Diffusion LMS over Multitask Networks

Jie Chen[†], *Member, IEEE*, Cédric Richard[†], *Senior Member, IEEE*
Ali H. Sayed[‡], *Fellow Member, IEEE*

[†] Université de Nice Sophia-Antipolis, UMR CNRS 7293, Observatoire de la Côte d'Azur
Laboratoire Lagrange, Parc Valrose, 06102 Nice - France
phone: (33) 492 076 394 fax: (33) 492 076 321
jie.chen@unice.fr cedric.richard@unice.fr

[‡] Electrical Engineering Department
University of California, Los Angeles, USA
phone: (310) 267 2142 fax: (310) 206 8495
sayed@ee.ucla.edu

EDICS: NET-ADEG, NET-DISP, MLR-DIST, SSP-PERF

Abstract

The diffusion LMS algorithm has been extensively studied in recent years. This efficient strategy allows to address distributed optimization problems over networks in the case where nodes have to collaboratively estimate a single parameter vector. Problems of this type are referred to as single-task problems. Nevertheless, there are several problems in practice that are multitask-oriented in the sense that the optimum parameter vector may not be the same for every node. This brings up the issue of studying the performance of the diffusion LMS algorithm when it is run, either intentionally or unintentionally, in a multitask environment. In this paper, we conduct a theoretical analysis on the stochastic behavior of diffusion LMS in the case where the so-called single-task hypothesis is violated. We explain under what conditions diffusion LMS continues to deliver performance superior to non-cooperative strategies in the multitask environment. When the conditions are violated, we explain how to endow the nodes with the ability to cluster with other similar nodes to remove bias. We propose an unsupervised clustering strategy that allows each node to select, via adaptive adjustments of combination weights, the neighboring nodes with which it can collaborate to estimate a common parameter vector. Simulations are presented to illustrate the theoretical results, and to demonstrate the efficiency of the proposed clustering strategy. The framework is applied to a useful problem involving a multi-target tracking task.

Index Terms

Multitask learning, distributed optimization, diffusion strategy, collaborative processing, stochastic performance, adaptive clustering.

The work of C. Richard was partly supported by the Agence Nationale pour la Recherche, France, (ODISSEE project, ANR-13-ASTR-0030). The work of A. H. Sayed was supported in part by NSF grant CCF-1011918. A short and preliminary version of this work appears in the conference publication [1].

I. Introduction

Distributed adaptive estimation is an attractive and challenging problem that allows a collection of interconnected nodes to perform preassigned tasks from streaming measurements, such as parameter estimation. Although centralized strategies may fully benefit from information collected throughout a network, in most cases, distributed strategies are more appropriate to solve inference problems in a collaborative and autonomous manner [2].

Most recent efforts in the study of distributed estimation problems have focused on scenarios where the entire network is employed to collectively estimate a single parameter vector. Several strategies have been proposed for this purpose for sequential data processing over networks, including consensus strategies [3]–[9], incremental strategies [10]–[14], and diffusion strategies [15], [16]. Diffusion strategies are particularly attractive due to their enhanced adaptation performance and wider stability ranges when constant step-sizes are used to enable continuous learning [17]. For this reason, we focus on this class of strategies in the remainder of the article. These strategies estimate a common parameter vector by minimizing, in a distributed manner, a global criterion that aggregates neighborhood cost functions. Nodes exchange information locally and cooperate only with their neighbors, without the need for sharing and requiring any global information. The resulting networks benefit from the temporal and spatial diversity of the data and end up being endowed with powerful learning and tracking abilities [17], [18]. Depending on whether the adaptation step is performed before or after the consultation step, two forms of diffusion can be derived: the adapt-then-combine (ATC) and the combine-then-adapt (CTA) strategies. The performance of the corresponding adaptive networks have been extensively studied in the literature, under favorable and unfavorable conditions such as model non-stationarities and imperfect communication [19], [20]. This framework has also been extended by considering more general cost functions and data models [18], [21]–[23], by incorporating additional regularizers [24]–[26], or by expanding its use to other scenarios [27]–[30].

The working hypothesis for these earlier studies on diffusion LMS strategies is that the nodes cooperate with each other to estimate a single parameter vector. We shall refer to problems of this type as *single-task* problems. However, many problems of interest happen to be *multitask*-oriented in the sense that there are multiple optimum parameter vectors to be inferred simultaneously and in a collaborative manner. The multitask learning problem is relevant in several machine learning formulations and has been studied in the machine learning community in several contexts [31]–[33]. Recently, this concept has also been studied in the context of distributed estimation and adaptation over networks [34]. Due to inaccurate modeling, or minor differences between tasks neglected intentionally, there may be situations in which the diffusion LMS algorithm is applied in a multitask environment. When these situations occur, the distributed implementation will lead to biased results that may be acceptable depending on the application at hand. This biased solution may still be beneficial compared to purely non-cooperative strategies provided that the local optimums are sufficiently close to each other. These observations motivate us to examine the performance of the diffusion LMS strategy when it is run, either intentionally or unintentionally, in a multitask environment. In this respect, we shall analyze the performance of the diffusion LMS in terms of its mean weight deviation and mean-square error in the case when the single-task hypothesis is violated. We shall also clarify under what conditions diffusion LMS continues to deliver performance superior to non-cooperative strategies in the multitask environment. When the conditions are violated, we shall explain how to endow the nodes with the ability to cluster with other similar nodes to remove the bias. In particular, we shall propose an unsupervised clustering strategy that allows each node to select, via adaptive adjustments of combination weights, the neighboring nodes with which it should collaborate to improve its estimation accuracy. In the related work [34], we formulate the multitask problem directly over networks with connected clusters of nodes. In that work, the clusters are assumed to be known beforehand and no clustering is proposed. We then derive extended diffusion strategies that enable adaptation and learning under these conditions. In the current work, on the other hand, the clusters are not assumed to be known. It then becomes necessary to examine how this lack of information influences performance. It also becomes necessary to endow the nodes with the ability to identify and form appropriate clusters to enhance performance. One clustering strategy was proposed in the earlier work [35]; its performance is dependent on the initial conditions used by the nodes to launch their adaptation rules. In this work, we provide a more general formulation and propose a more robust clustering strategy. We also provide a detailed performance analysis to support the conclusions.

This paper is organized as follows. Section II formulates the distributed estimation problem for multitask learning, and briefly introduces the diffusion LMS algorithm. Section III analyzes the theoretical performance of this algorithm in a multitask-oriented environment, in the mean and mean-square-error sense. Section IV introduces an unsupervised clustering strategy. In Section V, experiments and applications are presented to illustrate the performance of the approach. Section VI concludes this paper and gives perspectives on future work.

II. Multitask problems and diffusion LMS

Before starting our presentation, we provide a summary of some symbols used in the paper. Normal font letters x denote scalars. Boldface small letters $\boldsymbol{x}$ denote vectors. All vectors are column vectors. Boldface capital letters $\boldsymbol{X}$ denote matrices. The superscript $(\cdot)^\top$ represents the transpose of a matrix or a vector. Matrix trace is denoted by trace$\{\cdot\}$, Kronecker product is denoted by $\otimes$, and expectation is denoted by $E\{\cdot\}$. Identity matrix of size $N \times N$ is denoted by $\boldsymbol{I}_N$. We denote by $\mathcal{N}_k$ the set of node indices in the neighborhood of node k, including k itself, and $|\mathcal{N}_k|$ its cardinality. The operator col$\{\cdot\}$ stacks its vector arguments on the top of each other to generate a connected vector. The other symbols will be defined in the context where they are used.

A. *Modeling assumptions and Pareto solution*

We consider a connected network composed of N nodes. The problem is to estimate $L\times 1$ unknown vectors $\boldsymbol{w}_k^\star$ at each node k from collected measurements. Node k has access to temporal wide-sense stationary measurement sequences $\{d_k(n), \boldsymbol{x}_k(n)\}$, with $d_k(n)$ denoting a scalar zero-mean reference signal, and $\boldsymbol{x}_k(n)$ denoting an $L\times 1$ regression vector with a positive-definite covariance matrix $\boldsymbol{R}_{x,k} = E\{\boldsymbol{x}_k(n)\boldsymbol{x}_k^\top(n)\} > 0$. The data at node k are assumed to be related via the linear regression model:

$$d_k(n) = \boldsymbol{x}_k^\top(n)\,\boldsymbol{w}_k^\star + z_k(n) \tag{1}$$

where $z_k(n)$ is a zero-mean i.i.d. additive noise at node k and time n. Noise $z_k(n)$ is assumed to be independent of any other signals and has variance $\sigma_{z,k}^2$. Let $J_k(\boldsymbol{w})$ denote the mean-square-error cost at node k, namely,

$$J_k(\boldsymbol{w}) = E\left\{|d_k(n) - \boldsymbol{x}_k^\top(n)\,\boldsymbol{w}|^2\right\}. \tag{2}$$

It is clear from (1) that each $J_k(\boldsymbol{w})$ is minimized at $\boldsymbol{w}_k^\star$. Depending on whether the minima of all the $J_k(\boldsymbol{w})$ are achieved at the same location or not, referred to as tasks, the distributed learning problem can be single-task or multitask oriented [34].

In a single-task network, all nodes have to estimate the same parameter vector $\boldsymbol{w}^\star$. That is, in this case we have that

$$\boldsymbol{w}_k^\star = \boldsymbol{w}^\star, \quad \forall k \in \{1, ..., N\}. \tag{3}$$

Diffusion LMS strategies for the distributed estimation of $\boldsymbol{w}^\star$ under this scenario were derived in [2], [15], [16], [36] by seeking the minimizer of the following aggregate cost function:

$$J^{\text{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N} J_k(\boldsymbol{w}) \tag{4}$$

in a cooperative manner in order to improve estimation accuracy. In a multitask network, on the other hand, each node needs to determine its own parameter vector $\boldsymbol{w}_k^\star$. It will be assumed that some similarities or relationships exist among the parameter vectors of neighboring nodes so that cooperation can still be meaningful and useful, namely,

$$\boldsymbol{w}_k^\star \sim \boldsymbol{w}_\ell^\star \;\text{ if } \ell \in \mathcal{N}_k \tag{5}$$

where the symbol $\sim$ represents a similarity relationship in some sense, which can be promoted using appropriate regularization [34]. Since each cost function $J_k(\boldsymbol{w})$ may not be minimized at the same location, the minimizer of the aggregate cost (4) can be shown to correspond to a Pareto optimum solution for the multi-objective optimization problem [21]. That is, there would not exist another vector $\boldsymbol{w}$ that is able to reduce any individual cost without increasing some of the other costs [37]. Diffusion LMS thus leads to a compromise for the entire network.

B. Diffusion LMS

The diffusion LMS algorithm was originally designed for minimizing the cost function (4) in an adaptive and distributed manner [15], [16], [36], [38]. Let $\boldsymbol{w}_k(n)$ denote the estimate of the minimizer of (4) at node k and time instant n. The general structure of the algorithm consists of the following steps:

$$\boldsymbol{\phi}_k(n) = \sum_{\ell \in \mathcal{N}_k} a_{1,\ell k}\, \boldsymbol{w}_\ell(n) \tag{6}$$

$$\boldsymbol{\psi}_k(n+1) = \boldsymbol{\phi}_k(n) + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\big[d_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{\phi}_k(n)\big] \tag{7}$$

$$\boldsymbol{w}_k(n+1) = \sum_{\ell \in \mathcal{N}_k} a_{2,\ell k}\, \boldsymbol{\psi}_\ell(n+1) \tag{8}$$

The non-negative coefficients $a_{1,\ell k}$, $a_{2,\ell k}$ and $c_{\ell k}$ are the (ℓ, k)-th entries of two left-stochastic matrices, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, and a right-stochastic matrix $\boldsymbol{C}$, that is,

$$\begin{gathered} \boldsymbol{A}_1^\top \boldsymbol{1}_N = \boldsymbol{1}_N,\; \boldsymbol{A}_2^\top \boldsymbol{1}_N = \boldsymbol{1}_N,\; \boldsymbol{C}\boldsymbol{1}_N = \boldsymbol{1}_N \\ a_{1,\ell k} = 0,\; a_{2,\ell k} = 0,\; c_{\ell k} = 0 \quad \text{if} \quad \ell \notin \mathcal{N}_k \end{gathered} \tag{9}$$

Several adaptive strategies can be obtained as special cases of (6)–(8) through appropriate selections of $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{C}$. For instance, setting $\boldsymbol{A}_1 = \boldsymbol{I}_N$ yields the so-called adapt-then-combine (ATC) diffusion LMS. Setting $\boldsymbol{A}_2 = \boldsymbol{I}_N$ leads to the combine-then-adapt (CTA) diffusion LMS. By setting $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \boldsymbol{C} = \boldsymbol{I}_N$, the algorithm degenerates to non-cooperative LMS that will be considered in the sequel for comparison purposes.

When applying ATC diffusion LMS without information exchange, that is, with $\boldsymbol{C} = \boldsymbol{I}_N$, the agents converge toward the Pareto optimum with a bias of the order $\mathcal{O}(\mu_{\max})$, where $\mu_{\max}$ denotes the largest step size parameter across all nodes [21]. In this paper, rather than focus on this convergence point that can be perceived as a compromise, we shall study analytically how diffusion LMS (6)–(8) behaves in a multitask environment in relation to the optimum vectors $\boldsymbol{w}_k^\star$. Moreover, in order to generalize the analysis, we shall consider drifting optimums around a fixed value $\boldsymbol{w}_k^\star$, namely,

$$\boldsymbol{w}_k^\star(n) = \boldsymbol{w}_k^\star + \boldsymbol{\epsilon}_k(n) \tag{10}$$

where $\boldsymbol{\epsilon}_k(n)$ is a zero-mean random perturbation independent of any other signal, with zero-mean and covariance matrix $\sigma^2_{\epsilon,k}\,\boldsymbol{I}_L$. Even if $\boldsymbol{w}^\star_k = \boldsymbol{w}^\star_\ell$ for all k and ℓ, it is important to note that the problem is still multitask-oriented due to the random perturbations $\boldsymbol{\epsilon}_k(n)$ and $\boldsymbol{\epsilon}_\ell(n)$. The formulation (10), with a possible different $\boldsymbol{w}^\star_k$, across the nodes, makes the problem formulation different from the analysis of diffusion LMS in a non-stationary environment [20]. Under (10), model (1) is replaced by

$$d_k(n) = \boldsymbol{x}_k^\top(n)\,\boldsymbol{w}^\star_k(n) + z_k(n) \tag{11}$$

III. Performance analysis of diffusion LMS for multitask networks

We collect information from across the network into block vectors and matrices. In particular, we denote by $\boldsymbol{w}(n)$, $\boldsymbol{w}^\star$ and $\boldsymbol{w}^\star(n)$ the block weight estimate vector, the block optimum mean weight vector, and the instantaneous block optimum weight vector, all of size $LN \times 1$, that is,

$$\boldsymbol{w}(n) = \text{col}\{\boldsymbol{w}_1(n), \ldots, \boldsymbol{w}_N(n)\} \tag{12}$$

$$\boldsymbol{w}^\star = \text{col}\{\boldsymbol{w}^\star_1, \ldots, \boldsymbol{w}^\star_N\} \tag{13}$$

$$\boldsymbol{w}^\star(n) = \text{col}\{\boldsymbol{w}^\star_1(n), \ldots, \boldsymbol{w}^\star_N(n)\}. \tag{14}$$

The weight error vector for each node k at iteration n is defined by

$$\boldsymbol{v}_k(n) = \boldsymbol{w}_k(n) - \boldsymbol{w}^\star_k(n). \tag{15}$$

Let

$$\boldsymbol{v}^\star_k(n) = \boldsymbol{w}_k(n) - \boldsymbol{w}^\star_k \tag{16}$$

be the weight error vector between the estimated weight vector $\boldsymbol{w}_k(n)$ and the fixed weight vector $\boldsymbol{w}^\star_k$. The following relation holds

$$\boldsymbol{v}_k(n) = \boldsymbol{v}^\star_k(n) - \boldsymbol{\epsilon}_k(n) \tag{17}$$

This relation enables us to derive recursions with respect to $\boldsymbol{v}^\star_k(n)$, and then get back to $\boldsymbol{v}_k(n)$. The weight error vectors $\boldsymbol{v}_k(n)$ and $\boldsymbol{v}^\star_k(n)$ are also stacked on top of each other to get the block weight error vectors:

$$\boldsymbol{v}(n) = \text{col}\{\boldsymbol{v}_1(n), \ldots, \boldsymbol{v}_N(n)\} \tag{18}$$

$$\boldsymbol{v}^\star(n) = \text{col}\{\boldsymbol{v}^\star_1(n), \ldots, \boldsymbol{v}^\star_N(n)\} \tag{19}$$

To perform the theoretical analysis, we introduce the following independence assumption.

Assumption 1: (Independent regressors) The regression vectors $\boldsymbol{x}_k(n)$ arise from a zero-mean random process that is temporally stationary, white, and independent over space with $\boldsymbol{R}_{x,k} = E\{\boldsymbol{x}_k(n)\,\boldsymbol{x}_k^\top(n)\} > 0$. ■

A direct consequence is that $\boldsymbol{x}_k(n)$ is independent of $\boldsymbol{v}_\ell(m)$ for all ℓ and $m \leq n$. Although not true in general, this assumption is commonly used for analyzing adaptive constructions because it allows to simplify the derivation without constraining the conclusions. Moreover, various analyses in the literature have already shown that performance results obtained under this assumption match well the actual performance of adaptive algorithms when the step-sizes are sufficiently small [39].

