by making $\phi_k(n)$ closer to one, leading to the sampling of node k. In contrast, when node k is sampled ($\zeta_k = 1$) and the error is small in magnitude, $J_{\alpha_k}(n)$ is minimized by making $\phi_k(n)$ closer to zero, and the algorithm stops sampling node k [8]. However, when node k is not sampled ($\zeta_k = 0$), $J_{\alpha_k}(n)$ is minimized by making $\phi_k(n)$ closer to one again. This ensures that the sampling of node k eventually resumes. Hence, the sampling of the nodes does not cease permanently.

The DTRAS-dLMS algorithm is obtained by selecting

$$\rho_k(n) = \gamma \sum_{i \in \mathcal{N}_k} c_{ik} \widehat{\sigma}_{v_i}^2(n) \tag{9}$$

in (8), where $\gamma > 1$ is a parameter that the designer must choose that is common to every node in the network. For compactness of notation, we introduce $\widehat{\sigma}_{\mathcal{N}_k}^2 \triangleq \sum_{i \in \mathcal{N}_k} c_{ik} \widehat{\sigma}_{v_i}^2$, with $\widehat{\sigma}_{v_i}^2$ denoting an estimate of $\sigma_{v_i}^2$, which is obtained by the algorithm proposed in [20].

Then, by taking the derivative of (8) with respect to $\alpha_k(n)$, we get the following stochastic gradient descent rule [9]:

$$\alpha_k(n+1) = \alpha_k(n) + \mu_{\zeta_k}(n)\phi_k'(n) \times \left[\sum_{i \in \mathcal{N}_k} c_{ik} \varepsilon_i^2(n) - \gamma \widehat{\sigma}_{\mathcal{N}_k}^2(n)\zeta_k(n)\right]. \tag{10}$$

where $\mu_{\zeta_k}(n) > 0$ is a step size, $\varepsilon_i(n)$ is the last measurement of $e_i(n)$ that we have access to, given by $\varepsilon_i(n) = \zeta_i(n)e_i(n) + [1 - \zeta_i(n)]\varepsilon_i(n-1)$, and $\phi_k'(n) = \frac{d\phi[\alpha_k(n)]}{d\alpha_k(n)}$ [19].

It is suggested in [9] that, if we wish to cease the sampling of the nodes in at most Δn iterations after the steady state is achieved in terms of the Network MSE (NMSE), given by $\text{NMSE}(n) \triangleq \frac{1}{V}\sum_{k=1}^{V} \text{MSE}_k(n)$, we should adopt

$$\mu_{\zeta_k}(n) = \frac{1}{\widehat{\sigma}_{\mathcal{N}_k}^2(n)} \left\{ \frac{\alpha^+}{(\gamma - 1)(\phi_0' - \phi_{\alpha^+}')} \left[\left(\frac{\phi_0'}{\phi_{\alpha^+}'} \right)^{\frac{1}{\Delta n}} - 1 \right] \right\}, \tag{11}$$

where ϕ_0' and ϕ_{α^+}' denote ϕ_k' evaluated at $\alpha_k(n) = 0$ and $\alpha_k(n) = \alpha^+$, respectively.

DTRAS-dLMS also has a reset mechanism, which sets $\alpha_k(n) = \alpha^+$ if a change in the environment is detected. This is done to improve the tracking capability of the algorithm. However, in a stationary environment such as the one considered in this paper, this mechanism should not come into play. For this reason, and due to space limitations, we shall disregard it in this paper and in our analysis.

III. Theoretical Analysis

We are interested in analyzing the Network Mean-Squared Deviation (NMSD), a common performance metric given by

$$\mathrm{NMSD}(n) = \frac{1}{V}\sum_{k=1}^{V}\mathrm{MSD}_k(n), \tag{12}$$

where MSD_k denotes the MSD at each node k, given by $\mathrm{MSD}_k(n) = \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\}$, where $\widetilde{\mathbf{w}}_k(n) \triangleq \mathbf{w}_\mathrm{o} - \mathbf{w}_k(n)$ is the weight-error vector for node k [1], [2].

For compactness of notation, it is convenient to introduce

$$\beta_{ij}(n) \triangleq \mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_j(n)\} \tag{13}$$

for $i = 1, \cdots, V$ and $j = 1, \cdots, V$. It is worth noting that $\beta_{kk}(n) = \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\} = \mathrm{MSD}_k(n)$.

