Published in *Handbook on Array Processing and Sensor Networks*, S. Haykin and K. J. Ray Liu, Editors, Wiley, NJ, pp. 695–722, 2009.

Distributed Adaptive Learning Mechanisms

Ali H. Sayed and Federico S. Cattivelli

Abstract— **The chapter describes recent developments in distributed processing over adaptive networks. The resulting adaptive learning rules rely on local data at the individual nodes and on collaborations among neighboring nodes in order to exploit the space-time dimension of the data more fully. The ideas are illustrated by considering algorithms of the least-mean-squares type, although more general adaptation rules are also possible including least-squares rules and Kalman-type rules. Both incremental and diffusion collaboration strategies are considered.**

I. Introduction

Distributed networks linking sensors and actuators will form the backbone of future data communication and control networks. Applications will range from sensor networks to precision agriculture, environment monitoring, disaster relief management, smart spaces, target localization, as well as medical applications [1]–[5]. In all these cases, the distribution of the nodes in the field yields spatial diversity, which should be exploited alongside the temporal dimension in order to enhance the robustness of the processing tasks and improve the probability of signal and event detection.

Distributed processing techniques allow for the efficient extraction of temporal and spatial information from data collected at such distributed nodes by relying on local cooperation and data processing. For example, each node in the network could collect noisy observations related to a certain parameter of interest. The nodes would then interact with their neighboring nodes, as dictated by the network topology, in order to estimate the parameter. The objective is to arrive at an estimate that is as reliable as the one that would be obtained if each node had access to the information across the entire network.

In contrast, in the centralized approach to parameter estimation, the data from all nodes would be conveyed to a central processor where they would be fused and the vector of parameters estimated. Such an approach requires sufficient communications resources to transmit the data back and forth between the nodes and the central processor, which would limit the autonomy of the network besides adding a critical point of failure in the network due to the presence of a central node [1], [6].

This chapter describes recent development in distributed processing over adaptive networks. The presentation covers adaptive algorithms that allow neighboring nodes to communicate with each other at every iteration. At each node, estimates exchanged with neighboring nodes are fused and promptly fed into the local adaptation rules. In this way, an adaptive network is obtained where the structure as a whole is able to respond in real-time to the temporal and spatial variations in the statistical profile of the data. Different adaptation or learning rules at the nodes, allied with different cooperation protocols, give rise to adaptive networks of various complexities and potential.

Obviously, the effectiveness of any distributed implementation depends on the modes of cooperation that are allowed among the nodes. Figure 1 illustrates three such modes of cooperation.

In an incremental mode of cooperation (see Fig. 1a), information flows in a sequential manner from one node to the adjacent node. This mode of operation requires a cyclic pattern of collaboration among the nodes, and has the advantage that for the last node in the cycle, the data from the entire network are used to update the desired parameter estimate, thereby offering excellent estimation performance. Moreover, for every measurement, every node needs to communicate with only one neighbor. However, incremental cooperation has the disadvantage of requiring the definition of a cycle, and network processing has to be faster than the measurement process, since a full communication cycle is needed for every measurement. This may become prohibitive for large networks. Incremental networks are also less robust to node and link failures.

An alternative protocol is the diffusion implementation (see Fig. 1b) where every node communicates with all of its neighbors as dictated by the network topology. This approach has no topology constraints and is more robust to node and link failure (see, e.g., [7]). It will have some performance degradation compared to an incremental solution, and also every node will need to communicate with its neighbors for every measurement, possibly requiring more energy than the incremental case. Note, however, that when omnidirectional communications are used, the energy required to transmit to one neighbor may be the same as that required to transmit to

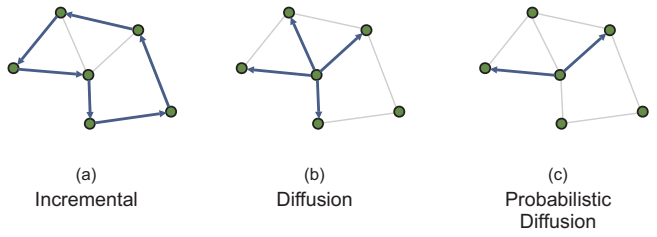

Fig. 1. Three modes of cooperation.

Published in *Handbook on Array Processing and Sensor Networks*, S. Haykin and K. J. Ray Liu, Editors, Wiley, NJ, pp. 695–722, 2009. This material was based on work supported in part by the National Science Foundation under awards ECS-0601266 and EECS-0725441. The authors are with the Electrical Engineering Department, University of California, Los Angeles, CA, 90095.

every neighbor.

The communication in the diffusion solution can be reduced by allowing each node to communicate only with a subset of its neighbors. This mode of cooperation is denoted probabilistic diffusion (see Fig. 1c). The choice of which subset of neighbors to communicate with can be randomized according to some performance criterion.

This chapter describes several developments in distributed processing over adaptive networks based on the works [13]–[23]. The resulting adaptive learning rules rely on local data at the individual nodes and on collaborations among neighboring nodes in order to exploit the space-time dimension of the data more fully. The ideas are illustrated by considering algorithms of the least-mean-squares type, although more general adaptation rules are also possible including least-squares rules and Kalman-type rules [16], [20], [21], [23]. Both incremental and diffusion strategies are considered in the sequel.

A. Notation

In the remainder of the chapter we use boldface letters for random quantities and normal font for non-random (deterministic) quantities. We also use capital letters for matrices and small letters for vectors. For example, $\boldsymbol{d}$ is a random quantity and d is a realization or measurement for it, and R is a covariance matrix while w is a weight vector. The notation * denotes complex conjugation for scalars and complex-conjugate transposition for matrices. The index i is used to denote iterations or time instants, and the indices k and ℓ are used to denote different nodes in a network with a total of N nodes.

II. Motivation

We motivate adaptive networks by examining an application in the context of data modeling. Thus, consider a set of N sensors scattered over a geographical area and observing some physical phenomenon of interest. Each node k collects a measurement $d_k(i)$ at time i. It is assumed that these measurements arise from an autoregressive (AR) model of the form:

$$d_k(i) = \sum_{m=1}^{M} \beta_m d_k(i-m) \; + \; v_k(i) \tag{1}$$

where $v_k(i)$ denotes additive zero-mean noise and the coefficients $\{\beta_m\}$ represent the underlying model. If we define the $M \times 1$ parameter vector

$$w^o = \text{col}\{\beta_1, \beta_2, \ldots, \beta_M\}, \quad (M \times 1)$$

and the $1 \times M$ regression vector

$$u_{k,i} = \left[\; d_k(i-1) \quad d_k(i-2) \quad \ldots \quad d_k(i-M) \; \right]$$

then we can express the measurement equation (1) at each node k in the equivalent form:

$$d_k(i) = u_{k,i} w^o + v_k(i) \tag{2}$$

The objective is to estimate the AR model coefficients $\{\beta_m\}$ or w^o from measurements $\{d_k(i)\}$ at all nodes. In other words, assuming that the $\{d_k(i)\}$ are realizations of a random variable

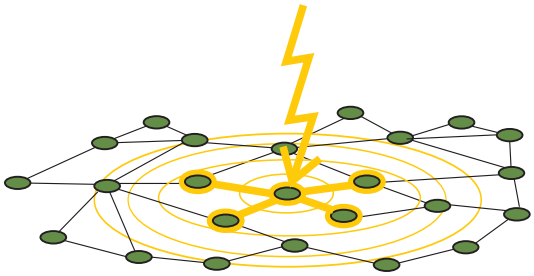

Fig. 2. A schematic representation of an adaptive network consisting of an interconnected system of adaptive nodes interacting with each other and with information flowing through the network in real-time.

$\boldsymbol{d}_k$ and the $\{u_{k,i}\}$ are realizations of a random vector $\boldsymbol{u}_k$, the objective is to find the vector w that minimizes the mean-square error

$$\frac{1}{N} \sum_{k=1}^{N} E|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2$$

One could employ individual adaptive filters at the nodes, with each node k estimating w^o independently of the other nodes by relying solely on its local data $\{d_k(i), \; u_{k,i}, \; i \geq 0\}$. In this case, each node will end up with a local estimate for w^o and the quality of this estimate will be dictated by the quality of the data at node k (such as the local SNR and noise conditions).

However, in situations where a multitude of nodes has access to data, and assuming some form of collaboration is allowed among the nodes, it is more useful to seek solutions that can take advantage of node cooperation. In addition, since the statistical profile of the data may vary with time and space, it is useful to explore cooperative strategies that are inherently adaptive. For example, the noise and signal-to-noise (SNR) conditions at the nodes may vary in time and space, and the model parameters $\{\beta_m\}$ themselves may vary with time as well. Under such conditions, it is helpful to endow the network of nodes with learning abilities so that it can function as an adaptive entity in its own right. By doing so, one would end up with an adaptive network where all nodes respond to data in real-time through local and cooperative processing, as well adapt to variations in the statistical properties of the data – see Fig. 2. It is expected that such cooperative adaptive schemes will result in improved performance over the decoupled individual filters in a non-cooperative implementation, and cooperation should help equalize the effect of varying SNR conditions across the network.

This chapter illustrates these ideas with several algorithms. We start by describing incremental adaptive networks in Section III, followed by diffusion networks in Section IV. In the process, we motivate and introduce the incremental LMS algorithm and two versions of the diffusion LMS algorithm: combine-then-adapt (CTA) and adapt-then-combine (ATC) diffusion LMS. We also comment on the performance of the algorithms via analysis and computer simulations.