A. Mean weight behavior analysis

Subtracting $\boldsymbol{w}_k^\star$ from both sides of the first step of diffusion LMS, namely equation (6), gives

$$\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star = \sum_{\ell \in \mathcal{N}_k} a_{1,\ell k}\, \boldsymbol{w}_\ell(n) - \boldsymbol{w}_k^\star \tag{20}$$

Defining $\boldsymbol{\mathcal{A}}_1 = \boldsymbol{A}_1 \otimes \boldsymbol{I}_L$ and using $\boldsymbol{w}_k^\star = \boldsymbol{w}_k(n) - \boldsymbol{v}_k^\star(n)$, expression (20) can be expressed in block-based form as follows

$$\boldsymbol{\phi}(n) - \boldsymbol{w}^\star = \boldsymbol{\mathcal{A}}_1^\top\, \boldsymbol{v}^\star(n) + (\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL})\boldsymbol{w}^\star \tag{21}$$

Note that the term $(\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL})\,\boldsymbol{w}^\star$, which does not appear for single-task networks, is inherited from the multitask context. The estimation error in the second step (7) of diffusion LMS can be rewritten as

$$\begin{aligned} d_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{\phi}_k(n) &= \boldsymbol{x}_\ell^\top(n)(\boldsymbol{w}_\ell^\star + \boldsymbol{\epsilon}_\ell(n)) + z_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{\phi}_k(n) \\ &= z_\ell(n) - \boldsymbol{x}_\ell^\top(n)\,(\boldsymbol{\phi}_k(n) - \boldsymbol{w}_\ell^\star) + \boldsymbol{x}_\ell^\top(n)\,\boldsymbol{\epsilon}_\ell(n) \end{aligned} \tag{22}$$

For single-task networks, $(\boldsymbol{\phi}_k(n) - \boldsymbol{w}_\ell^\star)$ in the above expression reduces to $(\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star)$ since $\boldsymbol{w}_k^\star = \boldsymbol{w}_\ell^\star$ for all k, ℓ. In the multitask context, we can establish the following relationship:

$$\begin{aligned} \boldsymbol{\phi}_k(n) - \boldsymbol{w}_\ell^\star &= (\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star) + (\boldsymbol{w}_k^\star - \boldsymbol{w}_\ell^\star) \\ &= (\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star) + \boldsymbol{u}_{k\ell}^\star \end{aligned} \tag{23}$$

where $\boldsymbol{u}_{k\ell}^\star$ is the difference between the fixed weight vectors $\boldsymbol{w}_k^\star$ and $\boldsymbol{w}_\ell^\star$. Incorporating this expression into (22) yields:

$$d_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{\phi}_k(n) = z_\ell(n) - \boldsymbol{x}_\ell^\top(n)\,(\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star) - \boldsymbol{x}_\ell^\top(n)\,\boldsymbol{u}_{k\ell}^\star + \boldsymbol{x}_\ell^\top(n)\,\boldsymbol{\epsilon}_\ell(n) \tag{24}$$

Subtracting $\boldsymbol{w}_k^\star$ from both sides of equation (7) and using the above relation, we have

$$\begin{aligned} \boldsymbol{\psi}_k(n+1) - \boldsymbol{w}_k^\star = (\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star) &- \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\,(\boldsymbol{\phi}_k(n) - \boldsymbol{w}_k^\star) + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\, z_\ell(n) \\ &- \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{u}_{k\ell}^\star + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{\epsilon}_\ell(n) \end{aligned} \tag{25}$$

Let us introduce the following $N \times N$ block diagonal matrices with blocks of size $L \times L$:

$$\boldsymbol{U} = \text{diag}\{\mu_1 \boldsymbol{I}_L, \dots, \mu_N \boldsymbol{I}_L\} \tag{26}$$

$$\boldsymbol{H}(n) = \text{diag}\Big\{ \sum_{\ell \in \mathcal{N}_1} c_{\ell 1}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n), \dots, \sum_{\ell \in \mathcal{N}_N} c_{\ell N}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n) \Big\}, \tag{27}$$

and the following vectors of length NL:

$$\boldsymbol{h}_u(n) = \text{col}\Big\{ \sum_{\ell \in \mathcal{N}_1} c_{\ell 1}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{u}_{\ell 1}^\star, \dots, \sum_{\ell \in \mathcal{N}_N} c_{\ell N}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{u}_{\ell N}^\star \Big\} \tag{28}$$

$$\boldsymbol{h}_\epsilon(n) = \text{col}\Big\{ \sum_{\ell \in \mathcal{N}_1} c_{\ell 1}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{\epsilon}_\ell(n), \dots, \sum_{\ell \in \mathcal{N}_N} c_{\ell N}\, \boldsymbol{x}_\ell(n)\boldsymbol{x}_\ell^\top(n)\, \boldsymbol{\epsilon}_\ell(n) \Big\} \tag{29}$$

$$\boldsymbol{p}_{zx}(n) = \text{col}\Big\{ \sum_{\ell \in \mathcal{N}_1} c_{\ell 1}\, \boldsymbol{x}_\ell(n)\, z_\ell(n), \dots, \sum_{\ell \in \mathcal{N}_N} c_{\ell N}\, \boldsymbol{x}_\ell(n)\, z_\ell(n) \Big\}. \tag{30}$$

Using the above notation and equations (21) and (25), the block weight error vector $\boldsymbol{\psi}(n+1)$ can be expressed as

$$\boldsymbol{\psi}(n+1) - \boldsymbol{w}^\star = (\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H}(n))\Big(\boldsymbol{\mathcal{A}}_1^\top\, \boldsymbol{v}^\star(n) + (\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL})\boldsymbol{w}^\star\Big) + \boldsymbol{U}\boldsymbol{p}_{zx}(n) - \boldsymbol{U}(\boldsymbol{h}_u(n) - \boldsymbol{h}_\epsilon(n)) \tag{31}$$

Let $\boldsymbol{\mathcal{A}}_2 = \boldsymbol{A}_2 \otimes \boldsymbol{I}_L$. The combination step (8) of diffusion LMS leads to

$$\boldsymbol{w}(n+1) = \boldsymbol{\mathcal{A}}_2^\top\, \boldsymbol{\psi}(n+1) \tag{32}$$

Subtracting $\boldsymbol{w}^\star$ from both sides of the above expression, we get

$$\boldsymbol{v}^\star(n+1) = \boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{\psi}(n+1) - \boldsymbol{w}^\star) + (\boldsymbol{\mathcal{A}}_2^\top - \boldsymbol{I}_{NL})\,\boldsymbol{w}^\star \tag{33}$$

Again, note that the term $(\boldsymbol{\mathcal{A}}_2^\top - \boldsymbol{I}_{NL})\,\boldsymbol{w}^\star$ does not appear for single-task networks and is inherited from the multitask context. Combing (31) and (33), we obtain the update relation for $\boldsymbol{v}^\star(n)$ in a single expression as follows:

$$\begin{aligned}\boldsymbol{v}^\star(n+1) = \boldsymbol{\mathcal{A}}_2^\top\,(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H}(n))\,\boldsymbol{\mathcal{A}}_1^\top \boldsymbol{v}^\star(n) + \boldsymbol{\mathcal{A}}_2^\top \boldsymbol{U}\boldsymbol{p}_{zx}(n) - \boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{U}(\boldsymbol{h}_u(n) - \boldsymbol{h}_\epsilon(n))\\ + \Big(\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{NL}(n) - \boldsymbol{U}\boldsymbol{H}(n))(\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL}) + (\boldsymbol{\mathcal{A}}_2^\top - \boldsymbol{I}_{NL})\Big)\,\boldsymbol{w}^\star\end{aligned} \tag{34}$$

In order to make the presentation clearer, we use the following notation for terms in expression (34):

$$\boldsymbol{B}(n) = \boldsymbol{\mathcal{A}}_2^\top\,(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H}(n))\,\boldsymbol{\mathcal{A}}_1^\top \tag{35}$$

$$\boldsymbol{g}(n) = \boldsymbol{\mathcal{A}}_2^\top \boldsymbol{U}\boldsymbol{p}_{zx}(n) \tag{36}$$

$$\boldsymbol{r}(n) = \underbrace{\boldsymbol{\mathcal{A}}_2^\top \boldsymbol{U}\boldsymbol{h}_u(n)}_{\boldsymbol{r}_u(n)} - \underbrace{\boldsymbol{\mathcal{A}}_2^\top \boldsymbol{U}\boldsymbol{h}_\epsilon(n)}_{\boldsymbol{r}_\epsilon(n)} - \underbrace{\Big(\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H}(n))(\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL}) + (\boldsymbol{\mathcal{A}}_2^\top - \boldsymbol{I}_{NL})\Big)\,\boldsymbol{w}^\star}_{\boldsymbol{r}_w(n)} \tag{37}$$

where the three terms that compose $\boldsymbol{r}(n)$ are denoted by $\boldsymbol{r}_u(n)$, $\boldsymbol{r}_\epsilon(n)$ and $\boldsymbol{r}_w(n)$, respectively. Then, recursion (34) can be rewritten as

$$\boldsymbol{v}^\star(n+1) = \boldsymbol{B}(n)\,\boldsymbol{v}^\star(n) + \boldsymbol{g}(n) - \boldsymbol{r}(n). \tag{38}$$

Let $\boldsymbol{H}$ be the expected value of $\boldsymbol{H}(n)$ given by

$$\boldsymbol{H} = \text{diag}\,\{\boldsymbol{R}_1,\,\dots,\,\boldsymbol{R}_N\} \tag{39}$$

in terms of the neighborhood covariance matrices:

$$\boldsymbol{R}_k = \sum_{\ell\in\mathcal{N}_k} c_{\ell k}\,\boldsymbol{R}_{x,\ell} \tag{40}$$

Let $\boldsymbol{h}_u$ be the expected value $E\{\boldsymbol{h}_u(n)\}$, that is,

$$\boldsymbol{h}_u = \text{col}\Big\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{R}_{x,\ell}\,\boldsymbol{u}_{1\ell}^\star,\,\dots,\,\sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{R}_{x,\ell}\,\boldsymbol{u}_{N\ell}^\star\Big\} \tag{41}$$

The independence assumption (Assumption 1), and the statistical properties of noise $z_k(n)$ and perturbations $\boldsymbol{\epsilon}_k(n)$, lead us to the following expected values for $\boldsymbol{B}(n)$, $\boldsymbol{g}(n)$ and $\boldsymbol{r}(n)$:

$$\boldsymbol{B} = \boldsymbol{\mathcal{A}}_2^\top\,(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H})\,\boldsymbol{\mathcal{A}}_1^\top \tag{42}$$

$$\boldsymbol{g} = 0 \tag{43}$$

$$\boldsymbol{r} = \underbrace{\boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{U}\,\boldsymbol{h}_u}_{\boldsymbol{r}_u} - \underbrace{\Big(\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H})(\boldsymbol{\mathcal{A}}_1^\top - \boldsymbol{I}_{NL}) + (\boldsymbol{\mathcal{A}}_2^\top - \boldsymbol{I}_{NL})\Big)\,\boldsymbol{w}^\star}_{\boldsymbol{r}_w} \tag{44}$$

where $\boldsymbol{r}_u$, $\boldsymbol{r}_\epsilon$ and $\boldsymbol{r}_w$ denote the expected values of $\boldsymbol{r}_u(n)$, $\boldsymbol{r}_\epsilon(n)$ and $\boldsymbol{r}_w(n)$, respectively. Note that the expected value $\boldsymbol{r}$ is expressed as $\boldsymbol{r} = \boldsymbol{r}_u - \boldsymbol{r}_w$ because $\boldsymbol{r}_\epsilon = 0$. Taking the expectation of both sides of (34), we get

$$E\{\boldsymbol{v}^\star(n+1)\} = \boldsymbol{B}E\{\boldsymbol{v}^\star(n)\} - \boldsymbol{r}_u + \boldsymbol{r}_w \tag{45}$$

Moreover, equation (17) tells us that

$$E\{\boldsymbol{v}(n+1)\} = E\{\boldsymbol{v}^\star(n+1)\} \tag{46}$$

Theorem 1: **(Stability in the mean)** Assume data model (1) and Assumption 1 hold. Then, for any initial condition, the diffusion LMS strategy (6)–(8) applied to multitask networks asymptotically converges in the mean if the step sizes are chosen to satisfy

$$0 < \mu_k < \frac{2}{\lambda_{\max}\{\boldsymbol{R}_k\}}, \qquad k = 1, \dots, N \tag{47}$$

where $\lambda_{\max}\{\cdot\}$ denotes the maximum eigenvalue of its matrix argument. In that case, it follows from (45) that the asymptotic mean bias is given by

$$E\{\boldsymbol{v}(\infty)\} = (\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1}\,(-\boldsymbol{r}_u + \boldsymbol{r}_w) \tag{48}$$

Proof: Since the last two terms on the RHS of (45) are constant, the convergence of this recursion requires that the matrix $\boldsymbol{B}$ be stable. As shown in [36], because $\boldsymbol{\mathcal{A}}_1$ and $\boldsymbol{\mathcal{A}}_2$ are left-stochastic, $\boldsymbol{\mathcal{A}}_2^\top(\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H})\boldsymbol{\mathcal{A}}_1^\top$ is stable if the block diagonal matrix $\boldsymbol{I}_{NL} - \boldsymbol{U}\boldsymbol{H}$ is stable, which yields condition (47). ■

Inspecting expression (48), we observe that the bias of diffusion LMS originates from the multiple local optimums $\boldsymbol{w}_k^\star(n)$ and information exchange among neighbors. This means that, even though the algorithm converges toward the Pareto optimum over multitask networks [21], the bias (48) can be large if the distance between the $\boldsymbol{w}_k^\star(n)$ is large, and if nodes cooperate to estimate them. On the other hand, if nodes estimate parameter vectors $\boldsymbol{w}_k^\star(n)$ that are close to each other, this bias is close to 0 since $-\boldsymbol{r}_u + \boldsymbol{r}_w$ characterizes the distances between the $\boldsymbol{w}_k^\star(n)$. In particular, if $\boldsymbol{w}_k^\star(n) = \boldsymbol{w}^\star(n)$ for all k, then $-\boldsymbol{r}_u + \boldsymbol{r}_w = 0$. If there is no information exchange between nodes, that is, $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \boldsymbol{C} = \boldsymbol{I}$, we get $-\boldsymbol{r}_u + \boldsymbol{r}_w = 0$. The algorithm is this case operates in non-cooperative mode.

B. Mean-square error behavior analysis

Using Assumption 1, equation (38), and $E\{\boldsymbol{g}(n)\} = 0$, the mean-square of the weight error vector $\boldsymbol{v}^\star(n+1)$ weighted by any positive semi-definite matrix $\boldsymbol{\Sigma}$ satisfies the following relation:

$$E\{\|\boldsymbol{v}^\star(n+1)\|^2_{\Sigma}\} = E\{\|\boldsymbol{v}^\star(n)\|^2_{\Sigma'}\} + \text{trace}\left\{\boldsymbol{\Sigma}\, E\{\boldsymbol{g}(n)\boldsymbol{g}^\top(n)\}\right\} + E\{\|\boldsymbol{r}(n)\|^2_{\Sigma}\} - 2E\{\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{B}(n)\,\boldsymbol{v}^\star(n)\} \tag{49}$$

where $\boldsymbol{\Sigma}' = E\{\boldsymbol{B}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{B}(n)\}$. In this expression, the freedom in selecting $\boldsymbol{\Sigma}$ will allow us to derive several performance metrics. Let

$$\begin{aligned}\boldsymbol{G} &= E\{\boldsymbol{g}(n)\boldsymbol{g}^\top(n)\}\\ &= \boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{U}\,\boldsymbol{\mathcal{C}}^\top\,\text{diag}\{\sigma_{z,1}^2\boldsymbol{R}_{x,1}, \dots, \sigma_{z,N}^2\boldsymbol{R}_{x,N}\}\,\boldsymbol{\mathcal{C}}\,\boldsymbol{U}\,\boldsymbol{\mathcal{A}}_2\end{aligned} \tag{50}$$

where $\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_L$. For the sake of clarity, let us introduce

$$\boldsymbol{f}(\boldsymbol{r}(n), \boldsymbol{\Sigma}, \boldsymbol{v}^\star(n)) = \|\boldsymbol{r}(n)\|^2_{\Sigma} - 2\,\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{B}(n)\,\boldsymbol{v}^\star(n). \tag{51}$$

Relation (49) can then be written as

$$E\{\|\boldsymbol{v}^\star(n+1)\|^2_{\Sigma}\} = E\{\|\boldsymbol{v}^\star(n)\|^2_{\Sigma'}\} + \text{trace}\,\{\boldsymbol{\Sigma}\boldsymbol{G}\} + E\{\boldsymbol{f}(\boldsymbol{r}(n), \boldsymbol{\Sigma}, \boldsymbol{v}^\star(n))\}. \tag{52}$$

Let vec$\{\cdot\}$ denote the operator that stacks the columns of a matrix on top of each other. Vectorizing both matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}'$ by $\boldsymbol{\sigma} = \text{vec}\{\boldsymbol{\Sigma}\}$ and $\boldsymbol{\sigma}' = \text{vec}\{\boldsymbol{\Sigma}'\}$, it can be checked that

$$\boldsymbol{\sigma}' = \boldsymbol{K}\,\boldsymbol{\sigma} \tag{53}$$

where $\boldsymbol{K}$ is the $(NL)^2 \times (NL)^2$ matrix given by

$$\begin{aligned}\boldsymbol{K} &= E\{\boldsymbol{B}^\top(n) \otimes \boldsymbol{B}^\top(n)\} \\ &= (\boldsymbol{\mathcal{A}}_1\boldsymbol{\mathcal{A}}_2) \otimes (\boldsymbol{\mathcal{A}}_1\boldsymbol{\mathcal{A}}_2) - (\boldsymbol{\mathcal{A}}_1\boldsymbol{H}\boldsymbol{U}\boldsymbol{\mathcal{A}}_2) \otimes (\boldsymbol{\mathcal{A}}_1\boldsymbol{\mathcal{A}}_2) - (\boldsymbol{\mathcal{A}}_1\boldsymbol{\mathcal{A}}_2) \otimes (\boldsymbol{\mathcal{A}}_1\boldsymbol{H}\boldsymbol{U}\boldsymbol{\mathcal{A}}_2) \\ &\quad + E\{(\boldsymbol{\mathcal{A}}_1\boldsymbol{H}(n)\,\boldsymbol{U}\boldsymbol{\mathcal{A}}_2) \otimes (\boldsymbol{\mathcal{A}}_1\boldsymbol{H}(n)\,\boldsymbol{U}\boldsymbol{\mathcal{A}}_2)\}\end{aligned} \tag{54}$$

For sufficiently small step sizes, the effect of the fourth term on the second RHS of equation (54) can be ignored since it is quadratic with respect to step sizes, $\{\mu_k\mu_\ell\}$. A reasonable approximate expression for $\boldsymbol{K}$ is then

$$\boldsymbol{K} \approx \boldsymbol{B}^\top \otimes \boldsymbol{B}^\top \tag{55}$$

since it only differs from (54) by a term that depends on the square of the step-sizes.

