MULTI-LEVEL DIFFUSION ADAPTIVE NETWORKS

Federico S. Cattivelli *Ali H. Sayed*

Department of Electrical Engineering
University of California, Los Angeles

ABSTRACT

We study the problem of distributed estimation, where a set of nodes are required to collectively estimate some parameter of interest from their measurements. Diffusion algorithms have been shown to achieve good performance, increased robustness and are amenable for real-time implementations. In this work we focus on multi-level diffusion algorithms, where a network running a diffusion algorithm is enhanced by adding special nodes that can perform different processing. These special nodes form a second network where a second diffusion algorithm is implemented. We illustrate the concept using diffusion LMS, provide performance analysis for multi-level collaboration and present simulation results showing improved performance over conventional diffusion.

Index Terms— Distributed estimation, cooperation, diffusion, adaptive network

1. INTRODUCTION

We study the problem of distributed estimation, where a set of nodes are required to collectively estimate some parameter of interest from their measurements. Consider a set of N nodes distributed over some region. At every time instant i, every node k takes a scalar measurement $d_k(i)$ of some random process $\boldsymbol{d}_k(i)$ and a $1 \times M$ regression vector, $u_{k,i}$, corresponding to a realization of a random process $\boldsymbol{u}_{k,i}$ which is correlated with $\boldsymbol{d}_k(i)$. The objective is for every node in the network to use the data $\{d_k(i), u_{k,i}\}$ to estimate some deterministic unknown vector w^o. Specifically, we seek the optimal linear estimator w^o that minimizes the following global cost function:

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

where E denotes the expectation operator.

In a centralized solution to the problem, every node transmits its data $\{d_k(i), u_{k,i}\}$ to a central fusion center for processing, requiring large amounts of energy for communication. We refer to the solution obtained by a fusion center as the *global* solution. An adaptive global LMS solution to (1) can be obtained as follows [1, 2]:

$$w_i = w_{i-1} + \mu \sum_{k=1}^{N} u_{k,i}^*(d_k(i) - u_{k,i}w_{i-1}) \qquad (2)$$

Distributed implementations avoid the use of a fusion center and distribute the processing and communication across the entire network. Two types of distributed estimation methods are known as *incremental* and *diffusion*. Diffusion algorithms are amenable for real-time implementations, more robust to node and link failure, and obtain good performance in terms of estimation accuracy. Several distributed estimation algorithms have been proposed, including incremental LMS and RLS [1], diffusion LMS [3] , diffusion RLS [4] and diffusion Kalman filtering [5] and smoothing [6]. Algorithms based on consensus have been proposed in [7, 8].

This work was supported in part by NSF grants ECS-0601266 and ECS-0725441. Author's emails: {fcattiv, sayed}@ee.ucla.edu.

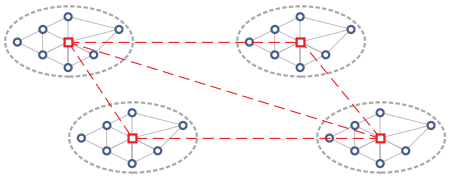

Fig. 1. Disjoint adaptive networks connected via supernodes.

In this work, we study multi-level diffusion algorithms, whereby an adaptive network is enhanced with a set of nodes with special abilities. These new nodes form a second network that also runs a diffusion algorithm, creating an adaptive network with two levels of adaptivity. We focus on the adapt-then-combine (ATC) version of diffusion LMS [9], though the results can be extended to other diffusion algorithms. We start by providing some motivation for multi-level diffusion, then propose an algorithm for multi-level diffusion, provide performance analysis for the case of diffusion LMS with small step-sizes, and show through simulation how adding a second layer of diffusion improves the overall performance of the network.

2. MOTIVATION

In our previous work we considered diffusion algorithms whereby nodes communicate with their neighbors in an isotropic manner. These algorithms usually include two updates: an adaptive update where nodes use their individual measurements to adapt their current estimates based on an adaptive filtering algorithm such as LMS or RLS, followed by a combination update where nodes combine the estimates from their neighbors to obtain the new estimate. This type of diffusion algorithms is fully distributed. If a node were to fail, the remaining nodes would continue operating, and the adaptive network would continue working.