III. Incremental Adaptive Solutions

Consider a network with N nodes and assume initially that at least one cyclic path can be established across the

network. The cyclic path should enable information to be moved from one node to a neighboring node around the network and back to the initial node (see Fig. 3). Obviously, some topologies may permit several possibilities for selecting such cyclic trajectories.

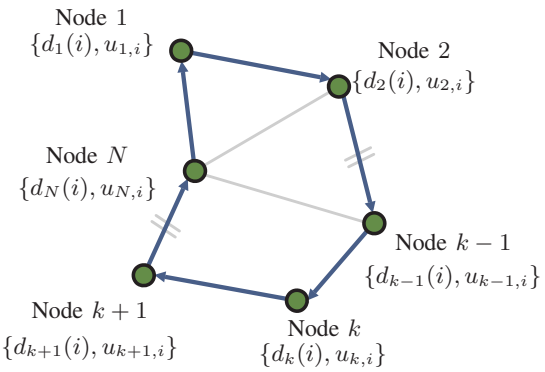

Fig. 3. A incremental network with N active nodes accessing space-time data.

Assume further that each node k has access to time-realizations $\{d_k(i), u_{k,i}\}$ of zero-mean data $\{\boldsymbol{d}_k, \boldsymbol{u}_k\}$, $k = 1, \ldots, N$, where each $\boldsymbol{d}_k$ is a scalar and each $\boldsymbol{u}_k$ is a $1 \times M$ (row) regression vector. We denote the $M \times M$ covariance matrices of the regression data by

$$R_{u,k} \triangleq E\, \boldsymbol{u}_k^* \boldsymbol{u}_k \quad \text{(at node } k\text{)} \tag{3}$$

and the $M \times 1$ cross-covariance vectors by

$$R_{du,k} \triangleq E\, \boldsymbol{d}_k \boldsymbol{u}_k^* \quad \text{(at node } k\text{)} \tag{4}$$

where E is the expectation operator. Observe that $\{R_{u,k}, R_{du,k}\}$ depend on k and, therefore, for generality, we are allowing the statistical profile of the data to vary spatially across the nodes. The special case of uniform statistical profile would correspond to assuming $R_{u,k} = R_u$ and $R_{du,k} = R_{du}$ for all k.

Our objective is to develop a mechanism that would allow the nodes to cooperate with each other in order to estimate some unknown $M \times 1$ vector w^o. We focus here on the mean-square error criterion and assume that the network seeks a vector w^o that solves the following estimation problem:

$$w^o \;=\; \arg\min_{w} \;\left(\frac{1}{N} \sum_{k=1}^{N} E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2 \right) \tag{5}$$

In other words, the optimal solution, w^o, should be such that it minimizes the average mean-square error (MSE) across the network. We shall refer to the optimal minimum cost as the resulting MSE network performance, namely,

$$\text{MSE}^{\text{network}} \;\triangleq\; \frac{1}{N} \sum_{k=1}^{N} E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w^o|^2 \tag{6}$$

Likewise, we shall denote the MSE performance at an individual node k by

$$\text{MSE}_k \;\triangleq\; E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w^o|^2 \tag{7}$$

Note that since N denotes the size of the network and is independent of the unknown w, then the optimization problem (5) is also equivalent to

$$w^o \;=\; \arg\min_{w} \;\sum_{k=1}^{N} E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2 \tag{8}$$

where the factor $1/N$ has been removed. We shall denote the cost function in (8) by

$$J(w) \;\triangleq\; \sum_{k=1}^{N} E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2 \tag{9}$$

It is worth noting that $J(w)$ decouples into a sum of individual cost functions, namely,

$$J(w) \;=\; \sum_{k=1}^{N} J_k(w) \tag{10}$$

where each individual $J_k(w)$ is given by

$$J_k(w) \;\triangleq\; E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2 \tag{11}$$

In optimization problems involving such decoupled cost functions, incremental methods have been used to seek the solution in a distributed manner [8]–[11], as we now explain.

A. Steepest-Descent Solution

To begin with, the traditional iterative steepest-descent solution for determining w^o in (8) takes the form

$$w_i = w_{i-1} - \mu\,[\nabla_w J(w_{i-1})]^* \tag{12}$$

where $\mu > 0$ is a step-size parameter and w_i is an estimate for w^o at iteration i. Moreover, $\nabla_w J$ denotes the complex gradient of $J(w)$ with respect to w, which is given by

$$\nabla_w J(w) \;=\; \sum_{k=1}^{N} (R_{u,k} w - R_{du,k})^*$$

Substituting into (12) leads to

$$w_i \;=\; w_{i-1} + \mu \sum_{k=1}^{N} (R_{du,k} - R_{u,k} w_{i-1}) \tag{13}$$

Thus observe that each iteration step in (13) involves evaluating a sum of N terms, namely,

$$\sum_{k=1}^{N} (R_{du,k} - R_{u,k} w_{i-1})$$

and adding the result to w_{i-1} in order to arrive at w_i. This same result can be achieved by splitting the update into N separate steps whereby each step adds one term, $R_{du,k} - R_{u,k} w_{i-1}$, at a time and gives an intermediate value, say as follows:

$$
\begin{aligned}
\psi_0^{(i)} &\leftarrow w_{i-1} \\
\\
\psi_1^{(i)} &= \psi_0^{(i)} + \mu(R_{du,1} - R_{u,1}w_{i-1}) \\
\psi_2^{(i)} &= \psi_1^{(i)} + \mu(R_{du,2} - R_{u,2}w_{i-1}) \\
\psi_3^{(i)} &= \psi_2^{(i)} + \mu(R_{du,3} - R_{u,3}w_{i-1}) \\
&\vdots \quad\quad \vdots \\
\psi_N^{(i)} &= \psi_{N-1}^{(i)} + \mu(R_{du,N} - R_{u,N}w_{i-1}) \\
\\
w_i &\leftarrow \psi_N^{(i)}
\end{aligned}
\tag{14}
$$

Observe that we are denoting the intermediate value at node k by $\psi_k^{(i)}$, and we are using $\psi_0^{(i)}$ to denote the initial condition at a virtual node 0. The procedure (14) defines a *cycle* visiting every node only once. At every iteration (or time i), the information cycles through all N nodes. Each $\psi_k^{(i)}$ represents a *local estimate* of w^o at node k and time i, and the process assumes that each node k has access to $\psi_{k-1}^{(i)}$, which is the local estimate of w^o at node $k-1$ – see Fig. 4.

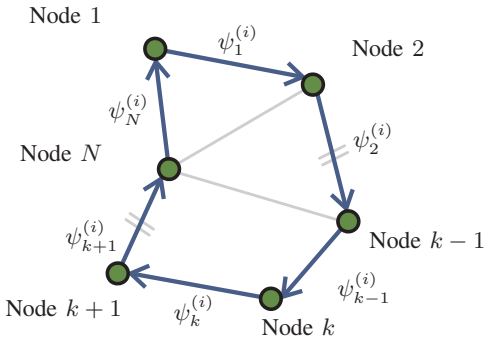

Fig. 4. A cycle covering nodes 1 through N.

The procedure (14) can be described more compactly as follows:

$$
\left\{
\begin{array}{l}
\psi_0^{(i)} \leftarrow w_{i-1} \\
\psi_k^{(i)} = \psi_{k-1}^{(i)} + \mu\left[R_{du,k} - R_{u,k}w_{i-1}\right] \\
\qquad k = 1, \ldots, N \\
w_i \leftarrow \psi_N^{(i)}
\end{array}
\right.
\tag{15}
$$

Note, in particular, that the iteration for $\psi_k^{(i)}$ is over the spatial index k. Note further that the update for $\psi_k^{(i)}$ requires knowledge of w_{i-1}, which enters into the computation of the update direction in (15), namely,

$$
R_{du,k} - R_{u,k}w_{i-1}
$$

The implication of this fact is that all N nodes will need to have access to w_{i-1}, which requires communicating the w_{i-1} to all nodes at each time i.

B. Incremental Solution

A distributed solution can be motivated by resorting to an approximation whereby the estimate w_{i-1} at each node in (15) is replaced by its local estimate, thus leading to what is known as an incremental solution. Specifically, if each node k relies solely on the local estimate $\psi_{k-1}^{(i)}$ received from node $k-1$, as opposed to requiring also w_{i-1}, then an incremental version of algorithm (15) would result in the following form:

$$
\left\{
\begin{array}{l}
\psi_0^{(i)} \leftarrow w_{i-1} \\
\psi_k^{(i)} = \psi_{k-1}^{(i)} + \mu\left[R_{du,k} - R_{u,k}\psi_{k-1}^{(i)}\right] \\
\qquad k = 1, \ldots, N \\
w_i \leftarrow \psi_N^{(i)}
\end{array}
\right.
\tag{16}
$$

where w_{i-1} in (15) has been replaced by $\psi_{k-1}^{(i)}$ in the update for $\psi_k^{(i)}$. Such incremental techniques have been studied extensively in the literature, and especially in the optimization literature and in works on distributed computational algorithms (e.g., [6], [8], [9], [12], [11]).

