Distributed sparse diffusion estimation with reduced communication cost

Hamid Shiri[1] ✉, *Mohammad Ali Tinati*[1], *Marian Codreanu*[2], *Ghanbar Azarnia*[1]
[1]*Department of Electrical and Computer Engineering, University of Tabriz, Tabriz 51666, Iran*
[2]*Center for Wireless Communications, University of Oulu, Oulu 90014, Finland*
email: hshiri@tabrizu.ac.ir

Abstract: The issue considered in the current study is the problem of adaptive distributed estimation based on diffusion strategy which can exploit sparsity in improving estimation error and reducing communications. It has been shown that distributed estimation leads to a good performance in terms of the error value, convergence rate, and robustness against node and link failures in wireless sensor networks. However, the main focus of many works in the field of distributed estimation research is on convergence speed and estimation error, neglecting the fact that communications among the nodes require a lot of transmissions. In this work, the focus is on a solution based on sparse diffusion least mean squares (LMS) algorithm, and a new version of sparse diffusion LMS algorithm is proposed which takes both communications and error cost into account. Also, the computation complexity and communication cost for every node of the network, as well as performance analysis of the proposed strategy, is provided. The performance of the proposed method in comparison with the existing methods is illustrated by means of simulations in terms of computational and communicational cost, and flexibility to signal changes.

1 Introduction

In this paper, we consider the problem of distributed diffusion estimation over an adaptive network [1] which can exploit sparsity in improving estimation error and reducing communications. In distributed adaptive network, the parameter values of interest are estimated using collaboration among the nodes, and the running estimation algorithm can easily adapt to the changes of network conditions quickly. Furthermore, the range of communications in the distributed adaptive network is small since the transmissions are allowed only between the neighbour nodes. Hence, the communication cost, bandwidth usage, and power consumption are lower than the centralised network. As a result, numerous applications, such as environmental monitoring, transportation, factory instrumentation, and new emerging usages in Internet of things can benefit from distributed estimation methods [2–4].

Consider a wireless sensor network (WSN) which is deployed in a field in order to estimate an interested unknown parameter vector $\boldsymbol{\omega}_0$ of size $M \times 1$ by allowing the neighbour nodes to share data and collaborate with each other. We assume that the vector $\boldsymbol{\omega}_0$ is sparse with $K \ll M$ non-zero coefficients. There are numerous researches on estimating the vector $\boldsymbol{\omega}_0$ using various kinds of collaboration among the nodes. Recently, a lot of researches [1, 5–7] have focused on this issue by using a diffusion mode of cooperation among the nodes. However, the number of communications can be reduced by using probabilistic diffusion mode of cooperation, where each node is allowed to communicate only with a subset of its neighbours [2]. Another typical mode of cooperation between nodes is incremental, where each node communicates only with its direct neighbour at each time instant over a cyclic path in the network [2]. These modes of cooperation are depicted in Fig. 1. However, in order to avoid the very costly processing of finding cyclic paths covering the whole network, which is sensitive to link and node failures, only diffusion mode of cooperation is considered. Moreover, various adaptation algorithms have been proposed to estimate the parameter vector $\boldsymbol{\omega}_0$. Diffusion least mean squares (LMS) [1], diffusion recursive least-squares (RLS) [8], incremental affine projection algorithm [9], and diffusion affine projection algorithm [6] are some of the researches performed in adaptive estimations networks.

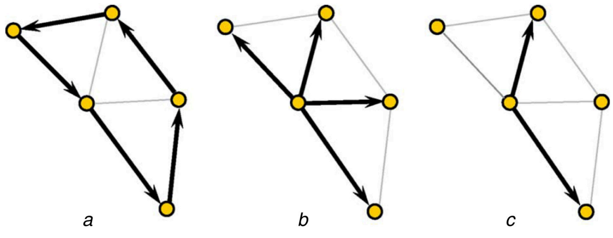

Fig. 1 *Modes of cooperation [2]*
(a) Incremental, ***(b)*** Diffusion, ***(c)*** Probabilistic diffusion

Additionally, when the intended parameter vector $\boldsymbol{\omega}_0$ is sparse, which is the case in many applications of distributed estimation, one can utilise this to improve the estimation error of adaptive algorithms as in [7, 10]. Knowledge about the sparsity is mainly used in compressed sensing (CS) to reconstruct the signal of interest [11]. Focus of most initial CS recovery solutions is on the batch methods, and the node(s) should recover the sparse signal utilising a collection of fixed number of measurements. However, batch algorithms are off-line, slow, and require high memory capacity. Therefore, many adaptive (on-line) methods have been proposed such that they are able to track changes of data over time [6, 7, 12–14]. These methods have improved the estimation performance of the distributed adaptive filtering for the purpose of recovering the sparse vector $\boldsymbol{\omega}_0$ but suffer from high communication cost of the diffusion strategy. As mentioned earlier, one method to alleviate this is to use probabilistic diffusion strategy. Although this strategy can reduce communication cost, it may lose the opportunity of using innovative measurements.

In this paper, the distributed diffusion estimation problem is considered which uses sparsity of parameter vector $\boldsymbol{\omega}_0$ for improving estimation error and reducing communications. The main contribution of this paper is to reduce communication cost at the time of estimating a sparse parameter vector in WSNs. First, a method to select and transmit the data necessary to recover the sparse vector $\boldsymbol{\omega}_0$ is proposed; so, the data transmission cost is reduced in the network. Then, the cost function, which is dependent on the proposed method for data transmission, is developed. Although the proposed method demands extra processing to find the proper data to be transmitted, additionally it is proposed to benefit from the wisely selected data to reduce processing cost, which can be interpreted as compensation for the extra amount of processing required for the proposed method. Next, the complexity of different operations needed to run the algorithm at every node of the network is computed. Then, the convergence condition, as well as the approximate steady-state Mean square deviation (MSD) level, is computed for the proposed algorithm. Finally, the performance of the proposed method is illustrated through simulations and compared with the well-known sparse diffusion LMS algorithm [7] which has low computational requirements.

We use the following general notations throughout the paper: $(\cdot)^*$ is the complex conjugate; $(\cdot)^{\mathrm{T}}$ is the transpose of a vector or matrix; $(\cdot)^{\mathrm{H}}$ is the Hermitian transpose; $(\cdot)^{(*)}$ indicates the solution to a problem; $\mathbb{E}$ is the expectation operator; the symbol $\boldsymbol{I}_N$ is the $N \times N$ identity matrix; $\mathbb{1}$ is $N \times 1$ vector with unit entries; $\|\cdot\|_0$, $\|\cdot\|_1$, and $\|\cdot\|$ are the l_0-norm, l_1-norm, and l_2-norm, respectively; $\lambda_{\max}(\cdot)$ is the largest eigenvalue of a matrix; $\mathrm{tr}(\cdot)$ is the trace of a matrix; $\mathrm{rank}(\cdot)$ is the rank of a matrix; $\otimes$ is the Kronecker product; $\mathrm{diag}\{\cdots\}$ is a diagonal matrix where its diagonal is formed from its arguments; $\mathrm{vec}\{\cdot\}$ is the column vector obtained by stacking the columns of the input matrix on top of each other; and $\mathbb{I}(\cdot)$ is the indicator function, where it returns 1 and 0 for non-zero and zero elements, respectively. We use boldface lower case letters for the vectors and boldface upper case letters for the matrices. Other symbols and notations will be explained in their context throughout the paper.

The remainder of this paper is organised as follows. In Section 2, the estimation problem is described and the general class of the distributed diffusion LMS algorithm is explained briefly. In Section 3, the proposed strategy to reduce amount of data which is to be transmitted is explained. In Section 4, the computational complexity as well as the required packet length to be broadcast are computed. In Section 5, the analysis of proposed distributed algorithm with reduced communication cost is studied. In Section 6, simulation results and comparisons are included. Finally in Section 7, conclusions are presented and some possible future extensions are pointed out.

2 Description of sparse diffusion LMS strategy

Consider a WSN with N nodes distributed over a given geographical area in order to estimate unknown sparse parameter vector

$$\boldsymbol{\omega}_0 = [\omega_{0,1}, \ldots, \omega_{0,M}]^{\mathrm{T}}. \tag{1}$$

It is assumed that the nodes in the network are connected to at least one other node in the network with a point-to-point link; in WSNs, two nodes are connected if they are in the transmission range of each other. In addition, every two nodes are assumed to be connected by some path in the network. These mean that the network is assumed to be partially connected.

At every time instant i, each node k takes a noisy scalar measurement $d_k(i)$ of the vector $\boldsymbol{\omega}_0$ as

$$d_k(i) = \boldsymbol{\omega}_0^{\mathrm{H}} \boldsymbol{u}_k(i) + v_k(i), \tag{2}$$

where $\boldsymbol{u}_k(i)$ is the $M \times 1$ random regression (or input) signal vector, and $v_k(i)$ is the scalar additive Gaussian noise with zero mean and variance $\sigma_{v,k}^2$. The cooperative sparse estimation problem can be stated as the distributed minimisation of the cost function [7]

$$J^{\mathrm{glob}}(\boldsymbol{\omega}) \triangleq \sum_{k=1}^{N} \mathbb{E}\left[|d_k(i) - \boldsymbol{\omega}^{\mathrm{H}} \boldsymbol{u}_k(i)|^2\right] + \rho f(\boldsymbol{\omega}), \tag{3}$$

where $\boldsymbol{\omega}$ is a $M \times 1$ vector, $f(\boldsymbol{\omega})$ is a sparsity promoting penalty function, and $\rho > 0$ is a regularisation parameter. Two common choices for function $f(\boldsymbol{\omega})$ are l_1-norm penalty $\|\boldsymbol{\omega}\|_1$, leading to zero attracting (ZA) update algorithms, and log-sum penalty $\sum_{j=1}^{m} \log(\varepsilon + |\boldsymbol{\omega}_j|)$, leading to reweighed ZA (RZA) update algorithms [15].

According to [1, 7], two kinds of adaptive diffusion algorithms can be developed based on the cost function (3), adapt-then-combine (ATC) and combine-then-adapt (CTA). However, we cannot possibly rehash and explain the main technical steps from [7] in this paper, because of space limitations. So, only the final resulted algorithms based on ATC and CTA strategies are represented.

