Diffusion LMS with Communication Constraints

Øyvind Lunde Rørtveit
Dept. of Electrical Engineering and Computer Science
University of Stavanger, Norway
Email: oyvind.l.rortveit@uis.no

John Håkon Husøy
Dept. of Electrical Engineering and Computer Science
University of Stavanger, Norway
Email: john.h.husoy@uis.no

Ali H. Sayed
Electrical Engineering Department
University of California, Los Angeles
Email: sayed@ee.ucla.edu

***Abstract*—Diffusion LMS is a distributed algorithm that allows a network of nodes to solve estimation problems in a fully distributed manner by relying solely on local interactions. The algorithm consists of two steps: a consultation step whereby each node combines in a convex manner information collected from its neighbors and an adaptation step where the node updates its local estimate based on local data and on the data exchanged with the neighbors. Various forms of diffusion algorithms are possible such as combine-then-adapt (CTA) and adapt-then-combine (ATC) forms, in addition to probabilistic implementations where consultations are performed only with a subset of the neighbors chosen at random. In this paper we propose an alternative protocol to reduce the communications cost during the consultation process. Each node is limited to selecting only one of its neighbors for consultation, and we propose a dynamic technique that enables the node to pick from among its neighbors that neighbor that is likely to lead to the best mean-square deviation (MSD) performance. In other words, rather than picking nodes at random, the proposed algorithm is meant to enable nodes to perform the selection in a more informed manner. The paper describes the proposed method and illustrates its behavior via simulations.**

I. Background

In a typical distributed estimation setup, a network of N spatially distinct nodes observes time data, and from these data wishes to estimate some vector-valued variable w^o. If the regression data at node k at time i is arranged into a row-vector $u_{k,i}$, and if the observed measurement is denoted by $d_k(i)$, then the estimation problem can be stated as that of solving:

$$\min_w E \sum_{k=1}^{N} \|\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} w\|^2 \qquad (1)$$

where E denotes the expectation operation. Note that we are using boldface letters to denote random quantities.

The existing distributed adaptive solutions can be roughly classified into incremental [1], [2], [3], [4], diffusion [5], [6], [7], [8], [9] and hiearchical [10], [11] algorithms. Our focus here is on the diffusion LMS algorithm of [5], [6], in which, at each iteration, each node performs an LMS update followed by a diffusion step. During diffusion, the current weight estimate is updated through a linear combination of the weight estimates of the node's neighbors. The algorithm exhibits excellent steady-state behavior, yet the discussion in [1], and particularly the results of [12], suggest that similar performance can be achieved with less communication. Our approach to reducing the amount of resources spent on communication is to introduce a constraint into the diffusion step, namely that each node should receive weight estimates from only one of its neighbors during the diffusion step. Moreover, the neighboring node is selected dynamically according to a procedure we develop further ahead. Some related literature exists in the context of distributed averaging, in the form of *gossiping* algorithms [13], [14]. These gossip algorithms usually select which nodes to communicate with in a random fashion and independent of the data. In contrast, we shall select the neighboring node on the fly by examining the real-time data and picking the node that is more likely to lead to lower mean-square deviation (MSD) performance. We pose an optimization problem, which requires additional communications. This means that the total amount of communication is generally not reduced as much as would have been the case with random selection. However, a significant reduction of communication is still achievable compared to standard diffusion where all nodes in a neighborhood are consulted. Simulations suggest that the proposed algorithm achieves steady-state performance that is comparable to the unconstrained algorithm, at the cost of a negligible penalty on convergence speed.

This work was performed while O. L. Rortveit was a visitor at the UCLA Adaptive Systems Laboratory. The work of A. H. Sayed was supported in part by NSF grants CCF-0942936, ECS-0601266, and CCF-1011918.