Let us now examine the term $E\{\boldsymbol{f}(\boldsymbol{r}(n), \boldsymbol{\Sigma}, \boldsymbol{v}^\star(n))\}$. Consider first the weighted norm $E\{\|\boldsymbol{r}(n)\|^2_\Sigma\}$:

$$E\{\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}(n)\} = E\{\boldsymbol{r}_u^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}_u(n)\} + E\{\boldsymbol{r}_\epsilon^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}_\epsilon(n)\} + E\{\boldsymbol{r}_w^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}_w(n)\} - 2\,E\{\boldsymbol{r}_u^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}_w(n)\} \tag{56}$$

We observe that the stochastic components in $\boldsymbol{r}_u(n)$, $\boldsymbol{r}_\epsilon(n)$ and $\boldsymbol{r}_w(n)$ are quadratic with respect to the step sizes with terms of the form $\{\mu_k\mu_\ell\}$ because of the presence of the step size matrix $\boldsymbol{U}$ (see (37) for their expressions). For sufficiently small step sizes, and following the same reasoning as for matrix $\boldsymbol{K}$, a reasonable approximation is then

$$\begin{aligned}E\{\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{r}(n)\} &\approx \boldsymbol{r}_u^\top\,\boldsymbol{\Sigma}\,\boldsymbol{r}_u + \boldsymbol{r}_\epsilon^\top\,\boldsymbol{\Sigma}\,\boldsymbol{r}_\epsilon + \boldsymbol{r}_w^\top\,\boldsymbol{\Sigma}\,\boldsymbol{r}_w - 2\,\boldsymbol{r}_u^\top\,\boldsymbol{\Sigma}\,\boldsymbol{r}_w \\ &= \|\boldsymbol{r}\|^2_{\boldsymbol{\Sigma}}\end{aligned} \tag{57}$$

Likewise, using the small step size assumption, with Assumption 1, a reasonable approximation for $E\{\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{B}(n)\,\boldsymbol{v}^\star(n)\}$ is

$$E\{\boldsymbol{r}^\top(n)\,\boldsymbol{\Sigma}\,\boldsymbol{B}(n)\,\boldsymbol{v}^\star(n)\} \approx \boldsymbol{r}^\top\,\boldsymbol{\Sigma}\,\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\} \tag{58}$$

It follows that $E\{\boldsymbol{f}(\boldsymbol{r}(n), \boldsymbol{\Sigma}, \boldsymbol{v}^\star(n))\}$ can be approximated by

$$\begin{aligned}E\{\boldsymbol{f}(\boldsymbol{r}(n), \boldsymbol{\Sigma}, \boldsymbol{v}^\star(n))\} &\approx \|\boldsymbol{r}\|^2_\Sigma - 2\,\boldsymbol{r}^\top\,\boldsymbol{\Sigma}\,\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\} \\ &= \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{\Sigma}, E\{\boldsymbol{v}^\star(n)\}).\end{aligned} \tag{59}$$

In this way, relation (52) can be approximated as follows:

$$E\{\|\boldsymbol{v}^\star(n+1)\|^2_\sigma\} = E\{\|\boldsymbol{v}^\star(n)\|^2_{K\sigma}\} + \text{vec}(\boldsymbol{G}^\top)^\top\boldsymbol{\sigma} + \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{\sigma}, E\{\boldsymbol{v}^\star(n)\}). \tag{60}$$

where we are using the notation $\|\cdot\|^2_{\boldsymbol{\Sigma}}$ and $\|\cdot\|^2_{\boldsymbol{\sigma}}$ interchangeably to refer to the same square weighted norm using $\boldsymbol{\Sigma}$ or its vector representation.

Theorem 2: **(Mean-square stability)** Assume data model (1) and Assumption 1 hold. Assume that the step sizes $\{\mu_k\}$ are sufficiently small such that condition (47) is satisfied and approximations (55) and (59) are justified by ignoring higher-order powers of $\{\mu_k\}$. Then, the diffusion LMS strategy (6)–(8) applied over multitask networks is mean-square stable if the matrix $\boldsymbol{K}$ is stable. Under approximation (55), the stability of $\boldsymbol{K}$ is guaranteed for sufficiently small step-sizes that also satisfy (47).

Proof: Iterating (60) starting from $n = 0$, we find that

$$E\{\|\boldsymbol{v}^\star(n+1)\|^2_\sigma\} = \|\boldsymbol{v}^\star(0)\|^2_{K^{n+1}\sigma} + \text{vec}(\boldsymbol{G}^\top)^\top\sum_{i=0}^{n}\boldsymbol{K}^i\boldsymbol{\sigma} + \sum_{i=0}^{n}\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{K}^i\boldsymbol{\sigma}, E\{\boldsymbol{v}^\star(n-i)\}) \tag{61}$$

with a deterministic initial condition $\boldsymbol{v}^\star(0) = \boldsymbol{w}(0) - \boldsymbol{w}^\star$. Provided that $\boldsymbol{K}$ is stable, the first and second terms on the RHS of (61) converge as $n \to \infty$, to zero for the former, and to a finite value for the latter. Consider now the third term on the RHS of (61). We know from (45) that $E\{\boldsymbol{v}^\star(n)\}$ is bounded because (45) is a BIBO stable recursion with a bounded driving term $-\boldsymbol{r}_u + \boldsymbol{r}_w$. Moreover, from the expression in (59),

$$\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{K}^i\boldsymbol{\sigma}, E\{\boldsymbol{v}^\star(n-1)\}) = \left(\text{vec}\{\boldsymbol{r}\boldsymbol{r}^\top\}\right)^\top \boldsymbol{K}^i\boldsymbol{\sigma} - 2\left(\boldsymbol{B}E\{\boldsymbol{v}^\star(n-i)\} \otimes \boldsymbol{r}\right)^\top \boldsymbol{K}^i\boldsymbol{\sigma}. \tag{62}$$

Since $\boldsymbol{K}$ is stable, there exists a matrix norm [36], denoted by $\|\cdot\|_\rho$, such that $\|\boldsymbol{K}\|_\rho = c_\rho < 1$. Applying this norm to the above equation and using the triangular inequality, we can deduce that

$$\|\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{K}^i\boldsymbol{\sigma}, E\{\boldsymbol{v}^\star(n-i)\})\|_\rho < \nu\, c_\rho^i \tag{63}$$

for some positive finite constant ν. It follows that the sum in the right-most term of (61) converges as $n \to \infty$. We conclude that $E\{\|\boldsymbol{v}^\star(n+1)\|^2_\sigma\}$ converges to a bounded value as $n \to \infty$, and the algorithm is mean-square stable for sufficiently small step-sizes. ■

Corollary 1: **(Transient MSD)** Considering sufficiently small step sizes μ_k that ensure mean and mean-square stability, and selecting $\boldsymbol{\Sigma} = \frac{1}{N}\boldsymbol{I}_{NL}$, then the MSD learning curve of the diffusion LMS algorithm in a multitask environment, defined by $\zeta(n) = \frac{1}{N}E\{\|\boldsymbol{v}(n)\|^2\}$, evolves according to the following recursion for $n \geq 0$

$$\zeta(n) = \zeta^\star(n) + \frac{L}{N}\sum_{k=1}^{N}\sigma^2_{\epsilon,k} \tag{64}$$

where $\zeta^\star(n)$ is evaluated as follows

$$\begin{aligned}\zeta^\star(n+1) = \zeta^\star(n) + \frac{1}{N}\Big(&\left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top \boldsymbol{K}^n\boldsymbol{\sigma}_I - \|\boldsymbol{v}^\star(0)\|^2_{(I_{(NL)^2}-K)K^n\sigma_I} + \|\boldsymbol{r}\|^2_{K^n\sigma_I} \\ &- 2\left(\boldsymbol{\Gamma}(n) + (\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\})^\top \otimes \boldsymbol{r}^\top\right)\boldsymbol{\sigma}_I\Big)\end{aligned} \tag{65}$$

$$\boldsymbol{\Gamma}(n+1) = \boldsymbol{\Gamma}(n)\boldsymbol{K} + (\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\})^\top \otimes \boldsymbol{r}^\top)(\boldsymbol{K} - \boldsymbol{I}_{(NL)^2}) \tag{66}$$

with $\boldsymbol{\sigma}_I = \text{vec}\{\frac{1}{N}\boldsymbol{I}_{NL}\}$, and the initial conditions $\zeta^\star(0) = \frac{1}{N}\|\boldsymbol{v}^\star(0)\|^2$ and $\boldsymbol{\Gamma}(0) = \boldsymbol{0}_{1\times(NL)^2}$.

Proof: The MSD at time instant n can be expressed as a function of $\boldsymbol{v}^\star(n)$ as

$$\begin{aligned}\zeta(n) &= \frac{1}{N}E\{\|\boldsymbol{v}(n)\|^2\} \\ &= \frac{1}{N}E\{\|\boldsymbol{v}^\star(n)\|^2\} + \frac{L}{N}\sum_{k=1}^{N}\sigma^2_{\epsilon,k}.\end{aligned} \tag{67}$$

Comparing (61) at time instants $n+1$ and n, we can relate $E\{\|\boldsymbol{v}^\star(n+1)\|^2_\Sigma\}$ and $E\{\|\boldsymbol{v}^\star(n)\|^2_\Sigma\}$ as follows:

$$\begin{aligned}E\{\|\boldsymbol{v}^\star(n+1)\|^2_\Sigma\} = E\{\|\boldsymbol{v}^\star(n)\|^2_\Sigma\} &+ \left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top \boldsymbol{K}^n\boldsymbol{\sigma} - \|\boldsymbol{v}^\star(0)\|^2_{(I_{(NL)^2}-K)K^n\sigma} \\ &+ \sum_{i=0}^{n}\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{K}^i\boldsymbol{\sigma}, E\{\boldsymbol{v}(n-i)\}) - \sum_{i=0}^{n-1}\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{K}^i\boldsymbol{\sigma}, E\{\boldsymbol{v}(n-1-i)\}).\end{aligned} \tag{68}$$

Expression (68) can be reformulated as (see [34, Theorem 3] for a similar derivation showing how to get from (68) to (70)):

$$\begin{aligned}E\{\|\boldsymbol{v}^\star(n+1)\|^2_\Sigma\} = E\{\|\boldsymbol{v}^\star(n)\|^2_\Sigma\} &+ \left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top \boldsymbol{K}^n\boldsymbol{\sigma} - \|\boldsymbol{v}^\star(0)\|^2_{(I_{(NL)^2}-K)K^n\sigma} + \|\boldsymbol{r}\|^2_{K^n\sigma} \\ &- 2\left(\boldsymbol{\Gamma}(n) + (\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\})^\top \otimes \boldsymbol{r}^\top\right)\boldsymbol{\sigma}\end{aligned} \tag{69}$$

$$\boldsymbol{\Gamma}(n+1) = \boldsymbol{\Gamma}(n)\boldsymbol{K} + (\boldsymbol{B}\,E\{\boldsymbol{v}^\star(n)\})^\top \otimes \boldsymbol{r}^\top)(\boldsymbol{K} - \boldsymbol{I}_{(NL)^2}) \tag{70}$$

with $\boldsymbol{\Gamma}(0) = \mathbf{0}_{(LN)^2}$. To derive the transient curve for the MSD, we replace $\boldsymbol{\sigma}$ by $\text{vec}\{\frac{1}{N}\boldsymbol{I}_{NL}\}$. ■

Corollary 2: **(Steady-state MSD)** If the step sizes are chosen sufficiently small to ensure mean and mean-square-error convergences, then the steady-state MSD for diffusion LMS in a multitask environment is given by

$$\begin{aligned}\text{MSD}^{\text{network}} &= \frac{1}{N}\left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top (\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\} \\ &\quad + \boldsymbol{f}\Big(\boldsymbol{r}, \frac{1}{N}(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\}, E\{\boldsymbol{v}^\star(\infty)\}\Big) + \frac{L}{N}\sum_{k=1}^{N}\sigma_{\epsilon,k}^2\end{aligned} \tag{71}$$

with $E\{\boldsymbol{v}(\infty)\}$ determined by (48).

Proof: The steady-state MSD is given by the limit

$$\begin{aligned}\text{MSD}^{\text{network}} &= \lim_{n\to\infty}\frac{1}{N}E\{\|\boldsymbol{v}(n)\|^2\} \\ &= \lim_{n\to\infty}\frac{1}{N}E\{\|\boldsymbol{v}^\star(n)\|^2\} + \frac{L}{N}\sum_{k=1}^{N}\sigma_{\epsilon,k}^2.\end{aligned} \tag{72}$$

Recursion (60) with $n \to \infty$ yields

$$\lim_{n\to\infty} E\{\|\boldsymbol{v}^\star(n)\|^2_{(I_{(NL)^2}-K)\sigma}\} = \left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top \boldsymbol{\sigma} + \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{\sigma}, E\{\boldsymbol{v}^\star(\infty)\}). \tag{73}$$

In order to use (73) in (72), we select $\boldsymbol{\sigma}$ to satisfy:

$$(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})\,\boldsymbol{\sigma} = \frac{1}{N}\text{vec}\{\boldsymbol{I}_{NL}\}. \tag{74}$$

This leads to expression (71). ■

The steady-state MSD can be expressed in an alternative form, which will facilitate the performance analysis. Since $\boldsymbol{K}$ is stable when the network is mean-square stable, we can write

$$(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1} = \sum_{i=0}^{\infty}\boldsymbol{K}^i = \sum_{i=0}^{\infty}(\boldsymbol{B}^\top \otimes \boldsymbol{B}^\top)^i. \tag{75}$$

Consider now the following formula involving the trace of a product of matrices and the Kronecker product [40]

$$\text{trace}\{\boldsymbol{X}_1^\top\,\boldsymbol{X}_2\,\boldsymbol{X}_3\,\boldsymbol{X}_4^\top\} = \text{vec}\{\boldsymbol{X}_1\}^\top(\boldsymbol{X}_4 \otimes \boldsymbol{X}_2)\,\text{vec}\{\boldsymbol{X}_3\} \tag{76}$$

where $\boldsymbol{X}_1$ to $\boldsymbol{X}_4$ denote matrices with compatible sizes. Using expansion (75) with (76), the first term on the RHS of (71) can be expressed as follows

$$\frac{1}{N}\left(\text{vec}\{\boldsymbol{G}^\top\}\right)^\top (\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\} = \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\{\boldsymbol{B}^j\,\boldsymbol{G}\,\boldsymbol{B}^{j\top}\} \tag{77}$$

Similarly, the second term on the RHS of (71) can be written as

$$\begin{aligned}&\boldsymbol{f}\Big(\boldsymbol{r}, \frac{1}{N}(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\}, E\{\boldsymbol{v}^\star(\infty)\}\Big) \\ &\quad = \text{vec}\{\boldsymbol{r}\boldsymbol{r}^\top\}^\top\frac{1}{N}(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\} - 2\,\text{vec}\{(\boldsymbol{B}E\{\boldsymbol{v}^\star(\infty)\}\,\boldsymbol{r}^\top)^\top\}\,\frac{1}{N}(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\} \\ &\quad = \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\{\boldsymbol{B}^j\,[\boldsymbol{r}\boldsymbol{r}^\top - 2\,\boldsymbol{B}E\{\boldsymbol{v}^\star(\infty)\}\,\boldsymbol{r}^\top]\,\boldsymbol{B}^{j\top}\}\end{aligned} \tag{78}$$

Substituting expression (48) of $E\{\boldsymbol{v}^\star(\infty)\}$ into this equation, we get

$$\boldsymbol{f}\Big(\boldsymbol{r}, \frac{1}{N}(\boldsymbol{I}_{(NL)^2} - \boldsymbol{K})^{-1}\text{vec}\{\boldsymbol{I}_{NL}\}, E\{\boldsymbol{v}^\star(\infty)\}\Big) = \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\{\boldsymbol{B}^j[\boldsymbol{I}_{NL} + 2\,\boldsymbol{B}(\boldsymbol{I} - \boldsymbol{B})^{-1}]\,\boldsymbol{r}\boldsymbol{r}^\top\boldsymbol{B}^{j\top}\} \tag{79}$$

Finally, we can express the steady-state MSD (71) as

$$\text{MSD}^{\text{network}} = \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\left\{\boldsymbol{B}^j\left(\boldsymbol{G}+(\boldsymbol{I}_{NL}+2\boldsymbol{B}(\boldsymbol{I}_{NL}-\boldsymbol{B})^{-1})\,\boldsymbol{r}\boldsymbol{r}^\top\right)\boldsymbol{B}^{j^\top}\right\}+\frac{L}{N}\sum_{k=1}^{N}\sigma_{\epsilon,k}^2. \tag{80}$$

In the sequel, this formulation will allow us to compare the performance of different algorithms.