Consider now a situation where the adaptive network is divided into non-connected sub-networks, as shown in Fig 1. This situation could arise in an application where the area to be sensed is too large, or where we wish to sense events located at points spatially far away from each other. One approach could be to scatter different adaptive networks over the area, each running a diffusion algorithm, say diffusion LMS. A question arises then about how to fuse the estimates from these sub-networks such that they benefit from interacting with each other. One solution would be to equip each one of the sub-networks with a supernode, which is able to communicate with the supernodes from the other networks (supernodes are shown as red squares in Fig. 1). This communication may be done through a special link or through multi-hop transmissions. Then the supernodes

would form a new network, and we could run a second diffusion algorithm on the supernodes. Thus, the supernodes would become a new adaptive network as shown by the red dashed links in Fig. 1.

Another scenario where multi-level diffusion algorithms could be useful is the case where we have a large adaptive network, and we want to improve its convergence speed. When an event occurs on one end of the network, it will take a number of iterations for this event to propagate. Again we can think of equipping the network with supernodes capable of communicating through larger distances. An example network is shown in Fig. 2. Events happening on one end of the network would propagate faster through the supernodes.

Thus, multi-level diffusion schemes are attractive in situations where the original adaptive network needs to be enhanced. It is important to note that multi-level diffusion algorithms should preserve the property of being fully distributed. If one of the supernodes fails, the network should be able to keep functioning without severe impact on the performance.

3. MULTI-LEVEL DIFFUSION

A multi-level diffusion algorithm consisting of two levels is discussed next. The algorithm gives special privileges to a subset of the nodes which we call *supernodes*. Thus, there will be a set of normal nodes running a diffusion algorithm, and a second level of diffusion running on the supernodes. Let $\mathcal{S}$ denote the set of nodes that are supernodes. Let $\mathcal{N}_k$ denote the neighborhood of node k, i.e., the set of nodes that are connected to node k, including k itself, and let $\mathcal{N}_k^{(1)}$ denote the set of supernodes connected to supernode k, including k itself. Consider matrices A, C and $A^{(1)}$ with individual non-negative real entries $a_{l,k}$, $c_{l,k}$ and $a_{l,k}^{(1)}$, respectively. These matrices satisfy

$$c_{l,k} = a_{l,k} = 0 \ \text{ if } l \notin \mathcal{N}_k \qquad a_{l,k}^{(1)} = 0 \ \text{ if } l \notin \mathcal{N}_k^{(1)} \tag{3}$$

$$\mathbb{1}^* A = \mathbb{1}^* \quad \mathbb{1}^* C = \mathbb{1}^* \quad C\mathbb{1} = \mathbb{1} \quad \mathbb{1}^* A^{(1)} = \mathbb{1}^* \tag{4}$$

where $\mathbb{1}$ denotes the $N \times 1$ vector with unit entries.

In a multi-level diffusion scheme, normal nodes (i.e., those that are *not* supernodes) will perform exactly the same updates as in a diffusion algorithm. In this case, we focus on the ATC diffusion LMS algorithm of [9], namely:

For every node k such that $k \notin \mathcal{S}$,

$$\begin{cases} \psi_{k,i} = w_{k,i-1} + \mu_k \sum\limits_{l \in \mathcal{N}_k} c_{l,k} u_{l,i}^* (d_l(i) - u_{l,i} w_{k,i-1}) \\ w_{k,i} = \sum\limits_{l \in \mathcal{N}_k} a_{l,k} \psi_{l,i} \end{cases} \tag{5}$$

The first and second equations of (5) are known as the adaptation and combination updates, respectively. The matrix C allows nodes to use measurements from their neighbors in the update, though for a choice $C = I$, no exchange of measurements is needed. In the latter case, at every time instant i, every node k updates the estimate in three steps. First, every node adapts its current estimate using its individual measurements $\{d_k(i), u_{k,i}\}$ to obtain $\psi_{k,i}$. Second, all nodes exchange their pre-estimates $\psi_{k,i}$ with their neighbors. Finally, every node combines (or averages) the pre-estimates to obtain the new estimate $w_{k,i}$.