Subtracting both sides of (3a) from $\mathbf{w}_\mathrm{o}$, and replacing (1) and (4) in the resulting equation, after some algebra, we get

$$\begin{aligned}\widetilde{\boldsymbol{\psi}}_k(n) = [\mathbf{I}_M - \mu_k\zeta_k(n)\mathbf{u}_k(n)\mathbf{u}_k^\mathrm{T}(n)]\widetilde{\mathbf{w}}_k(n-1)\\ - \mu\zeta_k(n)\mathbf{u}_k(n)v_k(n),\end{aligned} \tag{14}$$

where we have defined $\widetilde{\boldsymbol{\psi}}_k(n) \triangleq \mathbf{w}_\mathrm{o} - \boldsymbol{\psi}_k(n)$. Moreover, we observe from (3b) that

$$\widetilde{\mathbf{w}}_k(n) = \sum_{i\in\mathcal{N}_k} c_{ik}\widetilde{\boldsymbol{\psi}}_i(n). \tag{15}$$

If we multiply both sides of (15) by $\widetilde{\mathbf{w}}_k^\mathrm{T}(n)$ from the left, and use (3b) again, we get after some algebraic manipulations

$$\|\widetilde{\mathbf{w}}_k(n)\|^2 = \beta_{kk}(n) = \sum_{i\in\mathcal{N}_k}\sum_{j\in\mathcal{N}_k} c_{ik}c_{jk}\widetilde{\boldsymbol{\psi}}_j^\mathrm{T}(n)\widetilde{\boldsymbol{\psi}}_i(n). \tag{16}$$

Replacing (14) in (16) and taking the expectations, we get

$$\beta_{kk}(n) = \sum_{i\in\mathcal{N}_k}\sum_{j\in\mathcal{N}_k} c_{ik}c_{jk}x_{ji}(n), \tag{17}$$

where we have introduced

$$\begin{aligned}x_{ji}(n) \triangleq \mathrm{E}\Bigg\{&\Big\{[\mathbf{I}_M - \mu\zeta_j(n)\mathbf{u}_j(n)\mathbf{u}_j^\mathrm{T}(n)]\widetilde{\mathbf{w}}_j(n-1)\\ &- \mu\zeta_j(n)\mathbf{u}_j(n)v_j(n)\Big\}^\mathrm{T}\\ &\cdot\Big\{[\mathbf{I}_M - \mu\zeta_i(n)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)]\widetilde{\mathbf{w}}_i(n-1)\\ &- \mu\zeta_i(n)\mathbf{u}_i(n)v_i(n)\Big\}\Bigg\}.\end{aligned} \tag{18}$$

At this point, a few assumptions are necessary:

A1. All the nodes in the network employ the same step size, i.e., $\mu_1 = \cdots = \mu_V = \mu > 0$;

A2. The vectors $\widetilde{\mathbf{w}}_i(n-1)$ are statistically independent of $\mathbf{u}_j(n)$ for any pair i and j. This is a multi-agent version of the independence theory [11], [12];

A3. The measurement noise $v_k(n)$ is independent and identically distributed (iid), and independent from any other variable for $k = 1, \cdots, V$;

A4. The input signals are zero-mean and white Gaussian with variance $\sigma_{u_1}^2 = \cdots = \sigma_{u_V}^2 = \sigma_u^2 > 0$. In other words, the autocorrelation matrices $\mathbf{R}_{u_k} \triangleq \mathrm{E}\{\mathbf{u}_k(n)\mathbf{u}_k^\mathrm{T}(n)\}$, $k = 1, \cdots, V$ are the same, and are proportional to the identity matrix, i.e., $\mathbf{R}_{u_1} = \cdots = \mathbf{R}_{u_V} = \sigma_u^2\mathbf{I}_M$;

A5. At any time instant n, $u_i(n)$ is statistically independent from $u_j(n)$ for any pair of nodes i and j, $i \neq j$;

A6. For every node k, we shall consider $\zeta_k(n)$ independent from any other variable, and drawn from a Bernoulli distribution, such that $\zeta_k(n) = 1$ with probability $p_\zeta(n)$ and $\zeta_k(n) = 0$ with probability $1 - p_\zeta(n)$ for every node $k = 1, \cdots, V$. Moreover, for any $i \neq j$, $\zeta_i(n)$ is statistically independent from $\zeta_j(n)$.