ATC [7]: This strategy, as its name suggests, consists of two steps at every iteration i of the algorithm. The first step is the adaptation step, where each node $k = 1, \ldots, N$ aggregates the measurements $\{d_l(i), \boldsymbol{u}_l(i)\}$ from all of its neighbour nodes $l \in \mathcal{N}_k$, where $\mathcal{N}_k$ is the set of neighbour nodes of node k including node k itself. Then, each node k uses an adaptive algorithm such as LMS and RLS to obtain a local estimate $\boldsymbol{\psi}_k(i)$ of vector $\boldsymbol{\omega}_0$, i.e.

$$\begin{aligned}\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{u}_k(i) e_l^*(i) \\ - \mu_k \rho\, \partial f(\boldsymbol{\omega}_k(i-1)),\end{aligned} \tag{4}$$

where the LMS adaptive algorithm is chosen here and the strategy is called LMS-ATC, $\boldsymbol{\omega}_k(i-1)$ is the estimate vector of iteration $i-1$, index l indicates the neighbour node l linked to node k, the scalar value $e_l(i) = d_l(i) - \boldsymbol{\omega}_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$ denotes the error corresponding to the measurement $\{d_l(i), \boldsymbol{u}_l(i)\}$ received from neighbour $l \in \mathcal{N}_k$, vector $\partial f(\boldsymbol{x})$ is a sub-gradient of function f at point $\boldsymbol{x}$, the small positive coefficient $\{\mu_k\}$ is the step size of LMS algorithm, and the real non-negative weighting coefficients $\{c_{lk}\}$ satisfy

$$c_{lk} = 0 \quad \text{if } l \notin \mathcal{N}_k, \qquad \sum_{l \in \mathcal{N}_k} c_{lk} = 1, \tag{5}$$

which determine how much the measurements $\{d_l(i), \boldsymbol{u}_l(i)\}$ of nodes $l \in \mathcal{N}_k$ participate in the algorithm running at each node k.

The second step of the algorithm running at the iteration i begins when the nodes completed the adaptation step. This step is called the combination step, and each node k combines the collected local estimates $\psi_l(i)$ of all its neighbour nodes $l \in \mathcal{N}_k$ to generate the estimate $\boldsymbol{\omega}_k(i)$ of vector $\boldsymbol{\omega}_0$, i.e.

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} a_{lk} \boldsymbol{\psi}_l(i), \tag{6}$$

where the real non-negative weighting coefficients $\{a_{lk}\}$ satisfy

$$a_{lk} = 0 \quad \text{if } l \notin \mathcal{N}_k, \qquad \sum_{l \in \mathcal{N}_k} a_{lk} = 1, \tag{7}$$

which can be calculated through the uniform, the Metropolis, the Laplacian, or the relative degree rules [1]. Therefore, the sparse diffusion LMS-ATC strategy is written as

Sparse diffusion LMS-ATC strategy

Start with $\{\boldsymbol{\omega}_l(-1) = 0\}$ for all $l = 1, \ldots, N$

For each iteration $i \geq 0$, each node k

performs the update:

$$\begin{aligned}\boldsymbol{\psi}_k(i) &= \boldsymbol{\omega}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{u}_k(i) e_l^*(i) \\ &\quad - \mu_k \rho\, \partial f(\boldsymbol{\omega}_k(i-1)) \qquad \text{(Adaptation)}\end{aligned} \tag{8a}$$

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} a_{lk} \boldsymbol{\psi}_l(i) \qquad \text{(Combination)} \tag{8b}$$

CTA [7]: This strategy operates reversely, i.e. the combination step is performed before adaptation step at each iteration i of the algorithm. At the first step of CTA strategy, each node $k = 1, \ldots, N$ collects and combines the previous estimates $\boldsymbol{\omega}_l(i-1)$ of its neighbours $l \in \mathcal{N}_k$ to generate a local estimate $\boldsymbol{\psi}_k(i-1)$ of vector $\boldsymbol{\omega}_0$, i.e.

$$\boldsymbol{\psi}_k(i-1) = \sum_{l \in \mathcal{N}_k} a_{lk}\boldsymbol{\omega}_l(i-1), \tag{9}$$

where the real non-negative weighting coefficients $\{a_{lk}\}$ satisfy (7).

In the second step of the algorithm, each node k updates its local estimate $\boldsymbol{\psi}_k(i-1)$ by an adaptive algorithm, using aggregated measurements $\{d_l(i), \boldsymbol{u}_l(i)\}$ from its neighbours $l \in \mathcal{N}_k$, to obtain the estimate $\boldsymbol{\omega}_k(i)$ of vector $\boldsymbol{\omega}_0$, i.e.

$$\begin{aligned}\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk}\boldsymbol{u}_k(i)e_l^*(i) \\ -\mu_k\rho\,\partial f(\boldsymbol{\psi}_k(i-1)),\end{aligned} \tag{10}$$

where again LMS adaptive algorithm is chosen for the adaptation and the strategy is called LMS-CTA, the scalar value $e_l(i) = d_l(i) - \boldsymbol{\psi}_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$ denotes the error corresponding to the measurement $\{d_l(i), \boldsymbol{u}_l(i)\}$ received from neighbour $l \in \mathcal{N}_k$, the small positive coefficient $\{\mu_k\}$ is the step size of LMS algorithm, and the real non-negative weighting coefficients $\{c_{lk}\}$ satisfy (5). Thus, the sparse diffusion LMS-CTA strategy is written as

Sparse diffusion LMS-CTA strategy

Start with $\{\boldsymbol{\omega}_l(-1) = 0\}$ for all $l = 1, \dots, N$

For each iteration $i \geq 0$, each node k

performs the update:

$$\boldsymbol{\psi}_k(i-1) = \sum_{l \in \mathcal{N}_k} a_{lk}\boldsymbol{\omega}_l(i-1) \quad \text{(Combination)} \tag{11a}$$

$$\begin{aligned}\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk}\boldsymbol{u}_k(i)e_l^*(i) \\ -\mu_k\rho\,\partial f\left(\boldsymbol{\psi}_k(i-1)\right) \quad \text{(Adaptation)}\end{aligned} \tag{11b}$$

In order to save the space of the paper, instead of writing and explaining everything for each strategy separately, one general class of diffusion adaptation is formed which includes both the ATC and CTA strategies as special cases [1]. Therefore, the general class of distributed sparse LMS diffusion is written as

General class of distributed sparse LMS diffusion

Start with $\{\boldsymbol{\omega}_l(-1) = 0\}$ for all $l = 1, \dots, N$

For each iteration $i \geq 0$, each node k

performs the update:

$$\boldsymbol{\phi}_k(i-1) = \sum_{l \in \mathcal{N}_k} a_{1,lk}\boldsymbol{\omega}_l(i-1) \quad \text{(Combination-I)} \tag{12a}$$

$$\begin{aligned}\boldsymbol{\psi}_k(i) = \boldsymbol{\phi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk}\boldsymbol{u}_k(i)e_l^*(i) \\ -\mu_k\rho\,\partial f\left(\boldsymbol{\phi}_k(i-1)\right) \quad \text{(Adaptation)}\end{aligned} \tag{12b}$$

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} a_{2,lk}\boldsymbol{\psi}_l(i), \quad \text{(Combination-II)} \tag{12c}$$

where $\boldsymbol{\omega}_k(i)$ is an estimate of vector $\boldsymbol{\omega}_0$ at iteration i and at node k, $\boldsymbol{\phi}_k(i)$ and $\boldsymbol{\psi}_k(i)$ are $M \times 1$ local estimate vectors, index l indicates the neighbour node l linked to node k, $e_l(i) = d_l(i) - \boldsymbol{\phi}_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$ is the error value corresponding to measurement $\{d_l(i), \boldsymbol{u}_l(i)\}$ received from neighbour $l \in \mathcal{N}_k$, the small positive coefficient $\{\mu_k\}$ is the step size of LMS algorithm, and the real non-negative weighting coefficients $\{c_{lk}, a_{1,lk}, a_{2,lk}\}$ satisfy

$$c_{lk} = a_{1,lk} = a_{2,lk} = 0 \quad \text{if} \quad l \notin \mathcal{N}_k \tag{13a}$$

$$\mathbb{1}^{\mathrm{T}}\boldsymbol{C} = \mathbb{1}^{\mathrm{T}}, \quad \mathbb{1}^{\mathrm{T}}\boldsymbol{A}_1 = \mathbb{1}^{\mathrm{T}}, \quad \mathbb{1}^{\mathrm{T}}\boldsymbol{A}_2 = \mathbb{1}^{\mathrm{T}}, \tag{13b}$$

where $\boldsymbol{C}$, $\boldsymbol{A}_1$, and $\boldsymbol{A}_2$ are the $N \times N$ matrices with individual entries $\{c_{lk}, a_{1,lk}, a_{2,lk}\}$, respectively.

The sparse diffusion LMS-ATC algorithm can be obtained by substituting $\boldsymbol{A}_1 = \boldsymbol{I}_N$. Thus, the combination-I statement 12a becomes $\boldsymbol{\phi}_k(i-1) = \boldsymbol{\omega}_k(i-1)$, i.e. the combination-I statement is eliminated and only the adaptation 12b and combination-II 12c statements will remain. Similarly, the sparse diffusion LMS-CTA algorithm is obtained by substituting $\boldsymbol{A}_2 = \boldsymbol{I}_N$, which eliminates the combination-II statement 12c, i.e. $\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i)$, and only the combination-I 12a and adaptation 12b steps will be running.

The performance analysis of LMS-ATC and LMS-CTA algorithms has been well studied in [7]. However, these algorithms require that every node of the network transmits its measurement data and its (local) estimate vector at each iteration of the algorithm. The focus of the next section is on reducing the communications among the nodes assuming that the vector $\boldsymbol{\omega}_0$ is sparse.

3 Proposed strategy

In this section, the method to select only the necessary data to be shared with neighbours at each node is proposed; so the data transmission cost is reduced in the distributed network. As can be seen from 12, at each iteration i of the algorithm, each node k transmits its measurement $\{d_k(i), \boldsymbol{u}_k(i)\}$ as well as its local estimate $\boldsymbol{\psi}_k(i)$ for the ATC strategy or its estimate $\boldsymbol{\omega}_k(i-1)$ for the CTA strategy. When the signal to be estimated is long, use of algorithm 12 demands a great amount of data transmissions among the nodes to obtain a common and feasible estimation of the desired signal $\boldsymbol{\omega}_0$. Furthermore, the communication processes consume a great deal of time and energy resources in wireless networks. Therefore, we wish to reduce communication cost of the network. However, if the desired signal, which should be estimated, is sparse, the burden of long packet communications at the nodes can be reduced. Many different strategies can be proposed to reduce the communication cost. So the focus of this section is on a simple applicable method which exploits the knowledge about sparsity of the signal.

Inspired by the partial update adaptive filters [16] and distributed CS [17], a method to reduce the communication cost in a distributed estimation network is proposed. In partial update adaptive filters, some important taps of the filters which play an important role in adapting the elements are selected to be updated. However, here the most important elements is selected to be transmitted in both adaptation and combination steps. Therefore, our goal is reducing communication cost of the algorithm by transmitting the important taps of both the regression signals and local estimate vectors at every iteration such that each node k of the network is able to obtain an estimate vector

$$\boldsymbol{\omega}_k(i) = [\omega_{k1}(i), \dots, \omega_{kM}(i)]^{\mathrm{T}} \tag{14}$$

of the unknown sparse vector $\boldsymbol{\omega}_0$.

During running an estimation algorithm, there are two possible states for the nodes. First one is the uncertainty state when the distance between $\boldsymbol{\omega}_k(i)$ and $\boldsymbol{\omega}_0$ is high, i.e. the algorithms is in its initial steps and has not converged yet. At this step there is little information about the structure of $\boldsymbol{\omega}_0$, and therefore, it is difficult to decide which elements are more important than others and should be selected for transmission. After some iterations, the algorithm begins to converge to the desired $\boldsymbol{\omega}_0$, which is called certainty state in this paper; so, the nodes can find and send the elements which participate more in the estimation process. This is the basis of our proposed method. Here, certainty state is explained first.