It is instructive to compare the performance of the steepest-descent algorithm (13) or (15) and its incremental version (16) [14]. Thus, recall that the desired vector w^o in (8) is the solution to the normal equations [24], [25]:

$$
\left(\sum_{k=1}^{N} R_{u,k}\right) w^o = \sum_{k=1}^{N} R_{du,k}
\tag{17}
$$

We assume that the coefficient matrix is positive-definite,

$$
\sum_{k=1}^{N} R_{u,k} \; > 0
$$

so that a unique solution w^o exists. Let

$$
\tilde{w}_i = w^o - w_i
$$

denote the weight-error vector at iteration i. Subtracting w^o from both sides of the steepest-descent recursion (13) leads to

$$
\tilde{w}_i \;\; = \;\; \left[I - \mu \sum_{k=1}^{N} R_{u,k}\right] \tilde{w}_{i-1}
\tag{18}
$$

This recursion describes the dynamics of the weight-error vector; it is seen that the evolution of the weight-error vector is governed by the modes of the coefficient matrix

$$
F_{\rm sd} \;\triangleq\; \left[I - \mu \sum_{k=1}^{N} R_{u,k}\right] \quad \text{(steepest-descent)}
$$

Let us now examine the evolution of $\tilde{w}_i$ when evaluated by means of the incremental implementation (16). Subtracting w^o from both sides of (16) gives, for $k = N$,

$$
\tilde{\psi}_N^{(i)} = \tilde{\psi}_{N-1}^{(i)} - \mu\left[R_{du,N} - R_{u,N}\psi_{N-1}^{(i)}\right]
\tag{19}
$$

where

$$
\tilde{\psi}_k^{(i)} \;=\; w^o - \psi_k^{(i)}
$$

Replacing $\psi_{N-1}^{(i)}$ in (19) by its update in terms of $\psi_{N-2}^{(i)}$ (as given by (16)), and continuing in this manner, some algebra will show that the evolution for the weight-error vector is now described by a recursion of the form:

$$
\tilde{w}_i = \left[\prod_{k=1}^{N} (I - \mu R_{u,k})\right] \tilde{w}_{i-1} \; + \; O(\mu^2)
\tag{20}
$$

where $O(\mu^2)$ denotes terms that are independent of $\tilde{w}_{i-1}$ and of the order of μ^2 or higher powers in μ. In the special case when the statistical profile is uniform across all nodes, i.e., $R_{u,k} = R_u$ and $R_{du,k} = R_{du}$, it can be verified that the driving term denoted by $O(\mu^2)$ in (20) becomes zero.

Therefore, the evolution of the weight-error vector $\tilde{w}_i$ that is generated by the incremental solution (16) is governed by the modes of the coefficient matrix:

$$F_{\rm inc} \triangleq \left[\prod_{k=1}^{N}(I-\mu R_{u,k})\right] \quad \text{(incremental)}$$

In order to illustrate the difference in the dynamics of both implementations, consider the special case of uniform statistical profile across all nodes. Then

$$F_{\rm sd} = (I-\mu N R_u), \qquad F_{\rm inc} = (I-\mu R_u)^N$$

from which we find that the M modes of convergence of the algorithms are given by

$$\begin{aligned} \text{modes}^{\rm sd} &= \{1-\mu N\lambda_m\} \\ \text{modes}^{\rm inc} &= \{(1-\mu\lambda_m)^N\} \\ & m = 1,2,\ldots,M \end{aligned}$$

in terms of the eigenvalues $\{\lambda_m\}$ of R_u. In this case, a necessary condition for convergence in the steepest-descent case (18) is

$$\mu < \frac{2}{N\lambda_{\max}} \quad \text{(steepest-descent)}$$

whereas a necessary condition for convergence in the incremental case (20) is

$$\mu < \frac{2}{\lambda_{\max}} \quad \text{(incremental)}$$

where $\lambda_{\max}$ is the maximum eigenvalue of R_u. These conditions indicate that the incremental solution (16) converges over a wider range of the step-size.

Figure 5 shows the magnitudes of the modes of convergence for the case $R_u = \lambda I$ and $N = 6$ nodes, as a function of $\mu\lambda$, both for the steepest-descent and incremental algorithms (18) and (20), respectively. Note that for very small $\mu\lambda$, $(1-\mu\lambda)^N \approx 1 - N\mu\lambda$, and both algorithms have similar performance. As we increase $\mu\lambda$, the steepest-descent algorithm has faster convergence, though it quickly becomes unstable when $\mu\lambda = 2/N$. For larger step-sizes, the incremental algorithm has a faster convergence rate than the steepest-descent solution. Note further that the stability range for the incremental algorithm is wider, leading to a more robust implementation.

Still both algorithms tend to exhibit the same behavior for diminishing step-sizes. This can be seen from the weight-error recursion (20) for the incremental solution, where the coefficient matrix can be expressed as

$$\left[\prod_{k=1}^{N}(I-\mu R_{u,k})\right] = \left[I-\mu\sum_{k=1}^{N}R_{u,k}\right] + O(\mu^2)$$

so that for vanishingly small step-sizes,

$$F_{\rm inc} = F_{\rm sd} + O(\mu^2)$$

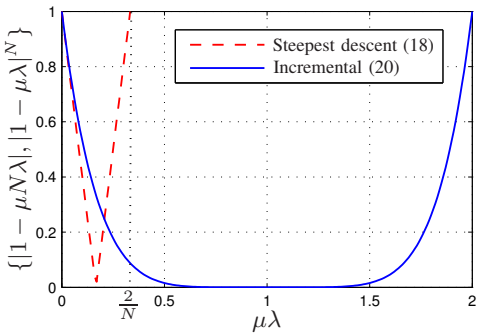

Fig. 5. Modes of convergence for algorithms (15) and (16) for $N = 6$.

and the weight-error vectors $\{\tilde{w}_i\}$ from both algorithms (15) and (16) evolve along similar dynamics. This same conclusion follows from examining the update equation for the weight estimates directly [14]. Indeed, note from (11) that

$$[\nabla J_k(w)]^* = -R_{du,k} + R_{u,k}w \tag{21}$$

Inspecting (21) we note that the following property holds for a scalar μ and any two column vectors x and y:

$$[\nabla J_k(x-\mu y)]^* = [\nabla J_k(x)]^* - \mu\,[\nabla J_k(y)]^* - \mu R_{du,k} \tag{22}$$

where ∇J_k is computed relative to w.

Now, if we iterate the incremental solution (16) starting with $\psi_0^{(i)} = w_{i-1}$, we get:

$$\psi_{k-1}^{(i)} = w_{i-1} - \mu\sum_{\ell=1}^{k-1}\left[\nabla J_\ell(\psi_{\ell-1}^{(i)})\right]^* \tag{23}$$

Substituting (23) into (16) gives

$$\psi_k^{(i)} = \psi_{k-1}^{(i)} - \mu\left[\nabla J_k\left(w_{i-1} - \mu\sum_{\ell=1}^{k-1}\left[\nabla J_\ell(\psi_{\ell-1}^{(i)})\right]^*\right)\right]^*$$

Using relation (22) with the choices

$$x = w_{i-1}, \qquad y = \sum_{\ell=1}^{k-1}\left[\nabla J_\ell(\psi_{\ell-1}^{(i)})\right]^*$$

leads to

$$\begin{aligned} \psi_k^{(i)} = {} & \underbrace{\psi_{k-1}^{(i)} - \mu\Big([\nabla J_k(w_{i-1})]^*\Big)}_{\text{steepest-descent as in (15)}} \\ & \underbrace{+\,\mu^2\left(\left[\nabla J_k\left(\sum_{\ell=1}^{k-1}\left[\nabla J_\ell(\psi_{\ell-1}^{(i)})\right]^*\right)\right]^* + R_{du,k}\right)}_{\text{extra term due to incremental procedure}} \end{aligned} \tag{24}$$

Therefore, the incremental algorithm can be written as a sum of the steepest-descent update plus extra terms. As $\mu \to 0$, the μ term dominates the μ^2 term and the incremental algorithm (16) and the steepest-descent algorithm (15) tend to exhibit the same behavior.

C. Adaptation: Incremental LMS

Now note that the incremental algorithm (16) requires knowledge of the second-order moments $\{R_{u,k}, R_{du,k}\}$. An adaptive implementation of (16) can be obtained by replacing these second-order moments by local instantaneous approximations, say of the LMS type, as follows:

$$R_{du,k} \approx d_k(i)u_{k,i}^*, \qquad R_{u,k} \approx u_{k,i}^* u_{k,i} \tag{25}$$

Obviously, more involved approximations are possible and they would lead to alternative adaptive implementations. Using the approximations (25) leads to the *incremental LMS* algorithm derived in [14], [18], where we additionally allow for the step-sizes to vary across the nodes:

Incremental LMS: Start with $w_{-1} = 0$. For each time $i \geq 0$, repeat:

$$\left\{ \begin{array}{l} \psi_0^{(i)} = w_{i-1} \\ \psi_k^{(i)} = \psi_{k-1}^{(i)} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\psi_{k-1}^{(i)} \right) \\ \qquad\qquad k = 1, \ldots, N \\ w_i = \psi_N^{(i)} \end{array} \right. \tag{26}$$

One question is how well the adaptive algorithm (26) performs. A detailed mean-square and stability analysis of the algorithm is performed in [14], [18]. The analysis relies on the following assumptions on the data: $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$:

1) The unknown vector w^o relates $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ as

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i) \tag{27}$$

   where $\boldsymbol{v}_k(i)$ is some white noise sequence with variance $\sigma_{v,k}^2$ and independent of $\{\boldsymbol{d}_\ell(j), \boldsymbol{u}_{\ell,j}\}$ for all ℓ, j.
2) $\boldsymbol{u}_{k,i}$ is independent of $\boldsymbol{u}_{\ell,i}$ for $k \neq \ell$ (spatial independence).
3) For every k, the sequence $\{\boldsymbol{u}_{k,i}\}$ is independent over time (time independence).
4) The regressors $\{\boldsymbol{u}_{k,i}\}$ arise from a source with circular Gaussian distribution with covariance matrix $R_{u,k}$.