The fact that normal nodes perform the same update as in a conventional diffusion algorithm adds robustness to the adaptive network. Normal nodes do not need to be aware that supernodes are present in the network, and will treat them as normal nodes. Thus, even if all the supernodes were to fail, this would be transparent to the rest of the nodes, and the adaptive network would keep functioning as a conventional single-level network.

For the supernodes, $k \in \mathcal{S}$, we adopt a diffusion mechanism in three steps as follows:

For every node k such that $k \in \mathcal{S}$,

$$\begin{cases} \phi_{k,i} = a_{k,k} w_{k,i-1} + \sum\limits_{l \in \mathcal{N}_k, l \neq k} a_{l,k} \psi_{l,i} \\ \varphi_{k,i} = \phi_{k,i} + \mu_k \sum\limits_{l \in \mathcal{N}_k} c_{l,k} u_{l,i}^* (d_l(i) - u_{l,i} \phi_{k,i}) \\ w_{k,i} = \psi_{k,i} = \sum\limits_{l \in \mathcal{N}_k^{(1)}} a_{l,k}^{(1)} \varphi_{l,i} \end{cases} \tag{6}$$

After the adaptive update of the normal nodes in (5), these nodes will transmit their pre-estimates $\psi_{k,i}$ to their neighbors. A supernode will receive these pre-estimates from its neighbors and will perform an initial combination step using these pre-estimates and its previous estimate $w_{k,i-1}$, to obtain $\phi_{k,i}$. Then it will run an adaptive step to obtain $\varphi_{k,i}$. In the third step, supernodes will exchange $\varphi_{k,i}$ with their neighboring supernodes $\mathcal{N}_k^{(1)}$, and will combine them to obtain $w_{k,i}$. The neighbors $\mathcal{N}_k$ of the supernode k will receive this estimate $w_{k,i} = \psi_{k,i}$ and will use it in the combination step of (5).

4. PERFORMANCE ANALYSIS

In this section we analyze the performance of the multi-level diffusion LMS algorithm. The analysis is similar to the one presented in [9], but the results are more general and include the ones presented in [9] as a special case.

In what follows we consider the estimates $\boldsymbol{w}_{k,i}$ to be random, and analyze their performance in terms of their expected behavior. Consider a diffusion LMS update consisting of two adaptive steps and two combination steps as follows:

$$\begin{cases} \boldsymbol{\psi}_{k,i} &= \boldsymbol{w}_{k,i-1} + \mu_{1,k} \sum\limits_{l=1}^{N} c_{1,l,k} \boldsymbol{u}_{l,i}^* [\boldsymbol{d}_l(i) - \boldsymbol{u}_{l,i} \boldsymbol{w}_{k,i-1}] \\ \boldsymbol{\phi}_{k,i} &= \sum\limits_{l=1}^{N} a_{1,l,k} \boldsymbol{\psi}_{l,i} \\ \boldsymbol{\varphi}_{k,i} &= \boldsymbol{\phi}_{k,i} + \mu_{2,k} \sum\limits_{l=1}^{N} c_{2,l,k} \boldsymbol{u}_{l,i}^* [\boldsymbol{d}_l(i) - \boldsymbol{u}_{l,i} \boldsymbol{\phi}_{k,i}] \\ \boldsymbol{w}_{k,i} &= \sum\limits_{l=1}^{N} a_{2,l,k} \boldsymbol{\varphi}_{l,i} \end{cases} \tag{7}$$

where the coefficients $a_{j,l,k}$ and $c_{j,l,k}$ are real, non-negative, and correspond to the $\{l,k\}$ entries of matrices A_j and C_j, respectively, for $j = 1, 2$, and the columns of A_j add up to one. The step-sizes $\mu_{1,k}$ and $\mu_{2,k}$ correspond to node k. Eq. (7) can be specialized to the multi-level diffusion LMS algorithm (5)-(6) by appropriately selecting these matrices - see (20)-(23).