3.1 Certainty state – partial transmission

As stated above, after some iterations, the algorithm starts to converge and the errors get smaller. Finally, the errors reach to a point where it is safe to say that the estimated vector at each node k is relatively close to the optimal vector $\boldsymbol{\omega}_0$.

Convergence at node k begins when the error value of estimated vector is below a predefined error constraint γ_k. In other words, convergence starts when the error value corresponding to both the data value $d_l(i)$ and input vector $\boldsymbol{u}_l(i)$ from node l as well as the estimated vector $\phi_l(i-1)$ of node k becomes smaller than γ_k, i.e.

$$|d_l(i) - \boldsymbol{\phi}_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)| \leq \gamma_k. \tag{15}$$

In its memory, each node l has only its own $\boldsymbol{\phi}_l(i-1)$, and not its neighbour's $\boldsymbol{\phi}_k(i-1)$. Since the data and estimated vectors are shared and combined in the network, their values get closer to each other when convergence occurs. Therefore, in the inequality (15) the vector $\boldsymbol{\phi}_k(i-1)$ is substituted with $\boldsymbol{\phi}_l(i-1)$ and the error value is compared with the constraint γ_l at node l before transmission. Then, the inequality (15) can be rewritten as

$$|d_l(i) - \boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)| \le \gamma_l\,. \tag{16}$$

When the node l is at this state, it interprets that the algorithm is converged enough and it can concentrate on reducing communication cost besides getting converged. So, it selects only $\tau^{(*)} \le M$ specific elements of the input vector $\boldsymbol{u}_l(i)$ to be transmitted and sets the rest to zero, i.e. it finds a vector $\boldsymbol{u}_l^{(\tau^{(*)})}(i)$ of the same size as vector $\boldsymbol{u}_l(i)$ with $\tau^{(*)}$ non-zero elements, and then transmits values and locations (or indexes) of the non-zero elements. The vector $\boldsymbol{u}_l^{(\tau^{(*)})}(i)$ should be computed in a way that the error which it results does not differ much with the error corresponding to the input vector $\boldsymbol{u}_l(i)$, i.e. it is limited to a constraint value $\gamma_l' = \gamma_l + \epsilon$, where ϵ is a small positive value. Let us define $\boldsymbol{S}_l$ at each node l as a diagonal $M \times M$ matrix whose diagonal elements are 1s or 0s and non-diagonal elements are 0s. Therefore, it is possible to find $\boldsymbol{u}_l^{(\tau)}(i) = \boldsymbol{S}_l\boldsymbol{u}_l(i)$ at each node l such that the error is limited to the constraint $\gamma_l' = \gamma_l + \epsilon$, i.e.

$$|d_l(i) - \boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)| \le \gamma_l + \epsilon\,. \tag{17}$$

A proper $\boldsymbol{S}_l$ to satisfy the inequality (17) (with a small ϵ value) is one such that $\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)$ contains most of the energy of $\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$, i.e.

$$\mathbb{E}\left[|\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)|^2\right] \ge \alpha_1\mathbb{E}\left[|\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)|^2\right]. \tag{18}$$

It is possible to rewrite the left hand side of the inequality (18) as

$$\begin{aligned}
&\mathbb{E}\left[|\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)|^2\right]\\
&\quad= \mathbb{E}\left[\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)\boldsymbol{u}_l^{\mathrm{H}}(i)\boldsymbol{S}_l\boldsymbol{\phi}_l(i-1)\right]\\
&\quad\overset{(a)}{=} \mathbb{E}\left[\mathrm{tr}\left(\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\boldsymbol{u}_l(i)\boldsymbol{u}_l^{\mathrm{H}}(i)\boldsymbol{S}_l\boldsymbol{\phi}_l(i-1)\right)\right]\\
&\quad\overset{(b)}{=} \mathbb{E}\left[\mathrm{tr}\left(\boldsymbol{u}_l(i)\boldsymbol{u}_l^{\mathrm{H}}(i)\boldsymbol{S}_l\boldsymbol{\phi}_l(i-1)\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\right)\right]\\
&\quad\overset{(c)}{=} \mathrm{tr}\left(\mathbb{E}\left[\boldsymbol{u}_l(i)\boldsymbol{u}_l^{\mathrm{H}}(i)\boldsymbol{S}_l\boldsymbol{\phi}_l(i-1)\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\right]\right)\\
&\quad= \mathrm{tr}\left(\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)}\boldsymbol{R}_{\boldsymbol{\phi},l}^{(\tau)}\right),
\end{aligned} \tag{19}$$

where $\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)} = \mathbb{E}\left[\boldsymbol{u}_l(i)\boldsymbol{u}_l^{\mathrm{H}}(i)\boldsymbol{S}_l\right]$, $\boldsymbol{R}_{\boldsymbol{\phi},l}^{(\tau)} = \mathbb{E}\left[\boldsymbol{\phi}_l(i-1)\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l\right]$, and τ is the rank (or number of non-zero elements) of diagonal matrix $\boldsymbol{S}_l$. Furthermore, the following facts are used in (19): (a) the trace of a scalar value x is equal to x itself, i.e. $\mathrm{tr}(x) = x$; (b) the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be switched inside the trace operator, i.e. $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$; (c) the trace and expectation are linear operators, hence $\mathbb{E}(\mathrm{tr}(\boldsymbol{A})) = \mathrm{tr}(\mathbb{E}(\boldsymbol{A}))$. Then, in order to select minimum number of transmissions, each node l should choose the matrix $\boldsymbol{S}_l$ with minimum rank, i.e.

$$\begin{aligned}
\boldsymbol{S}_l^{(\tau^{(*)})} &= \arg\min_{\boldsymbol{S}_l} \quad \mathrm{rank}(\boldsymbol{S}_l)\\
\text{s.t.}\quad &\mathrm{tr}\left(\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)}\boldsymbol{R}_{\boldsymbol{\phi},l}^{(\tau)}\right) \ge \alpha_1\mathrm{tr}\left(\boldsymbol{R}_{\boldsymbol{u},l}^{(M)}\boldsymbol{R}_{\boldsymbol{\phi},l}^{(M)}\right),
\end{aligned} \tag{20}$$

where s.t. stands for the term ‘subject to’, $\tau^{(*)}$ is the rank of the matrix $\boldsymbol{S}_l^{(\tau^{(*)})}$, and $\boldsymbol{S}_l^{(\tau^{(*)})}$ is the selected matrix by which the $\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{S}_l^{(\tau^{(*)})}\boldsymbol{u}_l(i)$ contains most of the energy of $\boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$.

However, each node l wishes to find an adaptive implementation for problem (20), i.e. it tries to find the instantaneous approximation $\boldsymbol{S}_l(i)$ to $\boldsymbol{S}_l$. So, let us represent the input vector $\boldsymbol{u}_l(i)$, and local estimate vector $\boldsymbol{\phi}_l(i-1)$ as

$$\boldsymbol{u}_l(i) = [u_{l1}(i), \ldots, u_{lM}(i)]^{\mathrm{T}} \tag{21a}$$

$$\boldsymbol{\phi}_l(i-1) = [\phi_{l1}(i-1), \ldots, \phi_{lM}(i-1)]^{\mathrm{T}}, \tag{21b}$$

where $u_{lj}(i)$ and $\phi_{lj}(i-1)$ for $j = 1, \ldots, M$ are elements of vectors $\boldsymbol{u}_l(i)$ and $\boldsymbol{\phi}_l(i-1)$, respectively, and assume the input vectors $\boldsymbol{u}_l(i)$ are zero-mean Gaussian distribution with a diagonal covariance matrices as

$$\boldsymbol{R}_{\boldsymbol{u},l}^{(M)} = \begin{bmatrix} \sigma_{u,l1}^2 & 0 & \cdots & 0\\ 0 & \sigma_{u,l2}^2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \sigma_{u,lM}^2 \end{bmatrix}, \tag{22}$$

where each variance $\sigma_{u,lj}^2$ corresponds to each element $u_{lj}(i)$; hence, it is possible to rewrite $\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)}$ in (19) with instantaneous $\boldsymbol{S}_l(i)$ as

$$\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)} = \boldsymbol{R}_{\boldsymbol{u},l}^{(M)}\boldsymbol{S}_l(i)\,. \tag{23}$$

Therefore, an adaptive implementation can be obtained by replacing the second-order moments by local instantaneous approximations, as follows:

$$\sigma_{u,lj}^2 \approx |u_{lj}(i)|^2\,. \tag{24}$$

Hence, the right hand side of constraint in (20) can be rewritten as

$$\mathrm{tr}\left(\boldsymbol{R}_{\boldsymbol{u},l}^{(M)}\boldsymbol{R}_{\boldsymbol{\phi},l}^{(M)}\right) = \sum_{j=1}^{M} |u_{lj}(i)\boldsymbol{\phi}_{lj}(i-1)|^2\,. \tag{25}$$

Similar expression can be written for the left hand side of (20) as

$$\mathrm{tr}\left(\boldsymbol{R}_{\boldsymbol{u},l}^{(\tau)}\boldsymbol{R}_{\boldsymbol{\phi},l}^{(\tau)}\right) = \sum_{j \in J} |u_{lj}(i)\phi_{lj}(i-1)|^2, \tag{26}$$

where J is the set of indexes as $J = \{j : [\boldsymbol{S}_l(i)]_{jj} = 1\}$. Therefore, problem (20) can be rewritten as

$$\begin{aligned}
\boldsymbol{S}_l^{(\tau^{(*)})}(i) &= \arg\min_{\boldsymbol{S}_l(i)} \quad \mathrm{rank}(\boldsymbol{S}_l(i))\\
\text{s.t.}\quad &\sum_{j \in J} |u_{lj}(i)\phi_{lj}(i-1)|^2\\
&\ge \alpha_1 \sum_{j=1}^{M} |u_{lj}(i)\phi_{lj}(i-1)|^2\,.
\end{aligned} \tag{27}$$

Since at the certainty state the vector $\boldsymbol{\phi}_l(i-1)$ is at the vicinity of the optimal vector $\boldsymbol{\omega}_0$, some elements $\phi_{lj}(i-1)$ of $\boldsymbol{\phi}_l(i-1)$ corresponding to the zero elements of the vector $\boldsymbol{\omega}_0$ are closer to zero. Then, the corresponding terms $u_{lj}(i)\phi_{lj}(i-1)$ tend to be closer to zero. Furthermore, whenever the elements $u_{lj}(i)$ are small and multiplied by elements $\phi_{lj}(i-1)$, which results in a small value of $u_{lj}(i)\phi_{lj}(i-1)$, then the elements $u_{lj}(i)$ will not have much effect on the adaptation step. Therefore, the node can refrain from sending the corresponding elements. However, the elements $u_{lj}(i)\phi_{lj}(i-1)$ have large values only when both elements $u_{lj}(i)$ and $\phi_{lj}(i-1)$ have large values. Therefore, to solve problem (27) each node l sorts the elements $|u_{lj}(i)\phi_{lj}(i-1)|$ in a descending manner, i.e.

$$|u_{l\theta_1}(i)\phi_{l\theta_1}(i-1)| \ge \cdots \ge |u_{l\theta_M}(i)\phi_{l\theta_M}(i-1)|, \tag{28}$$

where θ_j is an index representing the $\theta_j(i)$th element of $\boldsymbol{u}_l(i)$. Then, it solves an equivalent version of the problem (27) as

$$\begin{aligned} \tau_l^{(*)}(i) = \arg \min_{\tau} \quad & \tau \\ \text{s.t.} \quad & \sum_{j=1}^{\tau} |u_{l\theta_j}(i)\phi_{l\theta_j}(i-1)|^2 \\ & \geq \alpha_1 \sum_{j=1}^{m} |u_{l\theta_j}(i)\phi_{l\theta_j}(i-1)|^2 . \end{aligned} \tag{29}$$

As a result, each node l should transmit the elements which possibly participates in the energy of product $\phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)$ as stated in the constraint of (20).