It is worth noting that for linear data models of the form (27), the solution w^o of the mean-square-error criterion in (5) coincides with the desired unknown vector in (27) [24], [25].

The following results are simplifications of the general expressions derived in [14] assuming sufficiently small step-sizes. Define the error signals:

$$\widetilde{\boldsymbol{\psi}}_k^{(i)} \triangleq w^o - \boldsymbol{\psi}_k^{(i)} \tag{28}$$

$$\boldsymbol{e}_{a,k}(i) \triangleq \boldsymbol{u}_{k,i}\widetilde{\boldsymbol{\psi}}_{k-1}^{(i)} \tag{29}$$

where (28) denotes the weight-error vector and (29) denotes the *a priori* local error, both at node k and time i. Observe that we are now denoting $\widetilde{\boldsymbol{\psi}}_k^{(i)}$ and $\boldsymbol{e}_{a,k}(i)$ by boldface letters to highlight the fact that they are random quantities whose variances we are interested in evaluating.

For each node k, the mean-square deviation (MSD) and the excess mean-square error (EMSE) are defined as the steady-state values of the variances of these error quantities, namely,

$$\eta_k \triangleq E\left\|\widetilde{\boldsymbol{\psi}}_k^{(\infty)}\right\|^2 \qquad \text{(MSD)} \tag{30}$$

$$\zeta_k \triangleq E|\boldsymbol{e}_{a,k}(\infty)|^2 \qquad \text{(EMSE)} \tag{31}$$

In the case of small step-sizes, simplified expressions for the MSD and EMSE can be described as follows. For each node k, introduce the eigen-decomposition

$$R_{u,k} = U_k \Lambda_k U_k^*$$

where U_k is unitary and Λ_k is a diagonal matrix with the eigenvalues of $R_{u,k}$:

$$\Lambda_k = \text{diag}\{\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,M}\} \quad (\text{node } k)$$

Define further the quantities:

$$\begin{aligned} D &\triangleq 2\sum_{k=1}^{N} \mu_k \Lambda_k \quad \text{(diagonal matrix)} \\ b_k &\triangleq \text{diag}\{\Lambda_k\} \qquad \text{(column vector)} \\ a &\triangleq \sum_{k=1}^{N} \mu_k^2 \sigma_{v,k}^2 b_k \quad \text{(column vector)} \\ q &\triangleq \text{col}\{1, 1, \ldots, 1\} \end{aligned}$$

where $\sigma_{v,k}^2$ denotes the noise variance at node k. Then, according to the results from [13], [14]:

$$\eta_k \approx a^T D^{-1} q \qquad \text{(MSD)} \tag{32}$$

$$\zeta_k \approx a^T D^{-1} b_k \qquad \text{(EMSE)} \tag{33}$$

or, more explicitly,

$$\eta_k \approx \frac{1}{2}\sum_{j=1}^{M} \left(\frac{\sum_{\ell=1}^{N} \mu_\ell^2 \sigma_{v,\ell}^2 \lambda_{\ell,j}}{\sum_{\ell=1}^{N} \mu_\ell \lambda_{\ell,j}} \right) \tag{34}$$

$$\zeta_k \approx \frac{1}{2}\sum_{j=1}^{M} \left(\lambda_{k,j} \cdot \frac{\sum_{\ell=1}^{N} \mu_\ell^2 \sigma_{v,\ell}^2 \lambda_{\ell,j}}{\sum_{\ell=1}^{N} \mu_\ell \lambda_{\ell,j}} \right) \tag{35}$$

Moreover, the mean-square performance at each node is given by

$$\text{MSE}_k = \zeta_k + \sigma_{v,k}^2 \tag{36}$$

The fact that the expression (32) for the MSD is independent of k reveals an interesting behavior. Namely, there is an equalization effect on the MSD throughout the network.

In order to illustrate the adaptive network performance, we present a simulation example in Figs. 6 and 7. Fig. 6 depicts the network topology with $N = 15$ nodes, together with the network statistical profile. The regressors are zero-mean Gaussian, independent in time and space, with covariance matrices $R_{u,k}$. The background noise power is denoted by $\sigma_{v,k}^2$. Fig. 7 shows the steady-state performance for the incremental LMS algorithm (26), using a uniform $\mu = 0.01$. The results were averaged over 200 experiments, and the steady-state values were calculated by averaging the last 50 samples after convergence. The figure shows the simulated steady-state MSD and EMSE for every node in the network, and compares them with the theoretical results from (34) and (35).

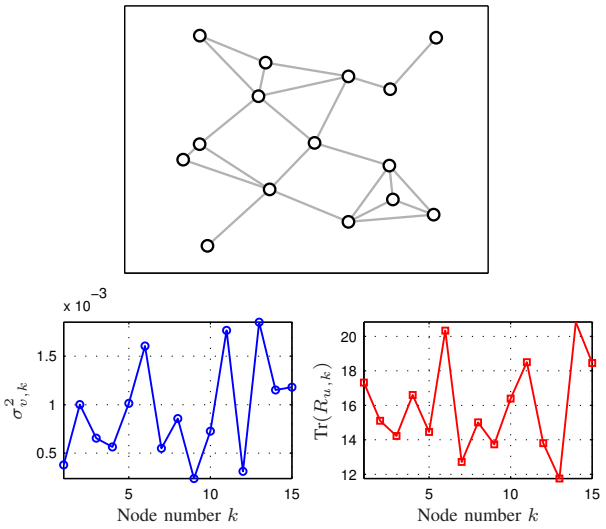

Fig. 6. Network topology (top), noise variances $\sigma^2_{v,k}$ (bottom, left) and trace of regressor covariances $\mathrm{Tr}(R_{u,k})$ (bottom, right) for $N = 15$ nodes.

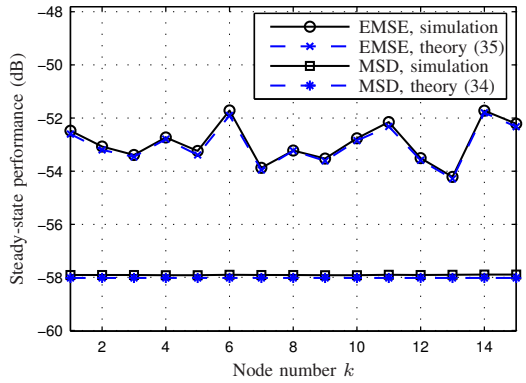

Fig. 7. Steady-state performance of the incremental LMS algorithm (26); theory and simulation.

IV. Diffusion Adaptive Solutions

The adaptive incremental solution (26) requires a cyclic trajectory across the entire network, which can limit its application and make the procedure less robust to node and link failures. In particular, observe that the updates progress sequentially from one node to another so that the generation of $\psi_k^{(i)}$ at node k can only happen after $\psi_{k-1}^{(i)}$ has been generated at node $k-1$.

However, when more communication resources are available, we should be able to take advantage of the network connectivity and devise more sophisticated cooperation rules among the nodes. For instance, node k does not need to rely solely on information from node $k-1$; it should also be able to rely on information from other nodes in its neighborhood. In addition, it should be possible for all nodes in the network to undergo updates simultaneously whenever possible without being limited by sequential processing.

Thus observe from (26) that the update for each node k relies on receiving the local estimate $\psi_{k-1}^{(i)}$ from its neighbor $k-1$. What if the network topology allows cooperation between node k and several other neighboring nodes? In this case, one could consider providing node k with a local estimate that is not only based on what node $k-1$ has to offer, but also on the information that the other neighboring nodes can offer. For example, one could consider replacing the local estimate $\psi_{k-1}^{(i)}$ in the incremental iteration (26) by some linear combination of the local estimates at the neighbors of node k, say, replace $\psi_{k-1}^{(i)}$ by

$$\phi_k^{(i-1)} \triangleq \sum_{\ell\in\mathcal{N}_k} a_{k\ell}\psi_k^{(i-1)} \tag{37}$$

The coefficients $\{a_{k\ell}\}$ are scaling factors that add up to one,

$$\sum_{\ell\in\mathcal{N}_k} a_{k\ell} = 1 \quad \text{(for each node } k\text{)} \tag{38}$$

and the notation $\mathcal{N}_k$ denotes the set of all nodes lying in the neighborhood of node k (including k itself), i.e., it is the set of all nodes ℓ that can communicate with node k:

$$\mathcal{N}_k = \{\text{set of nodes connected to } k \text{ including itself}\} \tag{39}$$

More generally, the neighborhood $\mathcal{N}_k$ could also vary with time, say as $\mathcal{N}_{k,i}$, but we are going to continue to work with $\mathcal{N}_k$ in this chapter for simplicity of exposition. We comment on choices for the combination coefficients $\{a_{k\ell}\}$ later in Section IV-C.