II. Problem formulation

The diffusion LMS algorithm is fully decentralized; therefore a description of the processing at a single node is sufficient to describe the algorithm. The following equations summarize the algorithm in its adapt-then-combine (ATC) version [6], from the perspective of node k:

$$\psi_{k,i} = w_{k,i-1} + \mu_k u_{k,i}^*(d_k(i) - u_{k,i} w_{k,i-1}) \qquad (2)$$

$$w_{k,i} = \sum_{l \in \mathcal{N}_k} a_{l,k}(i) \psi_{l,i}. \qquad (3)$$

In addition to the previously mentioned observations $u_{k,i}$ and $d_k(i)$, expressions (2)-(3) include the step sizes μ_k, the combination coefficients $a_{l,k}(i)$, the weight vector $w_{k,i}$ which is the current estimate of w^o at node k, the vector $\psi_{k,i}$ which represents an intermediate estimate in computing $w_{k,i}$, and $\mathcal{N}_k$ which is the set of neighbors of node k (that is, the set of nodes, including k, with which node k can exchange data

directly). It is assumed that the combination coefficients satisfy the constraint $\sum_{l\in\mathcal{N}_k} a_{l,k}(i) = 1$.

The choice of the combination coefficients $a_{l,k}(i)$ is of some importance for the performance of the algorithm. There are offline schemes for choosing the weights, such as the uniform [15], Laplacian [16], maximum degree [17], metropolis [18], relative degree [9], and relative degree-variance [6] rules. Different online schemes, where the coefficients are adapted at runtime based on the incoming data, have been proposed in [5] and [19].

Here, we study the problem of finding a set of coefficients $\{a_{l,k}(i)\}$ with the additional constraint that, for all i, all $a_{l,k}(i)$ equal zero except for two coefficients; $a_{k,k}(i)$ and $a_{s_k(i),k}(i)$, for some node index $s_k(i)$ to be selected from the neighborhood of node k.. This requirement leads to the following specialized form of the combination step (3):

$$w_{k,i} = \lambda_k(i)\psi_{k,i} + (1-\lambda_k(i))\psi_{s_k(i),i}, \tag{4}$$

where $\lambda_k(i) \triangleq (1 - a_{s_k(i),k}(i))$.

Our problem is that of choosing, for each node k and time i, the parameters $s_k(i) \in \mathcal{N}_k\backslash k$ and $\lambda_k(i) \in \mathbb{R}$ such that the resulting algorithm has desirable mean square performance.

III. Minimizing the network MSD

One useful performance measure for distributed adaptive problems is the network mean-square deviation, or network MSD, defined as:

$$\mathrm{MSD}_{\mathrm{nw}}(i) \triangleq \frac{1}{N}\sum_{k=1}^{N} E\|\tilde{\boldsymbol{w}}_{k,i}\|^2 \tag{5}$$

where $\tilde{\boldsymbol{w}}_{k,i} \triangleq \boldsymbol{w}_{k,i} - w^o$. Thus, consider the problem of minimizing the MSD at time i. At time i, $\lambda_k(j)$ and $s_k(j)$ have already been computed for all k and all $j < i$, that is, we are given a realization of the set

$$\boldsymbol{C}(i) \triangleq \{\boldsymbol{\lambda}_k(j), \boldsymbol{s}_k(j) : 0 < j < i, 0 < k \leq N\}, \tag{6}$$

and we wish to find the $\lambda_k(i)$ and $s_k(i)$ for all k that minimize the network MSD given (6). From the optimization viewpoint, $\lambda_k(i)$ and $s_k(i)$ are considered deterministic variables. Therefore, we can write the global cost function, using (5) and (4), as

$$\begin{aligned}
&\sum_{k=1}^{N} E\left[\|\tilde{\boldsymbol{w}}_{k,i}\|^2 \middle| C(i)\right] \\
&= \sum_{k=1}^{N} [\lambda_k^2(i)\nu_{k,k}(i) + (1-\lambda_k(i))^2 \nu_{s_k(i),s_k(i)}(i)] \\
&+ \sum_{k=1}^{N} 2\lambda_k(i)(1-\lambda_k(i))\nu_{k,s_k(i)}(i)
\end{aligned} \tag{7}$$

where

$$\nu_{k,l}(i) \triangleq E\left[\mathrm{Re}(\tilde{\boldsymbol{\psi}}_{k,i}^* \tilde{\boldsymbol{\psi}}_{l,i}) \middle| C(i)\right], \tag{8}$$

