Reduced-Communication Diffusion LMS Strategy for Adaptive Distributed Estimation

Reza Arablouei, Member, IEEE, Yih-Fang Huang, Fellow, IEEE, Stefan Werner, Senior Member, IEEE, and Kutluyl Doğançay, Senior Member, IEEE

Abstract—In diffusion-based algorithms for adaptive distributed estimation, each node of an adaptive network estimates a target parameter vector by creating an intermediate estimate and then combining the intermediate estimates available within its closed neighborhood. We propose a reduced-communication diffusion least mean-square (RC-DLMS) algorithm by allowing each node to only receive the intermediate estimates of a subset of its neighbors at each iteration. Therefore, the proposed RC-DLMS algorithm eases the usage of network communication resources and delivers a trade-off between estimation performance and communication cost. We examine the performance of the RC-DLMS algorithm analytically and show that it is stable and convergent in both mean and mean-square senses. We also calculate the theoretical steady-state mean-square deviation of the RC-DLMS algorithm and compute combination weights that optimize its performance in the small-step-size regime. Simulation results confirm the effectiveness of the RC-DLMS algorithm as well as a good match between theory and experiment.

Index Terms—Adaptive networks; communication reduction; diffusion adaptation; distributed estimation; least mean-square.

I. INTRODUCTION

The so-called diffusion strategies are effective methods for performing distributed estimation over adaptive networks. In a typical diffusion-based adaptive estimation algorithm, all network nodes concurrently make individual intermediate estimates of a target system parameters using the data locally accessible to them. Then, the nodes communicate with all their immediate neighbors to exchange their intermediate estimates. Subsequently, each node fuses the intermediate estimates received from its neighborhood together with its own to generate a new estimate. The procedure is repeated in all iterations [1]-[3]. This in-network cooperative processing helps the information propagate across the network so that all nodes can benefit from the observable data of the entire network. As a result, not only is the estimation performance of each network node significantly improved compared with the isolated case, but every node can asymptotically perform as well as in the often-theoretical fully-connected case [4], [5].

However, the benefits of diffusion-based estimation techniques come at the expense of increased internode communications. As all nodes transmit to and receive data from all their direct neighbors, depending on the average node-degree of the network, the total amount of required internode communications can become prohibitive. For example, in a network of $K = 20$ nodes where each node is connected to $D = 5$ other nodes on average, there will be total $KD = 100$ active internode links when using a diffusion strategy; whereas a centralized or incremental [6], [7] strategy will require only $K = 20$ node-to-hub or internode communication links. This large amount of internode communications required by the diffusion strategies may strain the valuable and often limited power, bandwidth, or hardware resources, particularly in wireless sensor networks. Moreover, in order to implement the conventional diffusion-based algorithms, each node should be able to communicate with all its neighbors simultaneously or within a certain time-frame. As a consequence, since different nodes can have different numbers of neighbors, they may require disparate hardware or consume power dissimilarly. This may in turn compromise the flexibility and efficiency of the network for ad-hoc deployment. Therefore, it is of practical importance to reduce the amount of internode communications in diffusion strategies while maintaining the benefits of cooperation.

There have been several attempts to reduce the communication complexity of the diffusion-based algorithms, particularly the diffusion least mean-square (DLMS) algorithm presented in [8], [9]. In the probabilistic diffusion LMS (P-DLMS) algorithm, each communication link is intermittently activated with a given probability [10]. Hence, the average amount of total internode communications taken place in the network is reduced. However, the total communication cost of the P-DLMS algorithm can vary in time. In [11], an approach for dynamically optimizing the link probabilities of the P-DLMS algorithm is proposed. The single-link diffusion LMS algorithms of [12] and [13] disconnect all links but one for each node to reduce the communications overhead. Each algorithm uses a different technique to select the neighbor with which each node communicates at each iteration by minimizing the steady-state network mean-square deviation (MSD). In [14], a set-theoretic diffusion-based algorithm is proposed, which can trade estimation performance and computational complexity for communication cost. The diffusion-based adaptive algorithm

This work was supported in part by the Academy of Finland.

R. Arablouei and K. Doğançay are with the School of Engineering and the Institute for Telecommunications Research, University of South Australia, Mawson Lakes SA 5095, Australia (email: reza.arablouei@unisa.edu.au, kutluyl.dogancay@unisa.edu.au).

Y.-F. Huang is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA (email: huang@nd.edu).