Assumptions **A1–A5** are common in the adaptive diffusion networks and adaptive filtering literature. In its turn, Assumption **A6** does not hold in practice, since the sampling of node k does depend on $e_i^2(n)$ of every sampled node i in its neighborhood. However, it is necessary for the tractability of the problem, and still leads to a good match between the simulations and the theoretical results, as shown in Sec. IV.

If $j = i$ in (18), using **A3** and **A6**, and observing that $\mathrm{E}\{\zeta_i(n)\} = \mathrm{E}\{\zeta_i^2(n)\} = p_\zeta(n)$, we can write

$$\begin{aligned}x_{ii}(n) =& \mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n-1)\widetilde{\mathbf{w}}_i(n-1)\}\\ &-2\mu p_\zeta(n)\mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_i(n-1)\}\\ &+\mu^2 p_\zeta(n)\mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_i(n-1)\}\\ &+\mu^2 p_\zeta(n)\sigma_{v_i}^2\mathrm{E}\{\mathbf{u}_i^\mathrm{T}(n)\mathbf{u}_i(n)\}.\end{aligned} \tag{19}$$

Using Assumptions **A2** and **A4**, and following similar procedures to those used in the analysis of the MSD of the LMS algorithm, we may write (see pages 803–807 of [21])

$$\mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_i(n-1)\} = \sigma_u^2\beta_{ii}(n-1) \tag{20}$$

and

$$\begin{aligned}&\mathrm{E}\{\widetilde{\mathbf{w}}_i^\mathrm{T}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_i(n-1)\} =\\ &\sigma_u^4(M+2)\beta_{ii}(n-1).\end{aligned} \tag{21}$$

Thus, (19) can be recast as

$$x_{ii}(n) = \theta(n)\beta_{ii}(n-1) + \mu^2 p_\zeta(n)M\sigma_u^2\sigma_{v_i}^2, \tag{22}$$

with $\theta(n)$ defined as

$$\theta(n) \triangleq 1 - 2\mu p_\zeta(n)\sigma_u^2 + \mu^2 p_\zeta(n)\sigma_u^4(M+2). \tag{23}$$

We shall now examine $x_{ji}(n)$ for $j \neq i$. To make this distinction clearer, we replace the index i by ℓ in the next expressions. From **A6**, we can observe that

$$\mathrm{E}\{\zeta_j(n)\zeta_\ell(n)\} = \mathrm{E}\{\zeta_j(n)\}\mathrm{E}\{\zeta_\ell(n)\} = p_\zeta^2(n). \tag{24}$$

Using (24), **A3** and **A6**, we can rewrite (18) for $\ell \neq j$ as

$$\begin{aligned}x_{j\ell}(n) =& \mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &- \mu p_\zeta(n)\mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^\mathrm{T}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &- \mu p_\zeta(n)\mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\mathbf{u}_\ell(n)\mathbf{u}_\ell^\mathrm{T}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &+\mu^2 p_\zeta^2(n)\mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^\mathrm{T}(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^\mathrm{T}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}.\end{aligned} \tag{25}$$

Using **A2**, **A4**, and **A5**, from (25) we can write

$$\begin{aligned}&\mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^\mathrm{T}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &= \mathrm{E}\{\widetilde{\mathbf{w}}_j^\mathrm{T}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^\mathrm{T}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &= \sigma_u^2\beta_{j\ell}(n-1)\end{aligned} \tag{26}$$

for any pair of nodes ℓ and j, $\ell \neq j$. Furthermore, we notice that the last expectation in the rhs of (25) can be calculated as

$$\begin{aligned}&\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &\quad=\sigma_u^4\beta_{j\ell}(n-1).\end{aligned} \tag{27}$$

Therefore, we may write

$$x_{j\ell}(n)=\tau(n)\beta_{j\ell}(n-1), \tag{28}$$

where we have introduced

$$\tau(n) \triangleq 1-2\mu p_\zeta(n)\sigma_u^2+\mu^2 p_\zeta^2(n)\sigma_u^4. \tag{29}$$

Hence, replacing (22) and (28) in (17), we obtain, after some algebraic manipulations,

$$\begin{aligned}\beta_{kk}(n)=&\,\theta(n)\sum_{i=1}^{V} c_{ik}^2\beta_{ii}(n-1)\\ &+\tau\sum_{j=1}^{V}\sum_{\substack{\ell=1\\ \ell\neq j}}^{V} c_{jk}c_{\ell k}\beta_{j\ell}(n-1)+\mu^2 p_\zeta(n)M\sigma_u^2\sum_{q=1}^{V} c_{qk}^2\sigma_{v_q}^2.\end{aligned} \tag{30}$$