C. Performance comparison with non-cooperative LMS

We shall now compare the performance of the ATC and CTA diffusion LMS algorithms with the non-cooperative LMS strategy when applied to a multitask network. We consider the case of uniform step sizes, $\mu_k = \mu$, for a meaningful comparison. Diffusion LMS degenerates to non-cooperative LMS by setting

$$\boldsymbol{C} = \boldsymbol{I}_N, \quad \boldsymbol{A}_1 = \boldsymbol{I}_N, \quad \boldsymbol{A}_2 = \boldsymbol{I}_N \tag{81}$$

from which the performance of the latter can be easily derived. In this case, matrices $\boldsymbol{B}$ and $\boldsymbol{G}$ reduce to

$$\boldsymbol{B}_{\text{lms}} = \boldsymbol{I}_{NL} - \mu\boldsymbol{H} \tag{82}$$

$$\boldsymbol{G}_{\text{lms}} = \mu^2\,\text{diag}\{\sigma_{z,1}^2\boldsymbol{R}_{x,1},\ldots,\sigma_{z,N}^2\boldsymbol{R}_{x,N}\} \tag{83}$$

where we use the subscript LMS for clarity. Note that in this case we have $\boldsymbol{r}_w(n) = 0$. In addition, since $\mathcal{N}_k = \{k\}$ and $\boldsymbol{u}_{kk}^\star = 0$, we have $\boldsymbol{r}_u(n) = 0$. This implies that $\boldsymbol{r} = 0$. The steady-state MSD for non-cooperative LMS is then given by:

$$\text{MSD}_{\text{lms}}^{\text{network}} = \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\left(\boldsymbol{B}_{\text{lms}}^j\,\boldsymbol{G}_{\text{lms}}\,\boldsymbol{B}_{\text{lms}}^{j^\top}\right)+\frac{L}{N}\sum_{k=1}^{N}\sigma_{\epsilon,k}^2. \tag{84}$$

It is useful to note that the matrices $\boldsymbol{B}$ and $\boldsymbol{G}$ for diffusion LMS can be expressed in terms of $\boldsymbol{B}_{\text{lms}}$ and $\boldsymbol{G}_{\text{lms}}$:

$$\begin{aligned}\boldsymbol{B} &= \boldsymbol{\mathcal{A}}_2^\top\,(\boldsymbol{I}_{NL}-\mu\boldsymbol{H})\,\boldsymbol{\mathcal{A}}_1^\top\\ &= \boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{B}_{\text{lms}}\,\boldsymbol{\mathcal{A}}_1^\top\\ \boldsymbol{G} &= \mu^2\,\boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{\mathcal{C}}^\top\,\text{diag}\{\sigma_{z,1}^2\boldsymbol{R}_{x,1},\ldots,\sigma_{z,N}^2\boldsymbol{R}_{x,N}\}\,\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{A}}_2\\ &= \boldsymbol{\mathcal{A}}_2^\top\,\boldsymbol{\mathcal{C}}^\top\,\boldsymbol{G}_{\text{lms}}\,\boldsymbol{\mathcal{C}}\,\boldsymbol{\mathcal{A}}_2\end{aligned} \tag{85}$$

with $\boldsymbol{A}_1 = \boldsymbol{I}_N$ for the ATC diffusion strategy, and $\boldsymbol{A}_2 = \boldsymbol{I}_N$ for the CTA diffusion strategy. Using the series expansions for $\text{MSD}^{\text{network}}$ and $\text{MSD}_{\text{lms}}^{\text{network}}$, the difference between the MSDs for non-cooperative LMS and diffusion LMS is given by

$$\begin{aligned}\text{MSD}_{\text{lms}}^{\text{network}} - \text{MSD}^{\text{network}} &= \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\left\{\boldsymbol{B}_{\text{lms}}^j\,\boldsymbol{G}_{\text{lms}}\,\boldsymbol{B}_{\text{lms}}^{j^\top} - \boldsymbol{B}^j\,\boldsymbol{G}\,\boldsymbol{B}^{j^\top}\right\}\\ &\quad - \frac{1}{N}\sum_{j=0}^{\infty}\text{trace}\left\{\boldsymbol{B}^j\,(\boldsymbol{I}_{NL}+2\boldsymbol{B}(\boldsymbol{I}_{NL}-\boldsymbol{B})^{-1})\,\boldsymbol{r}\boldsymbol{r}^\top)\,\boldsymbol{B}^{j^\top}\right\}.\end{aligned} \tag{86}$$

Note that the first term is the difference in performance between the non-cooperative LMS strategy and the cooperative diffusion strategy. It was first analyzed in [36] and, because it is not specific to the multitask context, it is denoted by $\Delta\text{MSD}^{\text{network}}$. Only the second term, which depends on $\boldsymbol{r}$, is specific to the multitask scenario. Thus, it is denoted by $\Delta\text{MSD}_{\text{multi}}^{\text{network}}(\boldsymbol{r})$. Therefore,

$$\text{MSD}_{\text{lms}}^{\text{network}} - \text{MSD}^{\text{network}} = \Delta\text{MSD}^{\text{network}} - \Delta\text{MSD}_{\text{multi}}^{\text{network}}(\boldsymbol{r}) \tag{87}$$

In order to obtain analytical results that allow some understanding of the algorithm behavior, we further assume that the matrices $\boldsymbol{C}$, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ in the diffusion implementation are doubly stochastic, and the regression covariance matrices are uniform across the agents, that is, $\boldsymbol{R}_{x,k} = \boldsymbol{R}_x$. With these assumptions, it was shown in [36] that the first term $\Delta\text{MSD}^{\text{network}}$ on the RHS of (86) is always nonnegative, namely,

$$\text{trace}\left\{\boldsymbol{B}^j_{\text{lms}}\,\boldsymbol{G}_{\text{lms}}\,\boldsymbol{B}^{j\top}_{\text{lms}} - \boldsymbol{B}^j\,\boldsymbol{G}\,\boldsymbol{B}^{j\top}\right\} \geq 0. \tag{88}$$

We need to check under which conditions the second term $\Delta\text{MSD}^{\text{network}}_{\text{multi}}(\boldsymbol{r})$ on the RHS of equation (86) is nonnegative so that it can be viewed as a degradation factor caused by the multitask scenario. Introduce the symmetric matrix $\boldsymbol{Z} = \boldsymbol{I}_{NL} + 2\boldsymbol{B}(\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1} = 2\,(\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1} - \boldsymbol{I}_{NL}$. We find that

$$\begin{aligned}\Delta\text{MSD}^{\text{network}}_{\text{multi}}(\boldsymbol{r}) &= \frac{1}{N}\,\boldsymbol{r}^\top\Big(\sum_{j=0}^{\infty}\boldsymbol{B}^{j\top}\boldsymbol{B}^j\Big)\,\boldsymbol{Z}\,\boldsymbol{r}\\ &= \frac{1}{N}\,\boldsymbol{r}^\top(\boldsymbol{I}_{NL} - \boldsymbol{B}^\top\boldsymbol{B})^{-1}\,\boldsymbol{Z}\,\boldsymbol{r}\end{aligned} \tag{89}$$

We conclude that expression (89) is non-negative for all $\boldsymbol{r}$ if, and only if, $(\boldsymbol{I}_{NL} - \boldsymbol{B}^\top\boldsymbol{B})^{-1}\,\boldsymbol{Z}$ is a symmetric positive semidefinite matrix. Now, we show that this condition is met for a large class of information exchange protocols. Assume, for instance, that either $\boldsymbol{A}_1$ or $\boldsymbol{A}_2$ is symmetric, depending on whether the focus is on the CTA or ATC strategy. Recalling conditions $\boldsymbol{R}_{x,k} = \boldsymbol{R}_x$ and $\mu_k = \mu$ for uniform data profile, it then holds that $\boldsymbol{B}$ is a symmetric matrix. Let us examine the eigenvalues of $(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}$ and $\boldsymbol{Z}$. They can be expressed in terms of the eigenvalues of $\boldsymbol{B}$ as:

$$\begin{aligned}\lambda_k\{(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}\} &= \frac{1}{1 - \lambda_k^2\{\boldsymbol{B}\}}\\ \lambda_k\{\boldsymbol{Z}\} &= \frac{2}{1 - \lambda_k\{\boldsymbol{B}\}} - 1\end{aligned} \tag{90}$$

for all k, where $\lambda_k\{\cdot\}$ denotes the k-th eigenvalue of its matrix argument. It follows that $(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}$ and $\boldsymbol{Z}$ are positive definite if $\boldsymbol{B}$ is stable. Now, we need to check that the product $(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}\boldsymbol{Z}$ is a symmetric positive semidefinite matrix. To proceed, we consider the following property:

Lemma 1: **(Positivity of a matrix product)** [41] Given two symmetric positive semidefinite matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ with compatible sizes. Then, $\boldsymbol{X}_1\boldsymbol{X}_2$ is symmetric positive semidefinite if, and only if, $\boldsymbol{X}_1\boldsymbol{X}_2$ is normal, that is, if the following equality is satisfied: $(\boldsymbol{X}_1\boldsymbol{X}_2)(\boldsymbol{X}_1\boldsymbol{X}_2)^\top = (\boldsymbol{X}_1\boldsymbol{X}_2)^\top(\boldsymbol{X}_1\boldsymbol{X}_2)$. ■

By setting $\boldsymbol{X}_1 = (\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}$ and $\boldsymbol{X}_2 = \boldsymbol{Z}$, observe that $\boldsymbol{X}_1\boldsymbol{X}_2$ is symmetric:

$$\begin{aligned}\boldsymbol{X}_1\boldsymbol{X}_2 &= (\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}[2\,(\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1} - \boldsymbol{I}_{NL}]\\ &= 2\,(\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1}(\boldsymbol{I}_{NL} + \boldsymbol{B})^{-1}(\boldsymbol{I}_{NL} - \boldsymbol{B})^{-1} - (\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1} = (\boldsymbol{X}_1\boldsymbol{X}_2)^\top\end{aligned} \tag{91}$$

It then holds that $(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}\,\boldsymbol{Z}$ is normal. By Lemma 1, $(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}\,\boldsymbol{Z}$ is a symmetric positive semidefinite matrix, which means that $\Delta\text{MSD}^{\text{network}}_{\text{multi}}(\boldsymbol{r})$ is nonnegative under the conditions specified above. It follows that this term can be viewed as a degradation factor caused by the cooperation of nodes performing different estimation tasks, which can be expressed as $\frac{1}{N}\|\boldsymbol{r}\|^2_{(\boldsymbol{I}_{NL} - \boldsymbol{B}^2)^{-1}\boldsymbol{Z}}$. We summarize the results in the following statement.

Theorem 3: **(Non-cooperative vs. cooperative strategies MSD)** Consider the same setting of Theorems 1 and 2, with the additional requirement that the conditions for a uniform data profile hold. The adaptive ATC or CTA diffusion strategies perform better than the non-cooperative strategy under doubly stochastic $\boldsymbol{A}$ and $\boldsymbol{C}$ if

$$\Delta\text{MSD}^{\text{network}} - \Delta\text{MSD}^{\text{network}}_{\text{multi}}(\boldsymbol{r}) \geq 0 \tag{92}$$

Assuming that matrix $\boldsymbol{A}$ is symmetric, then $\Delta\text{MSD}_{\text{multi}}^{\text{network}}(\boldsymbol{r})$ is positive. In this case, the multi-task environment leads to a degradation in MSD performance that is given by

$$\Delta\text{MSD}_{\text{multi}}^{\text{network}}(\boldsymbol{r}) = \frac{1}{N}\|\boldsymbol{r}\|^2_{(\boldsymbol{I}_{NL}-\boldsymbol{B}^2)^{-1}\boldsymbol{Z}} \tag{93}$$

where $\boldsymbol{Z} = 2\,(\boldsymbol{I}_{NL}-\boldsymbol{B})^{-1} - \boldsymbol{I}_{NL}$.

Although condition (92) allows to determine whether using the diffusion LMS is beneficial for multitask learning compared to the non-cooperative LMS strategy, it cannot be easily exploited to estimate appropriate combination coefficients because of its complexity and the need to handle dynamic problems. The aim of the next section is to derive an efficient strategy to estimate these coefficients.

IV. Node clustering via combination matrix selection

We now derive a clustering strategy where each node k can adjust the combination weights $a_{\ell k}$ in an online manner, for $\ell \in \mathcal{N}_k$, in order to adapt to multitask environments. It is sufficient to focus on the adapt-then-combine diffusion LMS defined by steps (7) and (8). For ease of presentation, the corresponding algorithm is summarized below:

$$\begin{cases} \boldsymbol{\psi}_k(n+1) = \boldsymbol{w}_k(n) + \mu_k \displaystyle\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\, \boldsymbol{x}_\ell(n)\big[d_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{w}_k(n)\big] \\ \boldsymbol{w}_k(n+1) = \displaystyle\sum_{\ell\in\mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_\ell(n+1) \end{cases} \tag{94}$$

where $a_{\ell k}$ is used instead of $a_{2,\ell k}$. As shown in the previous section, running (94) in a multitask environment leads to biased results. We now discuss how to cluster nodes in order to reduce this effect.

A. *Clustering via matrix $\boldsymbol{A}$ adjustments*

In the spirit of [35], we suggest to adjust matrix $\boldsymbol{A}$ in an online manner via MSD optimization. At each instant n, the instantaneous MSD at node k is given by

$$E\{\|\boldsymbol{v}_k^\star(n+1)\|^2\} = E\Big\{\|\boldsymbol{w}_k^\star - \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\, \boldsymbol{\psi}_\ell(n+1)\|^2\Big\} \tag{95}$$

provided that the optimum $\boldsymbol{w}_k^\star$ is known. Because the matrix $\boldsymbol{A}$ is assumed left-stochastic, this expression can be rewritten as

$$E\{\|\boldsymbol{v}_k^\star(n+1)\|^2)\} = \sum_{\ell\in\mathcal{N}_k}\sum_{p\in\mathcal{N}_k} a_{\ell k}\, a_{pk}\, E\left\{[\boldsymbol{w}_k^\star - \boldsymbol{\psi}_\ell(n+1)]^\top[\boldsymbol{w}_k^\star - \boldsymbol{\psi}_p(n+1)]\right\}. \tag{96}$$

Let $\boldsymbol{\Psi}_k$ be the matrix at each node k with (ℓ,p)-th entry defined as

$$[\boldsymbol{\Psi}_k]_{\ell p} = \begin{cases} E\left\{[\boldsymbol{w}_k^\star - \boldsymbol{\psi}_\ell(n+1)]^\top[\boldsymbol{w}_k^\star - \boldsymbol{\psi}_p(n+1)]\right\}, & \ell, p \in \mathcal{N}_k \\ 0, & \text{otherwise} \end{cases} \tag{97}$$

Let $\boldsymbol{a}_k = [a_{1k},\dots,a_{Nk}]^\top$. We can pose the following optimization problem for node k at time n:

$$\begin{aligned} &\min_{\boldsymbol{a}_k} \quad \boldsymbol{a}_k^\top\, \boldsymbol{\Psi}_k\, \boldsymbol{a}_k \\ &\text{subject to} \quad \boldsymbol{1}_N^\top\, \boldsymbol{a}_k = 1, \quad a_{\ell k} \geq 0, \\ &\qquad\qquad a_{\ell k} = 0 \ \text{ if } \ \ell \notin \mathcal{N}_k \end{aligned} \tag{98}$$

It is generally not possible to estimate the combination weights by solving this optimization problem because each node k needs to know the optimum $\boldsymbol{w}_k^\star$ and the cross-covariance terms, $[\boldsymbol{\Psi}_k]_{\ell p}$, which are not available beforehand. To make this problem tractable, we approximate the matrix $\boldsymbol{\Psi}_k$ by an instantaneous value, drop its off-diagonal entries, and use an approximation for $\boldsymbol{w}_k^\star$. Problem (98) is thus replaced by:

$$\begin{aligned} &\min_{\boldsymbol{a}_k} \quad \sum_{\ell=1}^{N} a_{\ell k}^2 \, \|\widehat{\boldsymbol{w}}_k^\star - \boldsymbol{\psi}_\ell(n+1))\|^2 \\ &\text{subject to} \quad \mathbf{1}_N^\top \boldsymbol{a}_k = 1, \quad a_{\ell k} \geq 0, \\ &\qquad\qquad a_{\ell k} = 0 \ \text{ if } \ \ell \notin \mathcal{N}_k \end{aligned} \tag{99}$$

where $\widehat{\boldsymbol{w}}_k^\star$ is some approximation for $\boldsymbol{w}_k^\star$. The solution to this problem is given by

$$a_{\ell k}(n+1) = \frac{\|\widehat{\boldsymbol{w}}_k^\star - \boldsymbol{\psi}_\ell(n+1))\|^{-2}}{\sum_{j\in\mathcal{N}_k} \|\widehat{\boldsymbol{w}}_k^\star - \boldsymbol{\psi}_j(n+1))\|^{-2}}, \qquad \text{for } \ell \in \mathcal{N}_k. \tag{100}$$