S. Werner is with the Department of Signal Processing and Acoustics, School of Electrical Engineering, Aalto University, Espoo, Finland (email: stefan.werner@aalto.fi).
of [15] mitigates the communication load by exchanging either a scalar or a single information bit generated from random projections of the intermediate estimate vector of each node. The works of [16]-[18] have utilized partial-update [19] or set-membership filtering [20] to alleviate the communication cost. In [21]-[23], two low-communication algorithms for adaptive distributed estimation have been proposed that employ the notion of partial diffusion where each node transmits a part of the entries of its intermediate estimate vector to its neighbors at each iteration.

In this paper, we propose a reduced-communication diffusion LMS (RC-DLMS) algorithm for distributed estimation over adaptive networks where, at each iteration, every node consults only a subset of its neighbors, i.e., receives the intermediate estimates of only a subset of its neighbors. The proposed algorithm reduces the total amount of internode communications in the network relative to the DLMS algorithm, where the nodes always receive the intermediate estimates of all their neighbors, with limited degradation in performance. We analyze the performance of the RC-DLMS algorithm utilizing the energy conservation argument [24]-[26]. We establish its stability and convergence in the mean and mean-square senses and derive a theoretical expression for its steady-state MSD. We also find combination weights that, with sufficiently small step-sizes, make the RC-DLMS algorithm perform close to the DLMS algorithm regardless of the number of consulted neighbors at each node. The presented results verify the usefulness of the RC-DLMS algorithm as well as the accuracy of the theoretical findings.

II. ALGORITHM DESCRIPTION

A. Diffusion Least Mean-Square Algorithm

Consider a connected network of $K \in \mathbb{N}$ nodes that collectively aim to estimate a parameter vector denoted by $h \in \mathbb{R}^{L \times 1}$ in an adaptive and collaborative manner. At each time instant $n \in \mathbb{N}$, every node $k \in \{1,2, ..., K\}$ observes a regressor vector $x_{k,n} \in \mathbb{R}^{L \times 1}$ and a scalar $y_{k,n} \in \mathbb{R}$ that are linearly related via

$$y_{k,n} = x_{k,n}^T h + v_{k,n}$$

where $v_{k,n} \in \mathbb{R}$ is the noise.

In the adapt-then-combine (ATC) diffusion least mean-square (DLMS) algorithm [8], each node produces an intermediate estimate using its previous estimate and recent observed data (adaptation phase):

$$z_{k,n} = w_{k,n-1} + \mu_k x_{k,n} (y_{k,n} - x_{k,n}^T w_{k,n-1})$$

where $\mu_k \in \mathbb{R}_{>0}$ is the step-size at node $k$. After sharing its intermediate estimate with its neighbors, each node creates a new estimate by combining the intermediate estimates available within its neighborhood (consultation phase):

$$w_{k,n} = \tilde{c}_{k,k} z_{k,n} + \sum_{l \in N_k} \tilde{c}_{k,l} z_{l,n}.$$  \hspace{1cm} (3)

The set $N_k$ denotes the open neighborhood of node $k$, i.e., $N_k$ comprises all nodes that are connected to node $k$ within one hop and excludes the node $k$ itself. The combination weights $\{\tilde{c}_{k,l} \in \mathbb{R}_{\geq 0}\}$ satisfy [1]

$$\forall k: \sum_{l=1}^{K} \tilde{c}_{k,l} = 1, \tilde{c}_{k,l} = 0 \text{ if } l \not\in N_k \cup \{k\},$$

$$\exists k: \tilde{c}_{k,k} > 0.$$  \hspace{1cm} (4)

Reversing the order of the two phases gives the combine-then-adapt (CTA) version of the DLMS algorithm [9]. In the above-mentioned ATC DLMS algorithm, only the local (node-specific) observation data is used in the adaptation phase. However, more general ATC and CTA DLMS algorithms have been proposed in [8] where the directly-connected nodes exchange their observation data as well as the intermediate estimates and use them in the adaptation phase. Since we aim to reduce the internode communications here, we only consider the ATC DLMS algorithm of (2)-(3), based on which our reduced-communication algorithm is derived. The proposed idea can also be applied to the CTA case in a similar fashion.