and $\tilde{\boldsymbol{\psi}}_{k,i} \triangleq \boldsymbol{\psi}_{k,i} - w^o$. Since no term containing $\lambda_k(i)$ and $s_k(i)$ also conains $\lambda_l(i)$ and/or $s_l(i)$ for any $l \neq k$, expression (7) can be minimized separately for each k, which means that the optimization with respect to $\lambda_k(i)$ and $s_k(i)$ can be performed locally at node k. Thus, introduce the local cost function:

$$J_k(l,\lambda) = \lambda^2\nu_{k,k} + (1-\lambda)^2\nu_{l,l} + 2\lambda(1-\lambda)\nu_{k,l}. \tag{9}$$

In the following, as in (9), we will drop the time index i to avoid cluttered notation wherever this leads to no confusion.

The optimization problem at node k becomes

$$(s_k^o, \lambda_k^o) = \underset{(l\in\mathcal{N}_k\backslash k, \lambda\in\mathcal{R})}{\arg\min} J_k(l,\lambda). \tag{10}$$

The problem can be solved stepwise as:

$$\lambda_{k,l}^o \triangleq \underset{\lambda}{\arg\min}\, J_k(l,\lambda) \tag{11}$$

$$s_k^o = \underset{l\in\mathcal{N}_k\backslash k}{\arg\min} J_{k,l} \tag{12}$$

$$\lambda_k^o = \lambda_{k,s_k^o}^o, \tag{13}$$

where $J_{k,l} \triangleq J_k(l, \lambda_{k,l}^o)$.

The minimization (11) is solved by differentiating $J_k(l,\lambda)$ with respect to λ, setting the result to zero, and solving for λ. This yields

$$\lambda_{k,l}^o = \frac{\nu_{l,l} - \nu_{k,l}}{\nu_{k,k} + \nu_{l,l} - 2\nu_{k,l}}. \tag{14}$$

Reinserting this result into (9) yields

$$J_{k,l} = \frac{\nu_{k,k}\nu_{l,l} - \nu_{k,l}^2}{\nu_{k,k} + \nu_{l,l} - 2\nu_{k,l}}. \tag{15}$$

We can now find s_k^o by evaluating $J_{k,l}$ for all $l \in \mathcal{N}_k\backslash k$, and simply choosing the minimizer. This means (14) does not need to be evaluated for other values than $l = s_k^o$, which gives λ_k^o directly.

IV. Proposed algorithm

Of course, the $\nu_{k,l}$ are not available for a given realization of the algorithm, but need to be estimated. Given such estimates $\hat{\nu}_{k,l}$, the proposed algorithm is fully defined by (2), (4), and the following equations, derived directly from the results of the preceding section:

$$\hat{J}_{k,l} \triangleq \frac{\hat{\nu}_{k,k}\hat{\nu}_{l,l} - \hat{\nu}_{k,l}^2}{\hat{\nu}_{k,k} + \hat{\nu}_{l,l} - 2\hat{\nu}_{k,l}}, \quad l \in \mathcal{N}_k\backslash k \tag{16}$$

$$s_k = \underset{l\in\mathcal{N}_k\backslash k}{\arg\min}\, \hat{J}_{k,l}, \tag{17}$$

$$\lambda_k = \frac{\hat{\nu}_{s_k,s_k} - \hat{\nu}_{k,s_k}}{\hat{\nu}_{k,k} + \hat{\nu}_{s_k,s_k} - 2\hat{\nu}_{k,s_k}}. \tag{18}$$

Estimating $\nu_{k,l}$ is difficult because it would require knowledge of the optimal weight vector w^o. In [19], this problem was avoided by assuming that $\boldsymbol{\psi}_{k,i}$ is unbiased for all k, i, that is, that $E\boldsymbol{\psi}_{k,i} = w^o$. Although the assumption is true only asymptotically, it produces good results even before the algorithm reaches steady state.