A. Adaptation: Node-Based Diffusion

Using (37), one can then consider replacing the incremental update (26) by the following recursion proposed in [15], [19] – see Fig. 8:

CTA Diffusion LMS: Start with $\left\{\psi_\ell^{(-1)} = 0\right\}$ for all ℓ. For each time $i \geq 0$ and for each node k, repeat:

$$\begin{aligned}\phi_k^{(i-1)} &= \sum_{\ell\in\mathcal{N}_k} a_{k\ell}\ \psi_\ell^{(i-1)} \qquad \text{(CTA version)} \qquad (40)\\ \psi_k^{(i)} &= \phi_k^{(i-1)} + \mu_k u_{k,i}^*\left(d_k(i) - u_{k,i}\phi_k^{(i-1)}\right)\end{aligned}$$

Note that the cyclic update through the nodes has been removed. Now, instead, at each iteration i, every node k performs a two-step procedure: an initial aggregation step to evaluate the aggregate (intermediate) estimate $\phi_k^{(i-1)}$ and a subsequent adaptation step to update the local node estimate to $\psi_k^{(i)}$. The aggregation step combines estimates $\{\psi_\ell^{(i-1)}\}$ from the *previous* time step $i-1$. In this way, all nodes across the network can perform their diffusion updates at the same time.

We refer to the above algorithm as *diffusion LMS* or, more specifically, as the Combine-then-Adapt (CTA) diffusion LMS version. The term diffusion is used to highlight the fact that information is being shared (or diffused) among the nodes in the neighborhood and, more generally, among the nodes in the entire network. This is because the aggregation step incorporates information from other neighborhoods into $\phi_k^{(i-1)}$.

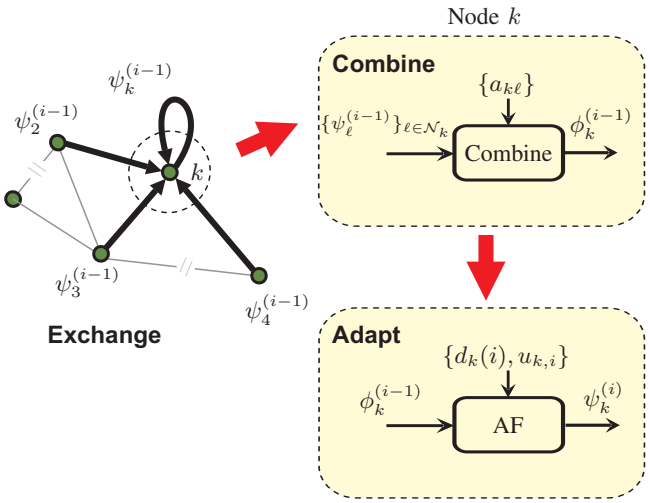

Fig. 8. A network with CTA diffusion strategy.

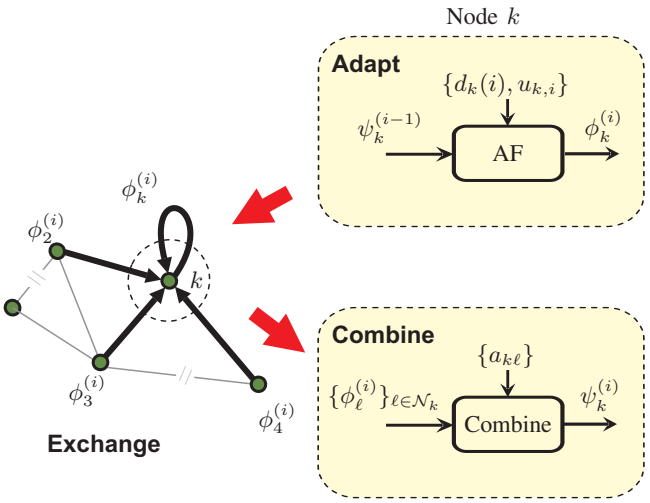

Fig. 9. A network with ATC diffusion strategy.

A useful alternative to the diffusion algorithm (40) is to perform the adaptation step first followed by the aggregation step, say (see Fig. 9):

ATC Diffusion LMS: Start with $\left\{\psi_\ell^{(-1)} = 0\right\}$ for all ℓ. For each time $i \geq 0$ and for each node k, repeat:

$$\begin{aligned} \phi_k^{(i)} &= \psi_k^{(i-1)} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\psi_k^{(i-1)}\right) \\ \psi_k^{(i)} &= \sum_{\ell\in\mathcal{N}_k} a_{k\ell}\ \phi_\ell^{(i)} \qquad \text{(ATC version)} \end{aligned} \tag{41}$$

We refer to (41) as the Adapt-then-Combine (ATC) diffusion LMS algorithm. Analysis and simulations show that ATC outperforms CTA. Intuitively, this is because the ATC version performs the adaptation step first, which incorporates the current data into the local weight estimates, $\phi_\ell^{(i)}$, before combining them. We should note that for both versions, the local weight vector estimate at node k and time i is taken as $\psi_k^{(i)}$.

One could also consider other diffusion schemes whereby the aggregation step involves more general functions of the local estimates, say as [15]:

$$\begin{aligned} \phi_k^{(i-1)} &= f_k\left(\psi_\ell^{(i-1)}; \ell\in\mathcal{N}_k\right) \\ \psi_k^{(i)} &= \phi_k^{(i-1)} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\phi_k^{(i-1)}\right) \end{aligned} \tag{42}$$

for some local combiner $f_k(\cdot)$. The combiners $f_k(\cdot)$ can be nonlinear or even time-variant, to reflect, for instance, changing topologies or to respond to non-stationary environments. For illustration purposes we continue to focus on the linear combination structures defined by (40) and (41).

B. Mean-Square-Error Optimization

The CTA and ATC diffusion LMS algorithms so described can be motivated formally in the same manner as the incremental LMS algorithm by starting from a mean-square cost function as follows. Consider node k and assume each node ℓ in its neighborhood has some initial estimate for the weight vector w, say $\{\psi_\ell,\ \ell\in\mathcal{N}_k\}$. We then formulate at node k the problem of estimating the weight vector w that solves:

$$\min_w \left(\delta \sum_{\ell\in\mathcal{N}_k} c_{k\ell}\|w-\psi_\ell\|^2 \ +\ E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2\right) \tag{43}$$

where $\delta > 0$ is a regularization parameter and the $\{c_{k\ell}\}$ are some weighting coefficients that add up to one:

$$\sum_{\ell\in\mathcal{N}_k} c_{k\ell} = 1$$

The second term in the above cost function is the same function $J_k(w)$ used earlier in (11); this term involves only local information and seeks that value of w that helps match $\boldsymbol{d}_k$ to $\boldsymbol{u}_k w$ in the mean-square error sense. The first term in the cost function (43) penalizes the distance between the solution w and the prior information represented by the available local estimates $\{\psi_\ell\}$. This is a useful term because it incorporates global information from other neighborhoods in the network; this is because the estimates $\{\psi_\ell\}$ are expected to have been influenced by data across the other neighborhoods.

Now note that the cost function in (43) decouples into the sum of two individual cost functions:

$$\delta \sum_{\ell\in\mathcal{N}_k} c_{k\ell}\|w-\psi_\ell\|^2$$

and

$$E\,|\boldsymbol{d}_k - \boldsymbol{u}_k w|^2$$

Thus, as before, an incremental approach can be used to carry out the optimization at node k. Let $\{\psi_k^{(i)},\ i\geq 0\}$ denote the successive iterates at node k that result from applying a steepest-descent approach to minimizing (43). Then the traditional steepest-descent solution, with the gradient vector of the cost function evaluated at the prior iterate $\psi_k^{(i-1)}$, is

given by

$$\begin{aligned}\psi_k^{(i)} &= \psi_k^{(i-1)} + \mu\left(R_{du,k} - R_{u,k}\psi_k^{(i-1)}\right) \\ &\quad - \mu\delta\sum_{\ell\in\mathcal{N}_k} c_{k\ell}\left(\psi_k^{(i-1)} - \psi_\ell\right)\end{aligned}$$

We can split this update into two incremental update steps, say as:

$$\phi_k^{(i)} = \psi_k^{(i-1)} + \mu\left(R_{du,k} - R_{u,k}\psi_k^{(i-1)}\right) \quad (44)$$

$$\psi_k^{(i)} = \phi_k^{(i)} - \mu\delta\sum_{\ell\in\mathcal{N}_k} c_{k\ell}\left(\psi_k^{(i-1)} - \psi_\ell\right) \quad (45)$$

with the intermediate variable denoted by $\phi_k^{(i)}$. And, just like we replaced w_{i-1} of (15) by the local estimate $\psi_{k-1}^{(i)}$ in (16), we can also replace $\psi_k^{(i-1)}$ in (45) by the local estimate $\phi_k^{(i)}$ from (44). This approximation leads to

$$\phi_k^{(i)} = \psi_k^{(i-1)} + \mu\left(R_{du,k} - R_{u,k}\psi_k^{(i-1)}\right) \quad (46)$$

$$\psi_k^{(i)} = \phi_k^{(i)} - \mu\delta\sum_{\ell\in\mathcal{N}_k} c_{k\ell}\left(\phi_k^{(i)} - \psi_\ell\right) \quad (47)$$

Recall that the $\{\psi_\ell\}$ in (47) are local estimates at the nodes ℓ in the neighborhood of k. One useful way to approximate these estimates is by replacing them by the values $\{\phi_l^{(i)}\}$ that are available at time i at these nodes, so that iteration (47) becomes

$$\begin{aligned}\psi_k^{(i)} &= \phi_k^{(i)} - \mu\delta\sum_{\ell\in\mathcal{N}_k} c_{k\ell}\left(\phi_k^{(i)} - \phi_\ell^{(i)}\right) \\ &= (1-\mu\delta+\mu\delta c_{kk})\phi_k^{(i)} + \sum_{\ell\in\mathcal{N}_k-\{k\}}\mu\delta c_{k\ell}\phi_\ell^{(i)}\end{aligned}$$

Introduce the coefficients

$$a_{kk} = (1-\mu\delta+\mu\delta c_{kk}), \quad a_{k\ell} = \mu\delta c_{k\ell},\ k\neq\ell$$

Note that the $\{a_{k\ell}\}$ defined in this manner add up to one. Note also that $a_{k\ell} = c_{k\ell}$ if we set $\delta = \mu^{-1}$. Then we obtain

$$\begin{aligned}\phi_k^{(i)} &= \psi_k^{(i-1)} + \mu\left(R_{du,k} - R_{u,k}\psi_k^{(i-1)}\right) \\ \psi_k^{(i)} &= \sum_{\ell\in\mathcal{N}_k} a_{k\ell}\phi_\ell^{(i)}\end{aligned}$$

If we apply the instantaneous approximations (25), and make the step-size node-dependent, then we arrive at the ATC diffusion LMS algorithm (41).