B. Communication Reduction

At each iteration of the DLMS algorithm, every node receives the intermediate estimates of all its neighbors, which are $d_k = |N_k|$ nodes where $|$ is the cardinality operator and $d_k$ is called the degree of node $k$. To reduce the internode communications, we allow each node to receive the intermediate estimates from $0 < m_k \leq d_k$ of its neighbors at each iteration. To this end, we define a selection variable called $a_{k,l,n}$ that specifies the status of neighbor $l$ of node $k$ at time instant $n$. This variable can be either 0 or 1 where $a_{k,l,n} = 1$ means that, at iteration $n$, node $k$ communicates with its neighboring node $l$ and receives its intermediate estimate to use in the consultation phase. Having $a_{k,l,n} = 0$ means that, at iteration $n$, node $k$ does not receive the intermediate estimate of its neighbor $l$.

We propose to select the consulted neighbors of each node at each iteration either arbitrarily or in a round-robin fashion so that the probability of receiving the intermediate estimate of each neighbor at node $k$ is the same for all its neighbors at all iterations and is expressed as

$$p_k = E[a_{k,l,n}] = \frac{m_k}{d_k}.$$  \hspace{1cm} (5)

Thus, we make the following remark regarding the proposed reduced-communication schemes:

1. $R1$: At any node $k$, the selection variable, $a_{k,l,n}$, is independent of the observation data $x_{k,n}$ and $y_{k,n}$ as well as the noise $v_{k,n}$. Moreover, the selection probability, $p_k$, is time-invariant and identical for all neighbors of node $k$.

As an example, an adaptive network with 10 nodes and 20 unidirectional communication links is shown in Fig. 1 where the links that are active in four consecutive iterations of the RC-DLMS algorithm are highlighted. In this example, each node receives the intermediate estimate of only one of its neighbors at each iteration.

To define the consultation phase of the DLMS algorithm at node $k$ in terms of the differences between the intermediate estimate of the node $k$ and those of its neighbors, we can
rewrite (3) as
\[ \mathbf{w}_{k,n} = \mathbf{z}_{k,n} + \sum_{l \in \mathcal{N}_k} \tilde{c}_{k,l} (\mathbf{z}_{l,n} - \mathbf{z}_{k,n}). \] (5)

When the intermediate estimates of only \( m_k \) neighbors are received at node \( k \), the other intermediate estimates are not available for consultation. We replace these unavailable intermediate estimates with each node’s own intermediate estimate and change (5) to
\[ \mathbf{w}_{k,n} = \mathbf{z}_{k,n} + \sum_{l \in \mathcal{N}_k} a_{k,l,n} c_{k,l} (\mathbf{z}_{l,n} - \mathbf{z}_{k,n}). \] (6)

Consequently, our proposed reduced-communication DLMS (RC-DLMS) algorithm utilizes (2) in the adaptation phase and (6) in the consultation phase. Note that we use \( \{c_{k,l}\} \), which also satisfy (4), rather than \( \{\tilde{c}_{k,l}\} \) for denoting the combination weights of the RC-DLMS algorithm. The reason will become clear in Section IV.

The expressions (3) and (6), which can also be written as
\[ \mathbf{w}_{k,n} = c_{k,k} \mathbf{z}_{k,n} + \sum_{l \in \mathcal{N}_k} c_{k,l} \left[ a_{k,l,n} \mathbf{z}_{l,n} + (1 - a_{k,l,n}) \mathbf{z}_{k,n} \right], \]
require an identical number of arithmetic operations, i.e., \((d_k + 1)L\) multiplications and \(d_k L\) additions per iteration at any computational node \( k \). Therefore, the RC-DLMS algorithm has the same computational complexity as the DLMS algorithm.

III. PERFORMANCE ANALYSIS

We study the performance of the RC-DLMS algorithm in this section. The analysis covers the non-cooperative LMS and DLMS algorithms as the special cases of \( m_k = 0 \) and \( m_k = d_k \), respectively.

A. Assumptions

For the analysis, we adopt the following assumptions, which are commonly used to facilitate the analytical studies [24], [27]:

A1: The regressor vector \( \mathbf{x}_{k,n} \) is temporally and spatially independent and
\[ E[\mathbf{x}_{k,n} \mathbf{x}_{k,n}^T] = \mathbf{R}_k \in \mathbb{R}^{L \times L} \ \forall k, n. \]

A2: The noise \( \nu_{k,n} \) is independent of \( \mathbf{x}_{k,n} \). In addition, it is temporally and spatially independent
\[ E[\nu_{k,n}] = 0 \text{ and } E[\nu_{k,n}^2] = \zeta_{k,\nu}^2 \in \mathbb{R} \ \forall k, n. \]

A3: The adaptation step-size at each node is sufficiently small so that its squared value is negligible.