Analogously, for a pair of distinct nodes j and ℓ, we get

$$\begin{aligned}\beta_{j\ell}(n)&=\theta(n)\textstyle\sum_{t=1}^{V} c_{tj}c_{t\ell}\beta_{tt}(n-1)\\ &\quad+\tau(n)\textstyle\sum_{r=1}^{V}\sum_{\substack{s=1\\ s\neq r}}^{V} c_{rj}c_{s\ell}\beta_{rs}(n-1)\\ &\quad+\mu^2 p_\zeta(n)M\sigma_u^2\textstyle\sum_{z=1}^{V} c_{zj}c_{z\ell}\sigma_{v_z}^2.\end{aligned} \tag{31}$$

We could aggregate the different quantities β_{kk} and $\beta_{j\ell}$ in a vector and recast (30) and (31) as a single vector equation. To this end, let us introduce $\boldsymbol{\beta}(n) \triangleq \mathrm{vec}\{\mathbf{B}(n)\}$, where the matrix $\mathbf{B}(n)$ is built in such a way that $[\mathbf{B}(n)]_{ij}=\beta_{ij}(n)$. In this case, introducing the vector $\mathbf{b} \triangleq \mathrm{vec}\{\mathbf{I}_V\}$, we may write

$$\mathrm{NMSD}(n)=\frac{1}{V}\mathbf{b}^{\mathrm{T}}\boldsymbol{\beta}(n). \tag{32}$$

From (30) and (31), we may write

$$\boldsymbol{\beta}(n)=\boldsymbol{\Phi}(n)\boldsymbol{\beta}(n-1)+\mu^2 p_\zeta(n)M\sigma_u^2\boldsymbol{\sigma}, \tag{33}$$

where $\boldsymbol{\Phi}(n)$ is a matrix whose k-th row determines how each $\beta_{ij}(n-1)$ influences the corresponding term in the current iteration, and $\boldsymbol{\sigma}$ is a vector that aggregates the information from the network topology and noise variance from the constant terms that appear in (30) and (31). Furthermore, it is worth noting that, if the algorithm is initialized with $\mathbf{w}_k(0)=\mathbf{0}_M$ for every node k, we have that $\widetilde{\mathbf{w}}_k(0)=\mathbf{w}_{\mathrm{o}}$. Thus, for any i and j, we have that $\beta_{ij}(0)=\mathrm{E}\{\widetilde{\mathbf{w}}_i^{\mathrm{T}}(0)\widetilde{\mathbf{w}}_j(0)\}=\mathrm{E}\{\mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{w}_{\mathrm{o}}\}=\|\mathbf{w}_{\mathrm{o}}\|^2$, and, consequently, $\boldsymbol{\beta}(0)=\|\mathbf{w}_{\mathrm{o}}\|^2\mathbf{1}_{V^2}$.

Let us now aggregate the combination weights into a $V\times V$ matrix $\mathbf{C}$, such that $[\mathbf{C}]_{ij}=c_{ij}$. Similarly, let us collect the noise variances in a $V\times V$ diagonal matrix $\mathbf{R}_v$, such that its k-th element is equal to $\sigma_{v_k}^2$, i.e., $\mathbf{R}_v=\mathrm{diag}\{\sigma_{v_1}^2,\sigma_{v_2}^2,\cdots,\sigma_{v_V}^2\}$. In this case, if we define $\boldsymbol{\Sigma}\triangleq\mathbf{C}\mathbf{R}_v\mathbf{C}^{\mathrm{T}}$, we observe that

$$\boldsymbol{\Sigma}=\begin{bmatrix}\sum c_{k1}^2\sigma_{v_k}^2 & \sum c_{k1}c_{k2}\sigma_{v_k}^2 & \cdots & \sum c_{k1}c_{kV}\sigma_{v_k}^2\\ \sum c_{k2}c_{k1}\sigma_{v_k}^2 & \sum c_{k2}^2\sigma_{v_k}^2 & \cdots & \sum c_{k2}c_{kV}\sigma_{v_k}^2\\ \vdots & \vdots & \ddots & \vdots\\ \sum c_{kV}c_{k1}\sigma_{v_k}^2 & \sum c_{kV}c_{k2}\sigma_{v_k}^2 & \cdots & \sum c_{kV}^2\sigma_{v_k}^2\end{bmatrix}, \tag{34}$$