A. Estimating $\nu_{k,k}$

In order to compute (16) through (18), each node k requires knowledge of $\nu_{l,l}$ for all $l \in \mathcal{N}_k$. We choose to achieve this task by letting each node k compute an estimate of $\nu_{k,k}$ and broadcast this estimate to all of its neighbors.

Using the unbiasedness assumption, we can estimate $\nu_{k,k}$ using an exponential moving average scheme as follows:

$$\hat{\nu}_{k,k}(i) = \alpha_1 \hat{\nu}_{k,k}(i-1) + (1-\alpha_1)\|\psi_{k,i} - \bar{\psi}_{k,i-1}\|^2, \quad (19)$$

where $\bar{\psi}_{k,i}$ is an estimate of $E\boldsymbol{\psi}_{k,i}$, which can be computed as

$$\bar{\psi}_{k,i} = \alpha_2 \bar{\psi}_{k,i-1} + (1-\alpha_2)\psi_{k,i}. \quad (20)$$

Here $0 < \alpha_1, \alpha_2 < 1$ are design parameters. A short discussion on how to choose these parameters is given in Section IV-C.

B. Estimating $\nu_{k,l}$ for $k \neq l$

Each node k also requires knowledge of $\nu_{k,l}$ for all $l \in \mathcal{N}_k \backslash k$ to compute (16) through (18). The simplest approach would be to assume that these cross terms are zero, but this does not yield good results. Notice that, under the unbiasedness assumption, $\nu_{k,l}$ is a sum of cross covariances between the elements of the vectors $\boldsymbol{\psi}_k$ and $\boldsymbol{\psi}_l$. Hence, it is reasonable to assume that $\nu_{k,l}$ decreases with time when nodes k and l do not communicate, and increases whenever they do. Therefore, the term in (9) containing $\nu_{k,l}$ is responsible for increasing the probability that node k will communicate with a neighbor that it has not communicated with in a long time. This in turn increases the flow of information through the network, and therefore leads to better cooperation and lower overall MSD. We will now show how the values $\nu_{k,l}$ can be estimated using only information present at node k, and therefore without requiring additional communication.

If we assume the signal model

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i), \quad (21)$$

where $\boldsymbol{v}_k(i)$ is additive noise, and use the common temporal and spatial independence assumptions

$$E[\boldsymbol{u}_{k,i}^* \boldsymbol{u}_{l,j}] = R_k \delta_{k-l} \delta_{i-j} \quad (22)$$

$$E[\boldsymbol{u}_{k,i}^* \boldsymbol{v}_l(j)] = 0, \quad \text{for all } i,j,k,l, \quad (23)$$

we can use (2) to evaluate $\nu_{k,l}(i)$ for $k \neq l$ as follows:

$$\nu_{k,l}(i) = E[\mathrm{Re}(\tilde{\boldsymbol{w}}_{k,i-1}^* Q \tilde{\boldsymbol{w}}_{l,i-1}) \big| C(i)], \quad (24)$$

where

$$Q = I - \mu_k R_k - \mu_l R_l + \mu_k \mu_l R_k R_l, \quad (25)$$

and where we used the fact that, by the above independence assumptions, $\boldsymbol{u}_{k,i}$ is independent from $\boldsymbol{C}(i)$. If we assume $\boldsymbol{u}_{k,i}$ is white for all k, that is, $R_k = \sigma_{u,k}^2 I$, expression (24) simplifies to

$$\nu_{k,l}(i) = (1 - \mu_k \sigma_{u,k}^2 - \mu_l \sigma_{u,l}^2 + \mu_k \mu_l \sigma_{u,k}^2 \sigma_{u,l}^2)\omega_{k,l}(i-1), \quad (26)$$

where

$$\omega_{k,l}(i-1) \triangleq E[\mathrm{Re}(\tilde{\boldsymbol{w}}_{k,i-1}^* \tilde{\boldsymbol{w}}_{l,i-1}) \big| C(i)] \quad (27)$$

Although we cannot expect whiteness in practice, it seems reasonable to approximate the development of $\nu_{k,l}$ as

$$\nu_{k,l}(i) \approx \beta \omega_{k,l}(i-1), \quad (28)$$

for some β smaller than but close to 1.