Let us now provide a reasonable approximation for $\boldsymbol{w}_k^\star$ to be used in (100). In order to reduce the MSD bias that results from the cooperation of nodes performing distinct estimation tasks, one strategy is to use the local one-step approximation:

$$\widehat{\boldsymbol{w}}_k^\star(n+1) = \boldsymbol{\psi}_k(n+1) - \mu_k \, \nabla J_k(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{\psi}_k(n+1)} \tag{101}$$

Since the true gradient of $J_k(\boldsymbol{w})$ at $\boldsymbol{\psi}_k(n{+}1)$ is not available in an adaptive implementation, we can approximate it by using the instantaneous value $\boldsymbol{q}_k(n) \triangleq e_k(n)\boldsymbol{x}_k(n)$ with $e_k(n) = [d_k(n) - \boldsymbol{x}_k^\top(n)\boldsymbol{\psi}_k(n+1)]$. This yields the following approximation:

$$\widehat{\boldsymbol{w}}_k^\star(n+1) = \boldsymbol{\psi}_k(n+1) + \mu_k \, \boldsymbol{q}_k(n) \tag{102}$$

Substituting this expression into (100), we get the combination rule

$$a_{\ell k}(n+1) = \frac{\|\boldsymbol{\psi}_k(n+1) + \mu_k \, \boldsymbol{q}_k(n) - \boldsymbol{\psi}_\ell(n+1)\|^{-2}}{\sum_{j\in\mathcal{N}_k} \|\boldsymbol{\psi}_k(n+1) + \mu_k \, \boldsymbol{q}_k(n) - \boldsymbol{\psi}_j(n+1)\|^{-2}}, \qquad \text{for } \ell \in \mathcal{N}_k. \tag{103}$$

This rule admits a useful interpretation. On the one hand, as mentioned above, it relies on the local estimate (101) in order to reduce the MSD bias effect caused by the cooperation of neighboring nodes estimating distinct parameter vectors. On the other hand, consider the inverse of the numerator of rule (103):

$$\begin{aligned} &\|\boldsymbol{\psi}_k(n+1) + \mu_k \, \boldsymbol{q}_k(n) - \boldsymbol{\psi}_\ell(n+1)\|^2 \\ &\qquad = \|\boldsymbol{\psi}_\ell(n+1) - \boldsymbol{\psi}_k(n+1)\|^2 + 2[\boldsymbol{\psi}_\ell(n+1) - \boldsymbol{\psi}_k(n+1)]^\top[-\mu_k \, \boldsymbol{q}_k(n)] + \mu_k^2 \, \|\boldsymbol{q}_k(n)\|^2. \end{aligned} \tag{104}$$

The first term $\|\boldsymbol{\psi}_\ell(n{+}1) - \boldsymbol{\psi}_k(n{+}1)\|^2$ on the RHS accounts for the distance of the current estimates between nodes k and ℓ; this term tends to decrease the combination weight $a_{\ell k}(n+1)$ if this distance is large, and to limit information exchange. Now, consider the first-order Taylor series expansion of $J_k(\boldsymbol{w})$ at $\boldsymbol{\psi}_k(n+1)$:

$$J_k(\boldsymbol{\psi}) \approx J_k(\boldsymbol{\psi}_k(n+1)) - [\boldsymbol{\psi} - \boldsymbol{\psi}_k(n+1)]^\top \boldsymbol{q}_k(n) \tag{105}$$

The second term $[\boldsymbol{\psi}_\ell(n{+}1) - \boldsymbol{\psi}_k(n{+}1)]^\top[-\mu_k \, \boldsymbol{q}_k(n)]$ on the RHS of (104) is proportional to $J_k(\boldsymbol{\psi}_\ell(n{+}1)) - J_k(\boldsymbol{\psi}_k(n{+}1))$. This term also tends to decrease the combination weight $a_{\ell k}(n{+}1)$ if $J_k(\boldsymbol{\psi}_\ell(n{+}1)) > J_k(\boldsymbol{\psi}_k(n{+}1))$. Indeed, in this case, it is not recommended to promote the combination of models $\boldsymbol{\psi}_k(n{+}1)$ and $\boldsymbol{\psi}_\ell(n{+}1)$ because the latter induces an increase of the cost function. Finally, the last term $\mu_k^2 \, \|\boldsymbol{q}_k(n)\|^2$ is the same for all $\ell \in \mathcal{N}_k$. To summarize this discussion, the combination rule (103) considers the closeness of the local estimate to the neighboring estimates, and the local slope of the cost function, to adjust the combination weights. This tends to promote information exchange between nodes that estimate the same optimum parameter vector, and thus to reduce the MSD bias and improve the estimation accuracy.

B. Algorithm

The flexibility of multitask networks may be exploited by considering distinct cost functions for each node. This raises the issue of sharing information via the exchange matrix $\boldsymbol{C}$, which can be simply set to the identity $\boldsymbol{I}_N$, i.e.,

$$\boldsymbol{C} = \boldsymbol{I}_N \tag{106}$$

However, the time-variant combination matrix $\boldsymbol{A}(n)$ determined by (103) describes how each agent combines the parameter vectors transmitted by its neighbors as a function of the estimated contrast between tasks. An additional way to exploit this information is that each agent uses the reciprocity principle defined by

$$\boldsymbol{C}(n+1) = \boldsymbol{A}^\top(n+1) \tag{107}$$

The rationale underlying this principle is that the magnitude of $a_{\ell k}$ reflects the similarity of the estimation tasks performed by nodes k and ℓ, as it is perceived by node k. It is reasonable that node ℓ should use this information, and weight the local cost function accordingly. The smaller $a_{\ell k}$ is, the smaller $c_{k\ell}$ should be because nodes k and ℓ do not address the same estimation problem. Other strategies, in the spirit of (103), may be considered to estimate the coefficients $c_{k\ell}$. Simulations will however confirm the effectiveness of (107). The ATC diffusion algorithm with adaptive clustering defined by time-variant combination matrices $\boldsymbol{A}(n)$ and $\boldsymbol{C}(n)$ is finally summarized in Algorithm 1.

Algorithm 1: ATC Diffusion LMS with adaptive clustering for multitask problems

Parameters: Preset non-negative step sizes μ_k for all nodes.

Initialization: Set the combination matrices $\boldsymbol{C}(0) = \boldsymbol{I}_N$ and $\boldsymbol{A}(0) = \boldsymbol{I}_N$.

Set initial weights $\boldsymbol{w}_k(0) = 0$ for all $k = 1, ..., N$.

Algorithm: At each time instant $n \geq 1$, and for each node k, update $\boldsymbol{\psi}_k(n+1)$:

$$\boldsymbol{\psi}_k(n+1) = \boldsymbol{w}_k(n) + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}(n) \left[d_\ell(n) - \boldsymbol{x}_\ell^\top(n)\boldsymbol{w}_k(n)\right] \boldsymbol{x}_\ell(n) \tag{108}$$

Update the combination coefficients:

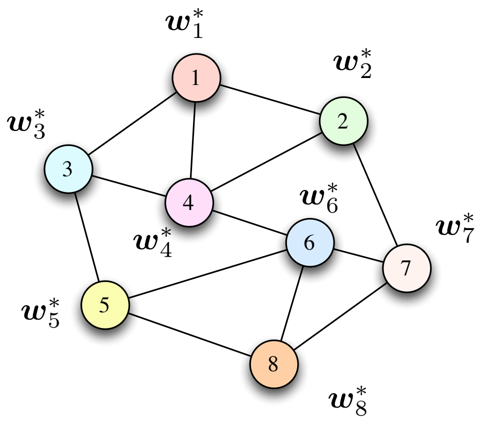
$$\begin{aligned} \boldsymbol{q}_k(n) &= \left[d_k(n) - \boldsymbol{x}_k^\top(n)\boldsymbol{\psi}_k(n+1)\right] \boldsymbol{x}_k(n) \\ a_{\ell k}(n+1) &= \frac{\|\boldsymbol{\psi}_k(n+1) + \mu_k\, \boldsymbol{q}_k(n) - \boldsymbol{\psi}_\ell(n+1))\|^{-2}}{\sum_{j \in \mathcal{N}_k} \|\boldsymbol{\psi}_k(n+1) + \mu_k\, \boldsymbol{q}_k(n) - \boldsymbol{\psi}_j(n+1))\|^{-2}} \\ \text{Optional:} &\quad c_{k\ell}(n+1) = a_{\ell k}(n+1) \end{aligned} \tag{109}$$

Combine weights:

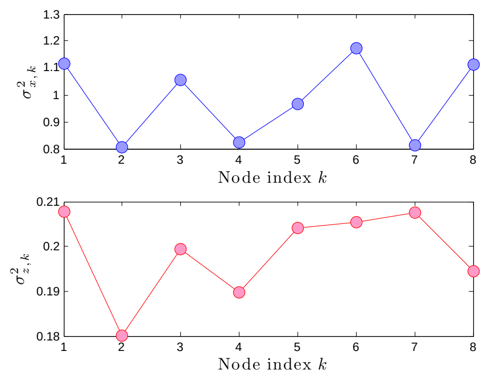
$$\boldsymbol{w}_k(n+1) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k}(n+1)\, \boldsymbol{\psi}_k(n+1) \tag{110}$$

V. Simulations

In this section, we first conduct simulations on a simple network in order to illustrate the accuracy of our mean weight and mean-square error behavior analyses. Next, we apply the adaptive clustering strategy on a multitask network with an inherent cluster structure, to illustrate its performance and effectiveness. Finally, an application-oriented example focused on multi-target



(a) Network topology.



(b) Input variances (top) and noise variances (bottom).

Fig. 1. (a) Network studied in Section V-A, with 8 nodes. (b) Input signal and noise variances for each sensor node.

tracking is considered to show how the proposed algorithm performs. In what follows, we consider the ATC diffusion LMS algorithm, that is, $\boldsymbol{A}_1 = \boldsymbol{I}_N$. In order to simplify the notation, we denote by $\boldsymbol{A}$ the combination matrix. All nodes are initialized with zero parameter vectors $\boldsymbol{w}_k(0)$, and all simulated curves are obtained by averaging over 100 Monte-Carlo realizations.

A. *Model validation*

We shall now provide simulation results to validate the theoretical derivations, and to discuss the behavior of diffusion LMS run in a multitask network. For simplicity, we consider a small network consisting of 8 nodes with interconnections shown in Figure 1 (left). The parameter vectors to be estimated are of length $L = 2$. The optimum mean vectors are uniformly distributed on a circle of radius r centered at $\boldsymbol{w}_o$, that is,

$$\begin{aligned}\boldsymbol{w}_k^\star &= \boldsymbol{w}_o + r\begin{pmatrix}\cos\theta_k\\ \sin\theta_k\end{pmatrix}\\ \theta_k &= 2\pi(k-1)/N + \pi/8\end{aligned} \tag{111}$$

The regression inputs $\boldsymbol{x}_k(n)$ were zero-mean 2×1 random vectors governed by a Gaussian distribution with covariance matrices $\boldsymbol{R}_{x,k} = \sigma^2_{x,k}\,\boldsymbol{I}_L$, and $\sigma^2_{x,k}$ shown on the top of Figure 1. The background noises $z_k(n)$ were i.i.d. zero-mean Gaussian random variables, and independent of any other signal. The corresponding variances $\sigma^2_{z,k}$ are depicted at the bottom of Figure 1. We considered the ATC diffusion LMS with measurement diffusion governed by a uniform matrix $\boldsymbol{C}$ such that $c_{\ell k} = |\mathcal{N}_k|^{-1}$ for all $\ell \in \mathcal{N}_k$. The combination matrix $\boldsymbol{A}$ simply averaged the estimates from the neighbors, that is, $a_{\ell k} = |\mathcal{N}_k|^{-1}$ for $\ell \in \mathcal{N}_k$. The step sizes were set to $\mu_k = 0.01$ for all nodes.

1) Stationary optimums: We first check the convergence analysis with stationary parameter vectors, that is, $\sigma^2_{\epsilon,k} = 0$ for all nodes. Four groups of coefficient vectors, centered at $\boldsymbol{w}_o = [1, -0.5]^\top$ with $r = 0$, $r = 0.03$, $r = 0.05$ and $r = 0.1$ were considered, as illustrated in Figure 2. Note that the case $r = 0$ corresponds to the single-task network where $\boldsymbol{w}_k^\star = \boldsymbol{w}_o$ for each node. Running ATC diffusion LMS with these four settings, we obtained the MSD curves shown in Figure 3. Observe that the theoretical and simulated transient MSD curves are accurately superimposed. The non-cooperative LMS algorithm was also considered. Since the average steady-state MSD of the non-cooperative LMS algorithm over all nodes is approximately

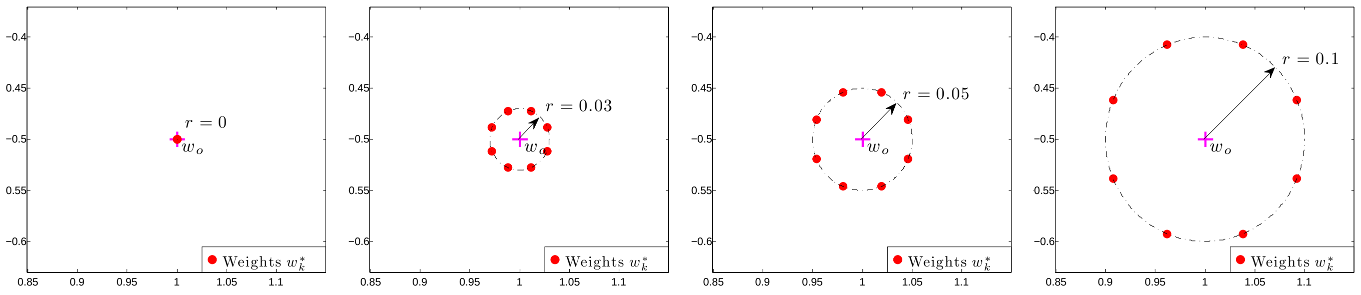


Fig. 2. Node coefficients centered at $\boldsymbol{w}_o = [1, -0.5]^\top$ with $r = 0$, $r = 0.03$, $r = 0.05$ and $r = 0.1$ (from left to right).

given by [38], [39]:

$$\text{MSD}_{\text{lms}}^{\text{network}} = \frac{1}{N}\sum_{k=1}^{N} \frac{\mu_k\, \sigma_{z,k}^2\, L}{2}, \tag{112}$$

then the MSD behavior with the different settings is almost the same, provided the other parameters remain unchanged. Consequently, the theoretical MSD curve for the non-cooperative LMS algorithm is only provided for $r = 0.05$. It can be observed that diffusion LMS can still be advantageous over non-cooperative LMS if the differences between local optimum weight vectors are sufficiently small, $r = 0$ and $r = 0.03$ in this simulation. However, when the contrast between the tasks increases, diffusion LMS provides lower performance than non-cooperative LMS due to the bias introduced by the algorithm, $r = 0.05$ and $r = 0.1$ in this simulation. We also provide the mean weight behavior of diffusion LMS and non-cooperative LMS for $r = 0.05$ in Figure 4. The Pareto optimal solution can be calculated as

$$\begin{aligned}\boldsymbol{w}_{\text{Pareto}}^\star &= \arg\min_{\boldsymbol{w}} \sum_{k=1}^{N} E\{|d_k(n) - \boldsymbol{x}_k^\top(n)\,\boldsymbol{w}|^2\}\\ &= \frac{\sum_{k=1}^{N}\sigma_{x,k}^2\;\boldsymbol{w}_k^\star}{\sum_{k=1}^{N}\sigma_{x,k}^2}\\ &= [1.011\;\; -0.5011]^\top\end{aligned} \tag{113}$$

Figure 4 (left) shows that diffusion LMS converged to $\boldsymbol{w}_{\text{Pareto}}^\star$ with a bias on the order of $\mathcal{O}(\mu_{\max})$ [21]. Figure 4 (right) shows that non-cooperative LMS converged towards the multiple optimums without bias.

2) Randomly perturbed optimums: We now consider the network described previously with $r = 0$ so that the differences between the optimum weight vectors $\boldsymbol{w}_k^\star(n)$ arise from the random perturbations $\boldsymbol{\epsilon}_k(n)$. Keeping all the other parameters unchanged, the variance of these perturbations was successively set to $\sigma_\epsilon^2 = 0$, 0.01, 0.05 and 0.1 for all the agents. MSD curves for diffusion LMS and non-cooperative LMS are provided in Figure 5. It can be observed that diffusion LMS always outperformed its non-cooperative counterpart. This experiment shows the advantage provided by cooperation. The relative performance gain becomes smaller as σ_ϵ^2 increases because weight lags caused by random perturbations dominate the estimation error.