B. Network Update Equation

Define
\[ \mathbf{z}_{k,n} = \mathbf{z}_{k,n} - \mathbf{h}, \]
\[ \mathbf{w}_{k,n} = \mathbf{w}_{k,n} - \mathbf{h}, \]
\[ \mathbf{z}_n = [\mathbf{z}_{1,n}^T, \ldots, \mathbf{z}_{K,n}^T]^T, \]
and
\[ \tilde{\mathbf{w}}_n = [\tilde{w}_{1,n}^T, \ldots, \tilde{w}_{K,n}^T]^T. \]
Subtracting \( \mathbf{h} \) from both sides of (2) and (6) while using (1) gives
\[ \hat{\mathbf{z}}_{k,n} = (\mathbf{I}_L - \mu_k \mathbf{P}_k \mathbf{x}_{k,n}^T) \tilde{\mathbf{w}}_{k,n-1} + \mu_k \mathbf{x}_{k,n} \nu_{k,n} \]
\[ \hat{\mathbf{w}}_{k,n} = \left(1 - \sum_{l \in \mathcal{N}_k} a_{k,l,n} c_{k,l}\right) \hat{\mathbf{z}}_{k,n} + \sum_{l \in \mathcal{N}_k} a_{k,l,n} c_{k,l} \mathbf{z}_{l,n} \]
and subsequently
\[ \hat{\mathbf{z}}_n = (\mathbf{I}_L - \mathbf{M} \mathbf{x}_n) \hat{\mathbf{w}}_{n-1} + \mathbf{M} \mathbf{g}_n \]
\[ \hat{\mathbf{w}}_n = \hat{\mathbf{B}}_n \hat{\mathbf{z}}_n \]
or the following recursion:
\[ \hat{\mathbf{w}}_n = \hat{\mathbf{B}}_n (\mathbf{I}_L - \mathbf{M} \mathbf{x}_n) \hat{\mathbf{w}}_{n-1} + \hat{\mathbf{B}}_n \mathbf{M} \mathbf{g}_n \] (7)
where
\[ \mathbf{M} = \text{bdiag} \{\mu_1 \mathbf{I}_L, \ldots, \mu_L \mathbf{I}_L\}, \]
\[ \mathbf{x}_n = \text{bdiag} \{\mathbf{x}_{1,n} \mathbf{x}_{1,n}^T, \ldots, \mathbf{x}_{K,n} \mathbf{x}_{K,n}^T\}, \]
\[ \mathbf{g}_n = [\mathbf{x}_{1,n}^T \mathbf{x}_{1,n}^T, \ldots, \mathbf{x}_{K,n}^T \mathbf{x}_{K,n}^T]^T, \]
\[ \hat{\mathbf{B}}_n = \mathbf{B}_n \otimes \mathbf{I}_L, \]
\[ \mathbf{B}_n = \begin{bmatrix} b_{1,1,n} & \cdots & b_{1,K,n} \\ \vdots & \ddots & \vdots \\ b_{K,1,n} & \cdots & b_{K,K,n} \end{bmatrix}, \]
\[ b_{i,j,n} = \begin{cases} 1 - \sum_{l \in \mathcal{N}_i} a_{l,i,n} c_{l,i} & \text{if } j = i \\ a_{l,i,n} c_{l,j} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise}, \end{cases} \]
and \( \otimes \) denotes the Kronecker product.

C. Mean Stability

From R1 and A1-A2, we deduce the following corollary:
C1: The vector \( \hat{\mathbf{w}}_{n-1} \) is statistically independent of \( \hat{\mathbf{B}}_n, \mathbf{x}_n, \) and \( \mathbf{g}_n \). Moreover, \( \hat{\mathbf{B}}_n \) is statistically independent of \( \mathbf{x}_n \) and \( \mathbf{g}_n \).

Taking the expectation on both sides of (7) while considering C1 and A2 results in
\[ E[\hat{\mathbf{w}}_n] = \left(\mathbf{B} \otimes \mathbf{I}_L\right) (\mathbf{I}_L - \mathbf{M} \mathbf{R}) E[\hat{\mathbf{w}}_{n-1}] \]
where
\[ \mathbf{R} = \text{blockdiag} \{\mathbf{R}_1, \ldots, \mathbf{R}_K\}, \]
\[ \hat{\mathbf{B}} = E[\hat{\mathbf{B}}_n] = \begin{bmatrix} \hat{b}_{1,1,n} & \cdots & \hat{b}_{1,K,n} \\ \vdots & \ddots & \vdots \\ \hat{b}_{K,1,n} & \cdots & \hat{b}_{K,K,n} \end{bmatrix}, \]
and
\[
\bar{b}_{i,j,n} = \begin{cases} 
1 - \sum_{l \in \mathcal{N}_i} p_i c_{i,l} & j = i \\
0 & j \in \mathcal{N}_i \\
-p_i c_{i,j} & j = i \\
0 & \text{otherwise}
\end{cases}
\]