The CTA version (40) can be obtained in a similar manner if we simply reverse the order by which the incremental split was done in (44)–(45).

C. Combination Rules

There are several ways by which the combination weights $\{a_{k\ell}\}$ can be selected. We list here some examples that have been used in the literature in the context of graph problems. We also motivate the case where the weights $\{a_{k\ell}\}$ can be adapted as well; in this case, the network gains another level of adaptation and the nodes are able to give less or more weight selectively to their neighbors according to their performance and reliability.

One of the simplest choices for the $\{a_{k\ell}\}$ is to average the neighboring estimates. Thus, let n_k denote the degree of node k, which is defined as the number of incident links at the node (including a link from the node onto itself). In other words, n_k is the size of the neighborhood of k:

$$\begin{aligned}n_k &\triangleq \text{number of neighbors of node } k \text{ including itself} \\ &= |\,\mathcal{N}_k\,|\end{aligned}$$

Then we may select (see, e.g., [26])

$$a_{k\ell} = \frac{1}{n_k} \quad \text{for each } \ell\in\mathcal{N}_k$$

In this case, each node is assigned the same weight and

$$\sum_{\ell\in\mathcal{N}_k} a_{k\ell} = 1$$

This scheme exploits network connectivity rather fully, leading to robust algorithms. If links or nodes eventually fail, the adaptive network can still react by relying on the remaining topology.

The so-called Laplacian rule is described as follows. In graph theory, the entries of the $N\times N$ Laplacian matrix $\mathcal{L}$ of a graph with N nodes is defined as [27]:

$$\mathcal{L}_{k\ell} = \begin{cases} -1 & \text{if } k\neq l \text{ are linked} \\ n_k - 1 & \text{for } k = l \\ 0 & \text{otherwise}\end{cases}$$

Note that for $k = l$, the entry of the Laplacian matrix is the number of incident links on node k. The Laplacian of a graph has several important properties. For example, it is always a nonnegative-definite matrix and the number of times that 0 occurs as an eigenvalue is equal to the number of connected components in the graph. The weights $A = [a_{k\ell}]$ in the Laplacian rule are chosen as follows (see, e.g., [28], [29]):

$$A = I_N - \gamma\mathcal{L}$$

for some constant γ. A possible choice is $\gamma = n_{\max}$ where $n_{\max}$ denotes the maximum degree across the network. In this case we get

$$a_{k\ell} = \begin{cases} 1/n_{\max} & \text{if } k\neq l \text{ are linked} \\ 1-(n_k-1)/n_{\max} & \text{for } k = l \\ 0 & \text{otherwise}\end{cases}$$

Another choice is the maximum-degree weights rule (e.g., [30]) which uses $\gamma = 1/N$, or equivalently,

$$a_{k\ell} = \begin{cases} 1/N & \text{if } k\neq \ell \text{ are linked} \\ 1-(n_k-1)/N & \text{for } k = \ell \\ 0 & \text{otherwise}\end{cases}$$

In this case, all links are assigned weights $1/N$ and each node complements the sum of the weights to 1.

The so-called Metropolis rule is described in [29] and motivated by earlier works on sampling methods [31], [32]. Let

n_k and n_ℓ denote the degrees of nodes k and ℓ, respectively. Then $a_{k\ell}$ is selected as follows:

$$a_{k\ell} = \begin{cases} 1/\max(n_k, n_\ell) & \text{if } k \neq \ell \text{ are linked} \\ 1 - \sum_{\ell \in \mathcal{N}_k - \{k\}} a_{k\ell} & \text{for } k = l \\ 0 & \text{otherwise} \end{cases}$$

In this case, the weighting assigned to a link is dependent on the degree of the node (i.e., on the number of incident links into that node).

Another choice is the relative-degree rule from [16], which does not yield symmetric weight matrices but generally yields better performance as illustrated in the examples further ahead:

$$a_{k\ell} = \begin{cases} n_\ell / \sum_{m \in \mathcal{N}_k} n_m & \text{if } k \text{ and } \ell \text{ are linked or } k = \ell \\ 0 & \text{otherwise} \end{cases} \tag{48}$$

In this case, every neighbor is weighted according to its degree. Reference [16] also suggests an optimal design procedure for the combination matrix A that is aimed at enhancing the network mean-square-error performance.

In the above rules, the combination weights are largely dictated by the sizes of the neighborhoods (or by the node degrees). When the neighborhoods vary with time, the degrees will also vary. However, for all practical purposes, these combination schemes are not adaptive in the sense that the schemes do not learn which nodes are more or less reliable so that the weights can be adjusted accordingly.

An adaptive combination rule along these lines can be motivated by the analysis results of [17]. The rule allows the network to assign convex combination weights to the local estimates and the aggregate estimate. Moreover, the combination weights can be adjusted adaptively so that the network can respond to node conditions and assign smaller weights to nodes that are subject to higher noise levels. For example, one possibility could be as follows [15]. Consider a set of coefficients $b_{k\ell}$ that add up to one when node k is excluded. These coefficients could be obtained from the coefficients $a_{k\ell}$ as follows:

$$b_{k\ell} = \begin{cases} \frac{a_{k\ell}}{\sum_{\ell \in \mathcal{N}_k - \{k\}} a_{k\ell}} & \text{if } k \neq \ell \text{ are linked} \\ 0 & \text{otherwise} \end{cases}$$

Now, we combine the local estimates at the neighbors of node k, say as before, but excluding node k itself. This step results in an intermediate estimate:

$$\overline{\psi}_k^{(i-1)} = \sum_{\ell \in \mathcal{N}_k - \{k\}} b_{k\ell} \psi_\ell^{(i-1)}$$

Then this aggregate estimate is combined *adaptively* with the local estimate at node k to provide the desired combination (compare with (40) and see Fig. 10):

$$\phi_k^{(i-1)} = \gamma_k(i)\ \psi_k^{(i-1)} + [1 - \gamma_k(i)]\ \overline{\psi}_k^{(i-1)}$$

where the coefficient $\{\gamma_k(i)\}$ is adapted in order to improve performance (such as reducing the mean-square error further whenever possible) [15], [17]. The idea is that the selection of γ_k will give more or less weight to the local weight as opposed to the combination from the neighbors depending on which source of information is more reliable (or less noisy);

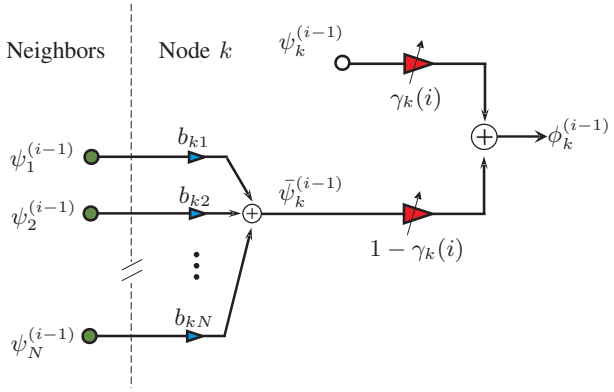

Fig. 10. An example of a network with an *adaptive* diffusion strategy.

we forgo the details of adapting the coefficient γ_k. Once this is done, we may continue to the adaptation step:

$$\psi_k^{(i)} = \phi_k^{(i-1)} + \mu\ u_{k,i}^* \big(d_k(i) - u_{k,i}\phi_k^{(i-1)}\big)$$

Alternatively, one could consider adapting all the coefficients $\{a_{k\ell}\}$ in the diffusion schemes (40)–(41) directly.

D. Simulation Examples

In order to illustrate the adaptive network performance, we present a simulation example. Fig. 6 depicts the network topology with $N = 15$ nodes, together with the network statistical profile. The regressors are zero-mean Gaussian, independent in time and space, with covariance matrices $R_{u,k}$. The background noise power is denoted by $\sigma_{v,k}^2$. Fig. 11 shows the learning behavior of several algorithms in terms of the network EMSE and MSD. These are evaluated as

$$\zeta^{\text{network}}(i) = \frac{1}{N}\sum_{k=1}^{N} \zeta_k(i) \quad \text{(EMSE)}$$

$$\eta^{\text{network}}(i) = \frac{1}{N}\sum_{k=1}^{N} \eta_k(i) \quad \text{(MSD)}$$

by averaging the corresponding curves across all nodes. For the diffusion and no-cooperation cases, a value of $\mu_k = 0.01$ was used, whereas for the incremental LMS algorithm, the value was $\mu_k = 0.01/N$; this is because the incremental algorithm uses N LMS-type iterations for every measurement time. The relative-degree weights (48) were used in the diffusion algorithms. The curves were averaged over 200 experiments, and the steady-state values were calculated by averaging the last 50 samples after convergence.