As in [9], we introduce the following assumptions:

- The measurements are related to the unknown vector as follows:

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^o + \boldsymbol{v}_k(i) \tag{8}$$

  where $\boldsymbol{v}_k(i)$ is a zero-mean random variable with variance $\sigma_{v,k}^2$, independent of $\boldsymbol{u}_{k,i}$ for all k and i, and independent of $\boldsymbol{v}_l(j)$ for $l \neq k$ or $i \neq j$.
- All regressors $\boldsymbol{u}_{k,i}$ are spatially and temporally independent with covariance matrices $R_{u,k}$.
- The step-sizes $\{\mu_{j,k}\}_{k=1}^N$ are sufficiently small (see [9] for details).

We define the error quantities $\tilde{\boldsymbol{w}}_{k,i} = w^o - \boldsymbol{w}_{k,i}$, $\tilde{\boldsymbol{\psi}}_{k,i} = w^o - \boldsymbol{\psi}_{k,i}$, $\tilde{\boldsymbol{\phi}}_{k,i} = w^o - \boldsymbol{\phi}_{k,i}$ and $\tilde{\boldsymbol{\varphi}}_{k,i} = w^o - \boldsymbol{\varphi}_{k,i}$, and the global vectors:

$$\tilde{\boldsymbol{w}}_i = \mathrm{col}\left[\tilde{\boldsymbol{w}}_{1,i}, \ldots, \tilde{\boldsymbol{w}}_{N,i}\right] \quad \tilde{\boldsymbol{\psi}}_i = \mathrm{col}\left[\tilde{\boldsymbol{\psi}}_{1,i}, \ldots, \tilde{\boldsymbol{\psi}}_{N,i}\right]$$
$$\tilde{\boldsymbol{\phi}}_i = \mathrm{col}\left[\tilde{\boldsymbol{\phi}}_{1,i}, \ldots, \tilde{\boldsymbol{\phi}}_{N,i}\right] \quad \tilde{\boldsymbol{\varphi}}_i = \mathrm{col}\left[\tilde{\boldsymbol{\varphi}}_{1,i}, \ldots, \tilde{\boldsymbol{\varphi}}_{N,i}\right]$$

We introduce the matrices below for $j = 1, 2$:

$$\mathcal{M}_j = \mathrm{diag}\left\{\mu_1^{(j)} I_M, \ldots, \mu_N^{(j)} I_M\right\} \tag{9}$$

$$\mathcal{A}_j = A_j \otimes I_M, \qquad \mathcal{C}_j = C_j \otimes I_M \tag{10}$$

$$\boldsymbol{D}_{j,i} = \mathrm{diag}\left\{\sum_{l=1}^{N} c_{l,1}^{(j)} \boldsymbol{u}_{l,i}^* \boldsymbol{u}_{l,i} \ , \ \ldots \ , \ \sum_{l=1}^{N} c_{l,N}^{(j)} \boldsymbol{u}_{l,i}^* \boldsymbol{u}_{l,i}\right\}$$
$$\boldsymbol{G}_i = \mathrm{col}\left\{\boldsymbol{u}_{1,i}^* \boldsymbol{v}_1(i) \ , \ \ldots \ , \ \boldsymbol{u}_{N,i}^* \boldsymbol{v}_N(i)\right\}$$