Thus far we have reduced the problem of estimating $\nu_{k,l}$ to that of estimating $\omega_{k,l}$. Inserting (4) into (27), we find that the latter is given by

$$\begin{aligned} \omega_{k,l} &= \lambda_k \lambda_l \nu'_{k,l} + \lambda_k(1-\lambda_l)\nu'_{k,s_l} \\ &+ (1-\lambda_k)\lambda_l \nu'_{s_k,l} + (1-\lambda_l)(1-\lambda_k)\nu'_{s_k,s_l}, \end{aligned} \quad (29)$$

where

$$\nu'_{k,l}(i) \triangleq E[\mathrm{Re}(\tilde{\boldsymbol{\psi}}_{k,i}^* \tilde{\boldsymbol{\psi}}_{k,i}) \big| C(i+1)]. \quad (30)$$

We may use the approximation $\nu'_{k,l} \approx \nu_{k,l}$, but even so (29) contains several quantities for which neither exact values nor estimates are available at node k. However, we can approximate (29) from available information only, by using the reasonable assumption that *communication between two nodes m and n does not change $\omega_{k,l}$ unless $\{m,n\}=\{k,l\}$*. Under this assumption, we can ignore the communication between k and s_k if $s_k \neq l$ and between l and s_l if $s_l \neq k$. Looking back at (4), we see that this is equivalent to assuming $\lambda_k = 1$ if $s_k \neq l$ and $\lambda_l = 1$ if $s_l \neq k$. Therefore, we have the following approximation for $\omega_{k,l}$:

$$\omega_{k,l} \approx \begin{cases} \nu_{k,l}, & \text{if } s_k \neq l \text{ and } s_l \neq k \\ \lambda_k \nu_{k,l} + (1-\lambda_k)\nu_{l,l}, & \text{if } s_k = l \text{ and } s_l \neq k \\ \lambda_l \nu_{k,l} + (1-\lambda_l)\nu_{k,k}, & \text{if } s_k \neq l \text{ and } s_l = k \\ 2\lambda_k(1-\lambda_k)\nu_{k,l} + \lambda_k^2 \nu_{k,k} & \\ \quad +(1-\lambda_k)^2 \nu_{l,l}, & \text{if } s_k = l \text{ and } s_l = k \end{cases} \quad (31)$$

Here we have assumed that, if $s_l = k$ and $s_k = l$, then $\lambda_l = (1-\lambda_k)$, which is true for the optimum λ_k^o and λ_l^o.

Comparing (31) to (9), it is clear that the last of the four cases is equal to $J_{k,l}$. In fact, when λ_k equals the optimum λ_k^o, the two middle cases are also equal to $J_{k,l}$, a result that is found by inserting (14) into the respective equations.

Before using the preceding results to compute our estimates, we will make one more simplification, namely, we disregard the third case of (31), and instead use the first case if $s_l = k$ but $s_k \neq l$ (this modification, in fact, improves performance, though space restrictions prevent us from discussing why). Thus, the estimate $\hat{\omega}_{k,l}$ of $\omega_{k,l}$ is computed as

$$\hat{\omega}_{k,s_k(i)}(i) = \hat{J}_{k,l}(i), \quad (32)$$

$$\hat{\omega}_{k,l}(i) = \hat{\nu}_{k,l}(i) \quad \text{for } l \neq s_k(i), \quad (33)$$

where we are now writing $J_{k,l}(i)$ to highlight the dependence of $J_{k,l}$ on time. Based on (28), $\hat{\nu}_{k,l}(i)$ is computed as

$$\hat{\nu}_{k,l}(i) = \beta \hat{\omega}_{k,l}(i-1). \quad (34)$$

Equations (33), (32) and (34) together define the estimation of $\nu_{k,l}(i)$ for $k \neq l$.