Note how the incremental and diffusion algorithms (26), (40), and (41) significantly outperform the non-cooperative case (where each node runs an individual filter). Also note that the ATC algorithm (41) outperforms the CTA version (40). Also shown is the incremental diffusion LMS algorithm (26), which outperforms the diffusion solutions. This behavior by the incremental solution is expected since the incremental algorithm uses data from the entire network at every iteration. Fig. 12 shows the steady-state network EMSE and MSD for every node in the network.

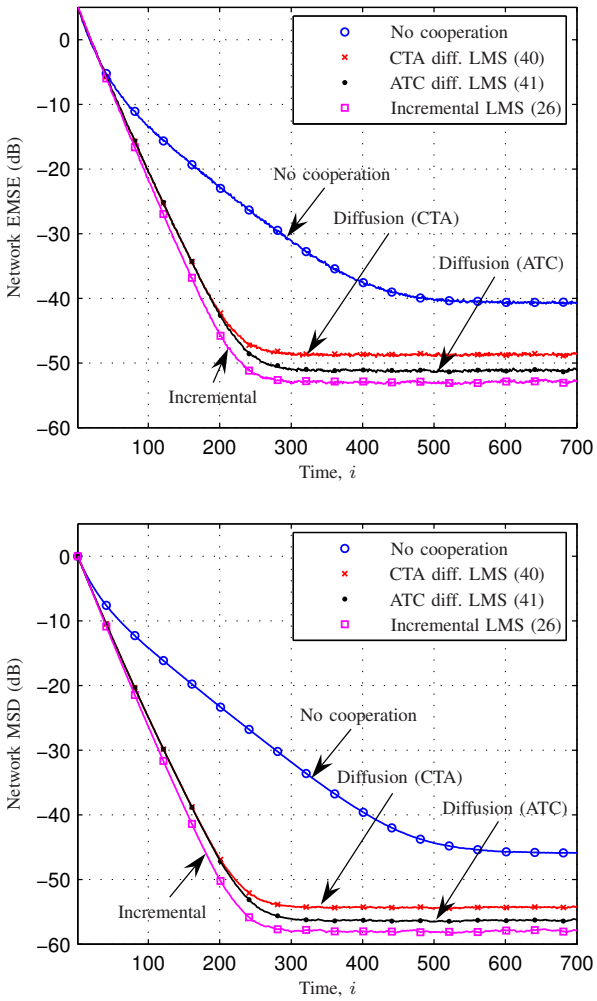

Fig. 11. Transient network EMSE (top) and MSD (bottom) for LMS without cooperation, CTA diffusion LMS, ATC diffusion LMS, individual LMS filters (no cooperation), and incremental LMS.

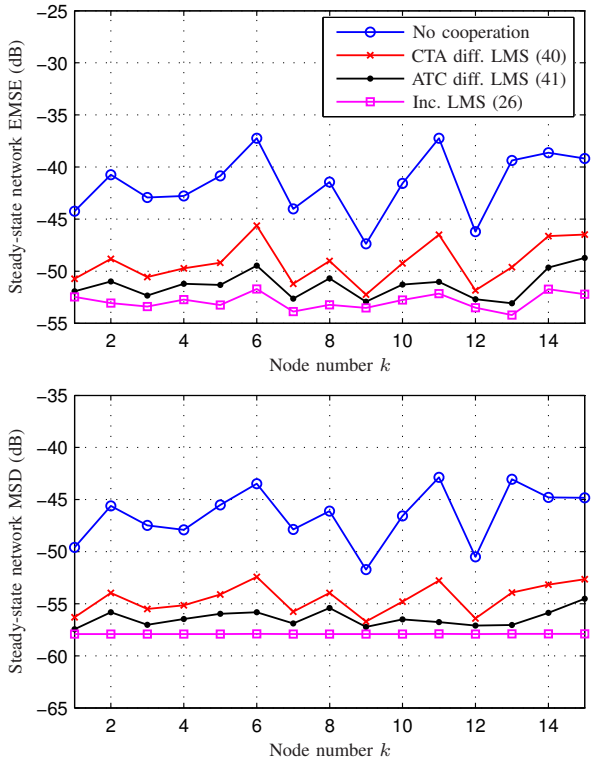

Fig. 12. Steady-state EMSE (top) and MSD (bottom) per node, for LMS without cooperation, CTA diffusion LMS, ATC diffusion LMS, and incremental LMS.

E. Cooperation Enhances Stability

We illustrate in this section a useful property of the diffusion algorithms, namely, that cooperation does not only enhance performance but it also enhances stability relative to the non-cooperative solution with individual filters at the nodes. Let us focus on the CTA version (40) of diffusion LMS.

The coefficients $a_{k\ell}$ give rise to an $N \times N$ combination matrix $A = [a_{k\ell}]$, which carries information about the network topology: a non-zero entry $a_{k\ell}$ means that nodes k and ℓ are connected. Note that A is a stochastic matrix, namely, it satisfies

$$Aq = q$$

where $q \triangleq \mathrm{col}\{1, \ldots, 1\}$. Let $X \otimes Y$ denote the Kronecker product of the matrices X and Y. Note in particular that if X and Y are both $M \times M$, then their Kronecker product is $M^2 \times M^2$. Moreover,

$$I_m \otimes X = \mathrm{diag}\{\underbrace{X, X, \ldots, X}_{m \text{ times}}\}$$

Introduce the global quantities

$$\begin{aligned} \tilde{\boldsymbol{\psi}}^i &\triangleq \mathrm{col}\{\tilde{\boldsymbol{\psi}}_1^{(i)}, \tilde{\boldsymbol{\psi}}_2^{(i)}, \ldots, \tilde{\boldsymbol{\psi}}_N^{(i)}\} \\ \mathcal{M} &\triangleq \mathrm{diag}\{\mu_1 I_M, \mu_2 I_M, \ldots, \mu_N I_M\} \\ \mathcal{A} &\triangleq A \otimes I_M \\ \mathcal{R}_u &\triangleq \mathrm{diag}\{R_{u,1}, R_{u,2}, \ldots, R_{u,N}\} \end{aligned}$$

where $\{\mathcal{M}, \mathcal{A}, \mathcal{R}_u\}$ are $NM \times NM$ matrices. Then, under the data assumptions described earlier in (27), some straightforward algebra will show that the mean of the extended weight-error vector evolves according to the following dynamics:

$$\boxed{E\tilde{\boldsymbol{\psi}}^i = \big(I_{NM} - \mathcal{M}\mathcal{R}_u\big)\mathcal{A}\; E\tilde{\boldsymbol{\psi}}^{i-1}} \tag{49}$$

For simplicity of notation, let

$$\mathcal{B} \triangleq I_{NM} - \mathcal{M}\mathcal{R}_u$$

Then, expression (49) shows that the adaptive network will be stable in the mean if, and only if, the spectral radius of $\mathcal{B}\mathcal{A}$ is strictly less than one, i.e.,

$$\boxed{\big|\lambda(\mathcal{B}\mathcal{A})\big| < 1} \tag{50}$$

In the absence of cooperation (i.e., when the nodes evolve independently of each other and therefore $\mathcal{A} = I_{NM}$), the mean-error vector would instead evolve according to

$$E\tilde{\boldsymbol{\psi}}^i = \big(I_{NM} - \mathcal{M}\mathcal{R}_u\big)\; E\tilde{\boldsymbol{\psi}}^{i-1}$$

with coefficient matrix $\mathcal{B}$ alone. Thus, in the diffusion network case, convergence in the mean also depends on the network topology (as represented by $\mathcal{A}$). Using matrix 2-norms we have

$$\|\mathcal{B}\mathcal{A}\|_2 \;\leq\; \|\mathcal{B}\|_2 \cdot \|\mathcal{A}\|_2 \tag{51}$$

However, for any matrix X it holds that

$$|\lambda_{\max}(X)| \leq ||X||_2 \tag{52}$$

with equality if X is Hermitian. Moreover, due to the block structure of $\mathcal{R}_u$, $\mathcal{B}$ is Hermitian, and recall that $\mathcal{A} = A \otimes I_M$. Hence, we have

$$|\lambda_{\max}(\mathcal{B}\mathcal{A})| \leq \|A\|_2 \cdot |\lambda_{\max}(\mathcal{B})| \tag{53}$$

That is, the network mean stability depends on the local data statistics (represented by $\mathcal{B}$) and on the cooperation strategy (represented by A). Whenever a combiner rule is picked so that $\|A\|_2 \leq 1$, the cooperative scheme will enforce robustness over the non-cooperative scheme. For combiners that render stochastic and symmetric matrices A, we have that $\|A\|_2 = 1$. As a result, we conclude that

$$\boxed{|\lambda_{\max}(\mathcal{B}\mathcal{A})| \leq |\lambda_{\max}(\mathcal{B})|} \tag{54}$$

In other words, the spectral radius of $\mathcal{B}\mathcal{A}$ is generally smaller than the spectral radius of $\mathcal{B}$. Hence, cooperation under the diffusion protocol (40) has a *stabilizing* effect on the network.

Figure 13 presents a simulation example for the network defined in Fig. 6, and a value $\mu_k = 0.05$. Here we show the magnitude of the network modes (i.e., the magnitude of the eigenvalues of $\mathcal{B}\mathcal{A}$) for all nodes in the network, for a total of MN modes. We present the modes when there is no cooperation ($A = I$), and when cooperation is introduced through the diffusion algorithms, for different choices of weighting matrices, namely, maximum-degree weights, metropolis weights, and relative-degree weights. Note how cooperation significantly decreases the eigenmodes of the mean weight error evolution, as compared with the non-cooperative scheme, thus yielding faster convergence. The top-right plot of Fig. 13 zooms on the largest eigenmodes, which generally determine the convergence speed. Again the diffusion algorithms outperform the no cooperation case. The bottom plot shows the MSD learning curves for different choices of weighting matrices.