All entries of \( \bar{B} \otimes I \) are real non-negative and all its rows add up to unity as we have

\[
(\bar{B} \otimes I_L) 1_{LK} = E[\bar{B}_n 1_K] \otimes 1_L = 1_K \otimes 1_L = 1_{LK}.
\]

Therefore, \( \bar{B} \otimes I \) is right-stochastic and has unit spectral radius. As a result, similar to the DLMS algorithm, the mean stability and asymptotic unbiasedness of the RC-DLMS is ensured if the spectral radius of \( I_{LK} - MR \) is smaller than one [8]. This can be realized by choosing the step-size of each node \( k \) such that:

\[
0 < \mu_k < \frac{2}{\lambda_{\max}(R_k)}
\]

where \( \lambda_{\max}(R_k) \) is the eigenvalue of \( R_k \) that has the largest absolute value.

### D. Variance Relation

Denote an arbitrary symmetric nonnegative-definite matrix by \( S \) and define the squared weighted Euclidean norm of a vector \( b \) with a weighting matrix \( A \) as

\[
\|b\|^2_{A} = \|b\|^2_{vec(A)} = b^T A b.
\]

Taking the expectation of the squared weighted Euclidean norm on both sides of (7) with CI in mind gives the following weighted variance relation:

\[
E[\|\bar{\Phi}_n\|_2^2] = E[\|\bar{\Phi}_{n-1}\|_2^2] + E[g_n^T M \bar{B}_n S \bar{B}_n M g_n]
\]

\[
T = E[(I_{LK} - M X_n) \bar{B}_n^T S \bar{B}_n (I_{LK} - M X_n)].
\]

Applying the vectorization operator to (10) together with using the property [28]

\[
vec(ABC) = (C^T \otimes A) vec(B)
\]

yields

\[
vec(T) = F D S
\]

where

\[
F = E[(I_{LK} - M X_n) \otimes (I_{LK} - M X_n)] \\
\approx (I_{LK} - MR) \otimes (I_{LK} - MR),
\]

\[
D = E[\bar{B}_n^T \otimes \bar{B}_n],
\]

and

\[
s = vec(S).
\]

The transpose of \( D \) is calculated as

\[
D^T = E[\bar{B}_n^T \otimes \bar{B}_n]
\]

\[
= \left[ I_L \otimes E[\bar{B}_{1,1,n}^T \bar{B}_n] \ldots I_L \otimes E[\bar{B}_{1,K,n}^T \bar{B}_n] \right] \otimes I_L
\]

using (12), shown at the bottom of the page, and

\[
E[a_{i,j,n} a_{k,l,n}] = \begin{cases} 
0 & i = k & j = l \\
p_i & i = j & l \neq k & l & d_i = 1 \\
p_k & i = k & j \neq l & & d_i > 1 \\
p_i p_k & i \neq k & j \neq l & &
\end{cases}
\]

In view of RL, the property [28]

\[
tr(A^T B) = vec^T(B) vec(A),
\]

and the fact that \( S \) is symmetric and deterministic, we have

\[
E[\bar{g}_n^T M \bar{B}_n S \bar{B}_n M g_n] = vec^T(H) D S
\]

where

\[
H = ME[\bar{g}_n^T g_n]M = \text{blockdiag}\{\mu_1^2 \xi_1^2 R_1, \ldots, \mu_K^2 \xi_K^2 R_K\}.
\]

Substituting (11) and (13) into (9) gives

\[
E[\|\bar{\Phi}_n\|_2^2] = E[\|\bar{\Phi}_{n-1}\|_2^2] + vec^T(H) D S.
\]