Then from (7) and the linear model assumption we have:

$$\begin{aligned}
\tilde{\boldsymbol{\psi}}_i &= \tilde{\boldsymbol{w}}_{i-1} - \mathcal{M}_1[\boldsymbol{D}_{1,i}\tilde{\boldsymbol{w}}_{i-1} + \mathcal{C}_1^T \boldsymbol{G}_{1,i}] \\
\tilde{\boldsymbol{\phi}}_i &= \mathcal{A}_1^T \tilde{\boldsymbol{\psi}}_i \\
\tilde{\boldsymbol{\varphi}}_i &= \tilde{\boldsymbol{\phi}}_i - \mathcal{M}_2[\boldsymbol{D}_{2,i}\tilde{\boldsymbol{\phi}}_i + \mathcal{C}_2^T \boldsymbol{G}_{2,i}] \\
\tilde{\boldsymbol{w}}_i &= \mathcal{A}_2^T \tilde{\boldsymbol{\varphi}}_i
\end{aligned}$$

or, equivalently,

$$\begin{aligned}
\tilde{\boldsymbol{w}}_i &= \mathcal{A}_2^T[I - \mathcal{M}_2\boldsymbol{D}_{2,i}]\mathcal{A}_1^T[I - \mathcal{M}_1\boldsymbol{D}_{1,i}]\tilde{\boldsymbol{w}}_{i-1} - \\
&\quad \mathcal{A}_2^T[I - \mathcal{M}_2\boldsymbol{D}_{2,i}]\mathcal{A}_1^T\mathcal{M}_1\mathcal{C}_1^T\boldsymbol{G}_i - \mathcal{A}_2^T\mathcal{M}_2\mathcal{C}_2^T\boldsymbol{G}_i
\end{aligned} \tag{11}$$

Using the assumption that the step-sizes are small for all filters, we drop the terms that include products of step-sizes from (11) to obtain

$$\begin{aligned}
\tilde{\boldsymbol{w}}_i &\approx \mathcal{A}_2^T[\mathcal{A}_1^T - \mathcal{M}_2\boldsymbol{D}_{2,i}\mathcal{A}_1^T - \mathcal{A}_1^T\mathcal{M}_1\boldsymbol{D}_{1,i}]\tilde{\boldsymbol{w}}_{i-1} - \\
&\quad \mathcal{A}_2^T\mathcal{A}_1^T\mathcal{M}_1\mathcal{C}_1^T\boldsymbol{G}_i - \mathcal{A}_2^T\mathcal{M}_2\mathcal{C}_2^T\boldsymbol{G}_i
\end{aligned} \tag{12}$$

Moreover, for $j = 1, 2$, let

$$\mathcal{D}_j \triangleq \mathrm{E}\,\boldsymbol{D}_{j,i} = \mathrm{diag}\left\{\sum_{l=1}^{N} c_{l,1}^{(j)} R_{u,l} \ , \ \ldots \ , \ \sum_{l=1}^{N} c_{l,N}^{(j)} R_{u,l}\right\}$$
$$\mathcal{G} \triangleq \mathrm{E}[\boldsymbol{G}_i\boldsymbol{G}_i^*] = \mathrm{diag}\left\{\sigma_{v,1}^2 R_{u,1} \ , \ \ldots \ , \ \sigma_{v,N}^2 R_{u,N}\right\} \tag{13}$$

We follow the energy conservation analysis of [10]. From the independence of the regressors $u_{k,i}$ we obtain that $\boldsymbol{D}_{j,i}$ is independent of $\tilde{\boldsymbol{w}}_{i-1}$. Evaluating the weighted norm of $\tilde{w}_i$ in (12) we obtain:

$$\mathrm{E}\,||\tilde{\boldsymbol{w}}_i||_\Sigma^2 = \mathrm{E}\,||\tilde{\boldsymbol{w}}_{i-1}||_{\Sigma'}^2 + \mathrm{Tr}[\Sigma K \mathcal{G} K^*] \tag{14}$$