F. Mean-Square Performance

We may also examine the mean-square performance of the diffusion schemes. A detailed mean-square and stability analysis of the CTA diffusion algorithm (40) is performed in [15]. The following results are simplifications of the general expressions derived in [15]; the simplification assumes a uniform statistical profile across the network, i.e., $R_{u,k} = R_u$, $R_{du,k} = R_{du}$, and $\sigma^2_{v,k} = \sigma^2_v$ for all k, as well as uniform and sufficiently small step-sizes, $\mu_k = \mu$. The results in [15] apply to the more general scenario of varying statistical profile and step-sizes across the nodes.

Introduce the vec$(\cdot)$ notation, which transforms an $M \times M$ matrix X into an $M^2 \times 1$ column vector x by stacking the columns of X on top of each other:

$$x \;=\; \text{vec}(X)$$

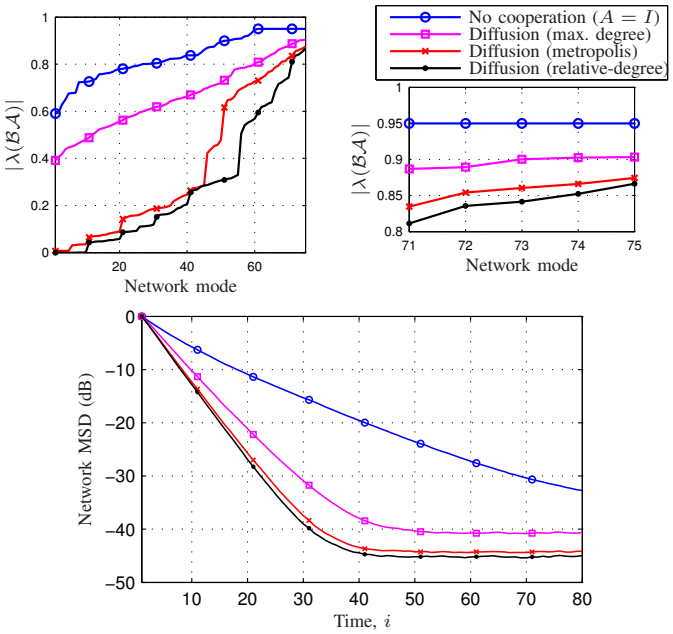

Fig. 13. Comparison of diffusion schemes using different weighting matrices and the case where there is no cooperation. The simulation uses $N = 15$, $M = 5$ (75 modes in total), $\mu_k = 0.05$ and network statistics as in Fig. 6. All diffusion schemes use the ATC diffusion LMS algorithm (41). Plots include the network modes for different choices of weighting matrices (top-left plot), a zoomed-in version showing the largest eigenmodes (top-right plot), and the MSD learning curves (bottom plot).

In the case of small step-size μ, simplified expressions for the MSD and EMSE for the CTA diffusion algorithm (40) can be described as follows. Introduce the quantities:

$$\begin{aligned} D &\triangleq I - A^T \otimes (I - \mu R_u^T) \otimes A^T \otimes (I - \mu R_u) \quad (55)\\ a &\triangleq \mu^2 \sigma_v^2 \cdot \text{vec}(I_N \otimes R_u) \\ b_k &\triangleq \text{vec}(\text{diag}(e_k) \otimes R_u) \\ q_k &\triangleq \text{vec}(\text{diag}(e_k) \otimes I_M) \end{aligned}$$

Then

$$\eta_k \;\approx\; a^T D^{-1} q_k \qquad \text{(MSD)} \tag{56}$$

$$\zeta_k \;\approx\; a^T D^{-1} b_k \qquad \text{(EMSE)} \tag{57}$$

where e_k denotes the $M \times 1$ basis vector with 1 corresponding to the position of the $k-$th node and zeros elsewhere. Observe how the network topology influences the performance through the matrix A, which appears in the expressions for the MSD and the EMSE.

Fig. 14 shows the steady-state performance of the CTA diffusion LMS algorithm (40), both for simulation and the theoretical results of (56) and (57), for a network with $N = 15$ nodes, $\mu_k = 0.02$, and uniform noise variances and regressor covariances across the nodes.

V. Concluding Remarks

This chapter describes several distributed and cooperative algorithms that endow networks with learning abilities. The algorithms address distributed estimation problems that arise

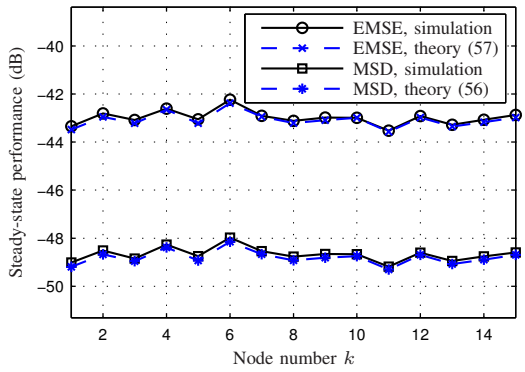

Fig. 14. Steady-state performance of the CTA diffusion LMS algorithm (40); theory and simulation.

in a variety of applications, such as environment monitoring, target localization and sensor network problems.

Although the chapter focused on algorithms of the LMS type, several other extensions are possible and have been pursued including algorithms of the least-squares type as well as Kalman filtering and smoothing procedures. The objective of this chapter has been to introduce the main ideas and to illustrate them by focusing on simpler algorithms for the benefit of clarity.

REFERENCES

[1] D. Estrin, G. Pottie and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. ICASSP*, pp. 2033-2036, Salt Lake City, UT, May 2001.

[2] D. Li, K. D. Wong, Y. H. Hu and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, March 2002.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, issue 8, pp. 102–114, August, 2002.

[4] L. A. Rossi, B. Krishnamachari and C.-C. J. Kuo, "Distributed parameter estimation for monitoring diffusion phenomena using physical models," *Proc. IEEE Conf. Sensor and Ad Hoc Comm. and Networks*, pp. 460-469, Santa Clara, CA, Oct. 2004.

[5] D. Culler, D. Estrin and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, Aug. 2004.

[6] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56-69, July 2006.

[7] C. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," *Proc. ICASSP*, pp. 3285–3288, Las Vegas, April 2008.

[8] D. Bertsekas, "A new class of incremental gradient methods for least squares problems," newblock *SIAM J. Optim.*, vol.7, no. 4, pp. 913-926, Nov. 1997.

[9] A. Nedic and D. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.

[10] J. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. AC-31, no. 9, pp. 650655, Sep. 1986.

[11] B. T. Poljak and Y. Z. Tsypkin, "Pseudogradient adaptation and training algorithms," *Automatic and Remote Control*, vol. 12, pp. 8394, 1978.

[12] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol.23, no.4, pp. 798–808, April 2005.

[13] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Trans. Fundamentals of Electronics,Communications and Computer Sciences*, vol. E90-A, no. 8, pp. 1504–1510, 2007.

[14] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064-4077, August 2007.

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

[16] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Processing*, vol. 56, no. 5, pp. 1865-1877, May 2008.

[17] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Trans. Signal Processing*, vol. 54, no. 3, pp. 1078–1090, March 2006.

[18] C. G. Lopes and A. H. Sayed, "Distributed adaptive incremental strategies: Formulation and performance analysis," *Proc. ICASSP*, vol. 3, pp. 584–587, Toulouse, France, May 2006.

[19] C. G. Lopes and A. H. Sayed, "Diffusion least-mean-squares over adaptive networks," *Proc. ICASSP*, vol. III, pp. 917–920, Honolulu, Hawaii, April 2007.

[20] A. H. Sayed and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," *Proc. Asilomar Conference on Signals, Systems and Computers*, pp. 233–237, Pacific Grove, CA, October 2006.

[21] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "A diffusion RLS scheme for distributed estimation over adaptive networks," *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, Helsinki, Finland, June 2007.

[22] F. S. Cattivelli and A. H. Sayed, "Diffusion mechanisms for fixed-point distributed Kalman smoothing," *Proc. EUSIPCO*, Lausanne, Switzerland, August 2008.

[23] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering: Formulation and performance analysis," *Proc. IAPR Workshop on Cognitive Information Processing*, Santorini, Greece, June 2008.

[24] A. H. Sayed, *Fundamentals of Adaptive Filtering,* Wiley, NJ, 2003.

[25] A. H. Sayed, *Adaptive Filters,* Wiley, NJ, 2008.

[26] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. Joint 44th IEEE Conf. on Decision and Control and European Control Conf. (CDC-ECC)*, pp. 2996-3000, Seville, Spain, Dec. 2005.

[27] W. Kocay and D. L. Kreher, *Graphs, Algorithms and Optimization,* Chapman & Hall/CRC Press, Boca Raton, 2005.

[28] D. S. Scherber and H. C. Papadopoulos, "Locally constructed algorithms for distributed computations in ad-hoc networks," *Proc. Information Processing in Sensor Networks (IPSN)*, pp. 11-19, Berkeley, CA, April 2004.

[29] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no.1, pp. 65-78, September 2004.

[30] L. Xiao, S. Boyd and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," *Proc. Information Processing in Sensor Networks*, pp. 63–70, Los Angeles, CA, April 2005.

[31] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087-1092, 1953.

[32] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97-109, 1970.