### E. Mean-Square Stability

The recursion (14) is stable if the matrix \( FD \) is stable [24]. The entries of \( D \) are all real-valued and non-negative. In addition, we have

\[
D^T 1_{L^2 K^2} = E[(\bar{B}_n \otimes I_L) 1_{LK} \otimes (\bar{B}_n \otimes I_L) 1_{LK}]
\]

\[
= E[\bar{B}_n 1_K \otimes I_L \otimes B_n 1_K \otimes I_L]
\]

\[
= 1_{L^2 K^2}.
\]

\[
E[b_{i,j,n} b_{k,l,n}] = \begin{cases} 
1 - p_i (1 - c_{i,l}) - p_k (1 - c_{k,l}) + \sum_{r \in \mathcal{N}_l \cup \mathcal{N}_k} \sum_{u \in \mathcal{N}_k} c_{l,r} c_{k,u} E[a_{i,r,n} a_{k,u,n}] & i = j \& k \neq l \\
p_k c_{k,l} - c_{k,l} \sum_{r \in \mathcal{N}_l} c_{i,r} E[a_{i,r,n} a_{k,l,n}] & i = j \& l \in \mathcal{N}_k \\
p_i c_{l,j} - c_{l,j} \sum_{r \in \mathcal{N}_l} c_{r,k} E[a_{l,j,n} a_{k,r,n}] & j \in \mathcal{N}_l \& k = l \\
0 & \text{otherwise}
\end{cases}
\]
This means that \( D \) is left-stochastic and has unit spectral radius. Hence, the spectral radius of \( FD \) is equal to the spectral radius of \( F \), which is the square of the spectral radius of \( I_{kK} - MR \). Therefore, the RC-DLMS algorithm is stable in the mean-square sense and converges to a steady state when the step-sizes satisfy (8).

**F. Steady-State Mean-Square Deviation**

Using (14), at the steady state, we can write

\[
\lim_{n \to \infty} E \left[ \left\| \mathbf{w}_n \right\|_2^2 (I_{2K}^2 - FD)s \right] = \text{vec}^T (H) D s. \tag{15}
\]

Setting

\[ s = (I_{2K}^2 - FD)^{-1} \text{vec} \{ f_{k} \} \]

in (15), the steady-state MSD of node \( k \), defined by

\[ \eta_k = \lim_{n \to \infty} E \left[ \left\| \mathbf{w}_n \right\|_2^2 (I_{2K}^2 - FD) \right] \]

can be calculated as

\[ \eta_k = \text{vec}^T (H) D (I_{2K}^2 - FD)^{-1} \text{vec} \{ f_{k} \} . \]

All the entries of \( f_{k} \in \mathbb{R}^{K \times K} \) are zero except the \((k, k)\)th entry that is one. Similarly, the steady-state network MSD is defined and calculated as

\[ \eta = \frac{1}{K} \sum_{k=1}^{K} \eta_k = \frac{1}{K} \text{vec}^T (H) D (I_{2K}^2 - FD)^{-1} \text{vec} \{ I_{2K} \} . \]

Note that the stability of \( FD \) implies that \( I_{2K}^2 - FD \) is invertible.

**IV. OPTIMAL COMBINATION**

It is shown in [5] that, under \( \lambda \), the steady-state MSD of all nodes of the DLMS algorithm can be well approximated to the first order in the step-sizes as

\[
\tilde{\eta}_k = \frac{1}{2} \text{tr} \left[ (H) M \mathbb{R} (I_{kK} \otimes I_{L}) \right]^{-1} \times \left[ \text{vec}^T (H) D (I_{2K}^2 - FD) \right] \text{vec} \{ f_{k} \}.
\]

where \( \mathbf{q} = [q_{1}, ..., q_{K}]^T \in \mathbb{R}^{K \times 1} \) is the normalized left Perron eigenvector of the combination matrix \( \mathbf{C} \), composed of the weights \( \{ c_{kl} \} \). Due to (4), \( \mathbf{C} \) is right-stochastic and primitive hence the Perron-Ferobinus theorem ensures that it has a unique eigenvalue at one and all the other eigenvalues are strictly smaller than one in the absolute value [29]. The left Perron eigenvector of such a matrix is its left eigenvector that corresponds to its eigenvalue at one. The Perron-Ferobinis theorem also ensures that all the entries of \( \mathbf{q} \) are positive and less than one as it is normalized, which means that its entries add up to one.