where the index k in the summations goes from $k=1$ to $k=V$. Then, we may write the $V^2\times 1$ vector $\boldsymbol{\sigma}$ in (33) as

$$\boldsymbol{\sigma}=\mathrm{vec}\{\boldsymbol{\Sigma}\}. \tag{35}$$

As for the matrix $\boldsymbol{\Phi}$, from (30) and (31) we obtain that

$$\boldsymbol{\Phi}(n)=\boldsymbol{\Omega}(n)\odot\boldsymbol{\Gamma}, \tag{36}$$

where we have introduced

$$\boldsymbol{\Gamma}\triangleq(\mathbf{C}\otimes\mathbf{C})^{\mathrm{T}} \tag{37}$$

and

$$\boldsymbol{\Omega}(n)=[\boldsymbol{\Omega}_1(n)\ \boldsymbol{\Omega}_2(n)\ \cdots\ \boldsymbol{\Omega}_V(n)], \tag{38}$$

in which $\boldsymbol{\Omega}_i(n)$ is a $V^2\times V$ matrix whose elements in the i-th column are all equal to $\theta(n)$, and whose other elements are all equal to $\tau(n)$, i.e.

$$\boldsymbol{\Omega}_i(n)=\underbrace{\begin{array}{c}\begin{array}{cccccc} & & \overset{i\text{-th column}}{\downarrow} & & & \end{array}\\ \begin{bmatrix}\tau(n)\cdots & \tau(n) & \theta(n) & \tau(n) & \cdots & \tau(n)\\ \tau(n)\cdots & \tau(n) & \theta(n) & \tau(n) & \cdots & \tau(n)\\ \vdots\ \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \tau(n)\cdots & \tau(n) & \theta(n) & \tau(n) & \cdots & \tau(n)\end{bmatrix}\end{array}}_{V\ \text{columns}}. \tag{39}$$

It has been shown in [9] that, in steady state, the maximum sampling probability for every node k is given by

$$p_{\mathrm{s.s.}_{\max}}=\frac{\lceil(\gamma-1)^{-1}\rceil}{\lceil(\gamma-1)^{-1}\rceil+\max\{1,\lfloor\gamma-1\rfloor\}}, \tag{40}$$

where the index "s.s." comes from "steady state". For simplicity, we shall assume that the DTRAS-dLMS simply switches the sampling probability p_ζ from 1 to $p_{\mathrm{s.s.}_{\max}}$ given by (40) at a certain iteration n_{switch}. Thus, we may write

$$p_\zeta(n)=\begin{cases}1, \text{ if } n<n_{\text{switch}},\\ p_{\mathrm{s.s.}_{\max}}, \text{ otherwise.}\end{cases} \tag{41}$$

By considering $p_\zeta(n)$ given by (41) in (23), (29), and (33), and taking (34)–(40) into account, we can predict the NMSD performance of the DTRAS-dLMS algorithm. The only question left is at which iteration n_{switch} the sampling probability transitions from unity to its steady-state value. To answer this, we must investigate when the algorithm achieves the steady state in terms of the NMSD. From assumptions **A2**–**A5**, we can write that $\mathrm{MSE}_k(n)\approx\sigma_u^2\mathrm{MSD}_k(n-1)+\sigma_{v_k}^2$, and, consequently,

$$\mathrm{NMSE}(n)\approx\sigma_u^2\mathrm{NMSD}(n-1)+\bar{\sigma}_v^2, \tag{42}$$

where we have introduced $\bar{\sigma}_v^2\triangleq\frac{\sum_{k=1}^{V}\sigma_{v_k}}{V}$. During the transient phase, we have $p_\zeta(n)=1$ and therefore, $\tau(n)=\tau_0=1-2\mu\sigma_u^2+\mu^2\sigma_u^4$. Moreover, if we adopt the approximation $\boldsymbol{\Omega}_i(n)\approx\tau_0\mathbf{1}_{V^2\times V}$ while the nodes are still sampled, we have that $\boldsymbol{\Phi}(n)\approx\boldsymbol{\Phi}_0\triangleq\tau_0\boldsymbol{\Gamma}$ during this period. Thus, using (32) and (33) with the previous approximations, and considering that $\boldsymbol{\beta}(0) = \|\mathbf{w}_\text{o}\|^2$, we conclude that, while the nodes are still sampled, we may write

$$\text{NMSD}(n) \approx \frac{\|\mathbf{w}_\text{o}\|^2\tau_0^n}{V} \cdot \mathbf{b}^\text{T}\boldsymbol{\Gamma}^n\mathbf{1}_{V^2} + \frac{\mu^2 M\sigma_u^2}{V}\mathbf{b}^\text{T}[\mathbf{I}_{V^2} - \tau_0\boldsymbol{\Gamma}]^{-1}[\mathbf{I}_{V^2} - \tau_0^n\boldsymbol{\Gamma}^n]\boldsymbol{\sigma}. \quad (43)$$