C. Choice of parameters

The cost function $J_k(l,\lambda)$ represents an energy quantity, and should therefore be nonnegative. For the true expectations $\nu_{k,k}$, $\nu_{l,l}$ and $\nu_{k,l}$, this is ensured by the Cauchy-Schwarz inequality: $\nu_{k,k}\nu_{l,l} \geq \nu_{k,l}^2$. The same inequality will in fact provably hold for our estimates as long as we choose $\beta \leq \alpha_1$, although the proof is not included here due to space restrictions. The inequality is sufficient for $\hat{J}_k(l,\lambda)$ to be convex with respect to λ, and sufficient and necessary for it to be nonnegative.

Through simulations we have seen that the performance of the algorithm improves as α_1 decreases and as β increases, thus it is natural to choose $\alpha_1 = \beta$. Our simulations show good results in the range $0.85 \leq \beta = \alpha_1 \leq 0.95$. The selection of α_2 is a tradeoff between high accuracy and fast convergence of the mean estimate. We have seen best results with $\alpha_2 \geq 0.995$.

V. Algorithm summary

The following pseudocode describes the processing running at node k.

Initialize $w_{k,0} = 0$, $\hat{\nu}_{k,k}(0) = 0$, $\bar{\psi}_{k,0} = 0$ and, for all $l \in \mathcal{N}_k \backslash k$, $\hat{\omega}_{k,l}(0) = 0$.
for $i = 1, 2, \ldots$ **do**
$e_k(i) = d_k(i) - u_{k,i} w_{k,i-1}$
$\psi_{k,i} = w_{k,i-1} + \mu_k u_{k,i}^* e_k(i)$
$\hat{\nu}_{k,k}(i) = \alpha_1 \hat{\nu}_{k,k}(i-1) + (1-\alpha_1)\|\psi_{k,i} - \bar{\psi}_{k,i-1}\|^2$
Send $\hat{\nu}_{k,k}(i)$ to and receive $\hat{\nu}_{l,l}(i)$
from all neighbors $l \in \mathcal{N}_k \backslash k$
$\bar{\psi}_{k,i} = \alpha_2 \bar{\psi}_{k,i-1} + (1-\alpha_2)\psi_{k,i}$
$\hat{\nu}_{k,l}(i) = \beta \hat{\omega}_{k,l}(i-1)$, $l \in \mathcal{N}_k \backslash k$
$\hat{J}_{k,l}(i) = \frac{\hat{\nu}_{k,k}(i)\hat{\nu}_{l,l}(i) - \hat{\nu}_{k,l}^2(i)}{\hat{\nu}_{k,k}(i) + \hat{\nu}_{l,l}(i) - 2\hat{\nu}_{k,l}(i) + \epsilon}$, $l \in \mathcal{N}_k \backslash k$
$s_k(i) = \arg\min_{l \in \mathcal{N}_k \backslash k} \hat{J}_{k,l}(i)$
Receive $\psi_{s_k(i),i}$, and send $\psi_{k,i}$ if requested
$\lambda_{k,i} = \frac{\hat{\nu}_{s_k,s_k}(i) - \hat{\nu}_{k,s_k}(i)}{\hat{\nu}_{k,k}(i) + \hat{\nu}_{s_k,s_k}(i) - 2\hat{\nu}_{k,s_k}(i) + \epsilon}$
$w_{k,i} = \lambda_{k,i}\psi_{k,i} + (1-\lambda_{k,i})\psi_{s_k,i}$
$\hat{\omega}_{k,s_k(i)}(i) = \hat{J}_{k,l}(i)$
$\hat{\omega}_{k,l}(i) = \hat{\nu}_{k,l}(i)$, for $l \in \mathcal{N}_k \backslash \{k, s_k(i)\}$
end for

Note that we have added a small positive constant ϵ to the denominators where there is danger of division by zero.