By solving for $\tau_l^{(*)}(i)$ in problem (29), node l should select and broadcast $\tau_l^{(*)}(i)$ number of elements of $\boldsymbol{u}_l(i)$ and their corresponding locations. At first, each node l computes the following vectors:

$$\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i) = \left[u_{l\theta_1}(i), u_{l\theta_2}(i), \ldots, u_{l\theta_{\tau^{(*)}}}(i)\right]^{\mathrm{T}} \tag{30a}$$

$$\boldsymbol{\theta}_l^{(\tau^{(*)})}(i) = [\theta_1(i), \theta_2(i), \ldots, \theta_{\tau^{(*)}}(i)]^{\mathrm{T}} \tag{30b}$$

$$\bar{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i) = [\theta_{\tau^{(*)}+1}(i), \ldots, \theta_M(i)]^{\mathrm{T}}, \tag{30c}$$

where $\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i)$ includes the selected elements of $\boldsymbol{u}_l(i)$ to be transmitted, $\boldsymbol{\theta}_l^{(\tau^{(*)})}(i)$ contains the corresponding (non-zero) location data, and $\bar{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)$ indicates the zero location data which is the complement of non-zero location data $\boldsymbol{\theta}_l^{(\tau^{(*)})}(i)$. Then, it should transmit data pair $\{\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$, where

$$\tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i) = \arg \min_{\boldsymbol{x} \in \left\{\boldsymbol{\theta}_l^{(\tau^{(*)})}(i), \bar{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\right\}} \| \boldsymbol{x} \|_0 \tag{31}$$

is the location data to be transmitted. In other words, when the size of non-zero location data vector $\boldsymbol{\theta}_l^{(\tau^{(*)})}(i)$ is smaller than $\tau_l^{(*)}(i) = M/2$, the whole size of data pair $\{\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$ is $2\tau_l^{(*)}(i)$ and hence communication is reduced. But whenever the size of non-zero location data $\boldsymbol{\theta}_l^{(\tau^{(*)})}(i)$ is greater than or equal to $M/2$, the size of data pair to be transmitted is M and no reduction in communication cost is obtained.

After transmitting the selected elements of $\boldsymbol{u}_l(i)$, node k constructs an estimation $\boldsymbol{u}_l^{(\tau^{(*)})}(i)$ of $\boldsymbol{u}_l(i)$ as a vector whose values at each location $\theta_j(i)$ are $u_{l\theta_j}(i)$, and rest of the elements are set to zero.

As sending a fraction of elements from node l causes change in the corresponding error at the node k, i.e. $e_l(i)$, we try to send measurements $d_l^{(\tau^{(*)})}(i)$ corresponding to reduced vector $\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i)$ and hence corresponding to $\boldsymbol{u}_{l,i}^{(\tau^{(*)})}$. To this end, the error $e_l(i)$ is rewritten as

$$\begin{aligned} e_l(i) &= d_l(i) - \phi_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i) \\ &\approx d_l(i) - \phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i) \\ &= d_l(i) - \phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l^{(\tau^{(*)})}(i) - \phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l^{(\tau')}(i) \\ &= d_l^{(\tau^{(*)})}(i) - \phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l^{(\tau^{(*)})}(i) \end{aligned} \tag{32}$$

where

$$\boldsymbol{u}_l(i) = \boldsymbol{u}_l^{(\tau^{(*)})}(i) + \boldsymbol{u}_l^{(\tau')}(i) \tag{33a}$$

$$d_l^{(\tau^{(*)})}(i) = d_l(i) - \phi_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l^{(\tau')}(i) . \tag{33b}$$

Therefore, the node l should send $d_l^{(\tau^{(*)})}(i)$ as the new measurement value besides the pair $\{\tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$. In other words, the data to be transmitted is $\{d_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$. However, computing this data demands extra processing at node l.

The reduced communication cost method is proposed at adaptation step of the algorithm. The wireless network can use the knowledge of sparsity of the signal to reduce communication cost at the combination step as well. Now we focus on combination step which occurs either before or after the adaptation step corresponding to CTA or ATC strategies, respectively.

First, we consider the ATC strategy where each node l, after performing adaptation step, prepares to send vectors $\boldsymbol{\psi}_l(i)$ to its neighbour k. But instead of $\boldsymbol{\psi}_l(i)$ itself, it finds a vector $\boldsymbol{\psi}_l^{(s)}(i)$ by selecting $s_l(i)$ largest elements of $\boldsymbol{\psi}_l(i)$ which contain at least α_2 per cent of the energy of vector $\boldsymbol{\psi}_l(i)$ and setting the rest equal to 0s, then puts its non-zero elements in vector $\tilde{\boldsymbol{\psi}}_l^{(s)}(i)$ to be transmitted. Therefore, in order to minimise the number of elements necessary to transmit to its neighbour nodes it should solve the following problem:

$$\begin{aligned} \boldsymbol{K}_l^{(s)}(i) = \arg \min_{\boldsymbol{K}} \quad & \| \mathbf{K}\boldsymbol{\psi}_l(i) \|_0 \\ \text{s.t.} \quad & \| \mathbf{K}\boldsymbol{\psi}_l(i) \|^2 \geq \alpha_2 \| \boldsymbol{\psi}_l(i) \|^2 , \end{aligned} \tag{34}$$

where $\boldsymbol{K}$ is a diagonal $M \times M$ matrix whose elements are 1s and 0s, and $\boldsymbol{\psi}_l^{(s)}(i) = \boldsymbol{K}_l^{(s)}(i)\boldsymbol{\psi}_l(i)$. Solving the problem (34) for $\boldsymbol{\psi}_l^{(s)}(i)$ is similar to solving the problem (20). Similar to the former explanations for reduced regressor vector transmission, each node l should transmit $\tilde{\boldsymbol{\psi}}_l^{(s)}(i)$ with the corresponding location data $\tilde{\boldsymbol{\theta}}_l^{(s)}(i)$. So, the data pair $\{\tilde{\boldsymbol{\psi}}_l^{(s)}(i), \tilde{\boldsymbol{\theta}}_l^{(s)}(i)\}$ is transmitted.

However, if the network is running with CTA strategy, the problem to reduce the communications at the combination step is

$$\begin{aligned} \boldsymbol{K}_l^{(s)}(i) = \arg \min_{\boldsymbol{K}} \quad & \| \boldsymbol{K}\boldsymbol{\omega}_l(i-1) \|_0 \\ \text{s.t.} \quad & \| \boldsymbol{K}\boldsymbol{\omega}_l(i-1) \|^2 \geq \alpha_2 \| \boldsymbol{\omega}_l(i-1) \|^2 , \end{aligned} \tag{35}$$

where $\boldsymbol{K}$ is a diagonal $M \times M$ matrix whose elements are 1s and 0s. Similarly, after solving the problem (35), the vector $\boldsymbol{\omega}_l^{(s)}(i-1) = \boldsymbol{K}_l^{(s)}(i)\boldsymbol{\omega}_l(i-1)$ can be obtained and the node l transmits the data pair $\{\tilde{\boldsymbol{\omega}}_l^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_l^{(s)}(i)\}$ to every neighbour k of node l, where the vector $\tilde{\boldsymbol{\omega}}_l^{(s)}(i-1)$ is the non-zero elements of $\boldsymbol{\omega}_l^{(s)}(i-1)$, and $\tilde{\boldsymbol{\theta}}_l^{(s)}(i)$ is the corresponding location data.

3.2 Uncertainty state

We define the uncertainty state as the state when the algorithm is not converged enough to use the reduced communication method explained. The uncertainty is generated when the problem (20) is not solvable, i.e. there is no matrix $\boldsymbol{S}_l$ such that the constraint in (20) hold. Since $\boldsymbol{S}_l$ is a diagonal matrix with elements 1s and 0s, this case happens only when the error at node l does not hold, i.e.

$$|d_l(i) - \boldsymbol{\phi}_l^{\mathrm{H}}(i-1)\boldsymbol{u}_l(i)| > \gamma_l . \tag{36}$$

When this happens, node l cannot use the proposed method explained at the certainty state, but if it wants to reduce transmission cost, it might use other methods. However, in order to maintain simplicity of the algorithm in this paper, no reduction in communications is considered in uncertainty state and $\tau_l^{(*)}(i) = M$ is set. Then, the node l transmits data $\{d_l^{(\tau^{(*)})}(i) = d_l(i),\ \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i) = \boldsymbol{u}_l(i),\ \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i) = \{\}\}$ at adaptation step. However, when the node l is in this state it will send

$\{\tilde{\boldsymbol{\psi}}_l^{(s)}(i) = \boldsymbol{\psi}_l(i),\ \tilde{\boldsymbol{\theta}}_l^{(s)}(i) = \{\}\}$ or $\{\tilde{\boldsymbol{\omega}}_l^{(s)}(i-1) = \boldsymbol{\omega}_l(i-1),\ \tilde{\boldsymbol{\theta}}_l^{(s)}(i) = \{\}\}$ at combination step of ATC or CTA strategies, respectively.