where Σ is any Hermitian positive-definite matrix and

$$\begin{aligned}
\Sigma' &\approx \mathcal{A}_1\mathcal{A}_2\Sigma\mathcal{A}_2^T\mathcal{A}_1^T - \\
&\quad \mathcal{A}_1\mathcal{A}_2\Sigma\mathcal{A}_2^T\mathcal{M}_2\mathcal{D}_2\mathcal{A}_1^T - \mathcal{A}_1\mathcal{D}_2^*\mathcal{M}_2\mathcal{A}_2\Sigma\mathcal{A}_2^T\mathcal{A}_1^T - \\
&\quad \mathcal{A}_1\mathcal{A}_2\Sigma\mathcal{A}_2^T\mathcal{A}_1^T\mathcal{M}_1\mathcal{D}_1 - \mathcal{D}_1^*\mathcal{M}_1\mathcal{A}_1\mathcal{A}_2\Sigma\mathcal{A}_2^T\mathcal{A}_1^T \\
K &= \mathcal{A}_2^T\mathcal{A}_1^T\mathcal{M}_1\mathcal{C}_1^T + \mathcal{A}_2^T\mathcal{M}_2\mathcal{C}_2^T
\end{aligned} \tag{15}$$

where again we ignored terms including products of step-sizes.

Let

$$\sigma = \mathrm{vec}(\Sigma), \qquad \Sigma = \mathrm{vec}^{-1}(\sigma)$$

where the $\mathrm{vec}(\cdot)$ notation stacks the columns of the matrix argument on top of each other. We also use the notation $\|\tilde{w}\|_\sigma^2$ to denote $\|\tilde{w}\|_\Sigma^2$. Using the Kronecker product property

$$\mathrm{vec}(P\Sigma Q) = (Q^T \otimes P)\mathrm{vec}(\Sigma)$$

we arrive at

$$\sigma' \triangleq \mathrm{vec}(\Sigma') = F\sigma$$

$$\begin{aligned}
F &= (\mathcal{A}_1 \otimes \mathcal{A}_1)\{\mathcal{A}_2 \otimes \mathcal{A}_2 - \mathcal{A}_2 \otimes (\mathcal{D}_2\mathcal{M}_2\mathcal{A}_2) - \\
&\quad (\mathcal{D}_2^T\mathcal{M}_2\mathcal{A}_2) \otimes \mathcal{A}_2\} - (\mathcal{D}_1^T\mathcal{M}_1\mathcal{A}_1\mathcal{A}_2) \otimes (\mathcal{A}_1\mathcal{A}_2) - \\
&\quad (\mathcal{A}_1\mathcal{A}_2) \otimes (\mathcal{D}_1^T\mathcal{M}_1\mathcal{A}_1\mathcal{A}_2)
\end{aligned} \tag{16}$$

Then, using the result that $\mathrm{Tr}(\Sigma X) = \mathrm{vec}(X^T)^T\sigma$ we arrive at

$$\mathrm{E}\,||\tilde{\boldsymbol{w}}_\infty||_{(I-F)\sigma}^2 = [\mathrm{vec}(K\mathcal{G}^T K^T)]^T\sigma \tag{17}$$

The MSD at node k can be obtained by weighting $\mathrm{E}\,||\tilde{\boldsymbol{w}}_\infty||^2$ with a block matrix that has an identity matrix at block $\{k, k\}$ and zeros elsewhere. Let us denote the vectorized version of this matrix by q_k, that is:

$$q_k = \mathrm{vec}(\mathrm{diag}(e_k) \otimes I_M)$$

Then the MSD becomes:

$$\mathrm{MSD}_k = \mathrm{E}\,||\tilde{\boldsymbol{w}}_\infty||_{q_k}^2 = [\mathrm{vec}(K\mathcal{G}^T K^T)]^T(I - F)^{-1}q_k \tag{18}$$

The EMSE at node k is obtained by weighting $\mathrm{E}\,||\tilde{\boldsymbol{w}}_\infty||^2$ with a block matrix that has R_{u_k} at block $\{k, k\}$ and zeros elsewhere, that is, by selecting

$$(I - F)\sigma = r_k = \mathrm{vec}(\mathrm{diag}(e_k) \otimes R_{u_k})$$

Then the EMSE becomes:

$$\mathrm{EMSE}_k = \mathrm{E}\,||\tilde{\boldsymbol{w}}_\infty||_{r_k}^2 = [\mathrm{vec}(K\mathcal{G}^T K^T)]^T(I - F)^{-1}r_k \tag{19}$$

Note that if we select $\mathcal{M}_1 = 0$, we obtain the same result as in [9], and therefore the case presented here is more general. The *network* MSD and EMSE are defined as the average MSD and EMSE, respectively, across all nodes in the network.