From (43), we see that the NMSD depends on the network topology due to the matrix $\boldsymbol{\Gamma}$. For simplicity, as an approximation, we consider instead that the network is given by a complete graph, i.e., one in which every pair of nodes is directly connected by an edge. In this case, adopting the Uniform or Metropolis rule leads to $\boldsymbol{\Gamma} \approx \boldsymbol{\Gamma}_C \triangleq \frac{1}{V^2}\mathbf{1}_{V^2\times V^2}$, where the index C stands for "complete". An interesting property of this matrix is that $\boldsymbol{\Gamma}_C^n = \boldsymbol{\Gamma}_C$ for any integer $n \geq 1$. Moreover, in this case we get that $\mathbf{b}^\text{T}\boldsymbol{\Gamma}_C\mathbf{1}_{V^2} = V$. Lastly, we remark that it is possible to obtain $\mathbf{b}^\text{T}[\mathbf{I}_{V^2} - \tau_0\boldsymbol{\Gamma}_C]^{-1}\boldsymbol{\sigma} = \frac{1}{1-\tau_0}\bar{\sigma}_v^2$ [10]. Thus, from (43) and (42) we can write

$$\text{NMSE}(n) \approx \sigma_u^2\|\mathbf{w}_\text{o}\|^2\tau_0^{n-1} + \left[\frac{\mu M\sigma_u^2}{(2-\mu\sigma_u^2)V} - \frac{\mu^2 M\sigma_u^4\tau_0^{n-1}}{V} + 1\right]\bar{\sigma}_v^2. \quad (44)$$

From (44), we notice that, while the nodes are sampled, and assuming $\tau_0 < 1$, the NMSE converges approximately to

$$\chi = \left[\frac{\mu M\sigma_u^2}{(2-\mu\sigma_u^2)V} + 1\right]\bar{\sigma}_v^2. \quad (45)$$

Next, we shall consider that the algorithm has achieved the steady in terms of the NMSE at the time instant n if $\text{NMSE}(n) \leq (1+\delta)\chi$, where $0 < \delta \ll 1$ is a constant. From (44) and (45), after some algebra, we conclude that this holds for

$$n \geq n_\text{s.s.} = 1 + \left\lceil\frac{\ln(\delta) + \ln(\chi) - \ln(\eta)}{\ln(\tau_0)}\right\rceil, \quad (46)$$

where we introduced $\eta \triangleq \sigma_u^2\left(\|\mathbf{w}_\text{o}\|^2 - \frac{\mu^2 M\sigma_u^2\bar{\sigma}_v^2}{V}\right)$ for convenience. Finally, the iteration at which the sampling probability transitions in (41) can be approximated by

$$n_\text{switch} \approx n_\text{s.s.} + \Delta n. \quad (47)$$

IV. Simulations

The results presented next were obtained from an ensemble average of 100 independent realizations. In each experiment, we consider the network topology presented in Fig. 1a. The x and y coordinates of each node were generated randomly from a Uniform distribution in the range $[-1, 1]$. Then, an edge was added between two nodes whenever the distance between them was less than 0.6. The input signal $u_k(n)$ and the measurement noise $v_k(n)$ follow Gaussian distributions with zero mean for each node k, with $\sigma_{u_k}^2 = \sigma_u^2 = 1$, whereas the noise variance $\sigma_{v_k}^2$ is drawn from a uniform distribution in the range $[0.001, 0.01]$ for $k = 1, \cdots, V$, as shown in Fig. 1b. We consider $M = 10$ for both the optimal solution and the diffusion algorithm, and adopt $\mu = 0.1$. The coefficients of the optimal solution $\mathbf{w}_\text{o}$ are drawn from a Uniform distribution in the range $[-1, 1]$, and are later normalized so as to obtain $\|\mathbf{w}_\text{o}\|^2 = 1$. As stated in Sec. II, we adopt Metropolis combination weights. For the theoretical model, we consider $\delta = 0.01$ in Eq. (46).