If we admit the following approximation:

\[ D = E \left[ \mathbf{B} \otimes \mathbf{B} \right] \approx E \left[ \mathbf{B} \otimes \mathbf{B} \right] \approx \mathbf{B} \otimes I_{L} \otimes \mathbf{B} \otimes I_{L}, \]

the RC-DLMS algorithm can be seen as a DLMS algorithm with \( \mathbf{B} \) in place of \( \mathbf{C} \). Thus, we can have the following approximate expression for the steady-state MSD of the RC-DLMS algorithm:

\[ \eta_k = \frac{1}{2} \text{tr} \left[ (H) M \mathbb{R} (I_{kK} \otimes I_{L}) \right]^{-1} \times \left[ \text{vec}^T (H) D (I_{2K}^2 - FD) \right] \text{vec} \{ f_{k} \} . \]

The rows of the system of linear equations (16) can be stated as

\[(1 - p_{k})\tilde{q}_{i} + p_{k}c_{i,j}\tilde{q}_{i} + \sum_{j \in I_{k}} p_{j}c_{j,i}\tilde{q}_{j} = \tilde{q}_{i}, \quad i = 1, ..., K, \]

which can in turn be expressed as

\[ \sum_{j \in I_{k}} c_{j,i}p_{j}\tilde{q}_{j} = p_{k}\tilde{q}_{i}, \quad i = 1, ..., K. \tag{17} \]

The manifestation of (17) in the matrix form is

\[ \mathbf{C} (p \otimes \mathbf{q}) = (p \otimes \mathbf{q}) \]

where \( \otimes \) denotes the Hadamard (element-wise) product and

\[ \mathbf{p} = [p_{1}, ..., p_{k}]^T \in \mathbb{R}^{K \times 1} \]

The equation (18) suggests that

\[ \mathbf{q} = \mathbf{p} \otimes \mathbf{q} \]

or

\[ \mathbf{q} = \mathbf{p} \otimes \mathbf{p} \]

where \( \mathbf{q} \) is the left Perron eigenvector of \( \mathbf{C} \) and \( \otimes \) denotes the element-wise division. Therefore, with any given \( \mathbf{C} \) and \( \mathbf{p} \), we may choose \( \mathbf{C} \) of the RC-DLMS algorithm so that its left Perron eigenvector is

\[ \mathbf{q} = \mathbf{p} \otimes \mathbf{q}. \]

With such choice, we have

\[ \mathbf{q} = \mathbf{p} \otimes \mathbf{q} \otimes \mathbf{p} = \mathbf{q} \]

hence \( \eta_k \approx \tilde{\eta}_k \), i.e., we can expect the RC-DLMS algorithm to perform very close to the DLMS algorithm. To construct \( \mathbf{C} \) such that it has the left Perron eigenvector \( \mathbf{q} = \mathbf{p} \otimes \mathbf{q} \), we can utilize the Hastings’ rule [30], [31] and compute the weights via
\[ c_{k,l} = \begin{cases} \frac{p_k^{-1} q_k^{-1}}{\max\{ (d_k + 1) p_k^{-1} q_k^{-1}, (d_l + 1) p_l^{-1} q_l^{-1} \} } & l \in \mathcal{N}_k \\ 1 - \sum_{m \in \mathcal{N}_k} c_{k,m} & l = k. \end{cases} \]  

(19)

V. SIMULATIONS

We consider an adaptive network of \( K = 20 \) nodes that are arbitrarily connected to each other and each node has between one to seven neighbors. The topology of the network is depicted in Fig. 2(a). The nodes collectively identify a parameter vector of length \( L = 4 \).

The regressor at each node is zero-mean multivariate Gaussian with an arbitrary covariance matrix. The additive noise at each node is also zero-mean Gaussian. The regressors and the noise are temporally and spatially independent of each other. In Fig. 2(b), we show the trace of the regressor covariance matrix and the variance of the noise at each node. We obtain the experimental results by taking the ensemble average over \( 10^5 \) independent trials and the steady-state quantities by averaging over 500 steady-state values. We also use the same step-size, denoted by \( \mu \), in all nodes. In the RC-DLMS algorithm, we determine the number of neighbors with which each node communicates at each iteration to receive their intermediate estimates via

\[ m_k = \min(M, d_k) \]

where \( 0 \leq M \leq N \leq d_k \) specifies the maximum number of consulted neighbors of every node at each iteration.

In Fig. 3, we plot the time-evolution of the network MSD of the RC-DLMS algorithm for different values of \( M \) when \( \mu = 0.01 \). We use the relative-degree weights [1] for \( \mathbf{C} \) in the consultation phase:

\[ c_{k,l} = \frac{d_l + 1}{\sum_{m \in \mathcal{N}_k \setminus \{k\}} (d_m + 1)}. \]

Note that the RC-DLMS algorithm becomes the non-cooperative LMS algorithm when \( M = 0 \) and the DLMS algorithm when \( M = 7 \). In Fig. 4, we compare the theoretical and experimental values of the steady-state network MSD of the RC-LMS algorithm for different values of \( M \) and \( \mu \). In Fig. 5, we compare the theoretical and experimental values of the steady-state MSD of all nodes for different values of \( M \) when \( \mu = 0.01 \). We observe in Fig. 3-5 that the RC-DLMS algorithm provides an effective trade-off between performance and communication cost. Moreover, there is a good agreement between the theoretical and experimental steady-state MSD values for a wide range of \( \mu \) and \( M \).