In order to apply the above analysis to the multi-level diffusion LMS algorithm (5)-(6), we need to appropriately select the matrices A_1, A_2, C_1 and C_2 in (7). Let γ denote a vector of size N with unity entries at position k if $k \in \mathcal{S}$, and zeros elsewhere. Then we have

$$C_1 = C \cdot \mathrm{diag}(\mathbb{1} - \gamma) \tag{20}$$
$$A_1 = A \cdot \mathrm{diag}(\gamma) + I_N \cdot \mathrm{diag}(\mathbb{1} - \gamma) \tag{21}$$
$$C_2 = C \cdot \mathrm{diag}(\gamma) \tag{22}$$
$$A_2 = A^{(1)}[I_N \cdot \mathrm{diag}(\gamma) + A \cdot \mathrm{diag}(\mathbb{1} - \gamma)] \tag{23}$$

Eq. (20) indicates that all nodes that are not supernodes will update their estimates using the adaptive update. Eq. (21) indicates that all the supernodes will combine the estimates from their neighbors, whereas the normal nodes will not update their estimates. Eq. (22) represents the step where the supernodes perform the adaptive update on the data from their neighbors. Finally, Eq. (23) represents the steps where the supernodes combine the estimates from their supernode neighbors (via $A^{(1)}$), combined with the step where all normal nodes combine the estimates from their neighbors (via A).

5. SIMULATIONS

In order to illustrate the performance of the multi-level diffusion algorithms, we present a simulation example in Figs. 2-4. We use a network with $N = 30$ nodes and a topology as in Fig. 2, where the supernodes are denoted by squares. Supernode neighbors are defined as those supernodes that are 3 or less hops away from each other. The regressors have size $M = 2$, are zero-mean Gaussian, independent

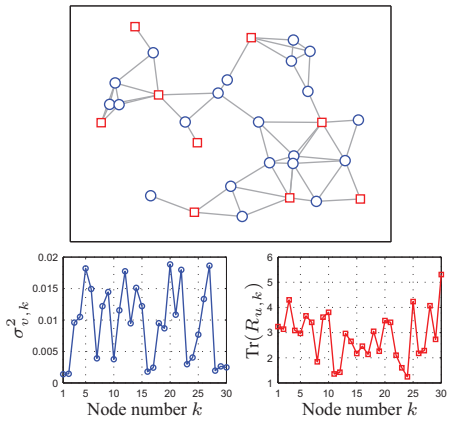

Fig. 2. Network topology (top), noise variances $\sigma^2_{v,k}$ (bottom, left) and trace of regressor covariances $\mathrm{Tr}(R_{u,k})$ (bottom, right).

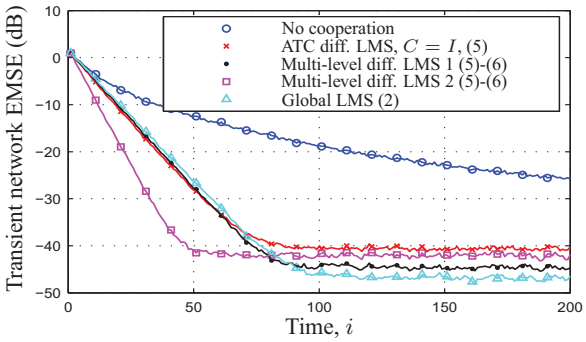

Fig. 3. Transient network EMSE (top) and MSD (bottom) for LMS without cooperation, CTA and ATC diffusion LMS, and global LMS.