3.3 General class of diffusion estimations for reduced communication method

As explained earlier, each node receives data from its neighbours to do the adaptation or combination operation in the diffusion mode. We use the methodology similar to [7] to develop the adaptation and combination steps at the proposed reduced-communication method. We know from the literature that the term $f(\boldsymbol{\omega})$ in (3) is added to the cost function to attract the elements towards zero. We eliminate this term in our case since as the algorithm is converging, the nodes try to share and combine the data which contain the sparsity knowledge of vector $\boldsymbol{\omega}_0$ in its structure. Therefore, the proposed cost function with the reduced communication is

$$J^{\text{glob}}(\boldsymbol{\omega}) \triangleq \sum_{k=1}^{N} \mathbb{E}\left[|d_k^{(\tau^{(*)})}(i) - \boldsymbol{\omega}^{\mathrm{H}} \boldsymbol{u}_k^{(\tau^{(*)})}(i)|^2 \right]. \tag{37}$$

As a result, similar to [1], the general class of the proposed distributed sparse LMS diffusion algorithm with reduced communication cost can be written as

---

General class of the proposed distributed sparse diffusion LMS with reduced communication cost

---

Start with $\boldsymbol{\omega}_k(-1) = 0$ for all $k = 1, \dots, N$

For each iteration $i \ge 0$, each node k

performs the update:

$$\boldsymbol{\phi}_k(i-1) = \sum_{l \in \mathcal{N}_k} a_{1,lk} \boldsymbol{\omega}_l^{(s)}(i-1) \tag{38a}$$

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\phi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{u}_l^{(\tau^{(*)})}(i) \left(e_l^{(\tau^{(*)})}(i) \right)^* \tag{38b}$$

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} a_{2,lk} \boldsymbol{\psi}_l^{(s)}(i), \tag{38c}$$

---

where $e_l^{(\tau^{(*)})}(i) = d_l^{(\tau^{(*)})}(i) - \boldsymbol{\phi}_k^{\mathrm{H}}(i-1) \boldsymbol{u}_l^{(\tau^{(*)})}(i)$. As mentioned earlier, the ATC and CTA sparse diffusion LMS algorithms can be obtained by substituting $\boldsymbol{A}_1 = \boldsymbol{I}_N$ or $\boldsymbol{A}_2 = \boldsymbol{I}_N$, respectively. Furthermore, the performance of our proposed algorithm in comparison to the conventional LMS diffusion algorithm (3) will be illustrated in the simulations section. The proposed algorithm for communication cost reduction is depicted schematically in Figs. 2 and 3 for ATC and CTA strategies, respectively.

4 Computational complexity and communication cost

In this section, the computational complexity and communication cost of the proposed method is compared with the sparse diffusion LMS strategy (3). It should be noted that since each node k knows the non-zero locations of received vectors $\boldsymbol{u}_{l,i}^{(\tau^{(*)})}(i)$ (and $\boldsymbol{\psi}_l^{(s)}(i)$ or $\boldsymbol{\omega}_l^{(s)}(i-1)$), we propose to use this information to reduce the processing cost as well. For example, the product $\boldsymbol{\phi}_k^{\mathrm{H}}(i-1)\boldsymbol{u}_l^{(\tau^{(*)})}(i)$ in 38 can be calculated only by multiplying the $\tau_l^{(*)}(i)$ non-zero elements of $\boldsymbol{u}_l^{(\tau^{(*)})}(i)$ by the corresponding elements of $\phi_k(i-1)$ which requires $\tau_l^{(*)}(i)$ instead of M multiplications as well as $\tau_l^{(*)}(i) - 1$ instead of $M - 1$ additions.

To calculate the required computations and communications at each node k with $|\mathcal{N}_k|$ neighbours, where $|\mathcal{N}_k|$ is the cardinality of neighbour set $\mathcal{N}_k$, let us define the average $\tau_{\text{avg},k}^{(*)}(i)$ and $s_{\text{avg},k}(i)$ as

$$\tau_{\text{avg},k}^{(*)}(i) = \frac{1}{|\mathcal{N}_k|} \sum_{l \in \mathcal{N}_k} \tau_l^{(*)}(i) \tag{39a}$$

$$s_{\text{avg},k}(i) = \frac{1}{|\mathcal{N}_k|} \sum_{l \in \mathcal{N}_k} s_l(i). \tag{39b}$$

Then, it is possible to compute the number of operations, such as number of additions, multiplications, divisions, comparisons, and sorting, and length of data to be transmitted at each node k, as shown in Table 1.

It will be shown in the simulation section that the proposed method does not require a large processing cost for the nodes of the network. On the contrary, when the desired signal has zero elements, the proposed method might even decrease processing as well as communication cost after some iterations of the algorithm.

5 Performance analysis

In this section, the convergence condition and theoretical approximate MSD of the proposed strategy are obtained. The analyses of current section benefit from the studies on the LMS-based diffusion algorithm in [1, 7]. To this end, suppose that the nodes detect $s \ge K$ locations of a vector $\boldsymbol{\omega}_0$ as the support of vector $\boldsymbol{\omega}_0$. The matrix $\boldsymbol{K}_l^{(s)}(i)$ is assumed to be equal to $\boldsymbol{K}^{(s)}$ for all the nodes l in the network. The matrix $\boldsymbol{S}_l^{(\tau^{(*)})}(i)$ is dependent on both $\boldsymbol{\omega}_0$ and the input vector $\boldsymbol{u}_l(i)$, but for the simplicity of analyses, its dependence on input vector is ignored. Similarly, matrix $\boldsymbol{S}_l^{(\tau^{(*)})}(i)$ is assumed to be equal to $\boldsymbol{K}^{(s)}$ for all the nodes l in the network. Later in the simulation section, it will be shown that these assumptions do not change the convergence condition and will have very little effect on the theoretical MSD value. To begin the analyses, first the following error vectors are defined:

$$\tilde{\boldsymbol{\phi}}_{s,k}(i) \triangleq \boldsymbol{K}^{(s)}(\boldsymbol{\omega}_0 - \boldsymbol{\phi}_k(i)), \tag{40a}$$

$$\tilde{\boldsymbol{\psi}}_{s,k}(i) \triangleq \boldsymbol{K}^{(s)}(\boldsymbol{\omega}_0 - \boldsymbol{\psi}_k(i)), \tag{40b}$$

$$\tilde{\boldsymbol{\omega}}_{s,k}(i) \triangleq \boldsymbol{K}^{(s)}(\boldsymbol{\omega}_0 - \boldsymbol{\omega}_k(i)). \tag{40c}$$

Next, various quantities across all nodes of the network are collected into block vectors as

$$\tilde{\boldsymbol{\phi}}_s(i) \triangleq \text{col}\{\tilde{\boldsymbol{\phi}}_{s,1}(i), \dots, \tilde{\boldsymbol{\phi}}_{s,N}(i)\}, \tag{41a}$$

$$\tilde{\boldsymbol{\psi}}_s(i) \triangleq \text{col}\{\tilde{\boldsymbol{\psi}}_{s,1}(i), \dots, \tilde{\boldsymbol{\psi}}_{s,N}(i)\}, \tag{41b}$$

$$\tilde{\boldsymbol{\omega}}_s(i) \triangleq \text{col}\{\tilde{\boldsymbol{\omega}}_{s,1}(i), \dots, \tilde{\boldsymbol{\omega}}_{s,N}(i)\}. \tag{41c}$$

Also, the extended weighting matrices are defined as

$$\mathscr{A}_{s1} \triangleq \boldsymbol{A}_1 \otimes (\boldsymbol{I}_M \boldsymbol{K}^{(s)}), \tag{42}$$

$$\mathscr{A}_{s2} \triangleq \boldsymbol{A}_2 \otimes (\boldsymbol{I}_M \boldsymbol{K}^{(s)}), \tag{43}$$

$$\mathscr{C}_s \triangleq \boldsymbol{C} \otimes (\boldsymbol{I}_M \boldsymbol{K}^{(s)}). \tag{44}$$

Furthermore, the following matrices and vectors are defined as

$$\mathscr{M}_s \triangleq \text{diag}\{\mu_1 \boldsymbol{I}_M \boldsymbol{K}^{(s)}, \dots, \mu_N \boldsymbol{I}_M \boldsymbol{K}^{(s)}\}, \tag{45}$$

$$\mathscr{D}_s(i) \triangleq \text{diag}\left\{ \sum_{l=1}^{N} c_{l1}(\boldsymbol{u}_l(i))(\boldsymbol{u}_l(i))^{\mathrm{H}} \boldsymbol{K}^{(s)}, \dots, \sum_{l=1}^{N} c_{lN}(\boldsymbol{u}_l(i))(\boldsymbol{u}_l(i))^{H} \boldsymbol{K}^{(s)} \right\}, \tag{46}$$

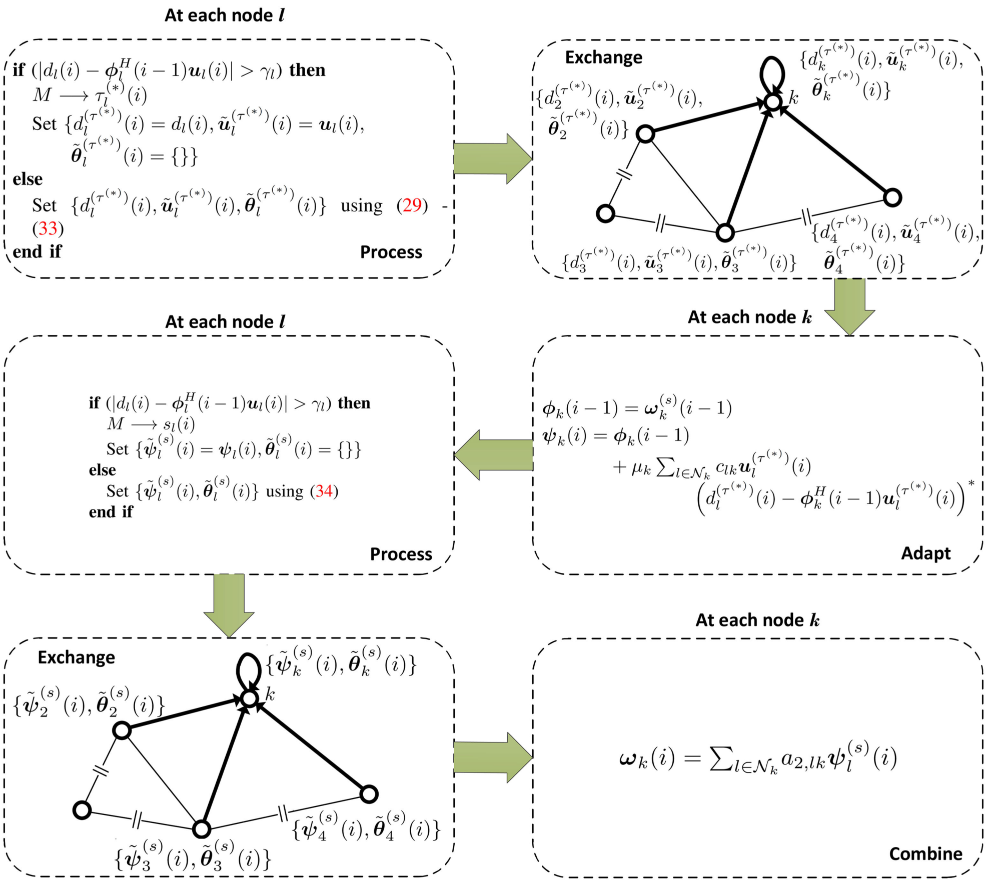
At each node l
if $(|d_l(i) - \boldsymbol{\phi}_l^H(i-1)\boldsymbol{u}_l(i)| > \gamma_l)$ then
$M \longrightarrow \tau_l^{(*)}(i)$
Set $\{d_l^{(\tau^{(*)})}(i) = d_l(i), \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i) = \boldsymbol{u}_l(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i) = \{\}\}$
else
Set $\{d_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$ using (29) - (33)
end if
Process
Exchange
$\{d_k^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_k^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_k^{(\tau^{(*)})}(i)\}$
$\{d_2^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_2^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_2^{(\tau^{(*)})}(i)\}$
k
$\{d_4^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_4^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_4^{(\tau^{(*)})}(i)\}$
$\{d_3^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_3^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_3^{(\tau^{(*)})}(i)\}$
At each node k
$\boldsymbol{\phi}_k(i-1) = \boldsymbol{\omega}_k^{(s)}(i-1)$
$\boldsymbol{\psi}_k(i) = \boldsymbol{\phi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{u}_l^{(\tau^{(*)})}(i) \left(d_l^{(\tau^{(*)})}(i) - \boldsymbol{\phi}_k^H(i-1)\boldsymbol{u}_l^{(\tau^{(*)})}(i)\right)^*$
Adapt
At each node l
if $(|d_l(i) - \boldsymbol{\phi}_l^H(i-1)\boldsymbol{u}_l(i)| > \gamma_l)$ then
$M \longrightarrow s_l(i)$
Set $\{\tilde{\boldsymbol{\psi}}_l^{(s)}(i) = \boldsymbol{\psi}_l(i), \tilde{\boldsymbol{\theta}}_l^{(s)}(i) = \{\}\}$
else
Set $\{\tilde{\boldsymbol{\psi}}_l^{(s)}(i), \tilde{\boldsymbol{\theta}}_l^{(s)}(i)\}$ using (34)
end if
Process
Exchange
$\{\tilde{\boldsymbol{\psi}}_k^{(s)}(i), \tilde{\boldsymbol{\theta}}_k^{(s)}(i)\}$
$\{\tilde{\boldsymbol{\psi}}_2^{(s)}(i), \tilde{\boldsymbol{\theta}}_2^{(s)}(i)\}$
k
$\{\tilde{\boldsymbol{\psi}}_4^{(s)}(i), \tilde{\boldsymbol{\theta}}_4^{(s)}(i)\}$
$\{\tilde{\boldsymbol{\psi}}_3^{(s)}(i), \tilde{\boldsymbol{\theta}}_3^{(s)}(i)\}$
At each node k
$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} a_{2,lk} \boldsymbol{\psi}_l^{(s)}(i)$
Combine