VI. Further reduction of communication

While the number of values communicated between neighbors is greatly reduced in the proposed algorithm compared to the traditional diffusion LMS, the number of communication sessions is actually increased because we require communication of the estimates $\hat{\nu}_{k,k}$ in addition to ψ_k. One strategy that reduces this number is to have the nodes communicate $\hat{\nu}_{k,k}$ more rarely, e.g., every L samples, and let each node k assume that $\nu_{l,l}(i)$, $l \in \mathcal{N}_k \backslash k$ develops as $\nu_{k,k}(i)$ otherwise, that is, at node k, replace $\hat{\nu}_{l,l}(i)$ by

$$\hat{\nu}_{l,l}^{(k)}(i) = \begin{cases} \hat{\nu}_{l,l}(i) & \text{if } L \text{ divides } i \\ \frac{\hat{\nu}_{k,k}(i)}{\hat{\nu}_{k,k}(i-1)} \hat{\nu}_{l,l}^{(k)}(i-1) & \text{otherwise.} \end{cases} \quad (35)$$

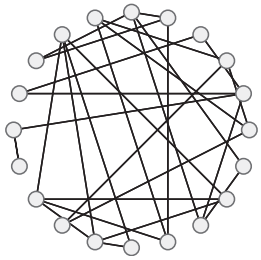

Fig. 1. Network topology in simulations

However, with this variation we can no longer guarantee that the property $\hat{\nu}_{k,l}^2 \leq \hat{\nu}_{k,k}\hat{\nu}_{l,l}$ holds. Therefore, whenever L divides i, we need to check whether $\hat{\nu}_{k,l}^2 \leq \hat{\nu}_{k,k}\hat{\nu}_{l,l}$, and let $\hat{\nu}_{k,l} \leftarrow \sqrt{\hat{\nu}_{k,k}\hat{\nu}_{l,l}}$ otherwise.

In the special case where $L = \infty$, λ_k stays constant at 0.5, while s_k visits all neighbors in a round robin fashion.

VII. Simulation results

Simulations were performed to illustrate the performance of the proposed algorithm. In the simulated example we used $N = 20$ sensors, each running an adaptive filter with $M = 30$ taps. The input signals $\boldsymbol{u}_{k,i}$ were generated as sample vectors $\boldsymbol{u}_{k,i} = [\boldsymbol{u}_k(i) \quad \boldsymbol{u}_k(i-1) \quad \ldots \quad \boldsymbol{u}_k(i-M+1)]$ of an AR-1 process of the form $\boldsymbol{u}_k(i) = \boldsymbol{x}_k(i) + \rho_k \boldsymbol{u}_k(i-1)$, where ρ_k is a correlation coefficient and $\boldsymbol{x}_k(i)$ is a white noise process with variance scaled such that the variance of $\boldsymbol{u}_k(i)$ becomes $\sigma_{u,k}^2$. The desired signal was modeled as $\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i)$, where $\boldsymbol{v}_k(i)$ is white noise with variance $\sigma_{v,k}^2$. We used $\alpha_1 = \beta = 0.9$ and $\alpha_2 = 0.999$. The plots were produced by averaging the results over 50 runs.

Figure 1 shows the topology of the network used in simulations. Figure 3 shows the parameters $\sigma_{u,k}^2$, $\sigma_{v,k}^2$ and ρ_k, as well as the SNR for each node. Figure 2 shows the network EMSE of the proposed algorithm compared to other algorithms, namely the full diffusion LMS with adaptive combiners (AC-DLMS, [19]) and the probabilistic diffusion LMS (P-DLMS, [5]). The first is unconstrained and thus it communicates significantly more than the proposed algorithm. We used a step size of 0.0005 for the adaption of the combiners. For the latter, we chose the probabilities of communication such that the expected number of weight vector exchanges per iteration is one per node, the same as for the proposed algorithm. We used simple averaging for the combiners. It is seen that the proposed algorithm outperforms the P-DLMS. It also compares well with the AC-DLMS at steady state, while using significantly less communication.

Figure 4 shows EMSE of the proposed algorithm at steady state with various values of L, as discussed in section VI. Even as $L \to \infty$, the proposed algorithm still outperforms the P-DLMS.