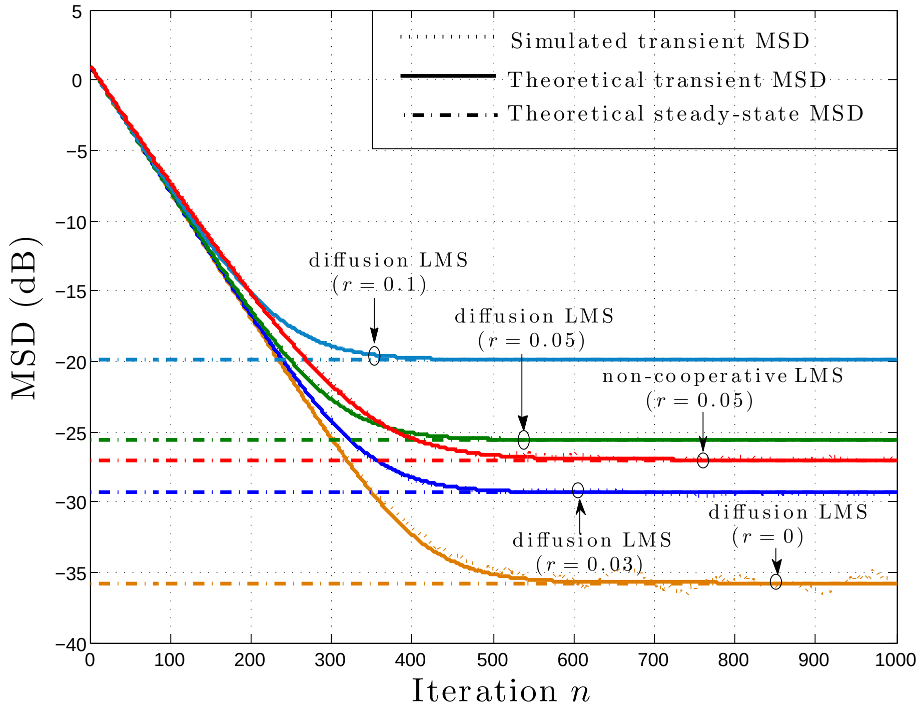


Fig. 3. Network MSD behavior for the deterministic case. Theoretical MSD curves were obtained by Theorem 3 and steady-state MSD values were obtained by Theorem 4. Simulated and theoretical transient MSD curves are perfectly superimposed.

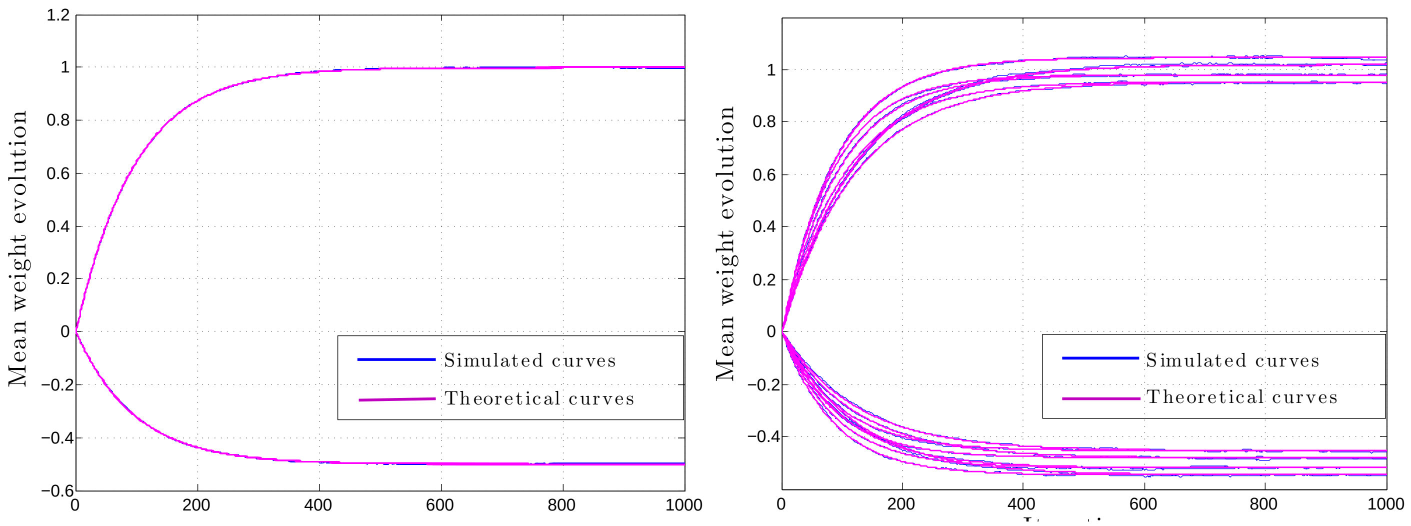


Fig. 4. Mean weight behavior for diffusion LMS (left) and non-cooperative LMS (right) with $r = 0.05$.

3) General settings: To check the convergence analysis in the case where tasks differ by both their mean value and random perturbations, diffusion LMS was tested in the following experimental settings pictorially described in Figure 6:

$$\begin{aligned} &\mathcal{S}_1 : \{r = 0.01, \sigma_\epsilon = 0.01\} \\ &\mathcal{S}_2 : \{r = 0.05, \sigma_\epsilon = 0.015\} \qquad (114) \\ &\mathcal{S}_3 : \{r = 0.1, \sigma_\epsilon = 0.015\} \end{aligned}$$

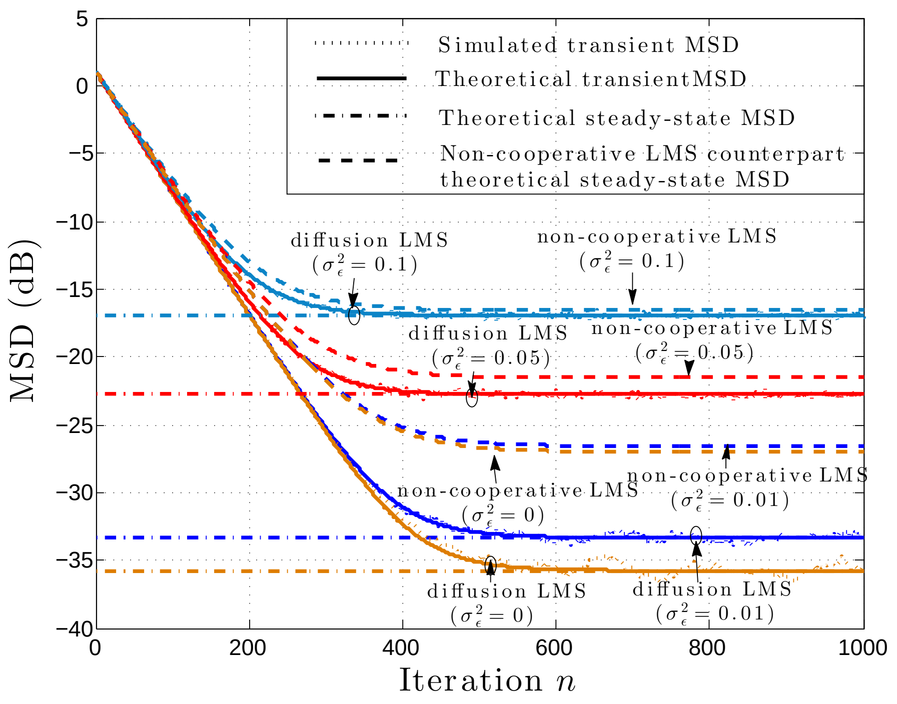


Fig. 5. Network MSD behavior for the perturbation-only case. Theoretical MSD curves were obtained by Theorem 3 and steady-state MSD values were obtained by Theorem 4. Note that simulated and theoretical transient MSD curves are superimposed.

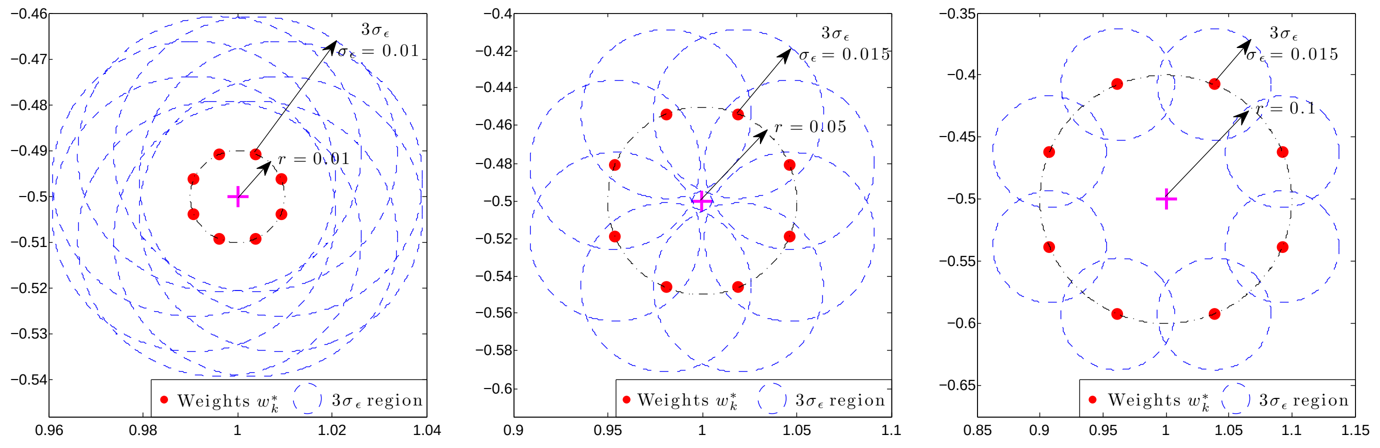


Fig. 6. Node coefficients centered at $\boldsymbol{w}_o = [1, -0.5]^\top$ with r and σ_ϵ^2 defined by $\mathcal{S}_1$ to $\mathcal{S}_3$ (from left to right).

The MSD curves shown in Figure 7 illustrates the validity of the analysis.

B. Adaptive clustering in multitask networks

We shall now illustrate the performance of diffusion LMS with adaptive clustering in a multitask environment. Our approach is compared with the strategy introduced in [35]. For the latter, as suggested in [35], the so-called smoothing factor γ was set to 0.1. A stationary problem is first considered. Next, a dynamic problem with time-varying clusters is introduced in order to confirm the reliability of our approach.

1) Stationary environment: Consider the network of 16 agents depicted in Figure 8(a). The regression inputs $\boldsymbol{x}_k(n)$ were zero-mean 2×1 random vectors governed by a Gaussian distribution with covariance matrices $\boldsymbol{R}_{x,k} = \sigma_{x,k}^2 \boldsymbol{I}_L$, and $\sigma_{x,k}^2$

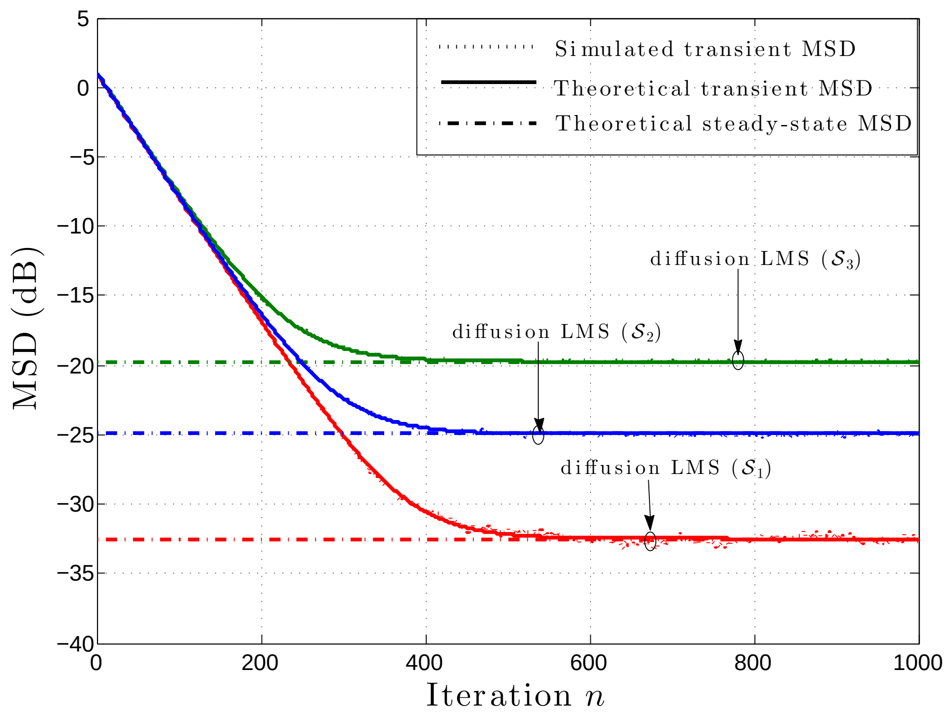


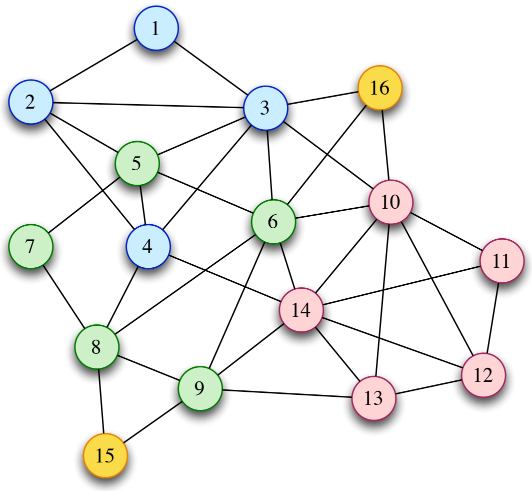
Fig. 7. Network MSD behavior for the general case with settings $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. Theoretical MSD curves were obtained by Theorem 3 and steady-state MSD values were obtained by Theorem 4. Simulated curves and theoretical curves are accurately superimposed.

shown at the top of Figure 8(b). The background noises $z_k(n)$ were i.i.d. zero-mean Gaussian random variables, independent of any other signals. The corresponding variances $\sigma_{z,k}^2$ are depicted at the bottom of Figure 8(b). The scenario under study is a multitask problem with a cluster structure. All agents in the same cluster share the same optimum weight vector. Nodes 1 to 4 belong to the first cluster. Nodes 5 to 9 are in the second cluster. Nodes 10 to 14 compose the third cluster, and nodes 15 and 16 are in the fourth cluster. The parameter vectors to be estimated are as follows:

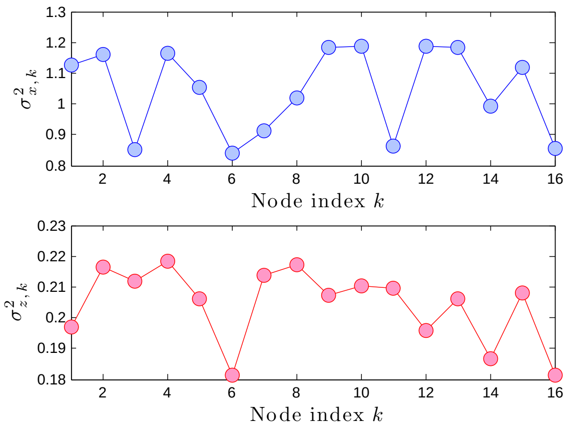
$$\boldsymbol{w}_k^\star = \begin{cases} [0.5\ \ -0.4]^\top & k = 1, \dots, 4 \qquad \text{Cluster 1} \\ [0.6\ \ -0.2]^\top & k = 5, \dots, 9 \qquad \text{Cluster 2} \\ [0.3\ \ -0.3]^\top & k = 10, \dots, 14 \qquad \text{Cluster 3} \\ [-0.8\ \ 0.5]^\top & k = 15, 16 \qquad \text{Cluster 4} \end{cases} \tag{115}$$

Note that optimums for cluster 1, 2 and 3 are quite close, whereas the optimum for cluster 4 acts as an abnormal interference. Moreover, although nodes 15 and 16 have the same parameter vector to estimate, they cannot cooperate since they are not connected. The following algorithms were considered for estimating the four optimum parameter vectors: 1) diffusion LMS with a uniform combination matrix $\boldsymbol{A}$, 2) non-cooperative LMS, 3) diffusion LMS with the clustering strategy introduced in [35], 4) diffusion LMS with our clustering strategy, with $\boldsymbol{C} = \boldsymbol{I}$ and $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$. The step size was set to $\mu = 0.01$ for all nodes.

Figure 9 presents the mean weight behavior for these algorithms. As expected, diffusion LMS with a uniform combination matrix made all nodes converge to the same Pareto optimum, so that it failed to determine multitask optimums. The non-cooperative LMS estimated the optimum weight vectors without bias. The algorithm from [35], which generally performs well for well-separated tasks and randomly well-separated initial weights, did not perform well in the setting under consideration. The proposed algorithm showed the same convergence behavior for $\boldsymbol{C} = \boldsymbol{I}$ and $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$. Only the case $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$ is presented here due to space limitation. It can be observed that, for each node, the parameter vector converged properly



(a) Network topology



(b) Input variances (top) and noise variances (bottom)

Fig. 8. Network topology in Section V-B1 and associated input variances and noise variances.