In Fig. 6, we compare the performance of the single-link diffusion LMS (SL-DLMS) [12], probabilistic diffusion LMS (P-DLMS) [10], RC-DLMS, and Diffusion LMS algorithms by plotting their network MSDs versus time in Fig. 6 and giving their steady-state MSDs at all nodes in Fig. 7. For the RC-DLMS algorithm, we set \( M = 1 \), which means that each node receives the intermediate estimate of only one of its neighbors at each iteration. For the P-DLMS algorithm, we select the probability of activeness of any link connecting node \( k \) to one of its neighbors as

\[ \hat{p}_k = \frac{1}{d_k}. \]

We use the relative-degree weights for the combination matrix \( \hat{\mathbf{C}} \) of the P-DLMS and DLMS algorithms. For the RC-DLMS algorithm, we consider both cases of having \( \mathbf{C} = \hat{\mathbf{C}} \) and constructing \( \mathbf{C} \) according to (19) for the given \( \hat{\mathbf{C}} \) and \( \mathbf{p} \), which is

\[ \mathbf{p} = \begin{bmatrix} 1 \\ d_1 \\ \vdots \\ d_K \end{bmatrix}. \]

We also set \( \mu = 0.001 \) for the P-DLMS, RC-DLMS, and Diffusion LMS algorithms and adjust it in the SL-DLMS algorithm to make all algorithms converge at almost the same speed. Figs. 6 and 7 show that the RC-DLMS algorithm outperforms the SL-DLMS and P-DLMS algorithms. Moreover, we observe that using the combination weights (19), the RC-DLMS algorithm can perform very close to the DLMS algorithm even when only a single neighbor of each node is consulted at each iteration.

In Fig. 8, we show the number of active links at each iteration for a single run of the P-DLMS and RC-DLMS algorithms. Note that there are a total of 80 unidirectional communication links in the considered network. The RC-DLMS algorithm utilizes 20 links at every iteration while the network utilization of the P-DLMS algorithm varies in time, ranging from 7 to 37 links in the experiment of Fig. 8. Nonetheless, in this scenario, the P-DLMS algorithm uses 20 links per iteration on average.

VI. CONCLUSION

We presented a reduced-communication diffusion LMS (RC-DLMS) algorithm for distributed adaptive parameter estimation. This algorithm enables reduced internode communications by allowing each node to receive the intermediate estimates of a subset of its neighbors at every iteration. Consequently, it offers an effective trade-off between estimation performance and communication cost. Employing the energy conservation principle, we analyzed the convergence performance of the proposed RC-DLMS algorithm and predicted its steady-state mean-square deviation. We also showed that a proper choice of the combination weights can make the performance of the RC-DLMS algorithm very close to that of the diffusion LMS algorithm, albeit for sufficiently small step-sizes. This is notably achieved with any non-zero number of neighbors consulted at each node and each iteration of the RC-DLMS algorithm.

REFERENCES


An adaptive network with 10 nodes and 20 unidirectional communication links. The active links are highlighted in four consecutive iterations of the RC-DLMS algorithm. At each time instant, each node only receives the intermediate estimate of one of its neighbors.

(a) The topology of the considered adaptive network. (b) Trace of the regressor covariance matrix and variance of the noise at each node.
Fig. 3. Network MSD curves of the RC-DLMS algorithm with different values of $M$ when $\mu = 0.01$.

Fig. 4. Theoretical and experimental steady-state network MSDs of the RC-DLMS algorithm versus $M$ for different values of $\mu$.

Fig. 5. Theoretical and experimental steady-state MSDs of the RC-DLMS algorithm at each node for different values of $M$ when $\mu = 0.01$.

Fig. 6. Network MSD curves of different algorithms when $M = 1$ and $\mu = 0.001$.

Fig. 7. Steady-state MSDs of different algorithms at each node when $M = 1$ and $\mu = 0.001$.

Fig. 8. Communication network utilization in a single run of the P-DLMS and RC-DLMS algorithms when $M = 1$ and $\beta_k = 1/d_k$. 