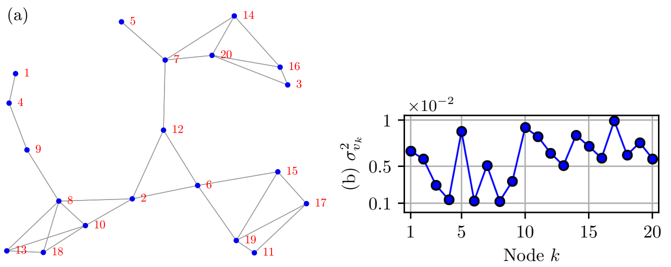

Fig. 1: (a) Network topology, and (b) noise variance profile considered in the simulations.

In Fig. 2, we show the theoretical curves as well as the simulation results obtained with the DTRAS-dLMS algorithm with $\gamma = 25$ and $\Delta n = 200$. These parameters were chosen so as to obtain a significant reduction in the sampling probability in steady state and a good transient performance. It is worth noting that in this case (40) yields $p_{\text{s.s.}_\text{max}} = 0.04$. For reference, we also show the results for the dLMS algorithm with fixed sampling probabilities $p_\zeta = 1$ and $p_\zeta = p_{\text{s.s.}_\text{max}}$. In Fig. 2a we show the NMSD curves, and in Fig. 2b the sampling probability along the iterations. We notice that, initially, the DTRAS-dLMS algorithm maintains every node sampled, and consequently presents the same convergence rate as the dLMS algorithm with $p_\zeta = 1$, which is captured by our theoretical model. Then, after $n_\text{switch} = 146$ iterations, the sampling probability of the DTRAS-dLMS suddenly decreases. We observe that the approximation given by Eq. (41) is reasonable in this case, and we notice that the NMSD of DTRAS-dLMS begins to decrease until it stabilizes at a steady-state level approximately 6 dB lower than that of the dLMS algorithm with every node sampled. This is also predicted by the model of Sec. III. Overall, we observe that the theoretical curves match the simulation results well. Lastly, we observe that the dLMS algorithm with a fixed sampling probability of $p_\zeta = 0.04$ converges approximately to the same level of steady-state NMSD as the DTRAS-dLMS algorithm, but at a much slower convergence rate. Simulation results obtained with other values of M and μ led to similar conclusions, but are omitted here due to space restrictions.

Lastly, in Fig. 3, we repeat the previous experiment, but considering $\Delta n = 1000$ for the DTRAS-dLMS algorithm. In this case, the theoretical model underestimates n_switch, and, as a result, there is a noticeable mismatch between the simulation results and the theoretical NMSD curve between $n \approx 2000$ and $n \approx 2800$. Overall, the simulations suggest that the theoretical model for n_switch is fairly accurate for relatively small values of Δn, but can lead to poor estimates if Δn is very large. Nonetheless, we remark that, typically, it is not desirable to choose excessively large values for Δn, since in this case it takes more iterations for the algorithm to cease the sampling of the nodes. Thus, the situation depicted in Fig. 3 is not of practical interest. Nevertheless, in future works we intend to

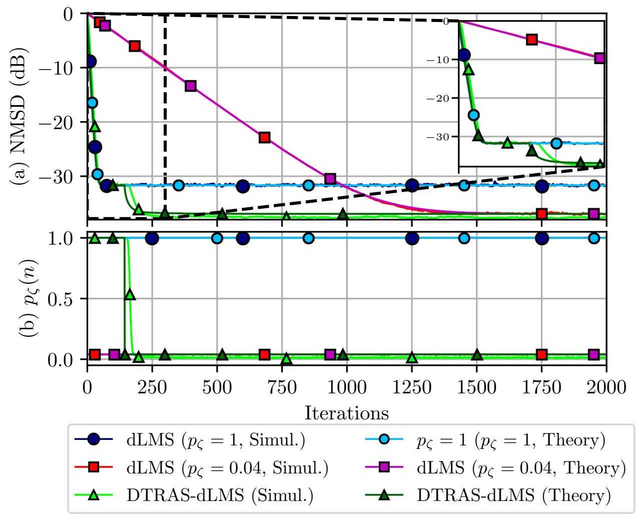

Fig. 2: Comparison between the theoretical models and the simulation results with $\gamma = 25$ and $\Delta n = 100$. (a) NMSD curves, and (b) sampling probability along the iterations.