Fig. 2 *Proposed sparse diffusion LMS-ATC estimation with reduced communication cost*

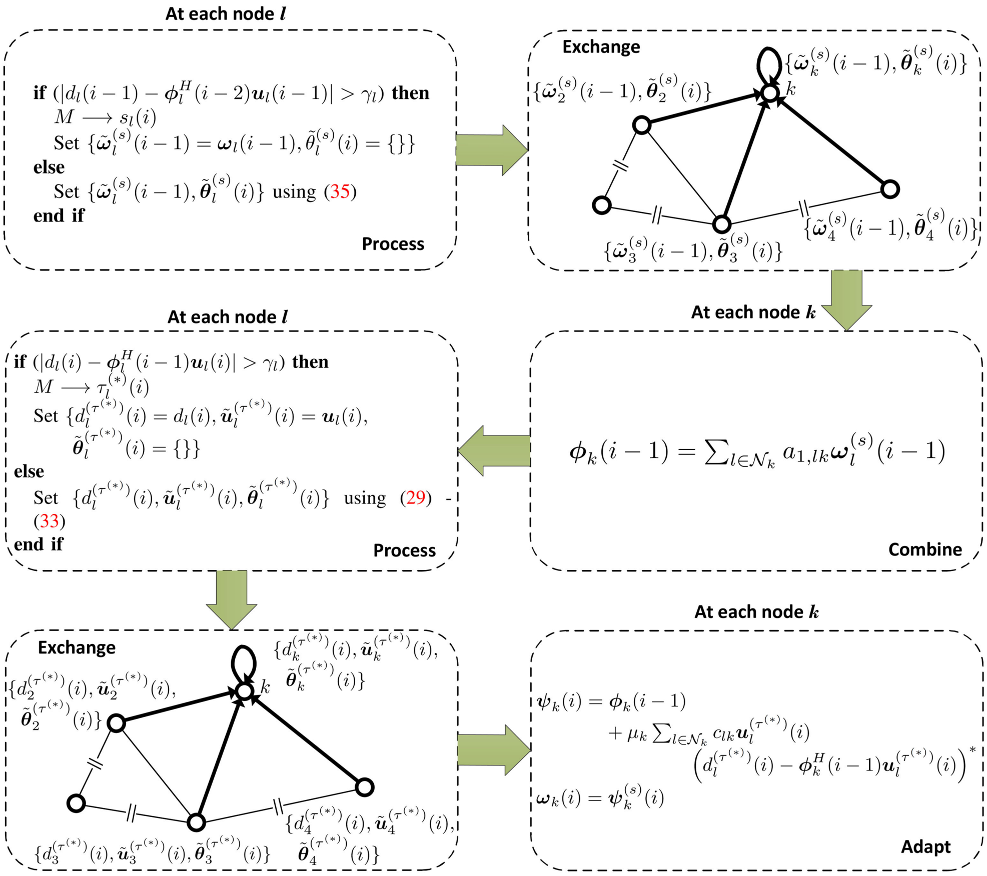
At each node l
if $(|d_l(i-1) - \boldsymbol{\phi}_l^H(i-2)\boldsymbol{u}_l(i-1)| > \gamma_l)$ then
$M \longrightarrow s_l(i)$
Set $\{\tilde{\boldsymbol{\omega}}_l^{(s)}(i-1) = \boldsymbol{\omega}_l(i-1), \tilde{\theta}_l^{(s)}(i) = \{\}\}$
else
Set $\{\tilde{\boldsymbol{\omega}}_l^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_l^{(s)}(i)\}$ using (35)
end if
Process
Exchange
$\{\tilde{\boldsymbol{\omega}}_k^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_k^{(s)}(i)\}$
$\{\tilde{\boldsymbol{\omega}}_2^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_2^{(s)}(i)\}$
k
$\{\tilde{\boldsymbol{\omega}}_4^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_4^{(s)}(i)\}$
$\{\tilde{\boldsymbol{\omega}}_3^{(s)}(i-1), \tilde{\boldsymbol{\theta}}_3^{(s)}(i)\}$
At each node k
$\boldsymbol{\phi}_k(i-1) = \sum_{l \in \mathcal{N}_k} a_{1,lk} \boldsymbol{\omega}_l^{(s)}(i-1)$
Combine
At each node l
if $(|d_l(i) - \boldsymbol{\phi}_l^H(i-1)\boldsymbol{u}_l(i)| > \gamma_l)$ then
$M \longrightarrow \tau_l^{(*)}(i)$
Set $\{d_l^{(\tau^{(*)})}(i) = d_l(i), \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i) = \boldsymbol{u}_l(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i) = \{\}\}$
else
Set $\{d_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_l^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_l^{(\tau^{(*)})}(i)\}$ using (29) - (33)
end if
Process
Exchange
$\{d_k^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_k^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_k^{(\tau^{(*)})}(i)\}$
$\{d_2^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_2^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_2^{(\tau^{(*)})}(i)\}$
k
$\{d_4^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_4^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_4^{(\tau^{(*)})}(i)\}$
$\{d_3^{(\tau^{(*)})}(i), \tilde{\boldsymbol{u}}_3^{(\tau^{(*)})}(i), \tilde{\boldsymbol{\theta}}_3^{(\tau^{(*)})}(i)\}$
At each node k
$\boldsymbol{\psi}_k(i) = \boldsymbol{\phi}_k(i-1) + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{u}_l^{(\tau^{(*)})}(i) \left(d_l^{(\tau^{(*)})}(i) - \boldsymbol{\phi}_k^H(i-1)\boldsymbol{u}_l^{(\tau^{(*)})}(i)\right)^*$
$\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k^{(s)}(i)$
Adapt

Fig. 3 *Proposed sparse diffusion LMS-CTA estimation with reduced communication cost*

Table 1 Number of calculations and communications at each node k with $|\mathcal{N}_k|$ neighbours

	Traditional sparse diffusion LMS [7]		Proposed	
	RZA	ZA	Adaptation and combination +	Process
additions	$(3\vert\mathcal{N}_k\vert + 1)M$	$3\vert\mathcal{N}_k\vert M$	$(2\vert\mathcal{N}_k\vert - 1)M + \vert\mathcal{N}_k\vert\tau^{(*)}_{avg,k}(i)$ +	$2M$
multiplications	$(3\vert\mathcal{N}_k\vert + 2)M + 2\vert\mathcal{N}_k\vert + 1$	$(3\vert\mathcal{N}_k\vert + 1)M + 2\vert\mathcal{N}_k\vert + 1$	$2\vert\mathcal{N}_k\vert\tau^{(*)}_{avg,k}(i) + \vert\mathcal{N}_k\vert s_{avg,k}(i) + 2\vert\mathcal{N}_k\vert$ +	$3M + 1$
divisions	*M*	0	0+	0
comparisons	*M*	*M*	1+	$2M$
sorting	0	0	0+	2 vectors of size *M*
broadcast	$2M + 1$	$2M + 1$	$2\tau_k^{(*)}(i)\mathbb{I}(\tau_k^{(*)}(i) < M/2) + M\mathbb{I}(\tau_k^{(*)}(i) \ge M/2)$ $+2s_k(i)\mathbb{I}(s_k(i) < M/2) + M\mathbb{I}(s_k(i) \ge M/2) + 1$	

$$\begin{aligned}\boldsymbol{h}_s(i) &\triangleq \mathscr{C}_s^{\mathrm{T}}\mathrm{col}\{(v_1(i))^*\boldsymbol{K}^{(s)}\boldsymbol{u}_l(i),\\ &\quad \ldots, (v_N(i))^*\boldsymbol{K}^{(s)}\boldsymbol{u}_l(i)\}\,.\end{aligned} \tag{47}$$

Moreover, let

$$\begin{aligned}\mathscr{R}_s &\triangleq \mathbb{E}[\mathscr{D}_s(i)]\\ &= \mathrm{diag}\left\{\sum_{l=1}^{N} c_{l1}\boldsymbol{R}_{u,l}^{(s)}, \ldots, \sum_{l=1}^{N} c_{lN}\boldsymbol{R}_{u,l}^{(s)}\right\},\end{aligned} \tag{48}$$

$$\begin{aligned}\mathscr{G}_s &\triangleq \mathbb{E}[(\boldsymbol{g}_s(i))(\boldsymbol{g}_s(i))^*]\\ &= \mathscr{C}_s^{\mathrm{T}} \cdot \mathrm{diag}\{\sigma_{v,1}^2\boldsymbol{R}_{u,1}^{(s)}, \ldots, \sigma_{v,N}^2\boldsymbol{R}_{u,N}^{(s)}\} \cdot \mathscr{C}_s,\end{aligned} \tag{49}$$

where $\boldsymbol{R}_{u,k}^{(s)} = \boldsymbol{R}_{u,k}^{(M)}\boldsymbol{K}^{(s)}$. Therefore, the recursion formula for network error vector can easily be obtained as

$$\begin{aligned}\tilde{\boldsymbol{\omega}}_s(i+1) &= \mathscr{A}_{s2}^{\mathrm{T}}(\boldsymbol{I} - \mathscr{M}_s\mathscr{D}_s(i))\mathscr{A}_{s1}^{\mathrm{T}}\tilde{\boldsymbol{\omega}}_s(i)\\ &\quad -\mathscr{A}_{s2}^{\mathrm{T}}\mathscr{M}_s\boldsymbol{g}_s(i)\,.\end{aligned} \tag{50}$$

In order to study the performance analysis, a few assumptions are considered about the measurement noise, regression vectors, and the step sizes.