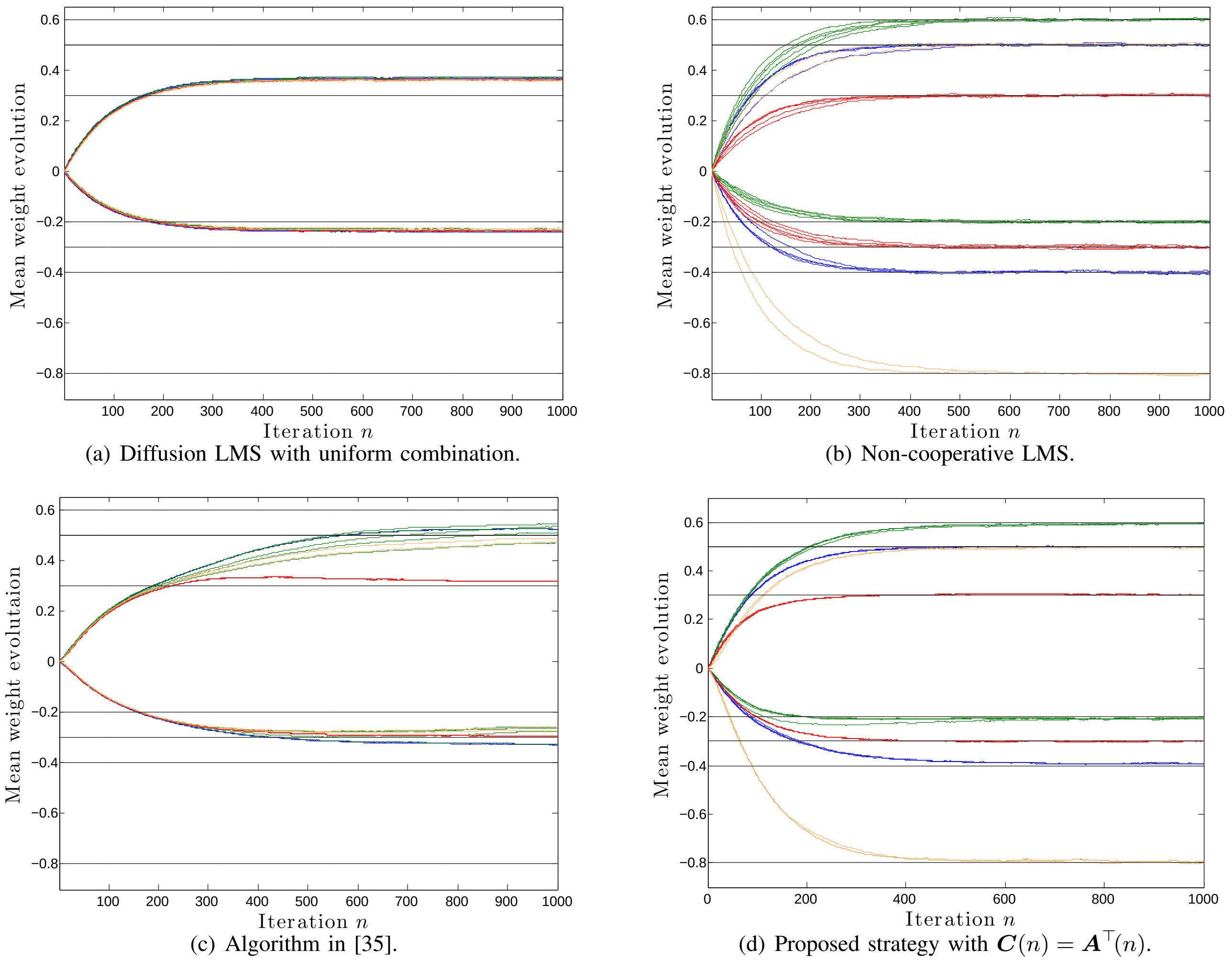
in accordance to the cluster structure. Figure 10(a) illustrates the MSD convergence behavior for these algorithms. Due to large bias of the estimated weights, diffusion LMS with a uniform combination matrix had large MSD. Non-cooperative LMS performed much better as it provides unbiased estimates. The proposed algorithm with $\boldsymbol{C} = \boldsymbol{I}$ achieved better performance, and $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$ led to additional performance gain due to information exchange. Finally, in order to provide a straightforward but visually-meaningful clustering result, we averaged the combination matrix $\boldsymbol{A}$ over the last 100 iterations of a single realization, and we considered that $a_{\ell k} > 0.05$ represents a one-way connection from ℓ to k. The estimated relationships between nodes provided in Figure 10(b) perfectly match the ground truth configuration.

2) Non-stationary environment: Consider now a more complex environment where clusters vary over time. Four stationary stages and three transient episodes were modeled in this experiment. Properties of input signals and noise were the same as those in the stationary case considered above. From instant $n = 1$ to 1000, the network consisted of one cluster with a unique optimum parameter vector to estimate. From $n = 1501$ to 2500, nodes were split into two clusters with two different optimums. From $n = 3001$ to 4000, nodes were split again to give four clusters. Finally, from instant $n = 4501$, nodes were aggregated into one cluster with a unique parameter vector to estimate, which was different from the first stage. Transient episodes were designed with linear interpolation between each steady-state stage over a period of 500 time samples. Taking, for example, the first component of the weight vector of node 1 over the time interval 1 to 2500, the time-variant optimum $w^\star_{1,1}(n)$ is expressed by

$$w^\star_{1,1}(n) = \begin{cases} 0.3, & n = 1, \ldots, 1000 \\ 0.3 + \frac{0.5-0.3}{500}(n-1000), & n = 1001, \ldots, 1500 \\ 0.5, & n = 1501, \ldots, 2500 \end{cases} \tag{116}$$

Cluster structures and optimum parameter vectors are illustrated in Figures 11 and 12, respectively.

The same four algorithms as before were considered for comparison. MSD learning curves are shown in Figure 14. Transient stages can be clearly observed on both weight behavior and MSD behavior curves. Diffusion LMS enforced the weight vectors estimated by each agent to converge to the same solution at each stage. As a consequence, the MSD learning curve shows poor



(a) Diffusion LMS with uniform combination.

(b) Non-cooperative LMS.

(c) Algorithm in [35].

(d) Proposed strategy with $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$.

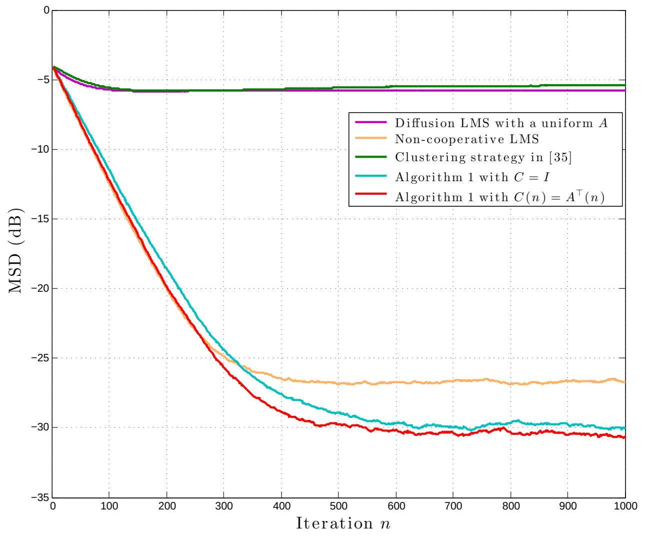
Fig. 9. Mean weight behavior of the various algorithms. Colors of curves are consistent with cluster colors in Figure 8(a). Black horizontal lines represent ground-truth optimum values. The weight iterates were initialized at the same value in plot (c). If random well-separated initial conditions are used across the nodes, then the performance of (c) becomes similar to that of (b).

performance due to large bias. Non-cooperative LMS converged without bias towards the optimum parameter vectors. The algorithm introduced by [35] showed some ability to conduct clustering but did not provide satisfactory results during transient episodes. During stages 1 and 4, it worked as well as diffusion LMS. However, during stages 2 and 3, it only performed slightly better than diffusion LMS. The proposed algorithm was able to track the system dynamic with correct clustering and appropriate convergence in the mean-square sense.

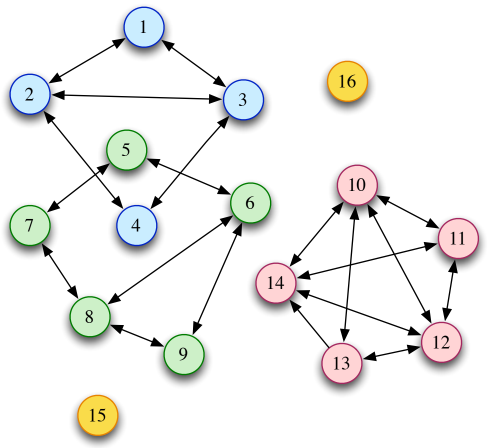
C. Multi-target tracking with sensor networks

Consider now a target tracking problem to illustrate our adaptive clustering strategy with diffusion LMS. We focused on a scenario involving four targets, numbered from $i = 1$ to 4, moving according to the state-transition equation

$$\boldsymbol{x}_i(n+1) = \boldsymbol{T}_i^\star\, \boldsymbol{x}_i(n) + \boldsymbol{z}_i(n) \qquad \text{for } \; i = 1, \ldots, 4, \quad n = 0, \ldots, 100. \tag{117}$$



(a) MSD behavior.



(b) Estimated cluster structure.

Fig. 10. Network MSD comparison in a stationary multitask environment, and estimated cluster structure by the proposed algorithm (averaged over the last 100 instants in one realization).

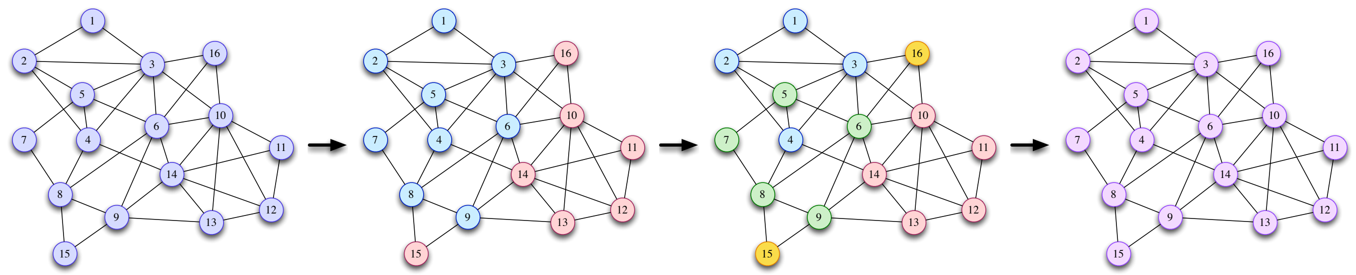


Fig. 11. Evolution of cluster structures of the network (1 cluster $\rightarrow$ 2 clusters $\rightarrow$ 4 clusters $\rightarrow$ 1 cluster).

where $\boldsymbol{x}_i(n)$ denotes the 2-dimensional coordinates for target i at instant n. Matrices $\boldsymbol{T}_i$ are 2×2 state-transition matrices defined as

$$\boldsymbol{T}_1^\star = \begin{pmatrix} 1.0019 & -0.0129 \\ 0.0187 & 1.0034 \end{pmatrix}, \quad \boldsymbol{T}_2^\star = \begin{pmatrix} 1.0149 & -0.0014 \\ 0.0033 & 1.0034 \end{pmatrix}, \quad \boldsymbol{T}_3^\star = \begin{pmatrix} 1.0128 & -0.0041 \\ 0.0156 & 1.0086 \end{pmatrix}, \quad \boldsymbol{T}_4^\star = \boldsymbol{T}_1^\star \tag{118}$$

and $\boldsymbol{z}_i(n)$ is the modeling error with i.i.d. zero-mean Gaussian distribution with covariance matrix $\sigma_z^2\,\boldsymbol{I}$. The standard deviation was set to $\sigma_z = 0.01$. The initial coordinates for the four targets were

$$\boldsymbol{x}_1(0) = \boldsymbol{x}_2(0) = \boldsymbol{x}_3(0) = [1 \ -1]^\top, \qquad \boldsymbol{x}_4(0) = [1.2 \ -1.2]^\top \tag{119}$$

Transition matrices (118) and initial coordinates (119) show that targets 1, 2 and 3 started from the same location with movements governed by distinct state-transition equations, while targets 1 and 4 had movements governed by the same state-transition equation but started from different positions. Figure 15(a) shows the trajectories of the four targets from instant $n = 0$ to 100. A network with $N = 100$ nodes was randomly deployed in a given area, with physical connections defined by the connectivity matrix in Figure 16(a).

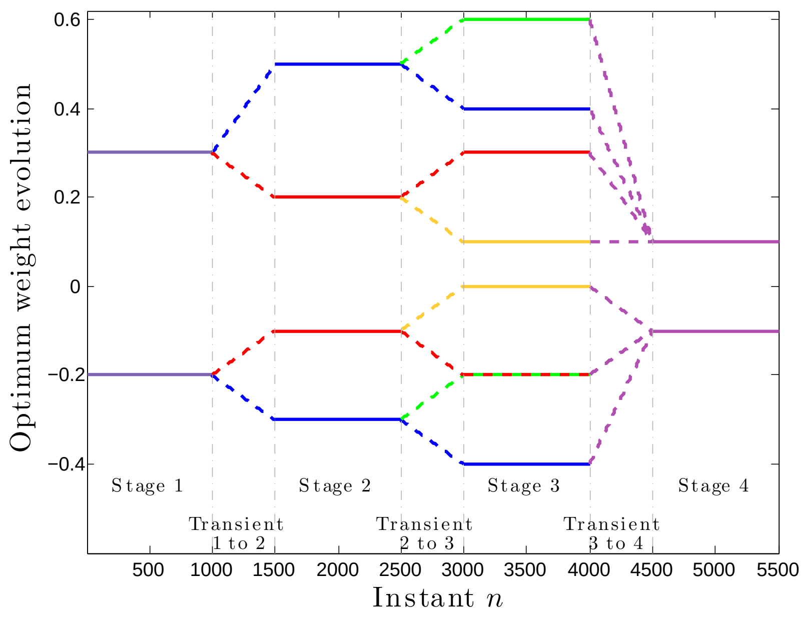


Fig. 12. Evolution of clusters. Colors are consistent with those of clusters in Figure 11. Dashed lines represent optimums during transient episodes.

We supposed that each node was able to track only one target during the experiment, with noisy observations $\tilde{\boldsymbol{x}}_k(n)$:

$$\tilde{\boldsymbol{x}}_k(n) = \boldsymbol{x}_k(n) + \boldsymbol{u}_k(n) \qquad \text{for } \; k = 1, \ldots, N \tag{120}$$

with $\boldsymbol{u}_k(n)$ an i.i.d. zero-mean Gaussian observation noise with covariance matrix $\sigma_u^2 \boldsymbol{I}$ and standard deviation $\sigma_u = 0.01$. For ease of presentation, we assume that nodes 1–25 tracked target 1, nodes 26–50 tracked target 2, nodes 51–75 tracked target 3, and nodes 76–100 tracked target 4.

Considering $\tilde{\boldsymbol{x}}(n)$ as input data and $\tilde{\boldsymbol{x}}(n+1)$ as desired output data for the learning algorithm, each nodes aimed to track a target or, equivalently, to estimate its transition matrix given input-output noisy data. Without cooperation, this task can be performed by each node k by minimizing the following cost function with respect to matrix $\boldsymbol{T}_k$:

$$J_k(\boldsymbol{T}_k) = E\|\tilde{\boldsymbol{x}}_k(n+1) - \boldsymbol{T}_k \, \tilde{\boldsymbol{x}}_k(n)\|^2 \qquad \text{for } k = 1, \ldots, N \tag{121}$$

Collaboration among nodes may be beneficial as several nodes are conducting the same task, including nodes that track the same target and nodes that track distinct targets with the same state-transition matrix. Clearly, diffusion LMS with a uniform combination matrix is not suitable within this context since neighboring nodes may not have the same task to conduct. This problem requires adaptive clustering to automatically aggregate nodes that perform a similar task.

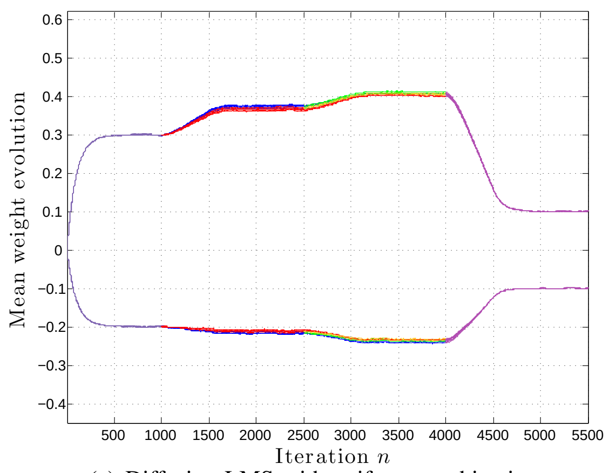
Algorithm 1 was run with $\boldsymbol{C} = \boldsymbol{I}_N$. Because matrices $\boldsymbol{T}_k$ have to be estimated, each node conducted the following steps

Instantaneous gradient:
$$\boldsymbol{Q}_k(n) = \tilde{\boldsymbol{x}}_k(n) \, [\tilde{\boldsymbol{x}}_k(n+1) - \boldsymbol{T}_k(n) \, \tilde{\boldsymbol{x}}_k(n)]^\top \tag{122}$$

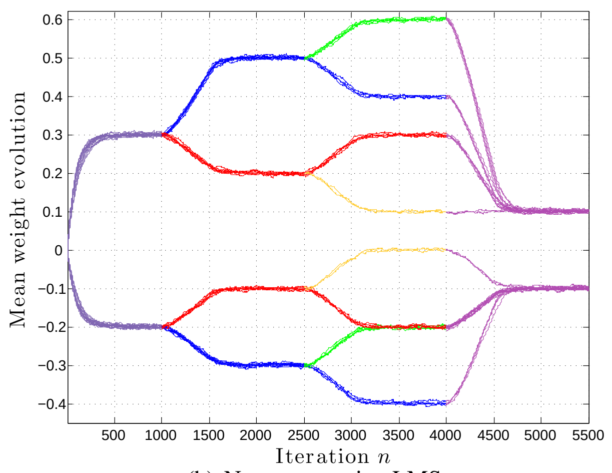
Local update:
$$\boldsymbol{\Psi}_k(n+1) = \boldsymbol{\Psi}_k(n) + \mu \, \boldsymbol{Q}_k(n) \tag{123}$$

Combination coefficients:
$$a_{\ell k}(n) = \frac{\|\boldsymbol{\Psi}_k(n+1) + \mu \, \boldsymbol{Q}_k(n) - \boldsymbol{\Psi}_\ell(n+1))\|_F^{-2}}{\sum_{j \in \mathcal{N}_k} \|\boldsymbol{\Psi}_k(n+1) + \mu_k \, \boldsymbol{Q}_k(n) - \boldsymbol{\Psi}_j(n+1))\|_F^{-2}} \tag{124}$$

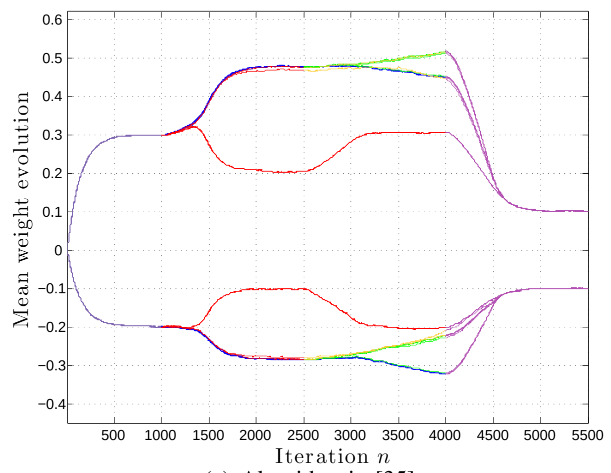
Combination:
$$\boldsymbol{T}_k(n+1) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k}(n) \, \boldsymbol{\Psi}_k(n+1) \tag{125}$$



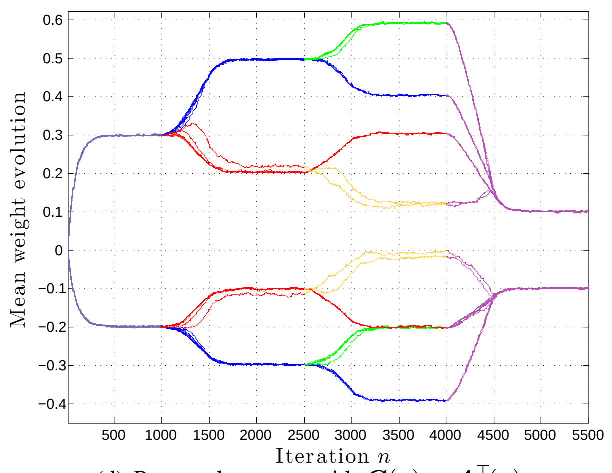
(a) Diffusion LMS with uniform combination.