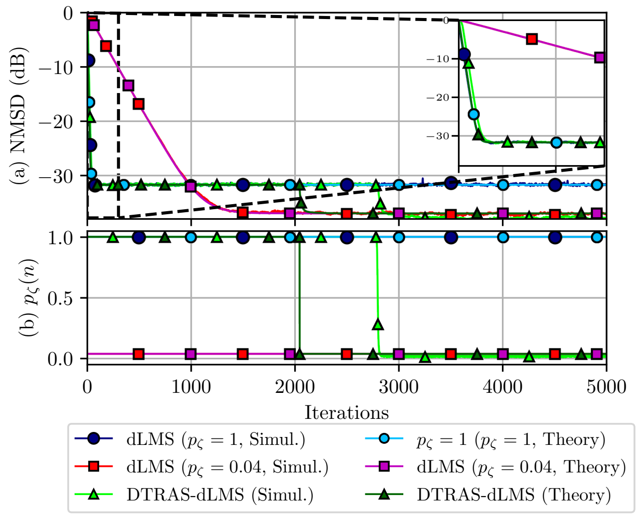

Fig. 3: Comparison between the theoretical models and the simulation results with $\gamma = 25$ and $\Delta n = 1000$. (a) NMSD curves, and (b) sampling probability along the iterations.

improve our estimate of n_{switch}. Despite this, it is interesting to note that the theoretical model still predicts the steady-state NMSD accurately in this scenario.

V. Conclusions

In this paper, we derived a theoretical model for the NMSD of the DTRAS-dLMS algorithm of [9]. The theoretical curves thus obtained match the simulation results well for relatively small values of Δn, which corresponds to the scenario of greater practical interest. For future works, we intend to refine our analysis for greater values of Δn. Our model shows that, by keeping the nodes sampled in the transient phase, and ceasing to sample them otherwise, it is possible to obtain a slightly improved steady-state performance in comparison with the case in which every node is permanently sampled, while preserving its convergence rate. Thus, our theoretical analysis shows that, by managing the sampling of the nodes in an intelligent manner, it is possible for adaptive diffusion networks to perform better with less, rather than more, data.

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] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing: array and statistical signal processing*, R. Chellapa and S. Theodoridis, Eds., vol. 3, chapter 9, pp. 323–456. Academic Press, 2014.
[3] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, pp. 460–497, Apr. 2014.
[4] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, 2008.
[5] 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, Jun. 2008.
[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] 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, May 2017.
[8] D. G. Tiglea, R. Candido, and M. T. M. Silva, "A low-cost algorithm for adaptive sampling and censoring in diffusion networks," *IEEE Trans. Signal Process.*, vol. 69, pp. 58–72, Jan. 2021.
[9] D. G. Tiglea, R. Candido, and M. T. M. Silva, "An adaptive algorithm for sampling over diffusion networks with dynamic parameter tuning and change detection mechanisms," *Digit. Signal Process.*, vol. 127, pp. 103587, Jul. 2022.
[10] D. G. Tiglea, R. Candido, and M. T. M. Silva, "On the impact of random node sampling on adaptive diffusion networks," Submitted for publication in the IEEE Transactions on Signal Processing. Available as arXiv:2403.17323 [eess.SP], Mar. 2024.
[11] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, NJ, 2008.
[12] S. Haykin, *Adaptive Filter Theory*, Pearson, Upper Saddle River, 5th edition, 2014.
[13] C. G. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," in *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, 2008, pp. 3285–3288.
[14] L. Lu and H. Zhao, "Diffusion leaky LMS algorithm: Analysis and implementation," *Signal Process.*, vol. 140, pp. 77–86, Nov. 2017.
[15] S. Werner, Y.-F. Huang, M. L. R. De Campos, and V. Koivunen, "Distributed parameter estimation with selective cooperation," in *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, 2009, pp. 2849–2852.
[16] 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.
[17] C.-K. Yu and A. H. Sayed, "A strategy for adjusting combination weights over adaptive networks," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, 2013, pp. 4579–4583.
[18] S.-Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proc. IEEE Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2011, pp. 317–320.
[19] 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.
[20] T. Strutz, "Estimation of measurement-noise variance for variable-step-size NLMS filters," in *Proc. European Signal Process. Conf. (EUSIPCO)*, 2019, pp. 1–5.
[21] V. H. Nascimento and M. T. M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing: Signal Processing Theory and Machine Learning*, R. Chellapa and S. Theodoridis, Eds., vol. 1, chapter 12, pp. 619–761. Academic Press, Chennai, 2014.