Assumption 1: Noise is an i.i.d. random variable and it is statistically independent of the input regression vectors.

Assumption 2: All the regression vectors $\boldsymbol{u}_{k,i}$ are spatially and temporally independent.

Assumption 3: It is assumed that the step sizes μ_k's are small; so the high order terms in the analyses will be ignored using this assumption.

5.1 Mean stability

By Assumption 1, it is inferred that $\mathscr{D}_s(i)$ and $\tilde{\boldsymbol{\omega}}_s(i)$ are independent. Hence, by taking the expectation from the recursion formula (48), the recursion formula for mean of network error vector can be calculated as

$$\mathbb{E}[\tilde{\boldsymbol{\omega}}_s(i+1)] = \mathscr{A}_{s2}^{\mathrm{T}}(\boldsymbol{I} - \mathscr{M}_s\mathscr{R}_s)\mathscr{A}_{s1}^{\mathrm{T}}\mathbb{E}[\tilde{\boldsymbol{\omega}}_s(i)]\,. \tag{51}$$

Therefore, similar to the methodology used in [1, 7], the condition for convergence of the algorithm is obtained as

$$0 < \mu_k < \frac{2}{\max\limits_{\substack{l=1,\ldots,N\\ s=1,\ldots,M}} \lambda_{\max}(\boldsymbol{R}_{u,l}^{(s)})}\,. \tag{52}$$

Since there is no prior information about the location of non-zero elements of the parameter vector $\boldsymbol{\omega}_0$, one can easily show that $\lambda_{\max}(\boldsymbol{R}_{u,k}^{(s)}) \le \lambda_{\max}(\boldsymbol{R}_{u,k}^{(M)})$ holds for $s = 1, \ldots, M$. Therefore, the convergence condition becomes

$$0 < \mu_k < \frac{2}{\max_{l=1,\ldots,N} \lambda_{\max}(\boldsymbol{R}_{u,l}^{(M)})}\,. \tag{53}$$

which is similar to the convergence condition of the diffusion LMS algorithm. This was expected since the proposed algorithm at worst case is equal to the diffusion LMS algorithm when the constraint γ_l is not properly selected.

5.2 Steady-state approximate MSD lower bound

The steady-state MSD of the diffusion LMS algorithm is defined as

$$\mathrm{MSD}_{\mathrm{Net},s} \triangleq \lim_{i\to\infty}\frac{1}{N}\sum_{k=1}^{N}\mathbb{E}\left[\,\|\tilde{\boldsymbol{\omega}}_{s,k}(i)\|^2\,\right]. \tag{54}$$

By using the methodology similar to [1, 7, 12] and eliminating the higher order terms according to Assumption 3, the network MSD of the proposed strategy is calculated as

$$\begin{aligned}\mathrm{MSD}_{\mathrm{Net},s} &= \frac{1}{N}\left[\mathrm{vec}\left(\mathscr{A}_{s2}^{\mathrm{T}}\mathscr{M}_s\mathscr{G}_s\mathscr{M}_s\mathscr{A}_{s2}\right)\right]^{\mathrm{T}}\\ &\quad \times(\boldsymbol{I} - \mathscr{F}_s)^{-1}\mathrm{vec}(\boldsymbol{I}),\end{aligned} \tag{55}$$

where

$$\mathscr{F}_s \approx \mathscr{B}_s^{\mathrm{T}} \otimes \mathscr{B}_s^{*}, \tag{56a}$$

$$\mathscr{B}_s \triangleq \mathscr{A}_{s2}^{\mathrm{T}}(\boldsymbol{I} - \mathscr{M}_s\mathscr{R}_s)\mathscr{A}_{s1}^{\mathrm{T}}\,. \tag{56b}$$

Now let us look at each mean square deviation term $\mathbb{E}\left[\,\|\tilde{\boldsymbol{\omega}}_{s,k}(i)\|^2\,\right]$ in (54). As the number of zero elements increases, less non-zero elements participate in this term, and thus its value decreases. Therefore, for $s \ge K$, the following inequality holds:

$$\mathrm{MSD}_{\mathrm{Net},s} \ge \mathrm{MSD}_{\mathrm{Net},K}\,. \tag{57}$$

This means that the lowest steady-state MSD of the network that the proposed method can obtain is approximately equal to $\mathrm{MSD}_{\mathrm{Net},K}$.

6 Simulation results

In this section, simulation results are provided to show the performance of the proposed method. We consider a connected network with $N = 20$ nodes. The input vectors $\boldsymbol{u}_k(i)$ are of size $M = 100$ drawn from zero-mean white Gaussian distribution with covariance matrices $\boldsymbol{R}_{u,k}^{(M)} = \sigma_{u,k}^2\boldsymbol{I}_M$ with $\sigma_{u,k}^2 = 1.1$. The noise power $\sigma_{v,k}^2$ affecting each node k is chosen from a uniform distribution between 0 and 0.01 or between 0 and 0.1. Furthermore, the constraint $\gamma_k = 0.5$ for lower noise variance and $\gamma_k = 1.0$ for higher noise variance, $\alpha_1 = \alpha_2 = 99.99\%$, and step size $\mu_k = 0.05$ are chosen for all the nodes in the experiments. However, since both

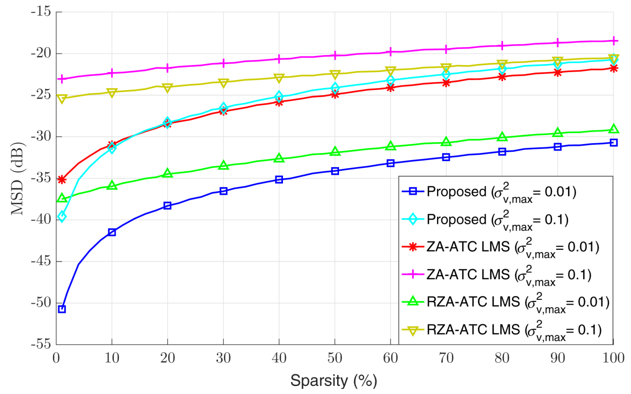

Fig. 4 *Steady-state MSD value versus different sparsity ratios for the proposed method, ZA-ATC LMS, and RZA-ATC LMS diffusion algorithms*

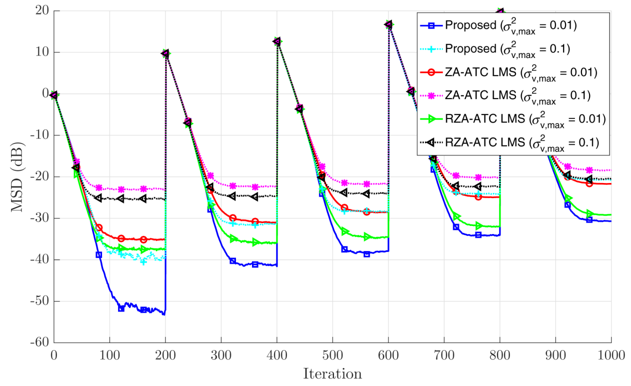

Fig. 5 *MSD value versus iteration with different sparsity ratios for the proposed method, ZA-ATC LMS, and RZA-ATC LMS diffusion algorithms*

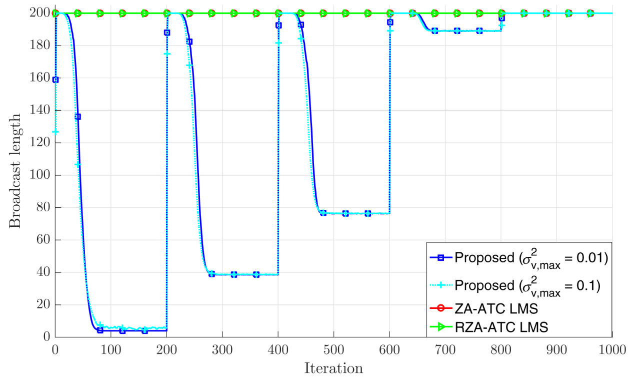

Fig. 6 *Broadcast length versus iteration with different sparsity ratios for the proposed method, ZA-ATC LMS, and RZA-ATC LMS diffusion algorithms*

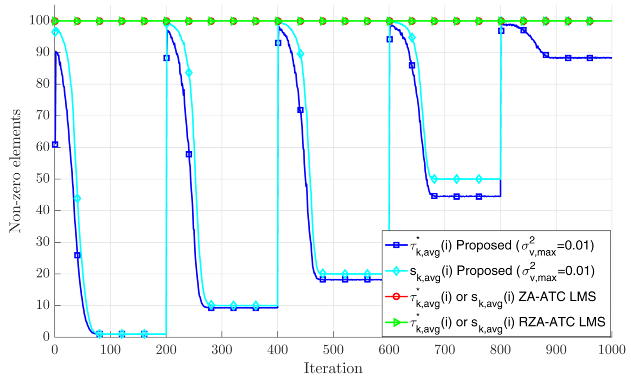

Fig. 7 *Average computed number of non-zero elements versus iteration with different sparsity ratios for the proposed method, ZA-ATC LMS, and RZA-ATC LMS diffusion algorithms*

ATC and CTA strategies have similar results, only the results for ATC strategy are presented in the experiments.

The aim of first experiment is to show the steady-state performance of the proposed algorithm to estimate the sparse signal and compare it with the sparse diffusion LMS algorithm of [7]. The value ρ is selected 7.5×10^{-3} for ZA and 2×10^{-3} for RZA strategy, which are obtained by plotting the MSD versus ρ and then selecting the proper ρ similar to [7]. These methods are denoted by ZA-ATC and RZA-ATC in the figures, respectively. Furthermore, the value of ε in RZA strategy is selected to be 0.1. The weighting coefficients for the adaptation step are chosen uniformly as $c_{lk} = 1/|\mathcal{N}_k|$ for all $l \in \mathcal{N}_k$; so the nodes exchange data with each other and participate at the adaptation step [1]. Moreover, coefficients for the combination step are chosen according to the relative degree combination rule as $a_{2,lk} = |\mathcal{N}_l| / \sum_{m \in \mathcal{N}_k} |\mathcal{N}_m|$ for all $l \in \mathcal{N}_k$ [1].

Fig. 4 depicts the network steady-state MSD with different sparsity ratios. For each sparsity ratio (e.g. 5/100), the vector $\boldsymbol{\omega}_0$ is set such that the sparsity per cent (e.g. 5%) of its elements are randomly selected and set them equal to 1 and set the rest of elements equal to 0. For each sparsity ratio, 100 simulations are performed and their average steady-state MSD value is calculated. As can be seen from Fig. 4, although our purpose is to reduce the communication cost, the proposed method can outperform the ZA-ATC LMS and RZA-ATC LMS algorithms considering the steady-state MSD value. Especially, when the sparsity ratio is small, the gap between them can increase much more. This shows that the proposed algorithm can properly perform estimation of a typical signal with various ranges of zero elements.