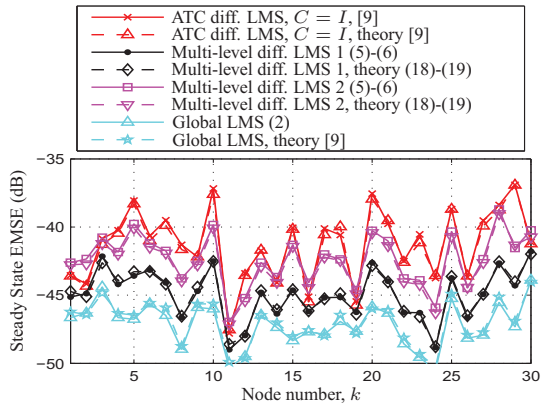

Fig. 4. Steady-state performance of different algorithms.

in time and space and have covariance matrices $R_{u,k}$. The background noise power is denoted by $\sigma^2_{v,k}$. Fig. 2 also shows $\sigma^2_{v,k}$ and $\mathrm{Tr}(R_{u,k})$ for every node k. The results were averaged over a total of 100 experiments of duration 200 iterations each. The unknown w^o was chosen randomly at the beginning of each experiment. The step-size $\mu_k = 0.05$ is constant for all nodes and algorithms (except for one of the muli-level diffusion algorithms as discussed next). For the diffusion algorithms, relative-degree weights [4] are used for the matrix A and for the matrix $A^{(1)}$, with the exception that supernodes are assigned ten times higher degree than their actual degree in order to compute the weights. In this way we give more weight to the estimates of the supernodes. We also used $C = I$ in all cases.

We compare five algorithms. The algorithm denoted "No cooperation" corresponds to the case where every node runs an LMS algorithm on its data and does not communicate at all with its neighbors. The algorithm denoted "Global LMS" is the global solution (2). "ATC diff LMS" refers to algorithm [9],(5) using $C = I$. "Multi-level ATC diffLMS 1" and "Multi-level ATC diffLMS 2" denote the multi-level diffusion LMS algorithm (5)-(6). The former algorithm uses a $\mu_k = 0.0083$ if k is a supernode, and $\mu_k = 0.05$ otherwise, whereas the latter uses a constant step-size of $\mu_k = 0.05$.

Fig. 3 shows the learning curves for the different LMS algorithms in terms of the network EMSE. We can observe that the multi-level algorithms outperform the conventional diffusion LMS algorithm. Moreover, the first multi-level algorithm attains comparable steady-state performance, yet converges much faster, whereas the second multi-level algorithm with a smaller step-size in the supernode level attains a comparable convergence rate, with improved steady-state performance. The steady-state is close to the one offered by the global solution. Fig. 4 shows the steady-state network EMSE for the same algorithms, showing agreement with the theoretical results from expressions (18) and (19). The steady-state values are obtained by averaging over 40 time samples after convergence.

6. CONCLUSIONS

We proposed a cooperation scheme denoted *multi-level diffusion* where nodes with enhanced characteristics are introduced in the network. We showed through theory and simulation that this method improves both the convergence and steady-state performance over conventional diffusion methods, at the expense of requiring special provisions to accommodate communication between the supernodes.

7. REFERENCES

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

[2] A. H. Sayed and F. Cattivelli, "Distributed adaptive learning mechanisms," Handbook on Array Processing and Sensor Networks, S. Haykin and K. J. Ray Liu, editors, Wiley, NJ, 2009.

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

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

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

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

[7] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc IPSN*, Nashville, TN, April 2006, pp. 168–176.

[8] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Stability analysis of the consensus-based distributed LMS algorithm," in *Proc. ICASSP*, Las Vegas, NV, March 2008, pp. 3289–3292.

[9] F. S. Cattivelli and A. H. Sayed, "Diffusion lms algorithms with information exchange," in *Proc. Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, October 2008.

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