(b) Non-cooperative LMS.



(c) Algorithm in [35].



(d) Proposed strategy with $\boldsymbol{C}(n) = \boldsymbol{A}^\top(n)$.

Fig. 13. Mean weight behavior of various algorithms in the non-stationary environment. Colors are consistent with cluster colors in Figure 8(a).

The algorithm was initialized with $\boldsymbol{T}_k(0) = \boldsymbol{I}_2$, and the step size μ was set to $\mu = 0.05$. Figure 15(b) shows the MSD learning curves of transition matrices estimated by non-cooperative LMS, and diffusion LMS with adaptive clustering strategy, averaged over 100 Monte-Carlo runs. The performance gain can be clearly seen from these figures. Figure 16(b) shows the connectivity matrix determined by the clustering strategy at iteration $n = 100$. To construct this figure, we considered that 2 nodes are connected if $a_{\ell k} > 0.05$. It can be seen that connections are distributed in 4 blocks on the diagonal, each one corresponding to a target, and 2 other blocks (upper-right and lower-left ones) where nodes track two distinct targets with the same state-transition matrix. This result is consistent with the experimental setup and shows the effectiveness of our clustering strategy.

VI. Conclusion and perspectives

Diffusion LMS algorithm is an efficient strategy to address distributed optimization problems over networks in the case where nodes have to collaboratively estimate a single parameter vector. However, many problems of interest happen to be multitask-oriented in the sense that there are multiple optimum parameter vectors to be inferred simultaneously and in a collaborative manner. In this paper, we studied the performance of the diffusion LMS algorithm when it is run, either intentionally or unintentionally, in a multitask environment. Accurate mean weight behavior model and mean square deviation model were

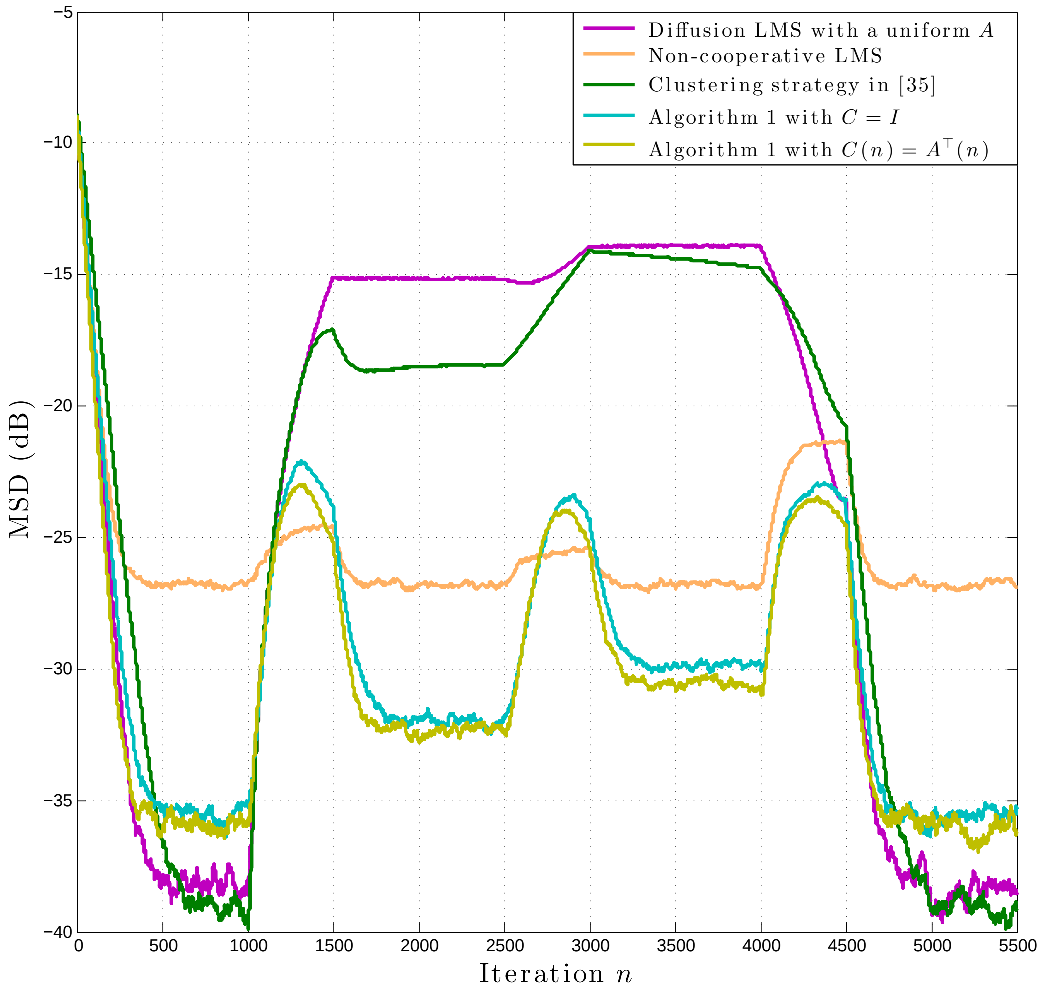


Fig. 14. Network MSD behavior comparison in the time variant multitask environment.

derived. Next, we proposed an unsupervised clustering strategy that allows each node to select, via adaptive adjustments of combination weights, the neighboring nodes with which it can collaborate to address a given task. Simulations were presented to demonstrate the efficiency of the proposed clustering strategy.

Several open problems still have to be solved for specific applications. For instance, it would be interesting to show which regularization can be advantageously used with our adaptive clustering strategy, and how they can be efficiently implemented in an adaptive manner. It would also be interesting to investigate asynchronous scenarios where adaptive multitask networks have to handle critical events such as random link failures, random data arrival times, and agents turning on and off randomly.

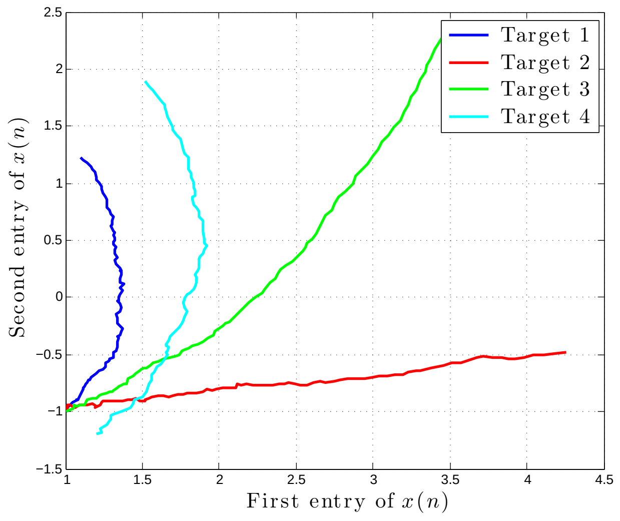
References

[1] J. Chen and C. Richard, "Performance analysis of diffusion LMS in multitask networks," in *Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Saint Martin, France, December 2013, pp. 137–140.

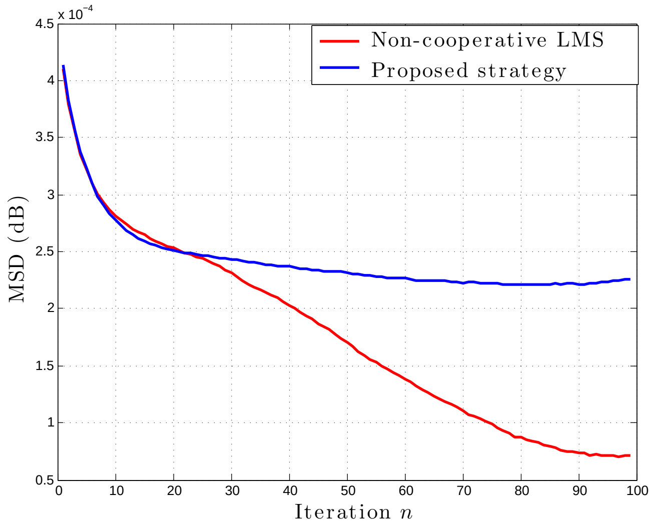
[2] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.

[3] J. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Transactions on Automatic Control*, vol. 29, no. 1, pp. 42–50, January 1984.

[4] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *System & Control letters*, vol. 53, no. 9, pp. 65–78, September 2004.

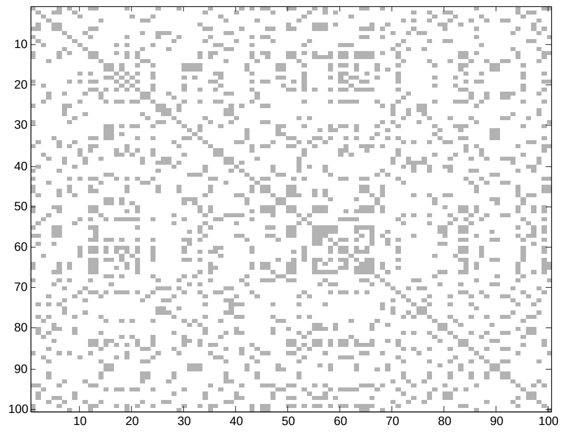


(a) Target trajectories.

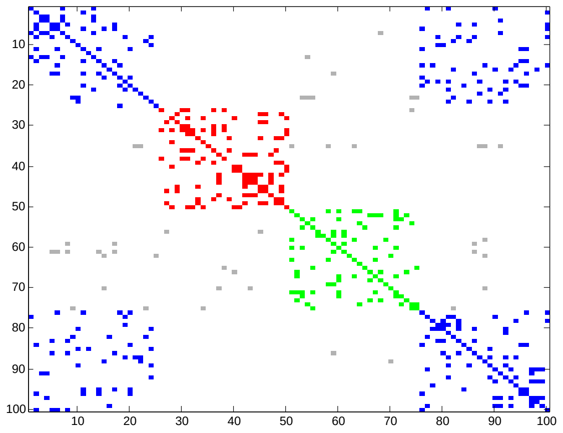


(b) MSD learning curves.

Fig. 15. (a) Trajectories of four targets with initial coordinates $\boldsymbol{x}_1(0) = \boldsymbol{x}_2(0) = \boldsymbol{x}_3(0) = [1, -1]^\top, \boldsymbol{x}_4(0) = [1.2, -1.2]^\top$, from $n = 0$ to 100. (b) Comparison of MSD learning curves for the estimate $\boldsymbol{T}_k$.



(a) Network physical connection matrix.



(b) Connections selected collaboratively by the algorithm.

Fig. 16. (a) Initial connectivity matrix, where gray elements represent physical connections. (b) Connectivity matrix resulting from our clustering strategy. Blue elements correspond to the connections used to estimate the transition matrices $\boldsymbol{T}_1^\star = \boldsymbol{T}_4^\star$. Red and green elements correspond to the connections used to estimate the transition matrices $\boldsymbol{T}_2^\star$ and $\boldsymbol{T}_3^\star$, respectively. Gray elements can be considered as false connections because they involve nodes that do not estimate the same transition matrix.

[5] P. Braca, S. Marano, and V. Matta, "Running consensus in wireless sensor networks," in *Proc. International Conference on Information Fusion (FUSION)*, Cologne, Germany, June-July 2008, pp. 1–6.

[6] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, January 2009.

[7] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Link failures and channel noise," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 355–369, January 2009.

[8] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing,"

Proceedings of the IEEE, vol. 98, no. 11, pp. 1847–1864, November 2010.

[9] K. Srivastava and A. Nedic, “Distributed asynchronous constrained stochastic optimization,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, August 2011.

[10] D. P. Bertsekas, “A new class of incremental gradient methods for least squares problems,” *SIAM Journal on Optimization*, vol. 7, no. 4, pp. 913–926, November 1997.

[11] A. Nedic and D. P. Bertsekas, “Incremental subgradient methods for nondifferentiable optimization,” *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, July 2001.

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

[13] D. Blatt, A. O. Hero, and H. Gauchman, “A convergent incremental gradient method with constant step size,” *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, February 2007.

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

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

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

[17] S.-Y. Tu and A. H. Sayed, “Diffusion strategies outperform consensus strategies for distributed estimation over adaptive netowrks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, December 2012.

[18] J. Chen and A. H. Sayed, “Diffusion adaptation strategies for distributed optimization and learning over networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, August 2012.

[19] A. Khalili, M. A. Tinati, A. Rastegarnia, and J. A. Chambers, “Steady-state analysis of diffusion LMS adaptive networks with noisy links,” *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 974–979, February 2012.

[20] X. Zhao, S.-Y. Tu, and A. H. Sayed, “Diffusion adaptation over networks under imperfect information exchange and non-stationary data,” *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3460–3475, July 2012.

[21] J. Chen and A. H. Sayed, “Distributed Pareto optimization via diffusion strategies,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 205–220, April 2013.

[22] O. N. Gharehshiran, V. Krishnamurthy, and G. Yin, “Distributed energy-aware diffusion least mean squares: Game-theoretic learning,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 5, pp. 1–16, October 2013.

[23] S. Chouvardas, K. Slavakis, and S. Theodoridis, “Adaptive robust distributed learning in diffusion sensor networks,” *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4692–4707, October 2011.

[24] Y. Liu, C. Li, and Z. Zhang, “Diffusion sparse least-mean squares over networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4480–4485, August 2012.

[25] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, “A sparsity-promoting adaptive algorithm for distributed learning,” *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5412–5425, October 2012.

[26] P. Di Lorenzo and A. H. Sayed, “Sparse distributed learning based on diffusion adaptation,” *IEEE Transactions on Signal Processing*, vol. 61, no. 6, pp. 1419–1433, March 2013.

[27] P. Chainais and C. Richard, “Distributed dictionary learning over a sensor network,” in *Proc. Conférence sur l’Apprentissage Automatique (CAP)*, Lille, France, July 2013, pp. 1–6.

[28] P. Chainais and C. Richard, “Learning a common dictionary over a sensor network,” in *Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Saint Martin, France, December 2013, pp. 1–4.

[29] J. Predd, S. Kulkarni, and H. Vincent Poor, “Distributed learning in wireless sensor networks,” *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 59–69, July 2006.

[30] P. Honeine, M. Essoloh, C. Richard, and H. Snoussi, "Distributed regression in sensor networks with a reduced-order kernel model," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, New Orleans, LO, USA, 2008, pp. 1–5.

[31] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for leaning shared structures from muliple tasks," in *Proc. 26th Annual International Conference on Machine Learning (ICML)*, Montreal, Canada, Juin 2009, pp. 137–144.

[32] O. Chapelle, P. Shivaswmy, K. Q. Vadrevu, S. Weinberger, Y. Zhang, and B. Tseng, "Multi-task learning for boosting with application to web search ranking," in *Proc. 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington DC, USA, July 2010, pp. 1189–1198.

[33] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, USA, August 2011, pp. 814–822.

[34] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion LMS over networks," *Submitted for publication*, Also available as arXiv:1311.4894 [cs.MA], November 2013.

[35] X. Zhao and A. H. Sayed, "Clustering via diffusion adaptation over networks," in *Proc. International Workshop on Cognitive Information Processing (CIP)*, Parador de Baiona, Spain, May 2012, pp. 1–6.

[36] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Libraray in Signal Processing*, R. Chellapa and S. Theodoridis, Eds., pp. 322–454. Elsevier, 2014. Also available as arXiv:1205.4220 [cs.MA], May 2012.

[37] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge university press, 2004.

[38] A. H. Sayed, S.-Y Tu, J. Chen, X. Zhao, and Z. Towfic, "Diffusion strategies for adaptation and learning over networks," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, May 2013.

[39] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, NJ, 2008.

[40] K. M. Abadir and J. R. Magnus, *Matrix Algebra*, Cambridge Univsersity Press, 2005.

[41] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas (Second Edition)*, Princeton University Press, 2009.