In the second experiment, we concentrate on different capabilities of the proposed method such as flexibility with the changes of the signal, and communication and processing cost. To this end, first a vector with sparsity 1% is considered; after 200, 400, 600, 800 iterations, the sparsity of the signal is changed to 10, 20, 50, 100%, respectively. Then, the proposed algorithm is applied to investigate if it can track the changes of signal quickly and if it can achieve a low error value when estimating the signal.

Fig. 5 shows the MSD values of the proposed algorithm versus the iteration. The ZA-ATC and RZA-ATC algorithms are also used to estimate the signal with various sparsities. As can be seen from the figure, the proposed method can adapt to the changes of signal quickly. This is an important criterion which shows the applicability of the proposed method. Furthermore, comparing it with the ZA-ATC and RZA-ATC strategies, its MSD value advantages over them can be seen once again.

Fig. 6 shows the average number of transmissions corresponding to the plots of Fig. 5, which is one of the main goals of the paper. It can be seen that as the algorithm converges, average broadcast length of the transmission data is reduced. If the time required for the network to perform all the transmissions in each iteration i is considered, it might even be possible to reduce this required time as well, and hence, speed up the convergence process. However, the criteria such as battery consumption and network life time can also be decreased due to shrinking packet lengths.

Now the payoff that this algorithm poses on the network, i.e. increasing processing complexity, is investigated. Fig. 7 shows the average $\tau^{(*)}_{\text{avg}}(i)$, and $s_{\text{avg}}(i)$ of the whole network that illustrates how the computed number of non-zero elements are decreased when the signal is sparse or contains zero elements. However, as explained earlier $\tau^{(*)}_{\text{avg},k}(i)$ and $s_{\text{avg},k}(i)$ are used to calculate average of every individual operation in each node k, as shown in Table 1.

Finally, the average number of each operation for the entire network is calculated and the results are presented in Fig. 8. Comparing the processing operations required to implement ZA-ATC LMS, RZA-ATC LMS, and the proposed algorithms, we deduce that the only main effective extra processing for the proposed method is implementing a sorting algorithm, which is executed at most twice at each node in every iteration. However, other important operations such as multiplications and divisions are decreased when the signal to be estimated contains a noticeable number of zeros. This shows that the proposed algorithm does not

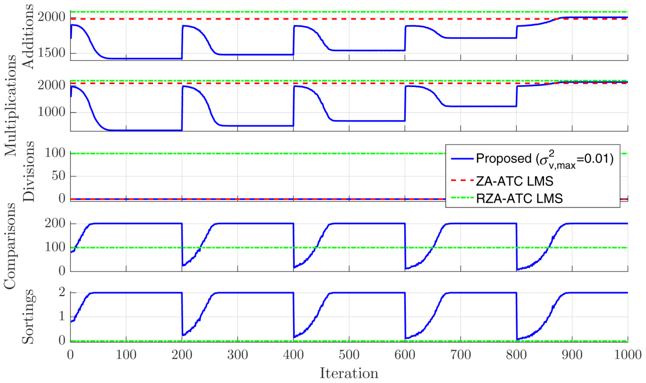

Fig. 8 *Processing complexities versus iteration with different sparsity ratios for the proposed method, ZA-ATC LMS, and RZA-ATC LMS diffusion algorithms*

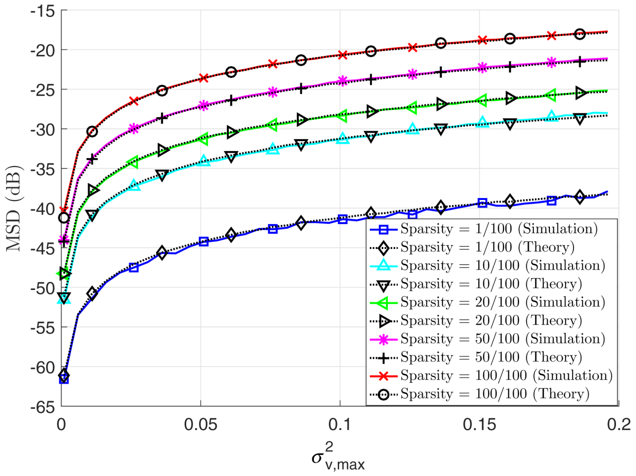

Fig. 9 *Steady-state MSD value versus different maximum noise powers for the proposed method with different sparsity ratios*

pose much processing burden for the network, and in contrary at some points, it may even reduce the processing complexity. This is another important feature of the proposed algorithm.

In the third experiment, we evaluate the steady-state MSD value of the proposed algorithm at different noise powers. To this end, the maximum noise power of the nodes is changed from 0.001 to 0.2 step by step, and the steady-state MSD value is obtained by simulations for different sparsities, i.e. $K = 1$, 10, 20, 50, and 100. Then, for the purpose of comparison, the corresponding theoretical MSD values are obtained by theoretical formula, i.e. $\mathrm{MSD}_{\mathrm{Net},K}$ for different K values. As it can be seen in Fig. 9, the simulations and theoretical results are very close to each other. Furthermore, it can be seen that the proposed algorithm uses sparsity to reduce the estimation error for different values of noise power. Also note that the theoretical MSD value is an approximation of its exact value and the simulation MSD values in the plots of Fig. 9 are sometimes a bit less than the theoretical values. The reason is that Assumption 3 is used to eliminate high order terms in (56a), and also $\boldsymbol{K}^{(S)}$ is used instead of the matrix $\boldsymbol{S}_l^{(\tau^{(*)})}(i)$ to obtain the theoretical MSD value. However, this difference is small enough to be ignored.

In addition, we have also performed the above simulations with different network sizes, i.e. $N = 100$ and $N = 1000$. However, the results showed little improvement in MSD value for all the algorithms, but the advantages of the proposed strategy still exist for these network sizes. Therefore, only the results for network size $N = 20$ are included in this paper.

7 Conclusion

In this paper, a method to decrease the communication cost of the diffusion based estimation algorithm is proposed. The proposed method relies on the well-known diffusion ATC and CTA strategies with an extra processing to select important elements and share them among the network nodes. However, it is seen that by using the proposed method the estimation accuracy is increased as well. Later, it is shown that since each node receives only a partial amount of data from its neighbours, the proposed method does not pose high processing cost to the network. The simulation results showed that running a sorting algorithm at most twice is the only noticeable extra processing operation, where other operations are reduced. On the other hand, the proposed method relies on the selection of a constraint which depends on the noise of the network. Moreover, the LMS-based algorithm is used to implement the reduced communication method, where other algorithms such as Affine projection (AP)-based algorithms, which need matrix inversion, cannot be used directly in the sparse case and should be altered accordingly. Therefore, selecting appropriate constraints, developing other adaptive algorithms, and reducing the communication cost in the uncertainty state, will be covered in future works.

8 References

[1] Cattivelli, F.S., Sayed, A.H.: ‘Diffusion LMS strategies for distributed estimation’, *IEEE Trans. Signal Process.*, 2010, **58**, (3), pp. 1035–1048

[2] Lopes, C.G., Sayed, A.H.: ‘Incremental adaptive strategies over distributed networks’, *IEEE Trans. Signal Process.*, 2007, **55**, (8), pp. 4064–4077

[3] Estrin, D., Girod, L., Pottie, G., *et al.*: ‘Instrumenting the world with wireless sensor networks’. Proc. IEEE Int. Conf. Acoustic, Speech, and Signal Processing (Cat. No.01CH37221), Salt Lake City, UT, 2001, pp. 2033–2036, vol. 4 doi: 10.1109/ICASSP.2001.940390

[4] Ghazanfari-Rad, S., Labeau, F.: ‘Formulation and analysis of LMS adaptive networks for distributed estimation in the presence of transmission errors’, *IEEE Internet Things J.*, 2016, **3**, (2), pp. 146–160

[5] Chen, J., Tu, S.Y., Sayed, A.H.: ‘Distributed optimization via diffusion adaptation’. 4th IEEE Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), San Juan, December 2011, pp. 281–284, doi: 10.1109/CAMSAP.2011.6136004

[6] Hu, W., Zhan, F.: ‘Sparse diffusion APA for distributed estimation’. IEEE 13th Int. Conf. Signal Processing (ICSP), Chengdu, November 2016, pp. 253–256, doi: 10.1109/ICSP.2016.7877835

[7] Lorenzo, P.D., Sayed, A.H.: ‘Sparse distributed learning based on diffusion adaptation’, *IEEE Trans. Signal Process.*, 2013, **61**, (6), pp. 1419–1433

[8] Cattivelli, F.S., Lopes, C.G., Sayed, A.H.: ‘Diffusion recursive least-squares for distributed estimation over adaptive networks’, *IEEE Trans. Signal Process.*, 2008, **56**, (5), pp. 1865–1877

[9] Li, L., Chambers, J.A., Lopes, C.G., *et al.*: ‘Distributed estimation over an adaptive incremental network based on the affine projection algorithm’, *IEEE Trans. Signal Process.*, 2010, **58**, (1), pp. 151–164

[10] Gu, Y., Jin, J., Mei, S.: ‘l0-norm constraint LMS algorithm for sparse system identification’, *IEEE Signal Process. Lett.*, 2009, **16**, (9), pp. 774–777

[11] Donoho, D.L.: ‘Compressed sensing’, *IEEE Trans. Inf. Theory*, 2006, **52**, (4), pp. 1289–1306

[12] Shiri, H., Tinati, M.A., Codreanu, M., *et al.*: ‘Distributed sparse diffusion estimation based on set membership and affine projection algorithm’, *Digit. Signal Process.*, 2018, **73**, pp. 47–61. Available at http://www.sciencedirect.com/science/article/pii/S105120041730249X

[13] Vaswani, N., Zhan, J.: ‘Recursive recovery of sparse signal sequences from compressive measurements: A review’, *IEEE Trans. Signal Process.*, 2016, **64**, (13), pp. 3523–3549

[14] Azarnia, G., Tinati, M.A., Rezaii, T.Y.: ‘Cooperative and distributed algorithm for compressed sensing recovery in wireless sensor networks’, *IET Signal Process.*, 2018, **12**, (3), pp. 346–357. DOI: 10.1049/iet-spr.2017.0093

[15] Chen, Y., Gu, Y., Hero, A.O.: ‘Sparse LMS for system identification’. Proc. IEEE Int. Conf. Acoustic, Speech, and Signal Processing, Taipei, April 2009, pp. 3125–3128, doi: 10.1109/ICASSP.2009.4960286

[16] Bhotto, M.Z.A., Antoniou, A.: ‘A new partial-update NLMS adaptive-filtering algorithm’. IEEE 27th Canadian Conf. Electrical and Computer Engineering (CCECE), Toronto, ON, May 2014, pp. 1–5, doi: 10.1109/CCECE.2014.6901048

[17] Mota, J.F.C., Xavier, J.M.F., Aguiar, P.M.Q., *et al.*: ‘Distributed basis pursuit’, *IEEE Trans. Signal Process.*, 2012, **60**, (4), pp. 1942–1956