VIII. Conclusions

A diffusion LMS algorithm with constraints on communication is presented. Through simulations, we studied the mean

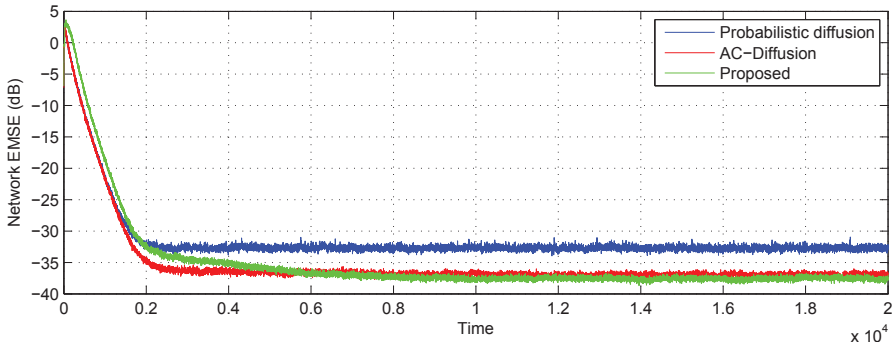

Fig. 2. Network EMSE of the proposed algorithm compared to full diffusion with adaptive combiners [19] and probabilistic diffusion LMS [5].

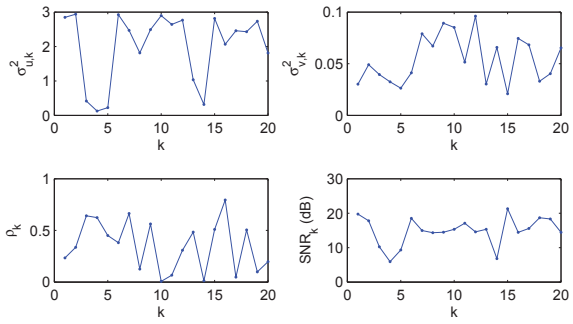

Fig. 3. Network settings; input signal variance $\sigma^2_{u_k}$, noise variance $\sigma^2_{v,k}$, correlation coefficients ρ_k and node-specific SNR.

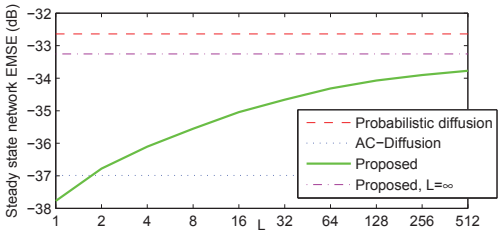

Fig. 4. Network EMSE of the proposed algorithm with various values of L, compared to probabilistic diffusion LMS.

square performance of the algorithm, and showed that EMSE remains unaffected by the reduced communication.

REFERENCES

[1] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4064 –4077, Aug 2007.

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

[3] L. Li, J. A. Chambers, C. G. Lopes, and A. H. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 151 –164, Jan 2010.

[4] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Stochastic incremental gradient descent for estimation in sensor networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*, Nov 2007, pp. 582 – 586.

[5] 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.

[6] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035 –1048, March 2010.

[7] S. S. Stankovic, M. S. Stankovic, and D. M. Stipanovic, "Decentralized parameter estimation by consensus based stochastic approximation," in *Proc. IEEE Conference on Decision and Control*, New Orleans, Dec 2007, pp. 1535 –1540.

[8] L. Li and J. A. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *Proc. IEEE/SP Workshop on Statistical Signal Processing*, Cardiff, Wales, UK, 31 2009-sept. 3 2009, pp. 757 –760.

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

[10] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[11] F. S. Cattivelli and A. H. Sayed, "Multi-level diffusion adaptive networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, April 2009, pp. 2789 –2792.

[12] C. G. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, USA, April 2008, pp. 3285 –3288.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508 – 2530, June 2006.

[14] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748 –2761, July 2009.

[15] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005, pp. 2996–3000.

[16] D. S. Scherber and H. C. Papadopoulos, "Locally constructed algorithms for distributed computations in ad-hoc networks," in *Proc. International Symposium on Information Processing in Sensor Networks*, Berkeley, USA, April 2004, pp. 11–19.

[17] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.

[18] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Information Processing in Sensor Networks*, Los Angeles, USA, Apr. 2005, pp. 63–70.

[19] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4795 –4810, Sep. 2010.